Université du Québec
Institut National de la Recherche Scientifique
Centre Énergie Matériaux Télécommunications

Human-Agent-Robot Teamwork Coordination in FiWi Based Tactile Internet Infrastructures

By

Mahfuzulhoq Chowdhury

Dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy (Ph.D.) in Telecommunications

December, 2018

Thesis Evaluation Committee

External Examiner	Prof. Christine Tremblay (École de Technologie Supérieure)
External Examiner	Prof. Zbigniew Dziong (École de Technologie Supérieure)
Internal Examiner	Prof. Jean-Charles Grégoire (INRS ÉMT)
Research Director	Prof. Martin Maier (INRS ÉMT)

This work is dedicated to my parents and teachers for their guidance, love, and encouragement throughout my life.

Acknowledgements

I want to express my sincere and wholehearted thanks to my adviser Prof. Martin Maier, who always gave me excellent guidance and support of my research project with brilliant ideas, encouragement, and great advice during my learning process. Without his excellent supervision, this thesis would not have been possible. I feel very fortunate to have had the chance to work under his guidance. I would like to extend my gratitude to Prof. Christine Tremblay, Prof. Zbigniew Dziong, and Prof. Jean-Charles Grégoire for agreeing to serve on my thesis evaluation committee and for their valuable observations. Finally, I would like to express my gratitude to the people of INRS, who helped me in some way. Thank you all!

Abstract

The demands of increasingly latency-sensitive applications create challenges for pervasive mobile devices/robots to execute the involved computation-intensive tasks in a resource-efficient manner. Cooperative human-agent-robot teamwork (HART) holds promise to serve as a powerful paradigm to tackle the challenges of real-time task execution of mobile devices/robots. Integrated fiber-wireless (FiWi) enhanced networks play a pivotal role in ensuring quality-of-service (QoS) for several HART-centric applications due to their coverage and capacity advantages. In this work, integrated FiWi access networks consist of optical fiber (Ethernet passive optical network or EPON) and wireless (wireless local area network or WLAN) Ethernet technologies, which are integrated with their cellular counterparts, namely, 4G Long Term Evolution Advanced (LTE-A), to give rise to FiWi enhanced LTE-A heterogeneous networks (HetNets).

To unleash the full potential of HART task coordination over FiWi enhanced 4G networks, this thesis first provides a detailed study of recent progress, enabling technologies, and briefly describes important open research challenges. To render the human-to-robot task allocation process more efficient, this thesis presents a local and non-local human-to-robot task allocation scheme for FiWi-based infrastructures according to several key design parameters such as the availability, skill set, distance to task location, minimum task processing time, and remaining energy of robots. To reduce failures during task execution, a neighboring robot assisted failure reporting mechanism is also proposed. Our obtained results show that, compared with traditional priority-based schemes, a task execution time efficiency of 18% can be achieved in our proposed local and non-local human-to-robot task allocation scheme.

Due to limited computing, energy, and storage resources, robots can not always meet the task execution time and energy consumption requirements of many delay-sensitive applications. To improve the energy efficiency of the selected host robot while satisfying a given task deadline, this thesis presents a collaborative task execution scheme, in which the sensing sub-task is conducted by a suitable host robot and the computation sub-task is offloaded onto one of the suitable collaborative nodes consisting of central cloud, cloudlets, and neighboring robots. The presented results demonstrate that for a typical task input size of 240 KB, the collaborative task execution scheme decreases the task response time by up to 8.75% and the

energy consumption by up to 14.98% compared to the only host robot based non-collaborative task execution scheme.

Taking the idea of task offloading a step further, task migration among mobile HART members has emerged as an important research topic to improve the quality of experience (QoE) of mobile users (MUs) by minimizing their task execution time. Task migration broadens the scope of conventional computation task offloading by not only transferring the task from an MU onto the cloud, but also from one cloud server to another one for execution. Note, however, that task migration incurs an additional migration delay. Hence, for a given task migration gain and latency overhead, the question of how and where an MU's task should migrate to is key. After describing the key features of physical vs. cognitive tasks and collaborative robot (cobot) vs. stand-alone robot types, this thesis next investigates the problem of whether and, if so, when and where a HART-centric task should be best migrated to. For resource-efficient task execution, a context-aware task migration scheme is presented, in which the suitable task migration decision is made by taking into account given task processing capabilities of cloud/cloudlet agents and cobots, task execution deadline, user mobility, energy consumption of involved collaborative robots (cobots) and mobile devices, and task migration latency. Our obtained results show that for a typical task input data size of 600 MB, the cobot-to-agent (c2a) task migration (cloudlet near task location) scheme exhibits up to 20% task response time and 23% energy efficiency improvements over the traditional task execution without migration scheme. The results also indicate that in the case of an agent node failure, intra-agent task migration offers a higher task response time gain than inter-agent migration.

Furthermore, to improve QoS for executing multiple HART tasks, the development of real-time task scheduling mechanisms has emerged as an interesting research issue by taking different real-time HART task properties, failure avoidance, and task processing capabilities into account. Thus, to improve the HART task execution process, this thesis next presents a community- and latency-aware HART task assignment scheme by using real-time information about arriving task requests for both isolated and clustered robots/agents. More specifically, a suitable multi-task scheduling scheme is presented for task on- and offloading based HART task execution with task prefetching and fault tolerance capabilities. To reap the benefits from task prefetching for executing multiple HART tasks, this thesis develops a novel prefetching-aware bandwidth allocation scheme that copes with both conventional broadband and task offloading data traffic at the same time. Next, a comprehensive analytical model is presented to investigate the performance of our proposed community- and latency-aware task offloading scheme in terms of mean task service time, delay and power saving ratio, task prefetching time efficiency, task service time gain to overhead ratio, among others. Our obtained results

show that for a typical system of 32 integrated optical network unit-mesh portal points (ONU-MPPs) and a polling cycle time of 100 ms, our proposed task offloading scheme achieves up to 31.3% and 32.7% task completion time gain over the task onloading scheme for nearby and remote task execution, respectively. The results demonstrate that for a typical task offload input data size of 500 MB, our proposed community- and latency-aware task offloading scheme with task prefetching capability offers a 11% higher task service time gain to overhead ratio than a conventional fetching based scheme. Our findings also suggest that for failure avoidance, the proposed fault tolerance mechanism is more effective in the considered task offloading scheme than the alternative failure recovery mechanism.

Given human users' different preferences for real-time HART task execution, e.g., lower delay and monetary cost, suitable HART task coordination has emerged as an important research problem, taking dynamically changing cloud agent/robot resources, network bandwidth utilization as well as delay-sensitive and delay-tolerant HART task properties into account. To cope with these challenges, this thesis explores the synergy between caching, computation, and communications for achieving cost-effective HART task execution. More precisely, to minimize task execution delay and monetary cost, this thesis presents a user preference-aware HART task coordination framework that selects the appropriate dedicated or non-dedicated robot and cloud agent for given caching and computing HART task execution requirements. To cope with varying bandwidth resources, this thesis proposes a proactive bandwidth allocation policy for the execution of both delay-sensitive and delay-tolerant HART tasks. To minimize the task execution delay of delay-sensitive users, our proposed delay cost saving (DCS) based scheme selects suitable actors by using both dedicated and non-dedicated actors. Conversely, to minimize the monetary cost for delay-tolerant policy users, our proposed monetary cost saving (MCS) scheme selects appropriate actors only from the set of dedicated actors. Furthermore, this thesis also presents a proactive bandwidth allocation scheme that assigns preemptive and non-preemptive bandwidth resources to DCS and MCS policy users, respectively. Unlike alternative approaches, our findings indicate that the maximum throughput and minimum delay (MTMD) based resource assignment policy is useful for both DCS and MCS policy users due to its minimum task execution time and monetary cost. Our obtained results show that for a typical number of 10 tasks and 8 available dedicated robots, the DCS (MTMD) policy exhibits a 30.5% higher task execution time saving ratio and a 63.6% lower monetary cost saving ratio than the MCS (MTMD) policy. Our proposed user preference aware HART task coordination policy thus represents a promising solution to reduce both task execution delay and monetary cost for emerging Tactile Internet applications.

Keywords: Caching, Cloud Computing, Computation Offloading, Collaborative Computing, Delay Cost Saving, Dynamic Bandwidth Allocation (DBA), Energy Efficiency, Fail-

ure Avoidance, Fiber-Wireless (FiWi) Enhanced Networks, Human-Agent-Robot Teamwork (HART), Human-to-Robot Communication (H2R), Human-Machine Co-activity, Internet-of-Things (IoT), Mobile-Edge Computing (MEC), Monetary Cost Saving, Tactile Internet, Task and Resource Scheduling, and Task Migration.

Statement of originality

I hereby certify that this thesis contains original work of the author. Some techniques employed from other authors are properly referenced herein.

Mahfuzulhoq Chowdhury
INRS, ÉMT, Montréal, QC
Université du Québec
Date: 01.12.2018

Résumé

Introduction et motivation

L'avènement de robots/machines télécommandés disponibles sur le marché peut être le précurseur d'une ère de convergence technologique, où les tâches de notre vie quotidienne (par exemple, l'assistance cognitive) seront de plus en plus souvent accomplies par des robots/machines qui nous permettent de voir, d'entendre, de toucher et de manipuler des objets dans des endroits où nous ne sommes pas physiquement présents. Dans divers systèmes cyber-physiques (CPSs, pour 'cyber-physical systems') qui exploitent l'interaction homme-machine en temps réel (par exemple, formation à distance, opérations de sauvetage essentielles à la mission), une latence aller-retour extrêmement faible est nécessaire pour faire correspondre l'interaction humaine à l'environnement. Cette vision de l'Internet est maintenant largement connue sous le nom d'Internet tactile, qui a récemment émergé pour diriger/contrôler les objets virtuels et physiques de notre entourage et de notre environnement et nous permettre de transmettre le toucher et l'action en temps réel [1],[2]. En offrant des communications à faible latence, l'Internet tactile devrait couvrir un large éventail de domaines d'application, y compris les soins de santé à distance, la conduite autonome ou assistée, le divertissement, et l'automatisation industrielle [3],[4]. Dans la plupart de ces secteurs verticaux, une latence très faible et une très grande fiabilité sont essentielles pour la réalisation d'applications immersives telles que la téléopération robotique [5], [6].

Il existe un chevauchement important entre l'Internet des objets (IoT, pour 'Internet of Things'), la 5G et la vision Tactile Internet, bien que chacune d'entre elles présente des caractéristiques uniques. Les principales différences peuvent être mieux exprimées en termes de paradigmes de communication sous-jacents et de dispositifs finaux habilitants. L'IoT repose sur la communication M2M (M2M, pour 'machine-to-machine'), l'accent étant mis sur les dispositifs intelligents (capteurs et actionneurs, par exemple). En coexistence avec la communication de type machine (MTC, pour 'machine type communication') émergente, 5G maintiendra son paradigme traditionnel de communication d'homme à homme (H2H, pour 'human-to-human') pour les services triple play conventionnels (voix, vidéo, données) avec un accent croissant sur l'intégration avec d'autres technologies sans fil (notamment le WiFi) et

la décentralisation. Inversement, l'Internet tactile sera centré sur les communications homme-machine/robot (H2M/R) (H2M, pour 'human-to-machine') en tirant parti des dispositifs tactiles/haptiques. Malgré leurs différences, l'IoT, le 5G et l'Internet Tactile semblent converger vers un ensemble commun d'objectifs de conception importants : très faible latence, ultra-haute fiabilité avec une disponibilité presque garantie de 99,999%, coexistence H2H/M2M, intégration de technologies centrées sur les données avec un accent particulier sur le WiFi et la sécurité. Contrairement à l'Internet mobile et à l'IoT, l'Internet tactile facilitera les communications haptiques en fournissant le moyen de transporter les sens haptiques (c'est-à-dire le toucher et l'actionnement) en temps réel en plus des données non haptiques classiques, de la vidéo et du trafic audio.

Les applications basées sur la communication H2M/R en temps réel sont sensibles à la latence de bout en bout vécue pendant le processus de communication entre les opérateurs humains et les robots/machines télécommandés. Pour réaliser des applications Internet tactile à faible latence de bout en bout, le cloud computing au bord du réseau d'accès radio mobile appelé cloudlet représente une solution prometteuse [7]. La recherche sur le cloudlet a eu tendance à se concentrer sur le WiFi dans le passé, bien qu'il y ait eu récemment un intérêt croissant parmi les opérateurs de réseaux cellulaires. L'importance des nuages peut être constatée dans de nombreuses applications centrées sur l'interaction homme-machine sensibles à la latence, telles que la réalité augmentée, l'assistance cognitive en temps réel ou la reconnaissance des visages sur les appareils mobiles. En septembre 2014, l'initiative de l'industrie de l'informatique mobile (MEC, pour 'mobile-edge computing') a introduit une architecture de référence afin d'identifier les défis qui doivent être surmontés pour faciliter la mise en œuvre des serveurs cloudlet [8]. MEC fournit des capacités informatiques et de l'informatique cloud dans le réseau d'accès radio (RAN, pour 'radio access network') à proximité des abonnés mobiles. On s'attend à ce que la mise en cache avancée, le déchargement des calculs et la gestion du trafic axée sur l'utilisateur à la périphérie des réseaux sans fil réduisent non seulement la charge de trafic de liaison terrestre, mais aussi améliorent la latence des applications Internet tactiles.

Les applications d'Internet tactile posent des exigences élevées pour les futurs réseaux d'accès en termes de latence, de fiabilité et de capacité. Pour atteindre les exigences clés du 5G et de l'Internet tactile de très faible latence et d'ultra-haute fiabilité, dans [9], les auteurs ont proposé le concept de réseaux LTE-A HetNets améliorés de FiWi qui unifie les réseaux mobiles 4G centrés sur la couverture et les réseaux d'accès à large bande à fibre optique et sans fil (FiWi, pour 'fiber-wireless') centrés sur la capacité, basés sur les technologies de fibre optique centrées sur les données et d'Ethernet sans fil. Au moyen d'analyses probabilistes et de simulations de vérification basées sur des traces récentes et complètes de smartphones, les

auteurs ont montré qu'une latence moyenne de bout en bout de 1 à 10 ms et une disponibilité presque garantie peuvent être obtenues grâce au partage de la fibre optique et aux capacités de déchargement WiFi. Notez, cependant, que seules les communications H2H conventionnelles ont été prises en compte dans [9] sans aucune communication H2R (H2R, pour 'human-to-robot')ou M2M coexistante. Pour réaliser des communications H2R à faible latence dans l'Internet Tactile, nous avons discuté dans [5] du rôle de plusieurs technologies habilitantes clés, y compris les réseaux LTE-A HetNets améliorés de FiWi, les cloudlets, la robotique dans les nuages, le codage de réseau, les réseaux définis par logiciel (SDN, pour 'software-defined networking'), et les communications de machine à cloud (M2C, pour 'machine-to-cloud'), entre autres. De plus, en tirant parti du haut débit, de la fiabilité, et en particulier, de la performance à retardement des réseaux HetNets LTE-A améliorés de FiWi, nous avons signalé que les infrastructures multirobots intégrées de FiWi basées sur des cloudlets décentralisés seront essentielles pour la coordination des applications Internet tactiles basées sur les communications H2R. Pour le déploiement rentable des applications Internet tactile, nous avons également identifié plusieurs défis de recherche importants tels que la conception de techniques adaptatives de gestion des ressources de bande passante pour le support du trafic H2H et H2R sur les réseaux améliorés FiWi avec une bonne coordination des services, des stratégies d'allocation des tâches H2R (planification optimale en ligne/hors ligne), le traitement des pannes et la gestion de la mobilité, entre autres. La figure R.1 résume les caractéristiques, les technologies habilitantes et les défis pour la réalisation d'applications Internet tactiles en temps réel.

Outre la communication haptique, un autre aspect distinct de l'Internet tactile est le fait qu'il devrait amplifier les différences entre les machines et les humains et entraîner la symbiose entre l'homme et la machine. En s'appuyant sur les zones où les machines sont fortes et les humains faibles, l'Internet tactile tire parti de leur autonomie "coopérative" et "collaborative" de telle sorte que les humains et les robots se complètent. À l'avenir, le travail en commun avec des machines (par exemple, des robots) favorisera les grappes géographiques de production locale ("inshoring") et nécessitera une expertise humaine dans la coordination de la symbiose homme-machine pour inventer de nouveaux emplois que les humains peuvent difficilement imaginer et ne savaient même pas qu'ils voulaient faire [6]. Contrairement à l'IoT qui s'appuie sur ses communications M2M sous-jacentes sans aucune implication humaine, l'Internet tactile implique la nature inhérente de l'interaction haptique HITL (HITL, pour 'inherent human-in-the-loop') et permet ainsi une approche de conception coopérative homme-machine vers la création et la consommation de nouvelles expériences immersives via l'Internet [10].

Dans *"DeepThinking : Where Machine Intelligence Ends and Human Creativity Begins,"* Garry Kasparov explique l'importance d'un processus supérieur dans la collaboration homme-machine, montrant que la faiblesse humaine + machine + meilleur processus est supérieur

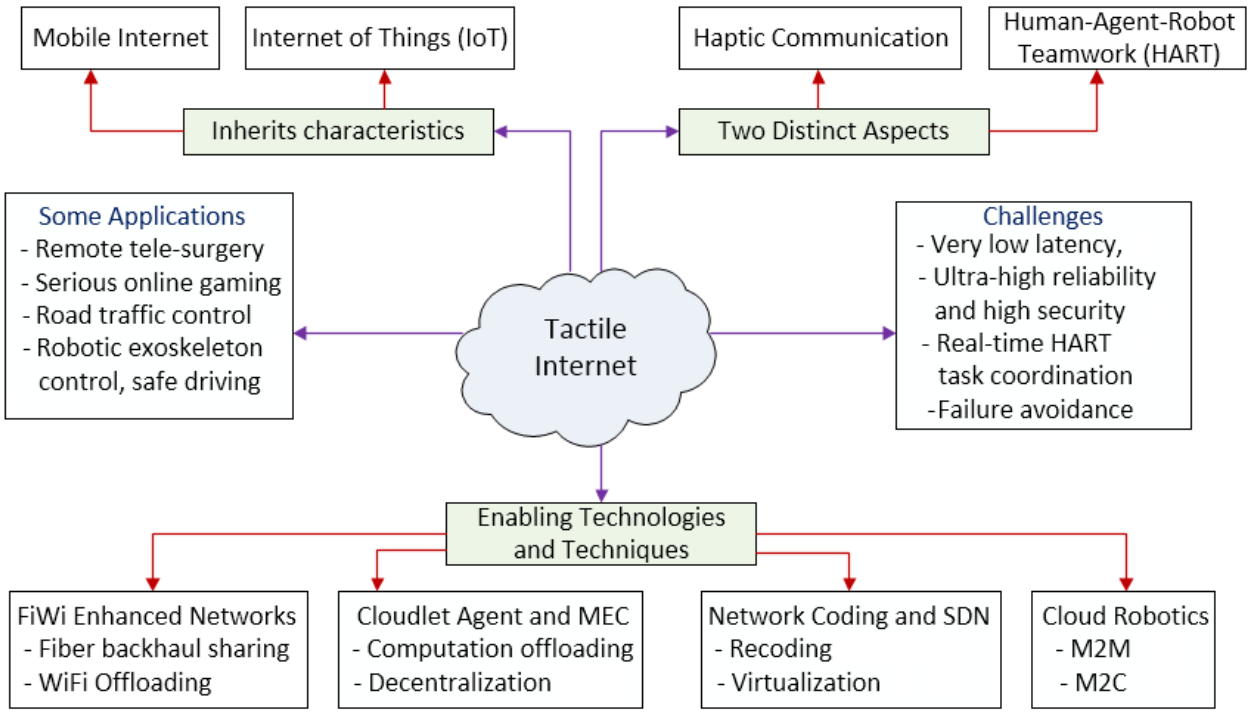


Figure R.1: L'Internet tactile : applications, défis, et technologies habilitantes.

à la force humaine + machine + processus inférieur. Ainsi, un processus intelligent bat des connaissances supérieures et une technologie supérieure.

Une approche prometteuse pour atteindre une coordination homme-machine avancée au moyen d'un processus supérieur pour orchestrer avec fluidité la co-activité homme-machine peut être trouvée dans le domaine encore jeune de la recherche sur le travail en équipe homme-agent et robot (HART, pour 'Human-Agent-Robot Teamwork'), dont l'objectif spécifique est de garder les humains dans la boucle plutôt qu'en dehors de la boucle [11]. Historiquement, HART étend l'approche dite humains-are-better-at/machines-are-better-at (HABA/MABA), qui assigne des tâches soit aux humainsou aux machines, alors que HART se concentre sur la façon dont les humains et les machines pourraient travailler ensemble. En ce qui concerne l'interaction homme-machine sous-jacente dans les applications d'Internet tactile HART, le principal défi est d'orchestrer la meilleure façon d'exécuter les tâches de concert. La collaboration et la communication entre les membres de HART sont essentielles pour faire face aux changements dynamiques dans l'environnement des tâches, améliorant ainsi la latence d'exécution des tâches. Il est à noter que les activités interdépendantes des membres de HART peuvent entraîner une complexité accrue et une consommation accrue de ressources. Pour faciliter l'exécution efficace des tâches HART, la recherche dans le domaine de la coordination centralisée/décentralisée des réseaux, la gestion adaptative des ressources en bande passante pour la coexistence du haut débit traditionnel et du trafic de déchargement, l'attribution des

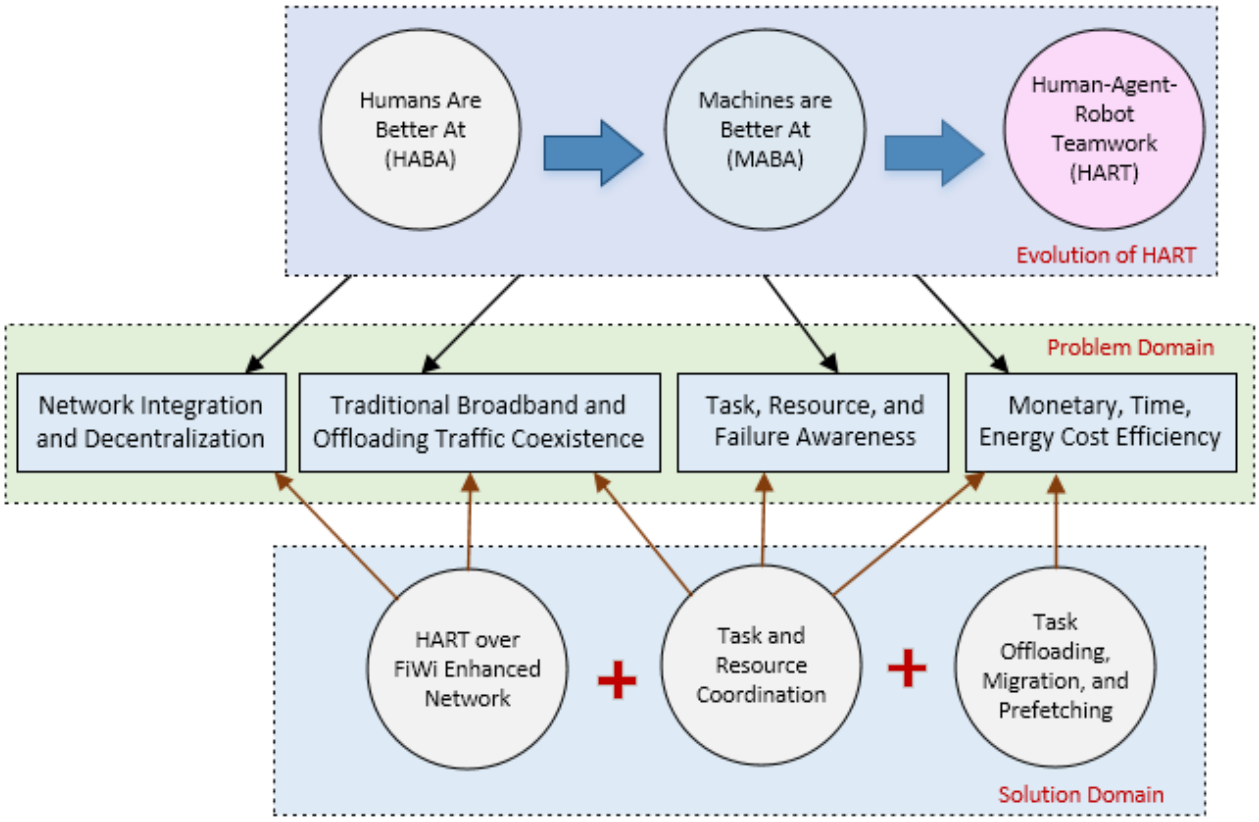


Figure R.2: Aperçu des principaux défis de la coordination HART sur l'Internet tactile.

tâches en cas de panne, l'énergie, le temps et les politiques de déchargement des tâches en fonction des coûts énergétiques, ainsi que la coordination des tâches et des ressources deviennent obligatoires. Les principaux défis de la coordination des tâches HART sur l'Internet tactile sont décrits à la figure R.2.

Objectifs

Les objectifs de cette thèse sont les suivants :

- L'obstacle crucial au déploiement réussi d'applications Internet tactiles locales et non locales est l'absence de stratégies appropriées de répartition des tâches entre les robots. La plupart des études existantes d'attribution de tâches multi-robots se concentrent sur un seul ou quelques paramètres pour la sélection du robot, par exemple, l'énergie d'un robot ou la distance jusqu'à l'emplacement de la tâche. Il est clair que les applications Internet tactiles basées sur les communications H2R en temps réel exigent des schémas avancés de sélection de robots, dans lesquels des paramètres supplémentaires doivent être pris en compte tels que les robots hétérogènes et les propriétés des tâches (par exemple, l'emplacement des robots et des tâches, la consommation d'énergie des robots, la charge

de travail des tâches, et les délais). De plus, l'absence de stratégies appropriées de surveillance des défaillances des robots pendant l'exécution des tâches et des stratégies d'allocation des ressources pourrait entraîner une augmentation des délais d'exécution des tâches et de la consommation d'énergie des robots. Un certain nombre de questions de recherche telles que (i) comment les demandes de tâches humaines sont arrivées au réseau de robots et (ii) comment les robots sont au courant de toutes les demandes de tâches ont été largement négligées dans les études précédentes. Ainsi, le premier objectif de ce travail est de concevoir un mécanisme efficace d'attribution de tâches H2R locales et non locales qui évitent les défaillances et un schéma unifié de gestion des ressources qui minimise le temps d'exécution des tâches et la consommation d'énergie des robots dans les infrastructures Internet tactiles basées sur FiWi.

- Avoir une sélection de robot appropriée pour satisfaire les demandes d'exécution de tâches des utilisateurs mobiles peut ne pas être suffisant pour éviter les échecs d'exécution de tâches en raison de contraintes de ressources données (par exemple, capacités de traitement des tâches, stockage, ou énergie restante) du robot sélectionné. Notez que les appareils mobiles/robots peuvent surmonter leur problème de pénurie de ressources en utilisant les ressources des agents cloud collaboratifs. Ce type d'exécution de tâche est également connu sous le nom de calcul collaboratif, où un robot aux ressources limitées transfère sa tâche de calcul à un autre agent cloud plus puissant ou à un robot proche pour exécution. Actuellement, la recherche dans le domaine du cloud computing d'infrastructure et de l'exécution de tâches collaboratives sans infrastructure HART centric sur les infrastructures FiWi fait défaut dans la littérature existante. Ainsi, pour améliorer le temps d'exécution des tâches et l'efficacité de la consommation d'énergie des robots/appareils mobiles à ressources limitées, le deuxième objectif de cette thèse est de proposer un schéma de calcul collaboratif qui sélectionne conjointement un nœud hôte approprié et un nœud d'agent cloud collaboratif pour exécuter différentes tâches HART. Un autre objectif majeur de cette partie de la thèse est d'étudier un schéma unifié d'allocation de bande passante pour gérer le trafic de données à large bande conventionnel et de déchargement des tâches de calcul sur des infrastructures Internet tactiles basées sur FiWi.

- Les services de cloud computing collaboratif permettent aux appareils mobiles à ressources limitées de décharger leurs tâches de calcul intensives sur des serveurs/substituts plus puissants pour le traitement. Par conséquent, l'un des principaux défis du cloud computing est de minimiser la latence d'exécution des tâches des utilisateurs mobiles. De plus, en raison des ressources variables dans le temps et des temps d'attente plus longs dans

un serveur cloud (agent), le serveur cloud initialement sélectionné peut ne pas toujours satisfaire aux exigences d'exécution des tâches de déchargement (p. ex., date limite). Ainsi, pour répondre aux exigences d'exécution des tâches de déchargement, la tâche déchargée d'une MU doit être migrée d'un serveur cloud à un autre pour être exécutée. Notez qu'en tenant compte de la charge du serveur cloud, des exigences des tâches, des latences de migration des tâches et de la mobilité des utilisateurs, l'une des questions de recherche fondamentales pour l'exécution de tâches centrées sur HART consiste à savoir si une tâche migre ou non avec la MU. Ainsi, en tenant compte de la mobilité des utilisateurs, des différentes tâches, et des propriétés des nœuds collaboratifs, le troisième objectif de cette thèse est de proposer une stratégie de migration des tâches en tenant compte du contexte pour l'exécution collaborative des tâches dans les infrastructures Internet tactiles basées sur FiWi.

- Les avantages du pré-transfert/pré-migration de tâches et de la connaissance des ressources des communauté-cluster n'ont pas été explicitement étudiés pour le déchargement de plusieurs HART tâches sur des infrastructures améliorées FiWi. De plus, l'ordre la planification optimale des tâches par rapport aux ressources et la sélection du service d'évitement des pannes pour l'exécution de tâches HART basées sur le chargement et le déchargement des tâches sont absentes de la littérature existante. Ainsi, en tenant compte à la fois des capacités de transfert préalable et de tolérance aux pannes ainsi que de la connaissance des ressources des cluster communautaires, le quatrième objectif de cette thèse est de concevoir un schéma de planification multi-tâches adapté à la communauté et à la latence pour l'exécution de tâches HART basées sur le chargement et le déchargement des tâches.

- Le cinquième objectif de cette thèse est de développer un schéma de coordination des tâches HART prenant en compte les préférences des utilisateurs. Notez que la recherche dans le domaine de l'exécution des tâches HART prenant en compte les préférences des utilisateurs en est encore à ses débuts. À l'heure actuelle, aucune étude existante ne traite du problème de la mise en cache et informatique sensibles aux délais et tolérants aux retards des HART tâches l'exécution en tenant compte de la connaissance des ressources des robots/agents dédié et non dédié et de l'allocation de bande passante basée sur la priorité préemptif. En tenant compte de différent préférences des MUs tels que le réduction des coûts monétaires et/ou retard pour l'exécution sensibles aux délais et tolérants aux retards des HART tâches, l'objectif final de cette thèse est de développer une stratégie appropriée de coordination des tâches HART et un schéma d'allocation des ressources proactif.

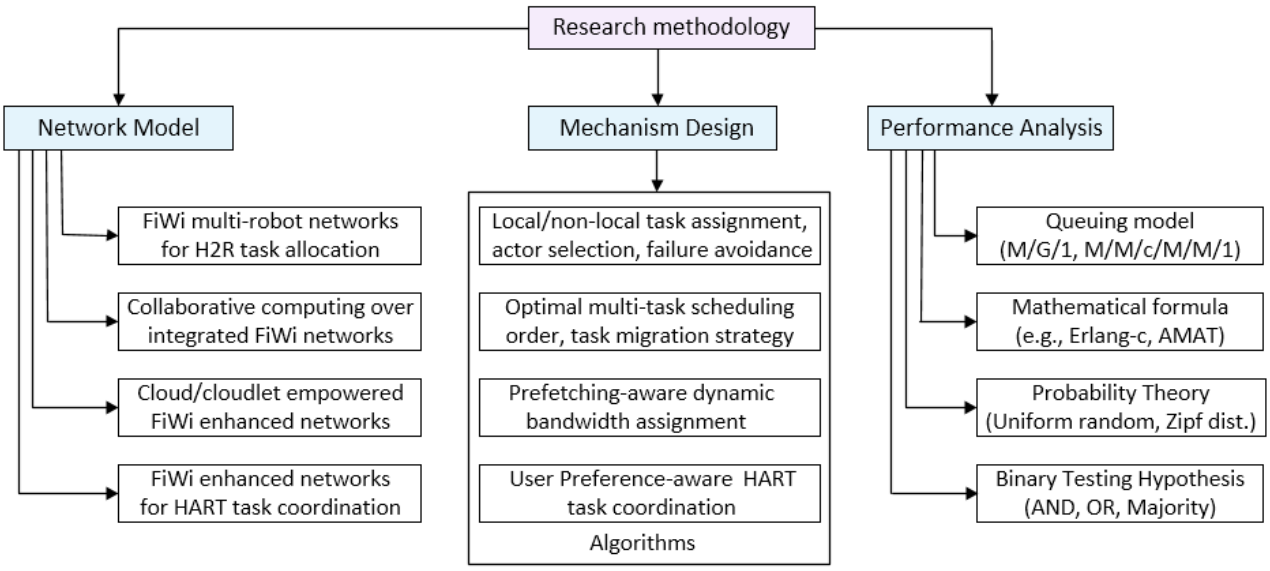


Figure R.3: Méthodologie de recherche.

Méthodologie de recherche

La méthodologie de recherche appliquée dans cette thèse comprend la modélisation des réseaux, la conception des mécanismes de coordination ainsi que la modélisation analytique et l'analyse des performances (voir Fig. R.3) et est décrite plus en détail ci-dessous:

- **Architecture de réseau:** Dans cette thèse, de multiples nouvelles architectures de réseau sont développées pour différents schémas de coordination des tâches centrés sur HART. Une approche descendante est envisagée, où les différentes exigences de tâches sont d'abord étudiées, puis l'infrastructure réseau est conçue pour répondre aux exigences de service. Il est important de noter que les fonctionnalités des réseaux de communication, les procédures d'attribution des tâches et des ressources, les technologies, et les protocoles sont étudiés. Les topologies basées sur les arbres et les mailles sont prises en compte dans la conception d'infrastructures Internet tactiles intégrées FiWi améliorées.

- **Conception de mécanismes:** Pour obtenir des performances optimales, différents algorithmes novateurs sont développés pour l'exécution collaborative de tâches HART dans les infrastructures améliorées FiWi. Plus particulièrement, les mécanismes proposés comprennent une stratégie unifiée d'allocation des ressources, la sélection d'un robot et d'un agent cloud pour exécuter différentes tâches HART, un ordre de planification multitâche optimal, un schéma de rapport de pannes ainsi qu'un algorithme d'allocation dynamique de bande passante (DBA, pour 'dynamic bandwidth allocation') tenant compte du transfert préalable au courant de tâches.

- **Analyse des performances:** Dans ce travail, une analyse du rendement est effectuée basé sur différents modèles de files d'attente (par exemple, M/G/1, M/M/M/1, M/M/1, M/M/c), d'outils analytiques, de formules mathématiques (par exemple, Erlang-C, distance euclidienne, temps moyen d'accès à la mémoire ou formule AMAT (AMAT, pour average memory access time), de distributions de probabilités (par exemple, aléatoires uniformes, Zipf), et d'hypothèses (par exemple, hypothèses de tests binaires). Pour évaluer la performance du système selon différents scénarios, la performance de simulation analytique et de vérification est examinée pour un large éventail d'indicateurs de performance et de paramètres de système variables.

Contributions de la thèse

Cette thèse est basée sur un total de huit publications scientifiques (revues et magazines de l'IEEE). Les principales contributions de cette thèse sont examinées ci-après.

Allocation de tâches H2R locales et non locales prenant en charge les évitement d'échec dans les infrastructures Multi-Robot FiWi

Pour le déploiement réussi d'applications H2R, une répartition efficace des tâches entre les robots est essentielle, ce qui s'est révélé être un sujet de recherche intéressant en tenant compte d'une grande variété de tâches et de types de robots, de l'emplacement des tâches, de la disponibilité des robots, de la capacité et de l'échec lors de l'exécution des tâches. Les solutions actuelles d'attribution des tâches robotiques souffrent généralement de plusieurs inefficacités au cours de l'exécution des tâches (par exemple, temps d'exécution des tâches et gaspillage d'énergie des robots) en raison de l'absence de mécanismes de robots sélection appropriés et de mécanismes de contrôle appropriés. L'hétérogénéité des robots et des types de tâches rend la répartition des tâches encore plus difficile. Pour accélérer le processus d'exécution des tâches robotiques en temps réel et réduire la consommation d'énergie des robots à ressources limitées, l'utilisation des services robotiques pour les tâches humaines doit se faire d'une manière plus efficace en termes de ressources. La plupart des études antérieures ne tiennent compte que d'un seul paramètre pour la sélection du robot, par exemple, la distance et l'énergie résiduelle. Un certain nombre de paramètres supplémentaires, tels que la compétence du robot, la disponibilité, et le temps d'exécution des tâches, doivent également être pris en compte. Dans le passé, un certain nombre d'aspects importants de l'allocation des ressources et de la mise en réseau des tâches robotiques liés à des questions clés de conception, y compris, mais sans s'y limiter: (*i*) comment les demandes de tâches humaines arrivent aux réseaux de robots, (*ii*) comment récupérer des pannes de robots, et (*iii*) comment s'assurer

que les robots sont conscients de toutes les demandes de tâches ont été largement négligés dans les études précédentes.

Pour surmonter les défis susmentionnés, ce travail développe une architecture réseau multi-robot basée sur FiWi qui coordonne l'attribution des tâches entre les humains, les robots, et les agents. Pour attribuer efficacement la tâche locale et non locale d'un utilisateur humain donné à un robot approprié, ce travail propose un algorithme de sélection du robot basé sur la distance, l'énergie résiduelle ainsi que sur la capacité, et la disponibilité du robot. Il introduit un mécanisme de signalement des pannes assisté par robot voisin pour éviter les échecs d'exécution des tâches. Pour faciliter l'attribution des tâches H2R locales et non locales en même temps sur notre infrastructure réseau FiWi proposée, ce travail propose un système unifié d'attribution des ressources basé sur l'accès multiple à répartition dans le temps (TDMA, pour 'time-division multiple access'). En utilisant l'hypothèse de test binaire, ce travail étudie le taux d'erreur de détection du robot de trois règles de fusion différentes (AND, OR, et Majorité) et examine leur efficacité respective pendant le processus de détection de défaillance du robot. Ce travail modélise le délai de transmission de trame maximum en amont (US, pour 'upstream') et en aval (DS, pour 'downstream') sur la base d'un modèle de file d'attente M/G/1 avec réservations et vacances. En tenant compte du délai de transmission des trames (US et DS) et du délai de sélection des robots, ce travail analyse à la fois le délai d'attribution des tâches locales et non locales de bout en bout. Un modèle analytique complet est présenté pour évaluer la performance de notre schéma proposé avec deux schémas d'allocation de tâches généralisées, c'est-à-dire la sélection de robot basée sur la distance minimale (MD) [12]-[13] et la sélection de robot basée sur la priorité (PS) [14]-[15]) en termes de débit, de délai d'allocation de tâches, de temps d'exécution, et d'énergie résiduelle. Ce travail étudie le compromis entre le délai d'attribution des tâches et le débit du système. De plus, la complexité temporelle de l'algorithme d'allocation des tâches proposé est également analysée. La figure R.4(a) compare l'efficacité du temps d'exécution des tâches de notre méthode MET proposée avec celle des approches traditionnelles MD et PS pour des charges de travail variables.

La figure montre que l'efficacité maximale réalisable du temps d'exécution des tâches de notre schéma de sélection de robot proposé par rapport au schéma de sélection traditionnel basé sur PS et MD est respectivement de 11% et 18%. Ceci est dû au fait que l'approche MD sélectionne un robot approprié pour chaque tâche en fonction de la distance la plus faible par rapport à un emplacement de tâche donné. Inversement, le schéma PS attribue une tâche donnée au robot ayant l'ID le plus bas ou en utilisant le robot et l'appariement des tâches. Notre approche proposée d'allocation des tâches MET sélectionne un robot sur la base du calcul préalable du temps minimum d'exécution des tâches qui inclut le délai d'allocation des tâches des robots, le temps de traversée de l'emplacement des tâches, le temps d'occupation des

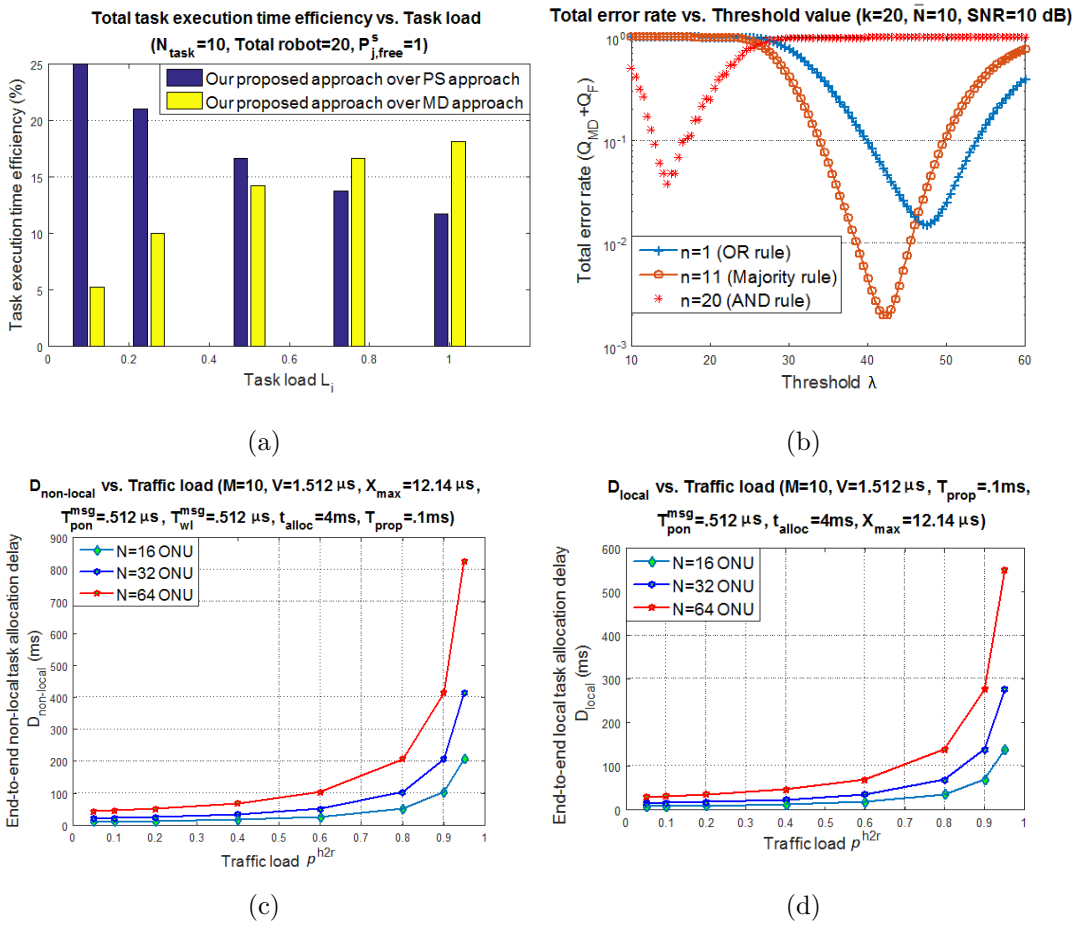


Figure R.4: Efficacité du temps d'exécution des tâches, délai d'allocation, et évaluation du taux d'erreur de détection.

robots, et le temps de traitement de la charge de travail des tâches. La figure R.4(b) illustre la variation du taux d'erreur de détection de défaillance des robots pour varier le seuil d'énergie de signal (λ) pour la détection de défaillance du robot sélectionné. Cette figure compare trois règles de fusion différentes (n sur k rapports de défaillance des voisins coopératifs) afin d'identifier leur efficacité respective. Nous observons que les règles AND ($n = k$), Majorité ($n \geq \frac{k}{2}$), et OR ($n = 1$) basées sur la détection coopérative des défaillances atteignent un taux d'erreur minimal pour un seuil d'énergie de signal de détection faible, moyen et grand, respectivement. Ainsi, les règles AND, Majorité et OR sont optimales pour un seuil d'énergie de signal de robot faible, moyen et grand, respectivement. Ensuite, les délais d'attribution des tâches de bout en bout non locales ($D_{non-local}$) et locales (D_{local}) pour différents nombres d'utilisateurs (M) et charges de trafic (ρ^{h2r}) sont évalués dans les Fig. R.4(c) et Fig. R.4(d), respectivement. Les deux retards (D_{local} and $D_{non-local}$) augmentent avec la charge de trafic (ρ^{h2r}) et le nombre différent d'ONU (N) dans le système que nous proposons. $D_{non-local}$ connaît un délai d'attribution des tâches plus élevé que D_{local}. Ceci est dû au fait qu'à côté

du délai de sélection du robot (t_{alloc}), $D_{non-local}$ dépend à la fois des délais de transmission des trames US (D_u) et DS (D_d), tandis que D_{local} dépend uniquement du délai de transmission des trames US (D_u).

l'informatique collaborative pour les communications Internet tactiles avancées H2R dans les infrastructures intégrées FiWi Multi-robot

Avec l'émergence de l'Internet tactile et l'avènement des robots télécommandés, la bonne répartition des tâches entre les robots a attiré une attention significative pour permettre des applications et des services robotiques basés sur le paradigme de communication H2R. Cependant, les ressources limitées de calcul, d'énergie et de stockage des robots peuvent entraver le lancement réussi de telles applications. Pour combler ces lacunes, les appareils mobiles/robots sollicitent de plus en plus souvent l'aide de nœuds de collaboration (p. ex., l'informatique cloud mobile, les communications entre appareils mobiles) pour exécuter leurs tâches de calcul, une tendance également connue sous le nom de cueillette informatique [16] ou d'informatique collaborative [17], [18]. Malgré les progrès récents dans l'exécution des tâches robotiques, l'impact des schémas d'exécution de tâches conjointes qui tiennent compte à la fois du robot hôte et des nœuds collaboratifs (p. ex., cloud central ou cloudlet) pour le processus d'exécution des sous-tâches de détection et de calcul indépendant de l'emplacement n'a pas été examiné suffisamment en détail auparavant. Pour plus de clarté, notez que la communication homme-robot-agent basée sur l'exécution complète des tâches se compose généralement de deux sous-parties. La première comprend le traitement initial ou la sous-tâche de surveillance physique dépendant de l'emplacement (par exemple, la capture d'une image), qui ne peut être exécutée que par le robot hôte sélectionné situé dans la zone de tâche donnée. La deuxième sous-partie de la tâche implique le calcul/traitement indépendant de l'emplacement des données capturées (p. ex., détection d'images/de visages), qui peut être effectué par le robot hôte lui-même ou ou bien être déchargé sur des nœuds cloud collaboratifs (indépendants de l'emplacement).

La plupart des études antérieures considéraient soit l'attribution d'une tâche complète (c'est à dire., les sous-parties de détection/surveillance physique et de calcul) à un robot, soit le déchargement du calcul (c'est à dire., la sous-partie de la tâche complète) sur des nœuds cloud (cloud central et cloudlet local) pour exécution, et non les deux. Par conséquent, la question de savoir comment assigner une tâche de détection/surveillance physique et de calcul locale ou non locale à un robot hôte et à un serveur cloud en tenant compte des différentes tâches et types de robots avec leur consommation d'énergie, leur disponibilité, la distance du robot à l'emplacement de la tâche, le traitement, et la vitesse de déplacement, la

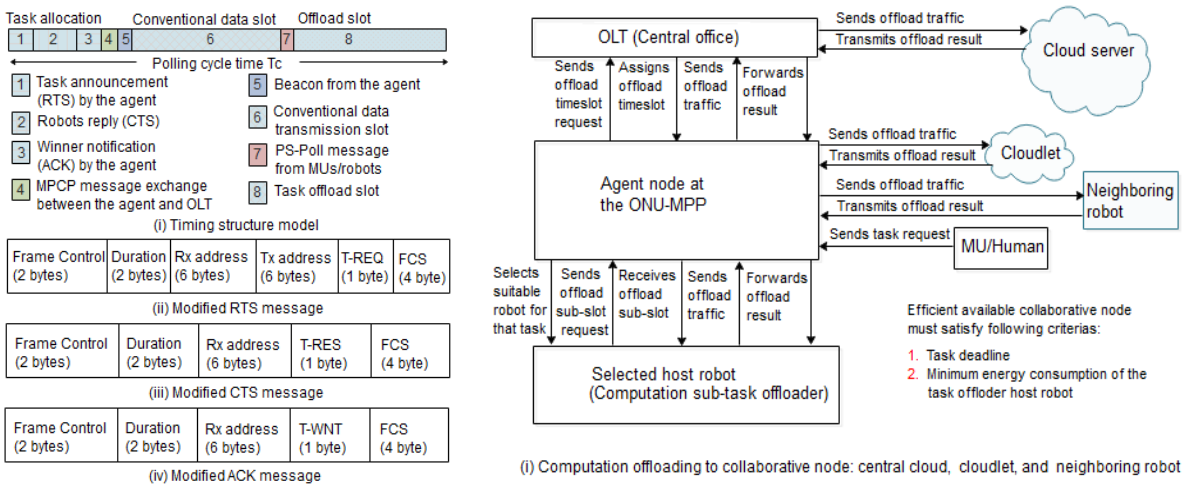
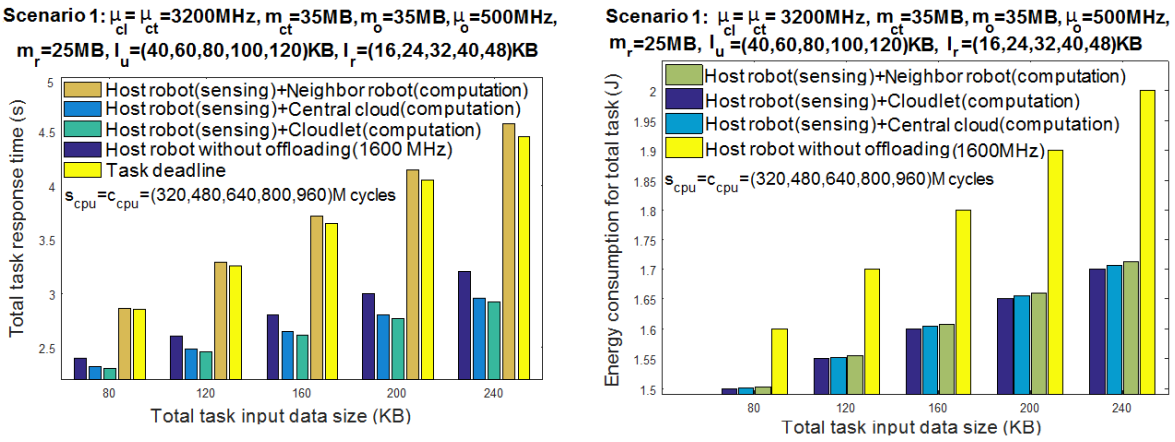


Figure R.5: (a) Structure temporelle et format de la trame de contrôle et (b) étapes opérationnelles du processus de déchargement des calculs.

disponibilité des ressources cloud demeure un défi de recherche ouvert. En outre, différentes situations difficiles doivent être étudiées pour obtenir de meilleures performances, lorsque les deux nœuds cloud/cloudlets collaboratifs peuvent satisfaire ou ne pas satisfaire aux exigences de déchargement des calculs.

Pour relever certains des défis susmentionnés, ce travail développe une stratégie efficace d'attribution des tâches qui inclut la sélection d'un robot hôte et d'un nœud collaboratif approprié dans les réseaux multi-robots intégrés de FiWi. Nous proposons d'utiliser non seulement le cloud central et les cloudlets locaux comme nœuds de collaboration, mais aussi les robots voisins disponibles pour le déchargement des sous-tâches de calcul. Pour réaliser des économies d'énergie maximales des robots et accomplir les tâches dans les délais requis, l'objectif principal de ce travail est de sélectionner la politique appropriée pour l'exécution des tâches demandées par les humains en évaluant la performance du schéma d'exécution des tâches non collaboratives, dans lequel le robot hôte sélectionné exécute la tâche complète (sous-tâche de détection et de calcul), et le schéma d'exécution collaborative/conjointe, dans lequel le robot hôte sélectionné exécute seulement la sous-tâche de détection tandis que le nœud collaboratif sélectionné exécute la sous-tâche de calcul via le déchargement de calcul. En outre, ce travail propose un schéma unifié de gestion des ressources capable de gérer le trafic à large bande conventionnel coexistant et le trafic de données déchargé par calcul. Le schéma de gestion des ressources proposé utilise un système d'accès multiple par répartition dans le temps à deux couches dans les sous-réseaux optiques et sans fil. La structure générale du chronométrage est divisée en trois parties: (*i*) la sélection initiale du robot pour l'attribution des tâches, (*ii*)



(a) (b)

Figure R.6: Temps de réponse aux tâches et évaluation de la consommation d'énergie des différents systèmes collaboratifs/conjoints et non collaboratifs.

le créneau horaire assigné pour le trafic à large bande conventionnel des utilisateurs associés, et (*iii*) les transmissions de données déchargées. La figure R.5 illustre la structure temporelle proposée et le processus de déchargement par calcul. Pendant la phase initiale d'attribution des tâches, l'agent situé à l'ONU-MPP échange trois messages de contrôle (`RTS`, `CTS`, and `ACK`)avec ses robots associés pour sélectionner un robot approprié pour chaque demande de tâche complète arrivée. L'agent sélectionne ensuite un robot hôte approprié avec un temps de réponse minimal pour chaque tâche qui contient à la fois des sous-parties de détection et de calcul en fonction de leur disponibilité, du seuil d'énergie pour effectuer la tâche et des critères de délai d'exécution de la tâche. Le robot hôte sélectionné exécute d'abord la partie détection/surveillance physique de la tâche (dépend de l'emplacement). Si le robot hôte envoie une demande de déchargement de sous-tâche de calcul à l'agent (emplacement indépendant), l'agent sélectionne un nœud collaboratif approprié pour l'exécution de la sous-tâche de calcul (sous-partie restante) sur la base des critères suivants: (*i*) le temps de réponse de la sous-tâche de calcul du nœud collaboratif est inférieur ou égal à la date limite de la sous-tâche de calcul, (*ii*) la disponibilité de ressources suffisantes, et (*iii*) la consommation d'énergie minimale.

Les figures R.6(a) et R.6(b) illustrent le temps total de réponse aux tâches (délai prévu de traitement des tâches après l'assignation des tâches) et la consommation d'énergie du robot hôte des différents schémas d'exécution des tâches pour le scénario 1. Dans le scénario considéré, on suppose que le cloud central et le cloudlet ont la même capacité de calcul/puissance du CPU. Les chiffres montrent que le temps de réponse aux tâches et la consommation d'énergie du robot hôte augmentent pour augmenter la taille des données d'entrée des tâches dans tous les schémas d'exécution des tâches proposés. La figure montre que le robot hôte et le robot voisin basé sur un schéma d'exécution conjointe de tâches montre un temps de réponse de

tâche plus élevé que le robot hôte et le cloud central basé schéma d'exécution conjointe de tâches et ne parvient pas à respecter la date limite de la tâche. C'est parce que la puissance CPU du robot voisin (500MHz) est inférieure à la puissance CPU du cloud central (3200MHz). Ainsi, le délai de traitement des sous-tâches de calcul est beaucoup plus élevé dans le robot voisin que celui de l'exécution du cloud central. En outre, les figures R.6(a) et R.6(b) montrent que le schéma d'exécution conjointe de tâches basé sur le robot hôte et le cloudlet dépasse le schéma conjoint basé sur le robot hôte et le cloud central en termes de temps de réponse aux tâches et de consommation d'énergie du robot hôte. Ceci est principalement dû au fait que le cloudlet implique un délai de déchargement de calcul plus court que le cloud central. Le schéma d'exécution conjointe de tâches basé sur le robot hôte et le cloudlet montre une augmentation de 36%, 8%, 2% du temps de réponse des tâches et une efficacité énergétique de 3%, 15%, 2% plus élevée que le robot hôte et le robot voisin schéma d'exécution conjointe, du robot hôte sans déchargement, et robot hôte et le cloud central schéma d'exécution conjointe, respectivement. Ainsi, le schéma d'exécution conjointe de tâches basé sur le robot hôte et le cloudlet est optimal pour le scénario considéré.

Schéma de migration de tâches HART centré sur des infrastructures Internet tactiles basées sur FiWi

En poussant plus loin l'idée du déchargement des tâches, la migration des tâches est apparue comme une approche prometteuse pour améliorer la qualité de l'expérience (QoE, pour 'quality of experience') des utilisateurs mobiles (MU, pour 'mobile users') en minimisant le temps d'exécution de leurs tâches [19]. La migration des tâches élargit la portée du déchargement des tâches de calcul conventionnel en transférant non seulement la tâche d'une MU sur le cloud, mais aussi d'un serveur cloud à un autre pour exécution. En général, la migration des tâches entre serveurs cloud n'est considérée comme bénéfique que si le temps d'exécution des tâches prévu au niveau du serveur cloud secondaire est inférieur à celui du serveur cloud primaire [20]. Notez cependant que la migration des tâches entraîne un délai de migration supplémentaire. Par conséquent, pour un gain de migration de tâche et un temps de latence donné, la question de savoir comment et où une tâche doit migrer est essentielle. Pour répondre à cette question, plusieurs critères de décision de migration doivent être pris en compte, tels que l'état des serveurs de destination actuels et provisoires, les propriétés des tâches et la latence de migration des tâches, entre autres.

À l'heure actuelle, il n'existe que quelques études sur la migration collaborative des tâches exploitant les agents basés sur le cloud, par exemple, la sélection des agents cloud pour la migration des tâches basée sur la prédiction de la charge [21], le délai de service [22], la distance [23], la disponibilité des ressources (c'est à dire., la vitesse du CPU et la charge de travail)

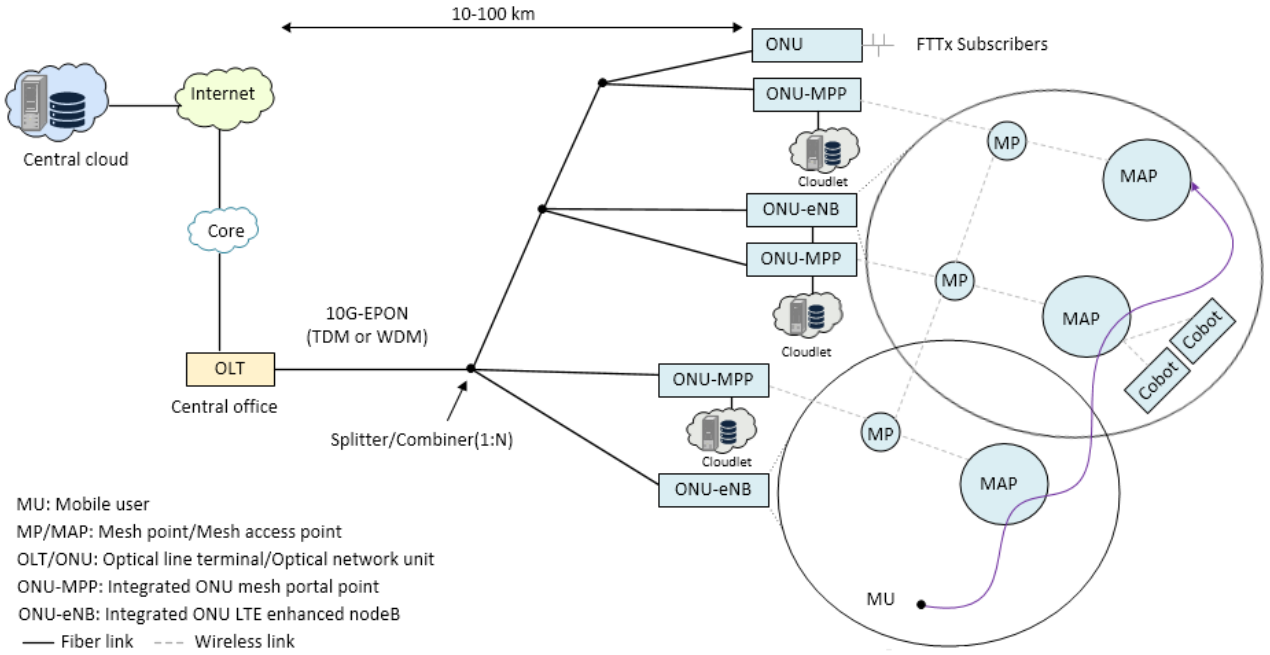


Figure R.7: Infrastructure Internet tactile basée sur FiWi, basée sur des cloud-cloudlets intégrés, des cobots, et des MU humaines pour la migration de tâches HART.

[24]-[25], l'emplacement de téléchargement des résultats des tâches des utilisateurs mobiles [19], et la consommation d'énergie [26]. Notez que ces études existantes sur la migration des tâches n'ont pris en compte que le problème de la migration des tâches d'une MU vers un robot approprié ou vers un agent cloud, plutôt que les deux. Aucune des études existantes ne s'est concentrée sur la participation/coopération active de tous les membres de HART, à savoir les MU (humains), les agents (cloud central/cloudlet), et les robots collaboratifs (cobots), ce qui est nécessaire pour l'exécution correcte des tâches HART impliquant à la fois des sous-tâches physiques et cognitives. Une autre question ouverte est de savoir comment coordonner la migration des tâches centrée sur HART de MUs vers les nœuds collaboratifs (cobots et agents) et entre les nœuds collaboratifs (cobot vers agent ainsi que d'agent à agent).

Cette partie de la thèse vise à aborder certains des défis de recherche ouverts susmentionnés dans le domaine de la migration des tâches. Nous introduisons d'abord une architecture Internet tactile intégrée à deux niveaux cloud-cloudlet basée sur FiWi pour l'exécution de tâches HART en tenant compte de la couverture cellulaire et WiFi (voir Fig. R.7). Après avoir décrit les caractéristiques clés des tâches physiques vs. tâches cognitives et robot collaborative (cobot) vs robot autonome, ce travail présente un schéma de migration de tâches HART approprié, prenant en compte les caractéristiques de différentes tâches (délai, charge de travail, taille des données) et les caractéristiques de nœud collaboratif (disponibilité, vitesse de traitement des tâches, énergie restante), et les caractéristiques de mobilité de l'utilisateur.

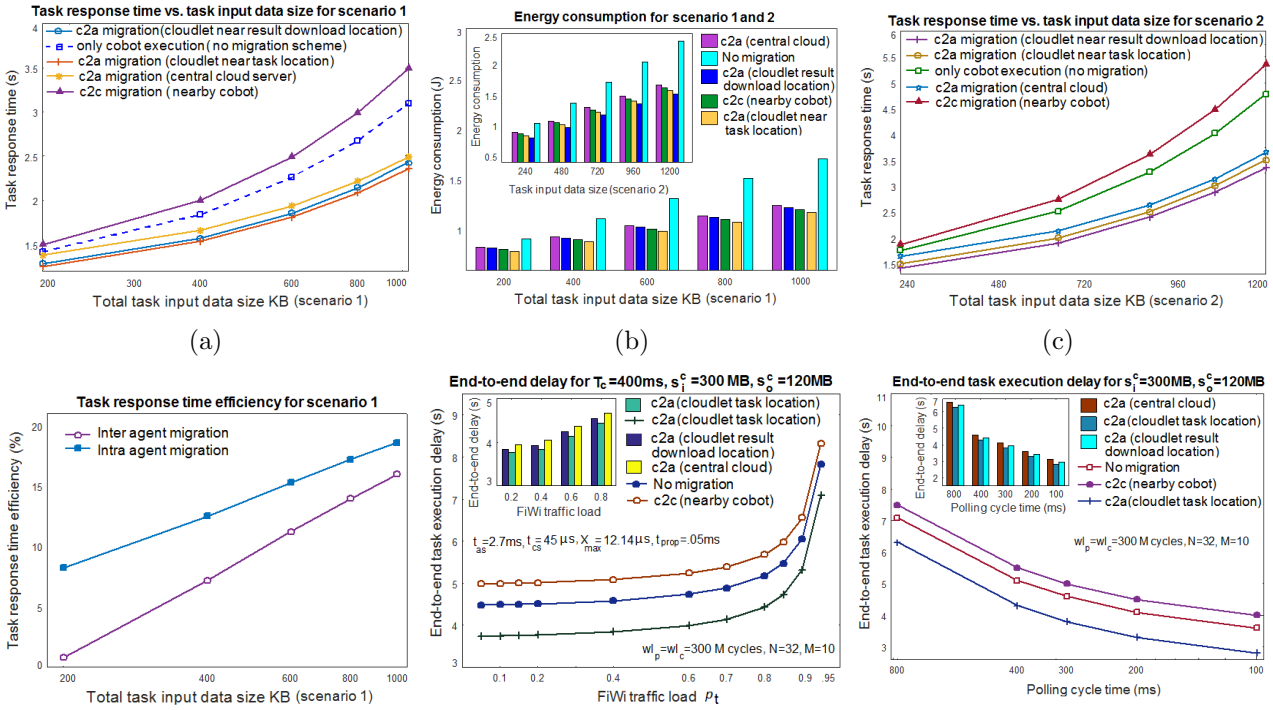


Figure R.8: Temps de réponse des tâches, consommation d'énergie, efficacité du temps de réponse des tâches, et évaluation des retards de bout en bout.

Plus précisément, ce travail analyse la performance de notre projet de migration de tâches HART, en tenant compte de la migration de tâches inter-agents (cloud à cloudlet et vice versa) et intra-agents (cloud à cloud et cloudlet à cloudlet) dans les infrastructures Internet tactiles basées sur FiWi. Pour déterminer le schéma optimal de migration des tâches, nous étudions les types suivants de schéma de migration des tâches en utilisant un certain nombre de mesures de performance spécifiques à HART: (*i*) schéma *c2a* (cobot à un emplacement de tâche donné vers un agent cloudlet qui est proche de l'emplacement de la tâche), (*ii*) schéma *c2a* (cobot vers agent cloudlet qui est proche de l'emplacement de téléchargement des résultats de tâche de l'MU), (*iii*) schéma *c2a* (cobot vers cloud distant), (*iv*) pas de migration, (*v*) schéma *c2c* (cobot vers voisin cobot) migration.

Les figures R.8(a)-(c) évaluent le temps de réponse aux tâches et l'évaluation de la consommation d'énergie des différents schémas de migration des tâches en fonction de la taille totale des données d'entrée des tâches. La figure montre que le temps de réponse aux tâches et la consommation d'énergie de tous les schémas comparés augmentent pour augmenter la taille des données d'entrée des tâches. Notez, cependant, que les schémas *c2a* (cloudlet près de l'emplacement de la tâche) et *c2a* (cloudlet près de l'emplacement de téléchargement des résultats) atteignent le temps de réponse minimum et la consommation d'énergie minimum de

la tâche dans les scénarios 1 et 2, respectivement. Ceci est dû au fait que dans le scénario 1, la taille des données d'entrée des tâches migrées (s_i^c) est plus grande que la taille des données de sortie des tâches migrées (s_o^c), alors que dans le scénario 2, la relation entre la taille des données d'entrée des tâches migrées et la taille des données de sortie est inversée. La relation inverse entre la taille totale des données d'entrée et de sortie des tâches se traduit par une latence minimale de migration des tâches (à la fois en amont et en aval) pour le schéma $c2a$ (clouldet près de l'emplacement des tâches) et $c2a$ (cloudlet près de l'emplacement de téléchargement des résultats) dans les scénarios 1 et 2, respectivement. Pour mettre en évidence l'impact de la migration des tâches entre deux agents cloud, la figure R.8(d) compare l'efficacité du temps de réponse des tâches des deux schémas différents : la migration inter-agent (du cloudlet vers le cloud central) et intra-agent (du cloudlet vers un autre cloudlet). La figure montre que la migration de l'agent intra-cloud offre un meilleur temps de réponse aux tâches que son homologue de l'agent inter-cloud.Cela s'explique par le fait que la migration intra-agent souffre d'une surcharge de communication de migration de tâches plus faible (dans le scénario 1). Ainsi, la migration intra-agent est plus préférable lorsqu'un échec se produit pendant l'exécution de la tâche de l'agent.

La figure R.8(e) montre que le délai d'exécution de bout en bout des différents schémas de migration des tâches reste faible pour une faible charge de trafic FiWi ρ_t, mais augmente rapidement pour une charge de trafic plus élevée ρ_t. Notez que le délai d'exécution des tâches de bout en bout est minimal dans le schéma de migration $c2a$ (cloudlet près de l'emplacement des tâches). Par exemple, pour une taille de données d'entrée de 300 Mo et une charge de trafic FiWi de 0.8, la migration $c2a$ (cloudlet près de l'emplacement de la tâche) offre un délai d'exécution des tâches de bout en bout de 15% and 21% inférieur à celui du schéma de migration de non-migration et $c2c$, respectivement. La figure R.8(f) illustre le retard d'exécution de bout en bout des différents schémas de migration des tâches pour différents temps de cycle d'interrogation T_c. La figure montre que pour les grands T_c, le délai d'exécution des tâches des différents schémas de migration des tâches reste élevé, mais diminue rapidement pour les petits T_c. Notamment, le schéma de migration $c2a$ (cloudlet près de l'emplacement des tâches) surpasse ses homologues en termes de délai d'exécution des tâches de bout en bout et convient donc mieux à l'exécution de tâches HART sensibles aux retards.

Schéma de planification multitâche prenant en compte la communauté et la latence et le schéma d'allocation de bande passante basé pré-transfert dans les réseaux améliorés FiWi

La majorité des politiques de planification des tâches existantes se concentrent sur le déchargement de tâches numériques à forte intensité de calcul sur des serveurs cloud ou des appareil-

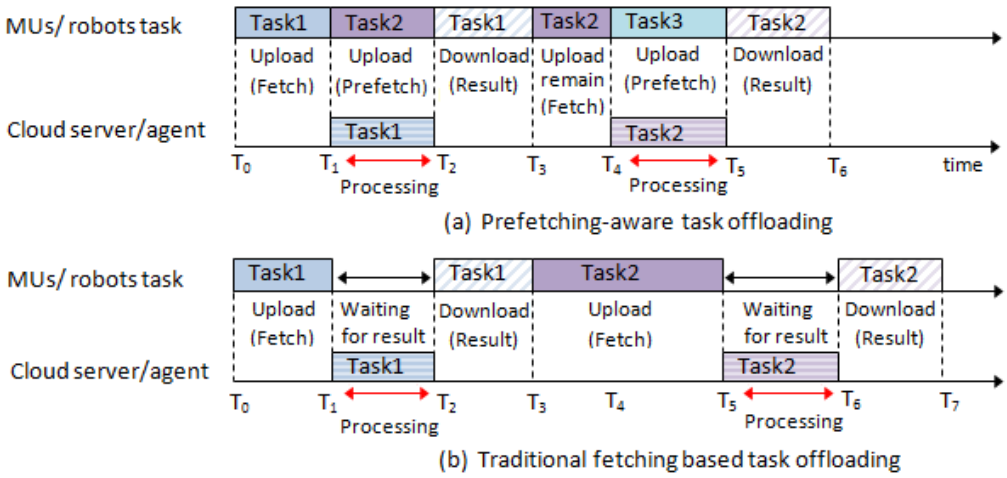


Figure R.9: Allocation de ressources basée sur le transfert préalable pour le déchargement multitâche.

s mobiles plutôt que sur les deux, ce qui fait qu'un nombre important de ces politiques s'appliquent à la planification hors ligne. Pour ce faire, le planificateur de tâches doit disposer d'informations a priori sur les tâches futures (p. ex., heure d'arrivée, date limite). Les schémas d'ordonnancement hors ligne sont bien adaptés aux tâches périodiques, mais deviennent moins adaptés à l'exécution de tâches apériodiques en temps réel [25],[27]. En raison de leurs besoins incertains en ressources cloud, l'exécution des tâches apériodiques en temps réel exige un schéma de planification des tâches en ligne (dynamique) approprié pour maintenir les assurances de qualité de service (QoS, pour 'quality of service'). Les schémas de planification des tâches en ligne inadéquats souffrent d'une latence importante de migration des tâches en raison de l'indisponibilité des ressources des acteurs pour le traitement des tâches déchargées [28]. De plus, l'absence d'une stratégie appropriée d'attribution de la bande passante peut entraîner des temps d'attente plus longs pour la transmission des données et la réception des résultats pendant le déchargement des tâches. Ainsi, l'un des défis de l'ordonnancement des tâches en ligne est de minimiser la latence d'exécution des tâches, y compris le traitement des tâches et le délai de communication de déchargement, en atténuant l'incertitude de la gestion des ressources en nuage/bande passante et l'évitement des pannes [29].

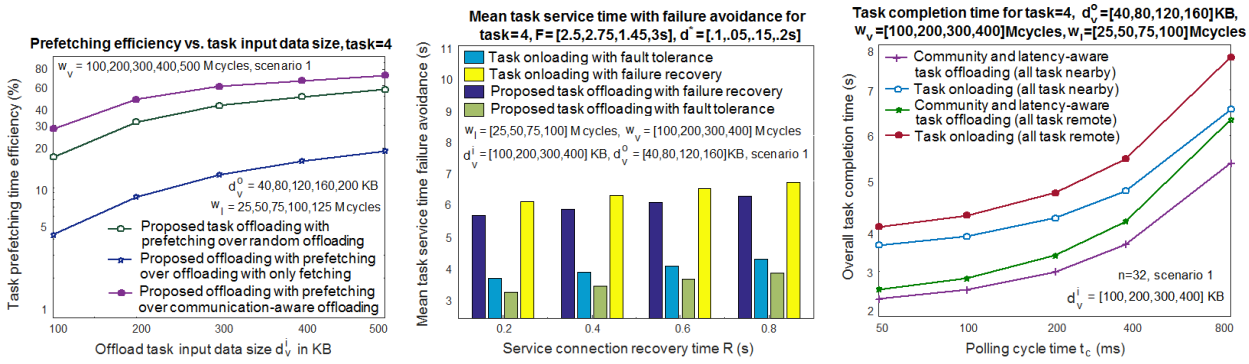
La plupart des études existantes sur le déchargement des tâches de calcul [17],[30],[25],[29] appliquent la technique de migration conventionnelle, où la prochaine tâche de calcul d'une MU donnée ne peut être transférée au serveur cloud qu'une fois la tâche précédemment déchargée. Par conséquent, le transfert/migration conventionnel souffre d'une latence de déchargement multitâche accrue. Pour surmonter ces lacunes dans le schéma du transfert/migration conventionnel, nous proposons un schéma de gestion des ressources de bande passante conscient du pré-transfert/pré-migration de tâches (voir Fig. R.9) pour décharger plusieurs tâches HART,

où la totalité ou une partie des données d'entrée de la tâche suivante de MU est transférée au serveur cloud pendant la période de calcul de la tâche précédemment déchargée.

Pour réduire la latence d'exécution des tâches, cette partie de la thèse propose un schéma d'ordonnancement de tâches multiples à ressources qui prend en compte non seulement la conscience de ressources l'hôte isolé et des cluster communautaire (robot/agent de cloud), mais aussi bien le pré-transfert/ pré-migration du déchargement des tâches et le schéma d'évitement des défaillances approprié pour l'exécution des tâches HART. Pour déterminer l'ordre optimal de planification des tâches, cette partie de la thèse compare les schémas suivants: Premier arrivé premier servi (FCFS), premier délai limite de la tâche en premier (EDF), et politique concurrente (CP). Nous comparons la performance de nos schémas proposés de déchargement de tâches en fonction des conscience de la communauté et conscience de la latence avec un schéma de chargement de tâches, un schéma de déchargement de tâche à la aléatoire, et des schéma de déchargement de tâche sensible à la communication en termes d'une variété de métriques de performance spécifiques à HART.

Contrairement aux travaux précédents, notre système de planification des tâches proposé effectue simultanément la sélection du robot/agent et l'assignation de la bande passante, réduisant ainsi la latence d'exécution des tâches multiples. De plus, dans ce travail, la durée totale du service des tâches HART est calculée en tenant compte du traitement de la charge de travail, de la transmission et du délai d'attente. Afin d'éviter des frais supplémentaires de traitement des tâches en raison d'une défaillance (par exemple, l'inaccessibilité des robots/agents), nous concevons un schéma de sélection optimal pour éviter les défaillances. Plus spécifiquement, dans notre schéma de chargement de tâches, les échecs d'exécution des tâches peuvent se produire pendant le traitement complet des tâches d'un robot et le processus de transfert des résultats. Inversement, dans notre schéma de déchargement des tâches, les échecs d'exécution des tâches peuvent se produire pendant le traitement physique d'un robot et le traitement des sous-tâches numériques d'un agent ou pendant le processus de téléchargement des sous-tâches numériques et de téléchargement des résultats. Pour détecter les défaillances pendant l'exécution des tâches, le planificateur de tâches de l'ONU-MPP diffuse périodiquement des messages de pulsation à tous les robots/agents et attend leurs réponses à des points/moments de prédéfinis. Le planificateur de tâches est capable de détecter les pannes d'inaccessibilité d'un robot/agent lorsque les réponses sont absentes à plusieurs points de contrôle ultérieurs. Nous appliquons un schéma d'évitement des défaillances, qui peut être une récupération après échec ou un système de tolérance aux pannes (sélectionné en fonction du délai de service minimum prévu).

Ensuite, pour démontrer l'impact de notre schéma d'attribution de bande passante tenant compte du pré-transfert/pré-migration de tâches, la figure R.10(a) compare l'efficacité du



(a) (b) (c)

Figure R.10: Temps de service moyen des tâches, temps de réalisation, et évaluation de l'efficacité du temps de transfert préalable.

temps de pré-transfert de tâches (p_g) de notre schéma de déchargement de tâches proposé (avec le pré-transfert) vers d'autres schémas pour varier la taille des données d'entrée des tâches de déchargement (d_v^i). Ce chiffre indique clairement que pour les valeurs plus élevées et plus faibles de la taille des données d'entrée des tâches de déchargement, une plus grande efficacité du temps de pré transfert des tâches est obtenue dans notre schéma de déchargement des tâches avec pré-migration, qui est supérieur à celui des autres schémas, y compris le schéma de déchargement de tâches avec transfert/migration conventionnel, schéma de déchargement de tâches aléatoires, et des schéma de déchargement de tâche sensible à la communication.

Ceci est dû au fait que, contrairement à notre schéma de déchargement proposé (conscient de la communauté et de la latence), tous les schémas alternatifs reposent sur la transfert conventionnelle pour le déchargement, souffrant ainsi d'une latence de déchargement multitâche plus élevée. De plus, dans le schéma de déchargement des tâches que nous proposons, chaque tâche est assignée à un nœud de traitement des tâches approprié (hôte et communautaire cluster robot/agent) en tenant compte non seulement des temps de traitement des tâches plus courts, mais aussi des délais de transmission et d'attente. Inversement, dans le schéma de chargement des tâches, la tâche HART complète n'est traitée que par le robot hôte initialement sélectionné, souffrant ainsi d'un retard de traitement numérique de sous-tâche plus élevé que le schéma de déchargement des tâches proposé. Dans le schéma de déchargement de tâche aléatoire et de communication, les nœuds de traitement de tâches sont sélectionnés sur la base d'une base aléatoire et d'un délai de communication inférieur. Ainsi, les schémas de déchargement des tâches aléatoires et de communication traditionnels ne peuvent pas améliorer le délai de service moyen de notre schéma de déchargement de tâche proposé en raison de leur surcharge de traitement des tâches déchargées plus élevée.

La figure R.10(b) montre la performance optimale de la sélection du service de défaillance

de notre schéma de déchargement des tâches proposé. La figure révèle que le temps moyen de service de tâche augmente pour augmenter le temps de récupération de la connexion de service (R) dans tous les schémas considérés. La figure montre également que notre schéma de déchargement de tâche proposé avec tolérance de panne permet d'obtenir le délai de service moyen le plus bas. Notez que les schémas de déchargement de tâches et de chargement de tâches avec reprise après échec présentent une durée de service moyenne des tâches plus faible que les schémas avec tolérance de panne. Ceci est dû au fait que dans le schéma de tolérance aux pannes, l'exécution de la tâche reprend à partir du dernier point de contrôle après récupération à partir de la défaillance de la connexion. Ainsi, nos résultats suggèrent que, pour éviter une défaillance, notre mécanisme de tolérance de panne proposé est plus efficace dans le schéma de déchargement de tâche considéré que les autres mécanismes de récupération d'échec.

La figure R.10(c) illustre le temps global d'achèvement des tâches de notre proposition de déchargement des tâches et et de chargement de tâches schémas en fonction du temps de cycle de scrutin (t_c). La figure montre que le délai global d'achèvement des tâches est plus élevé dans tous les schémas comparés pour les gros t_c. Il est important de noter que ce chiffre montre que les temps de réalisation des tâches à distance ($\bar{t}^k_{tct,r}$) et temps de réalisation des tâches à proximité ($\bar{t}^k_{tct,n}$) sont minimes dans notre proposition de schéma de déchargement des tâches tenant compte en fonction des conscience de la communauté et conscience de la latence. Par exemple, pour $t_c = 400$ ms and $k = 4$, le gain du temps d'achèvement des tâches à proximité et à distance réalisé dans notre schéma de déchargement des tâches proposé est d'environ 23.7% and 24.1% plus élevé que dans le schéma de chargement des tâches, respectivement. Ainsi, notre schéma de déchargement des tâches prenant en charge la latence et la communauté proposé, qui s'appuie à la fois sur la capacité de tolérance aux pannes et la capacité de prétransfert des tâches, est une solution prometteuse pour une collaboration HART à faible latence dans l'Internet tactile émergent.

Coordination des tâches en fonction des préférences de l'utilisateur et allocation proactif de la bande passante dans les infrastructures d'un réseau FiWi

La recherche dans le domaine de l'exécution des tâches HART en fonction des préférences de l'utilisateur en est encore à ses débuts. Au meilleur de notre connaissance, aucune étude existante ne traite du problème de la mise en cache et du calcul de tâches HART sensibles aux délais et tolérants aux retards en tenant compte à la fois de la connaissance des ressources du robot/agent dédié et non dédié et de l'allocation préemptif de la bande passante. À cette fin,

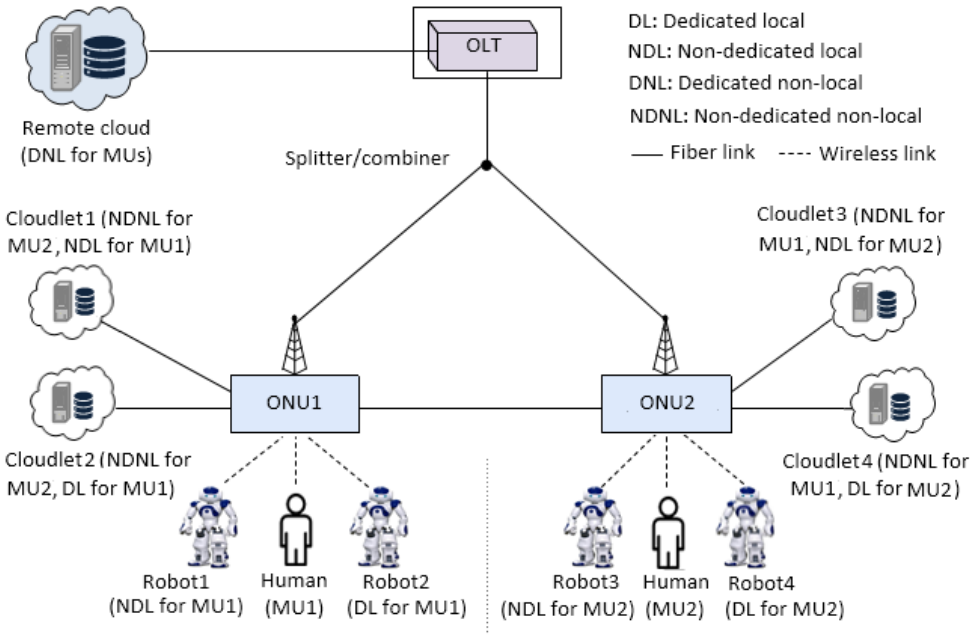
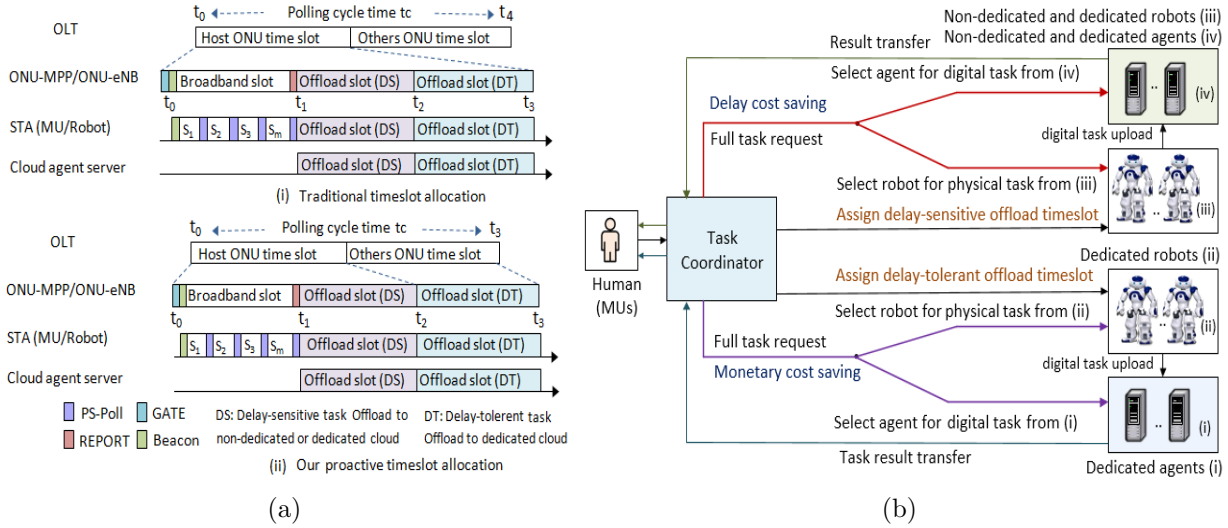


Figure R.11: Robots et agents dédiés et non dédiés locaux et non locaux.

en plus d'une sélection appropriée de l'agent cloud, les schémas de déchargement de tâches précédemment proposés [31], [32], [33] visait à résoudre le problème de la sélection de l'interface sans fil appropriée (4G LTE Advanced ou WiFi) pour le données de tâche de transfert ou le données de tâche de téléchargement, mais pas les deux en même temps. Pour éviter des délais et des coûts d'monétaires supplémentaires tout en atténuant les différentes demandes de tâches d'une utilisateurs mobiles, nous présentons un cadre de coordination des tâches HART conscient en fonction des préférences de l'utilisateur qui sélectionne le robot/agent (dédié ou non dédié) appropriés pour la mise en cache et ou calcul des exigences d'exécution des tâches HART. De plus, pour faire face à la variation des ressources en bande passante, ce travail décrit une politique proactif d'allocation de bande passante pour l'exécution de tâches HART sensibles aux délais et tolérants aux retards.

De plus, pour examiner le compromis de performance entre les schémas de réduction des coûts de retard (DCS) et de réduction des coûts monétaires (MCS) pour l'exécution de différentes tâches HART, cette partie de la thèse développe un cadre analytique en prenant en compte les agents dédié/non dédié avec/sans capacités de cache et en comparant les trois schémas de déchargement multitâches DCS et MCS suivants: (i) débit maximale et retard minimale (MTMD), (ii) débit maximale seulement (MT), and (iii) retard minimale seulement (MD). Contrairement aux études existantes, ce travail prend en compte les acteurs locaux et non locaux (dédié/non dédié) pour l'exécution des tâches. Contrairement aux acteurs non locaux du cloud/robot (dédié/non dédié), les acteurs locaux (dédié/non dédié) sont situés



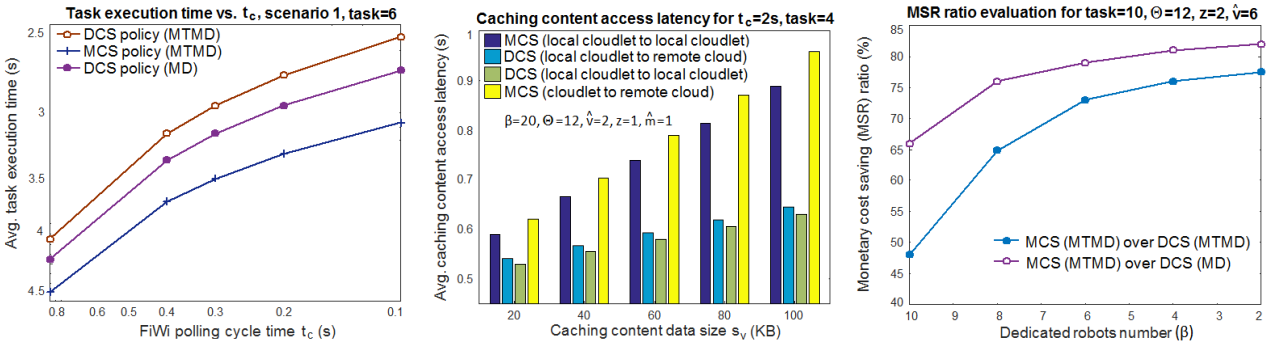
(a) (b)

Figure R.12: (a) Schéma d'allocation de bande passante proactif et (b) schéma de coordination des tâches.

dans la zone de couverture de la station de base hôte (ONU_1 dans la Fig. R.11), où la tâche physique doit être exécutée. Alors que les acteurs non locaux sont situés sous n'importe quelle couverture de station de base (ONU_2 dans la Fig. R.11) sauf la station de base hôte.

La figure R.12(a) illustre plus en détail notre système d'allocation de bande passante proactif à deux couches TDMA pour l'exécution de différentes tâches HART. Dans le schéma que nous proposons, nous divisons les utilisateurs de déchargement de tâches en deux groupes (voir Fig. R.12.b), à savoir les utilisateurs sensibles aux retards (appliquant la politique DCS) et les utilisateurs tolérants aux retards (appliquant la stratégie MCS). Notez que la sous-tâche physique/numérique des utilisateurs de la politique DCS est assignée à un acteur approprié qui peut être un robot/agent de cloud dédié ou non dédié, alors que l'assignation de sous-tâche physique/numérique des utilisateurs de la politique MCS est limitée à un robot/agent de cloud dédié. De plus, dans le schéma que nous proposons, les utilisateurs de la politique DCS déchargent leurs sous-tâches numériques sensibles au retard vers des agents dédiés ou non dédiés appropriés pendant le période de déchargement de l'ONU associé. Inversement, les utilisateurs de la politique MCS déchargent leurs sous-tâches numériques tolérantes au délai à des agents dédiés appropriés uniquement pendant la période de temps d'une autre ONU en utilisant les liaisons par fibre optique d'interconnexion point à point (IF). Ainsi, en effectuant le déchargement de tâche tolérant les délais pendant la période de temps d'une autre ONU, notre schéma proposé est capable d'économiser à la fois la bande passante et le coût monétaire pour les utilisateurs de la politique MCS.

La figure R.13(a) illustre l'impact de la durée du cycle de scrutin (t_c) sur le délai moyen d'exécution des tâches de nos politiques DCS et MCS proposées. La figure montre que le délai



(a) (b) (c)

Figure R.13: Temps moyen d'exécution des tâches, latence d'accès au contenu de la mise en cache, et évaluation de la performance du ratio d'économie monétaire.

moyen d'exécution des tâches augmente avec l'augmentation de t_c. La figure révèle que pour les petites et grandes valeurs de t_c, la politique DCS (MTMD) permet d'obtenir un gain de temps moyen d'exécution des tâches plus élevé que les politiques alternatives. Ceci est dû au fait que les utilisateurs de la politique DCS donnent un accès préemptif aux acteurs et aux ressources en bande passante pour exécuter leurs tâches sensibles au retard. Par exemple, pour $t_c = 0.3$ s, le gain moyen de temps d'exécution des tâches de la politique DCS (MTMD) par rapport à la politique MCS (MTMD) est de 15,42%, contre seulement 6,03% pour la politique DCS (MD). Ce résultat indique que pour l'exécution de tâches sensibles au retard, la politique DCS (MTMD) est la solution supérieure. La figure R.13(b) illustre le délai moyen d'accès au contenu cache pour nos politiques DCS et MCS policies. La figure montre que pour une taille croissante des données de mise en cache (s_v), le délai d'accès au contenu de la mise en cache augmente rapidement dans tous les schémas comparés. De plus, la figure indique que le délai d'accès au contenu du cache devient le plus bas dans la politique DCS, si le cloudlet local de l'hôte récupère le contenu mis en cache à partir d'un autre cloudlet local. La figure montre également que le délai d'accès au contenu du cache devient le plus élevé dans la politique MCS, si le cloudlet local de l'hôte récupère le contenu mis en cache sur le serveur cloud distant. Par exemple, pour $s_v = 80$ Mo, $\hat{m} = 1$, et $n_t = 4$, le schéma DCS (cloudlet local vers un autre mise en cache de cloudlet local) permet d'obtenir un gain de retard d'accès au contenu du cache d'environ 25% et 31% plus élevé que les schémas MCS (cloudlet local vers un autre mise en cache de cloudlet local) et MCS (cloudlet local vers un cloud distant), respectivement. La figure R.13(c) montre clairement qu'une pénurie de robots dédiés (β) a un impact négatif sur la performance en matière d'économie monétaire de nos politiques proposées en matière de DCS et de MCS. La figure révèle que pour différents β, le ratio d'économie monétaire est maximum dans la politique de MCS (MTMD). Il est à noter que l'utilisation d'acteurs dédiés et non dédiés entraîne des coûts monétaires supplémentaires

dans la politique DCS, par opposition à la politique MCS (MTMD). Par exemple, lorsque le nombre de tâches (n_t) est de 10 et $\beta = 6$, le ratio d'économie monétaire dans la politique MCS (MTMD) par rapport aux politiques DCS (MTMD) et DCS (MD) est respectivement de 74% et 80%.

Conclusions

Contrairement à l'IoT sans aucune implication humaine dans ses communications machine-machine sous-jacentes, l'Internet tactile implique la collaboration centrée sur HART et permet ainsi une approche de conception centrée sur l'humain pour créer et consommer de nouvelles expériences immersives via l'Internet. Cette thèse a tenté de faire la lumière sur l'augmentation (c'est à dire., l'extension des capacités) de l'humain par le biais du cadre d'exécution des tâches collaboratives centré sur HART. Pour récolter les bénéfices de la convergence homme-machine, cette thèse a présenté un cadre de coordination des tâches approprié pour orchestrer efficacement la collaboration en temps réel entre les utilisateurs mobiles humains, les agents informatiques centralisés et décentralisés (cloud/cloudlets), et les robots collaboratifs (cobots) à travers les infrastructures réseau convergentes FiWi. A la lumière de l'émergence de l'Internet tactile qui s'oriente vers une décentralisation basée sur l'informatique de pointe, les stations de base intelligentes, l'informatique cloud collaboratif (robots et cloudlets), les capacités de traitement et de stockage distribuées inhérentes aux réseaux améliorés FiWi ont été exploitées pour l'exécution de tâches locales et non locales centrées sur HART. La thèse de doctorat portait sur la coordination des tâches HART sur les réseaux améliorés FiWi en se concentrant sur trois questions majeures, à savoir l'attribution des tâches en fonction de la puissance et de la latence, l'évitement des pannes et l'attribution des ressources de bande passante en fonction du pré-transfert. Pour l'exécution rentable des tâches HART, le premier chapitre de la thèse a examiné les défis de recherche existants, les principales technologies habilitantes et les différentes techniques de communication et de calcul.

Pour rendre plus efficace le processus d'attribution des tâches de l'homme à robot, nous avons proposé dans le chapitre 2 un schéma d'attribution des tâches locales et non locales pour l'exécution des tâches demandées par les utilisateur mobile selon plusieurs paramètres de conception clés tels que la disponibilité, l'ensemble des compétences, la distance par rapport à l'emplacement des tâches, et l'énergie restante des robots. De plus, pour réduire les défaillances pendant l'exécution des tâches, nous avons présenté un mécanisme de signalement des défaillances assisté par robot voisin. Nos résultats montrent que le schéma de sélection du robot basée sur le temps d'exécution minimum estimé surpasse les schémas de sélection traditionnels basés sur la distance minimale et la priorité en termes de délai de bout en bout

et d'énergie résiduelle moyenne. De plus, nous avons observé que le délai d'attribution des tâches non locales est plus élevé que le délai d'attribution des tâches locales.

Au chapitre 3, nous avons présenté une stratégie de calcul collaboratif qui combine la sélection d'un robot hôte approprié pour l'exécution de sous-tâches de surveillance physique (détection) et la sélection de nœuds collaboratifs pour le déchargement de sous-tâches de calcul. Nous avons exploité des serveurs cloud conventionnels, des cloudlets décentralisés et des robots voisins en tant que nœuds collaboratifs pour le déchargement des calculs en support de l'exécution des sous-tâches de calcul demandées par un robot hôte. Les résultats des schémas d'exécution des tâches collaboratives et non collaboratives démontrent que pour un scénario typique, le schéma d'exécution des tâches collaboratives améliore le temps de réponse des tâches jusqu'à 8.75% et la consommation d'énergie jusqu'à 14.98% par rapport au schéma d'exécution des tâches non collaboratives.

Pour une exécution efficace des tâches, le chapitre 4 proposait un schéma de migration des tâches tenant compte du contexte pour orchestrer efficacement la collaboration en temps réel entre les utilisateurs mobiles humains, les agents informatiques centraux et décentralisés (cloud/cloudlets), et les robots collaboratifs (cobots, pour 'collaborative robots') à travers les infrastructures de communication FiWi convergentes. Nous avons examiné la question de savoir si et, dans l'affirmative, quand et où une tâche centrée sur HART devrait être migrée au mieux. Pour une exécution efficace des tâches, la décision de migration est prise en fonction des capacités de traitement des tâches des agents et des cobots, du délai d'exécution des tâches, de la consommation d'énergie des cobots et des appareils mobiles concernés et de la latence de migration des tâches. Nos résultats montrent que pour une sous-tâche cognitive typique de 600 Mo, le schéma de migration des tâches cognitives de cobot à agent (cloudlet près de l'emplacement des tâches) permet d'améliorer de plus de 20% le temps de réponse des tâches et d'économiser 23% d'énergie par rapport au schéma traditionnel de non-migration. Les résultats montrent également que la migration sous-tâches cognitives intra-agent permet d'obtenir un gain de temps de réponse aux tâches plus élevé que la migration inter-agent.

Dans le chapitre 5, nous avons étudié un schéma de planification de tâches HART multiple sensible à la latence et à la communauté en utilisant des informations en temps réel sur les demandes de tâches arrivant pour les robots/agents en cluster isolés et communautaires. Plus précisément, nous avons étudié l'ordre optimal de planification multitâches et la stratégie d'affectation des ressources pour l'exécution de tâches HART basées sur le chargement et le déchargement, avec des capacités de pré-transfert de tâches et de tolérance de pannes. Pour récolter les bénéfices du pré-transfert de tâches pour l'exécution de plusieurs tâches HART, nous avons présenté un nouveau schéma d'allocation de bande passante tenant compte du pré-transfért/pré-migration qui permet de gérer simultanément le trafic de données à large bande

et de déchargement de tâches. Les résultats que nous avons présentés montrent que pour un système type de 32 ONU-MPP et un temps de cycle de vote de 100 ms, notre proposition de schéma de déchargement de tâche pré-transfert permet d'obtenir un gain de temps d'exécution des tâches allant jusqu'à 31.3% et 32.7% par rapport au schéma de chargement des tâches pour l'exécution des tâches HART à proximité et à distance, respectivement.

Enfin, au chapitre 6, pour réduire au minimum les délais d'exécution des tâches et les coûts monétaires, nous avons mis au point un cadre de coordination des tâches HART en fonction des préférences de l'utilisateur qui sélectionne les dédié/non-dédié robot/cloud appropriés pour exécuter différentes tâches HART sensibles aux délais et tolérants aux retards (mise en cache et informatique tâches HART). Pour faire face à des ressources en bande passante limitées, nous avons proposé une politique proactif d'allocation de bande passante pour l'exécution de tâches HART sensibles aux délais et tolérants aux retards. Nous avons observé que pour un nombre de tâches de 10 et 8 robots dédiés disponibles, notre politique DCS (MTMD) proposée présente un ratio d'économie de temps jusqu'à 30.5% plus élevé et un ratio d'économie d'monétaire de 63.6% plus bas que la politique alternative MCS (MTMD).

La dernière partie de la thèse décrit enfin les orientations futures de la recherche qui s'appuient sur les projets que nous proposons. Pour libérer le plein potentiel des applications HART, une orientation de recherche future implique le développement de schémas de coordination des tâches HART pour l'exécution de tâches physiques et numériques basés sur l'utilisation partagée de robots/agents appartenant à l'utilisateur et au réseau. Plus précisément, la question de savoir quand, comment et dans quelles circonstances la propriété des robots mobiles et des agents dans le cloud devient bénéfique en termes de dépenses opérationnelles (OPEX, pour 'operational expenditures') par tâche exécutée représente un problème de recherche intéressant. De plus, la recherche dans le domaine de la migration des tâches conjointes basées sur la communication O2O (O2O, pour 'online to offline or offline to online') ainsi que des schémas appropriés de partage de la bande passante, est une autre direction prometteuse. Pour le déploiement rentable d'applications de réalité mixte, le développement de techniques adaptatives de synchronisation en temps réel, de communication et de calcul ouvre une multitude d'opportunités de recherche futures.

Contents

Acknowledgements ii
Abstract iii
Statement of originality vii
Résumé viii
List of Figures xl
List of Tables xliii
List of Abbreviations xliv

1 Introduction **1**
1.1 Background and Motivation 1
1.1.1 The Tactile Internet: Vision and Requirements 1
1.1.2 Recent Progress 5
1.1.3 FiWi Enhanced Network Infrastructures 8
1.1.4 Human-Agent-Robot Teamwork (HART) 11
1.1.5 Collaborative Computation and Communication Techniques 13
1.2 Objectives 18
1.3 Research Methodology 20
1.4 Contributions of the Thesis 21
1.4.1 Failure-Aware Local and Non-local H2R Task Allocation in FiWi Multi-Robot Infrastructures 21
1.4.2 Collaborative Computing over FiWi Based Tactile Internet Infrastructures 22
1.4.3 HART-centric Task Migration Scheme over FiWi Based Tactile Internet Infrastructures 22
1.4.4 Community- and Latency-Aware Multi-Task Scheduling in FiWi Enhanced Networks 23
1.4.5 User Preference Aware Task Coordination and Resource Allocation in a FiWi Network Infrastructures 24
1.5 List of Publications 25
1.6 Thesis Organization 26

2 Failure-Aware Local and Non-local H2R Task Allocation in FiWi Multi-Robot Infrastructures 28
2.1 Preamble . . . 28
2.2 Introduction . . . 28
2.3 Motivation and Contributions . . . 30
2.4 FiWi Infrastructure for H2R task allocation . . . 31
2.5 Analytical Model . . . 36
2.6 Results . . . 42
2.7 Conclusions . . . 48

3 Collaborative Computing over FiWi Based Tactile Internet Infrastructures 50
3.1 Preamble . . . 50
3.2 Introduction . . . 50
3.3 FiWi Multi-Robot Network infrastructure for H2R Task Allocation . . . 54
3.3.1 Network Architecture . . . 54
3.3.2 Resource Management Scheme . . . 56
3.3.3 Proposed Task Allocation Algorithm . . . 59
3.4 Analytical Model . . . 62
3.4.1 Assumptions . . . 62
3.4.2 Task Allocation Delay . . . 62
3.4.3 Task Execution Time Without Offloading . . . 63
3.4.4 FiWi End-to-End Task Allocation and Offloading Delay . . . 64
3.4.5 Performance Analysis of Computation Offloading and Collaborative Task Execution . . . 66
3.4.6 Task Response Time and Energy Consumption Efficiency . . . 69
3.5 Results . . . 69
3.5.1 Collaborative vs. Non-collaborative Task Execution . . . 71
3.6 Conclusions . . . 79

4 HART-centric Task Migration Scheme over FiWi Based Tactile Internet Infrastructures 80
4.1 Preamble . . . 80
4.2 Introduction . . . 80
4.3 Task Migration: State of the Art and Open Challenges . . . 83
4.4 FiWi Based Tactile Internet Infrastructure for HART-Centric Task Migration . 84
4.4.1 Network Architecture . . . 84
4.4.2 Mobility of Cobots and Human MUs . . . 86

4.5 Cobots and Tasks: Characteristics and Assumptions . . . 88
4.5.1 Characteristics . . . 89
4.5.2 Assumptions . . . 90
4.6 Context-Aware Task Migration Scheme . . . 91
4.7 Analysis . . . 95
4.7.1 Polling Cycle Time and Task Migration Subslot . . . 95
4.7.2 Task Migration Packet Delay . . . 96
4.7.3 Task Response Time . . . 97
4.7.4 Energy Consumption . . . 99
4.7.5 Task Response Time and Energy Efficiency . . . 100
4.7.6 Migration Gain Overhead Ratio . . . 100
4.7.7 Deadline Miss Ratio . . . 100
4.7.8 Task Blocking Probability . . . 101
4.7.9 Communication-to-Computation Ratio (CCR) . . . 101
4.7.10 End-to-End Task Execution Delay . . . 101
4.8 Results . . . 103
4.9 Conclusions . . . 110

5 Community- and Latency-Aware Multi-Task Scheduling and Prefetching-Aware DBA in FiWi Enhanced Networks 112
5.1 Preamble . . . 112
5.2 Introduction . . . 112
5.3 Related Work and Open Challenges . . . 115
5.4 FiWi Enhanced Tactile Internet Infrastructure for HART Task Scheduling . . . 116
5.4.1 Network Architecture . . . 117
5.4.2 Adaptive HART Task Scheduling Scheme . . . 118
5.4.3 Prefetching-Aware Dynamic Bandwidth Allocation . . . 122
5.5 Performance Analysis . . . 125
5.5.1 HART Task Service Time Analysis . . . 125
5.5.2 Transmission Time Delay . . . 126
5.5.3 Waiting Time Delay . . . 127
5.5.4 Task Service Time With Failure Recovery . . . 128
5.5.5 Task Service Time With Fault-Tolerance . . . 129
5.5.6 Task Completion Time . . . 130
5.5.7 Power Saving Ratio . . . 131
5.5.8 Task Prefetching Time Efficiency and TSO Ratio . . . 131
5.5.9 Satisfactory Ratio, Speed Up, and PSR . . . 132

5.6 Results . . . 132
5.7 Conclusions . . . 139

6 User Preference Aware Task Coordination and Proactive Bandwidth Allocation in a FiWi Network Infrastructures 141
6.1 Preamble . . . 141
6.2 Introduction . . . 141
6.3 Prior Art and Open Challenges . . . 143
6.4 FiWi Enhanced Tactile Internet Infrastructure for Preference Aware HART Task Coordination . . . 144
6.4.1 Network Architecture . . . 144
6.4.2 Preference Aware HART Task Coordination . . . 145
6.4.3 Proactive Bandwidth Allocation Scheme . . . 149
6.5 Performance Analysis . . . 150
6.5.1 Delay Analysis . . . 150
6.5.2 Caching Content Access Latency . . . 151
6.5.3 Mean Task Offload Delay . . . 152
6.5.4 Communication Delay . . . 154
6.5.5 Monetary Cost . . . 154
6.5.6 Time and Monetary Cost Saving Ratios . . . 156
6.5.7 Energy Cost . . . 156
6.5.8 Communication to Computation Ratio . . . 157
6.5.9 Task Offload Gain to Overhead Ratio . . . 157
6.6 Results . . . 159
6.7 Conclusions . . . 165

7 Conclusions and Future Research 167
7.1 Conclusions . . . 167
7.2 Future Research . . . 169

Bibliography 173

List of Figures

R.1 L'Internet tactile : applications, défis, et technologies habilitantes. xi
R.2 Aperçu des principaux défis de la coordination HART sur l'Internet tactile. . . xii
R.3 Méthodologie de recherche. xv
R.4 Efficacité du temps d'exécution des tâches, délai d'allocation, et évaluation du taux d'erreur de détection. xviii
R.5 (a) Structure temporelle et format de la trame de contrôle et (b) étapes opérationnelles du processus de déchargement des calculs. xx
R.6 Temps de réponse aux tâches et évaluation de la consommation d'énergie des différents systèmes collaboratifs/conjoints et non collaboratifs. xxi
R.7 Infrastructure Internet tactile basée sur FiWi, basée sur des cloud-cloudlets intégrés, des cobots, et des MU humaines pour la migration de tâches HART. . xxiii
R.8 Temps de réponse des tâches, consommation d'énergie, efficacité du temps de réponse des tâches, et évaluation des retards de bout en bout. xxiv
R.9 Allocation de ressources basée sur le transfert préalable pour le déchargement multitâche. xxvi
R.10 Temps de service moyen des tâches, temps de réalisation, et évaluation de l'efficacité du temps de transfert préalable. xxviii
R.11 Robots et agents dédiés et non dédiés locaux et non locaux. xxx
R.12 (a) Schéma d'allocation de bande passante proactif et (b) schéma de coordination des tâches. xxxi
R.13 Temps moyen d'exécution des tâches, latence d'accès au contenu de la mise en cache, et évaluation de la performance du ratio d'économie monétaire. xxxii

1.1 (a) Evolutionary leap of the Tactile Internet; (b) the three lenses of IoT, 5G, and the Tactile Internet: commonalities and differences. 3
1.2 The Tactile Internet: applications, challenges, and enabling technologies. . . . 6
1.3 (a) Cloudlet Empowered Ethernet based FiWi Enhanced 4G LTE-A HetNets; (b) Coexistence of Cloudlet (D-RAN) and C-RAN over FiWi Enhanced 4G LTE-A HetNets. 9

1.4 Overview of key challenges in HART over the Tactile Internet 12
1.5 Research methodology 20

2.1 FiWi communication infrastructure for H2R task allocation. 31
2.2 Resource management scheme for H2R task allocation in FiWi multi-robot networks. 32
2.3 Proposed task allocation mechanism: (a) timing structure and (b) modified control frames. 36
2.4 Throughput, task execution time, and average residual energy variation. 43
2.5 Performance comparison of our proposed method with MD and PS based approach. 46
2.6 Total failure sensing error rate evaluation. 47
2.7 Local and non-local task allocation end-to-end delay evaluation. 48

3.1 Integrated FiWi multi-robot network architecture for coordinating local and non-local H2R task allocation. 55
3.2 Proposed resource management scheme. 56
3.3 (a) Timing structure and control frame format, and (b) operational steps of computation offloading process. 57
3.4 Task response time and energy consumption variation of collaborative and non-collaborative task execution schemes versus total task input data size for three different scenarios. 73
3.5 Average task response time and energy consumption comparison of our collaborative (host robot-central cloud, host robot-cloudlet, host robot-neighbor robot) and non-collaborative (host robot without offloading) schemes with existing schemes. 76
3.6 (a) Task response time efficiency versus total task input data size; (b) energy consumption efficiency versus total task input data size; (c) offload gain overhead ratio versus computation sub-task input data size. 77
3.7 End-to-end local and non-local task response time variation versus FiWi traffic load. 78

4.1 FiWi based Tactile Internet infrastructure based on embedded cloudlets, cobots, and human MUs for HART-centric task migration. 85
4.2 Random waypoint (RWP) model of human MUs for predicting location of task request transmission and task result reception. 87
4.3 HART-centric task migration: (a) timing structure and (b) operational steps. . 94
4.4 Task migration packet delay components. 96

4.5 Task response time, energy consumption, task response time, energy efficiency, and migration gain-overhead ratio vs. task input data size (s_i) evaluation of different task migration schemes under scenario 1 and 2. 105
4.6 Communication-to-computation ratio (CCR), average utilization of collaborative node, task blocking probability, deadline-miss ratio, and end-to-end task execution delay evaluation of different task migration schemes. 108
4.7 Task response time efficiency of proposed task migration scheme over non-migration scheme for Scenario 3 task workload settings. 109

5.1 Resource block allocation in task offloading: (a) prefetching vs. (b) conventional fetching. 114
5.2 FiWi enhanced Tactile Internet infrastructure for adaptive HART task scheduling. 117
5.3 (a) Adaptive batch model for proposed HART task scheduling and (b) community-cluster architecture. 118
5.4 HART task execution model with failure avoidance. 120
5.5 Space-time diagram of prefetching-aware dynamic bandwidth allocation (DBA). 121
5.6 Illustration of prefetching-aware task offload sub-slot assignment. 124
5.7 Mean task service time, power consumption, delay and power saving ratio, prefetching time efficiency, and satisfactory ratio evaluation for scenario 1 and scenario 2. 136
5.8 Task service time gain to overhead ratio (TSO), speed up ratio, mean task service time with failure avoidance, PSR, and overall task completion time evaluation of our proposed community and latency-aware task offloading scheme. 138

6.1 FiWi enhanced Tactile Internet infrastructure for preference aware HART task coordination. 145
6.2 Local and non-local dedicated/non-dedicated robots and agents. 147
6.3 (a) Proactive bandwidth allocation scheme and (b) task coordination scheme. . 149
6.4 Mean offload packet delay components in DCS policy. 153
6.5 Mean offload packet delay components in MCS policy. 154
6.6 Average task execution time, monetary cost, task execution time cost saving ratio, monetary cost saving ratio, and total energy consumption cost performance for scenario 1 161
6.7 Mean Task offload delay, TGO ratio, C2R ratio, caching content access latency, and average task execution time performance. 164

List of Tables

2.1 Parameters and default values for local and non-local H2R task allocation . . . 42

3.1 Comparison of proposed scheme with existing task allocation schemes 53
3.2 Notation and default values for evaluation of collaborative computing based task execution scheme 70

4.1 Cobots vs. robots 88
4.2 Cognitive vs. physical tasks 90
4.3 Parameters and default values for evaluation of context-aware task migration strategies 102

5.1 Parameters and default values for community- and latency-aware multi-task onloading and offloading scheme evaluation 133

6.1 Parameters and default values for evaluation of user preference-aware task and resource assignment scheme 158

List of Abbreviations

AMAT	Average Memory Access Time
ANDSF	Access Network Discovery and Selection Function
BBU	Baseband Units
BS	Base Station
C2A	Cobot-to-Agent
C2C	Cobot-to-Cobot
CCDF	Complementary Cumulative Distribution Function
CCR	Communication-to-Computation Ratio
CP	Concurrent Policy
CPS	Cyber Physical System
CRAN	Centralized Cloud Based Radio Access Network
DBA	Dynamic Bandwidth Allocation
DCS	Delay Cost Saving Policy
DIFS	Distributed Coordination Function (DCF) Interframe Space
DRAN	Decentralized Radio Access Network
DS	Downstream
EDF	Earliest Deadline First
FCFS	First Come First Served
FiWi	Fiber-Wireless

H2H	Human-to-Human
H2M/R	Human-to-Machine/Robot
HART	Human-Agent-Robot Teamwork
HCCA	Hybrid Coordination Function Controlled Channel Access
HetNets	Heterogenous Networks
HITL	Human In The Loop
IoT	Internet of Things
KB	Kilo Bytes
M2C	Machine-to-Cloud
M2M	Machine-to-Machine
MAC	Medium Access Control
MCC	Mobile Cloud Computing
MCS	Monetary Cost Saving Policy
MD	Min Workload Processing Delay
MEC	Mobile Edge Computing
MPCP	Multi-Point Control Protocol
MPP	Mesh Portal Point
MRTA	Multi-Robot Task Allocation
MT	Maximum Throughput
MTC	Machine Type Communications
MTMD	Maximum Throughput and Minimum Delay
NFV	Network Function Virtualization
OLT	Optical Line Terminal
ONU	Optical Network Unit

OPEX	Operational Expenditure
PON	Passive Optical Network
PSR	Processing-to-Service Time Ratio
QoE	Quality of Experience
QoS	Quality of Service
R&F	Radio and Fiber
RLNC	Random Linear Network Coding
RoF	Radio over Fiber
RRH	Remote Radio Heads
RWP	Random Waypoint
SDN	Software Defined Networks
SIFS	Short Interframe Space
TDD	Time Division Duplexing
TDM	Time Division Multiplexing
TDMA	Time Division Multiple Access
TSO	Task Service Time Gain to Overhead Ratio
US	Upstream
WDM	Wavelength Division Multiplexing
WMN	Wireless Mesh Network

Chapter 1

Introduction

1.1 Background and Motivation

1.1.1 The Tactile Internet: Vision and Requirements

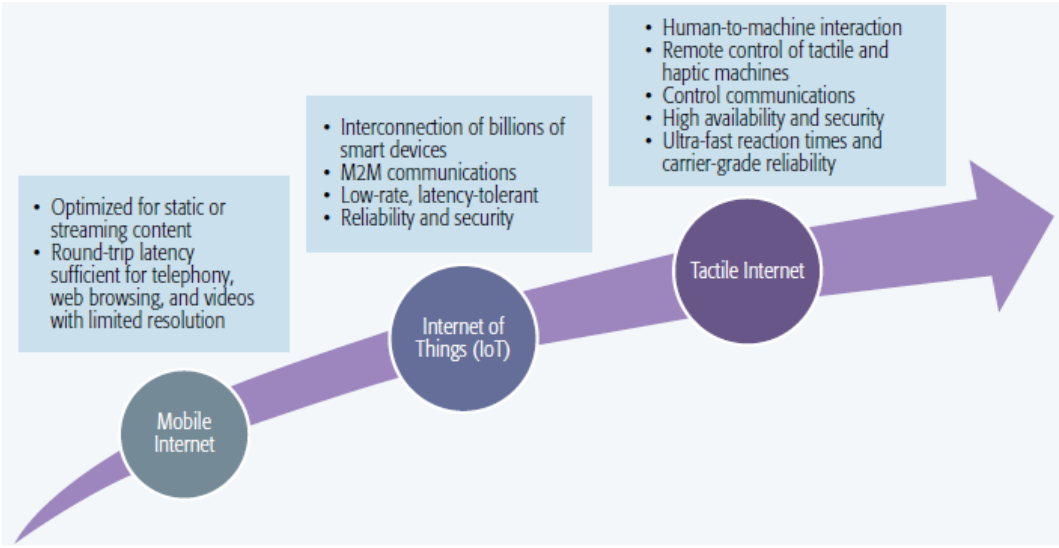
In the last decade, rapid technological advancement has changed people's way of living and their future expectations. We have witnessed tremendous improvements of mobile Internet technology to connect people anytime and anywhere [1]. Beside voice and data communications, the mobile Internet enables real-time access to richer content (e.g., video streaming, instant messaging, file sharing) [2]. In order to provide ubiquitous connectivity for machines and devices, the research focus of mobile communications has been shifting towards the emerging Internet-of-Things (IoT), which enables applications of machine-to-machine (M2M) or machine type communication (MTC) with a focus on smart devices such as robots, sensors, actuators, and wearable devices [3]. Once machines/robots become connected to the Internet, the next natural leap is to control them remotely for delivering low-latency human-machine interaction centric services (e.g., 3D gaming, powered exoskeleton). This vision of the Internet is now widely known as the so-called Tactile Internet, which has recently emerged to steer/control virtual and physical objects of our surroundings and environments and allow one to transmit touch and actuation in real-time [4].

With the advent of commercially available remote-controlled robots/machines, the Tactile Internet may be the precursor of an age of technological convergence, where tasks of our everyday life (e.g., cognitive assistance in household activities) will be increasingly done by robots/machines that allow us to see, hear, touch, and manipulate objects in places where we are not physically present. In various cyber-physical systems (CPSs) that harness real-time human-machine interaction (e.g., remote training, mission-critical rescue operations), including virtual and augmented reality, an extremely low round-trip latency is required to match human interaction with the environment [3]. An important CPS example is the smart grid and its fast response time requirements of 1 ms in the event of (cascading) power network failures.

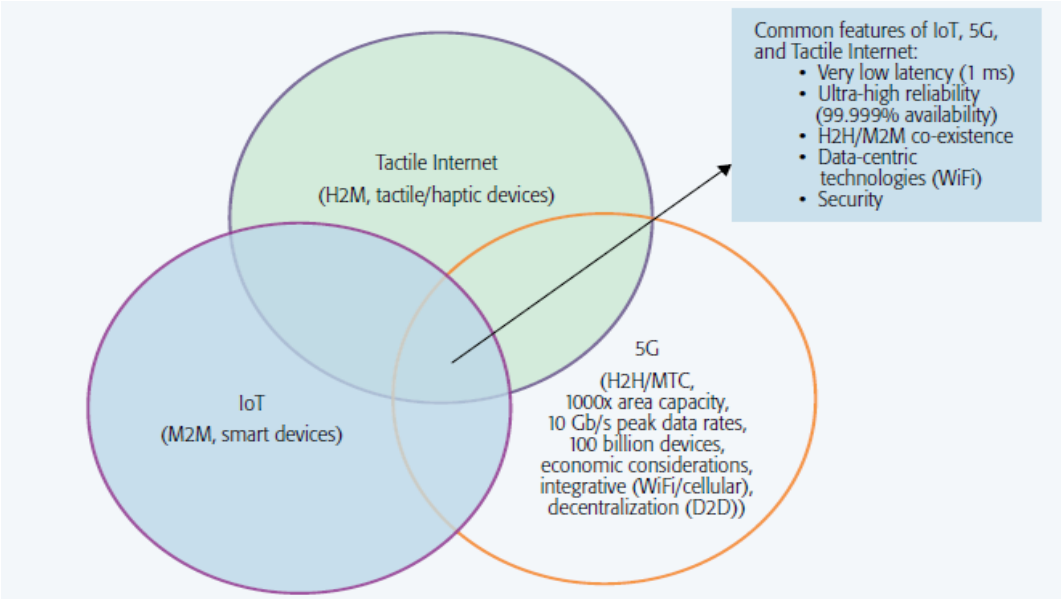
Current cellular and WLAN systems miss this target by at least one order of magnitude. A very low round-trip latency in conjunction with ultra-high reliability and essentially guaranteed availability for control communications have the potential to move today's mobile broadband experience into the new world of the Tactile Internet for a race with machines (rather than against) [5]. By offering low-latency communications, the Tactile Internet is expected to cover a wide range of application fields, including remote health-care, autonomous/assisted driving, entertainment, and industry automation. In most of these industry verticals, very low latency and ultra-high reliability are key for realizing immersive applications such as robotic tele-operation [6]. From the business perspective, a recent market study has predicted that the Tactile Internet could create commercial value of up to US$20 trillion worldwide, which is around 20% of today's overall GDP [34].

The evolutionary leap of the Tactile Internet is shown in Fig. 1.1(a). The conventional mobile Internet facilitates voice and data communications and provides the medium for audio/visual transport. Conversely, the Tactile Internet will enable remote real-time human-to-machine/robot (H2M/R) interaction by delivering tactile/haptic sensations [35]. It holds promise of an Internet that will enable the delivery of skills in digital form globally and provide a true paradigm shift from traditional content-delivery networks to labor/skill-set delivery networks via tactile/haptic devices [36]. The Tactile Internet envisions to enable reliable and adaptive networked control systems, where master and slave domains are connected and highly dynamic processes are controlled remotely. Unlike the mobile Internet and IoT, the Tactile Internet will facilitate haptic communications by providing the medium for transporting haptic senses (i.e., touch and actuation) in real-time in addition to conventional non-haptic data, video, and audio traffic. Unlike auditory and visual senses, the sense of touch occurs bilaterally in haptic communications, i.e., it is sensed by imposing a motion on the environment and feeling the environment by a distortion or reaction force. Haptic information is composed of two distinct types of feedbacks: kinesthetic (i.e., force, torque, position, velocity) and tactile feedback (i.e., texture, friction). The key difference between haptic and non-haptic control is that haptic feedback is exchanged through a global control loop with stringent latency constraints, whereas non-haptic feedback is only audio/visual and there is no notion of a closed control loop [37].

Beside haptic communication, another distinct aspect of the Tactile Internet is the fact that it should amplify the differences between machines and humans and drive the symbiosis between man and machine. This design approach is known as *Human-Agent-Robot Teamwork (HART)*, in which humans, agents (e.g., cloud servers), and robots work collaboratively to accomplish different types of task (e.g., remote medical assistance, remote sensing, remote food supply) [11]. HART differs from the traditional approaches (either human-only activity



(a)



(b)

Figure 1.1: (a) Evolutionary leap of the Tactile Internet; (b) the three lenses of IoT, 5G, and the Tactile Internet: commonalities and differences.

or machine-only activity), which have been viewing humans and machines as rivals, each side fighting for the other's job. In HART, humans work with machines to exploit what each party does best. On the one hand, humans may train and manage machines to perform challenging tasks, explain machine outcomes, and sustain machines in a responsible manner. On the other hand, machines may amplify humans' insights and their intuition by leveraging data analytics, interact with humans at scale using novel interfaces, and embody physical attributes that help extend a person's capabilities. Details on characteristics and challenges of HART-centric task

execution will be further discussed in Section 1.1.4.

To facilitate a better understanding of the Tactile Internet, Fig. 1.1(b) depicts the commonalities and subtle differences between IoT, 5G, and the Tactile Internet. The high availability, ultra-fast reaction times, and carrier-grade reliability of the Tactile Internet will add a new dimension to human-machine interaction by enabling tactile and haptic sensations. On the other hand, future 5G networks will have to be able to cope with the unprecedented growth of mobile data traffic as well as the huge volumes of data generated by smart devices enabling the IoT. Towards this end, the 5G technology vision foresees 1000-fold gains in area capacity, 10 Gb/s peak data rates, and connections for at least 100 billion devices. The key challenge of 5G wireless access and core network architectures is to make it possible to address novel machine-centric use cases such as mission-critical traffic safety and control of critical infrastructures (e.g., smart power grids), which are currently not addressed by cellular networks. Some of these envisioned 5G use cases require very low latency and ultra-high reliability with essentially guaranteed availability. Thus, beside very low latency, 5G has to enable connectivity, whose reliability will have to be orders of magnitude higher than in current radio access networks. Unlike the previous four generations, 5G will also be highly integrative. The integrative vision of 5G will lead to an increasing integration of cellular and WiFi technologies and standards. Another important aspect of the 5G vision is decentralization by evolving the cell-centric architecture into a device-centric one and exploiting edge intelligence in close proximity to wireless end users (humans or machines).

Clearly, the discussion above shows that there is a significant overlap among IoT, 5G, and the Tactile Internet, though each one of them exhibits unique characteristics, as shown in Fig. 1.1(b). The major differences may be best expressed in terms of underlying communications paradigms and enabling end devices. IoT relies on M2M communications with a focus on smart devices (e.g., sensors and actuators). In co-existence with emerging MTC, 5G will maintain its traditional human-to-human (H2H) communications paradigm for conventional triple-play services (voice, video, data) with a growing focus on the integration with other wireless technologies (most notably WiFi) and decentralization. Conversely, the Tactile Internet will be centered around H2M communications leveraging tactile/haptic devices. More importantly, despite their differences, IoT, 5G, and the Tactile Internet seem to converge toward a common set of important design goals: very low latency on the order of 1 ms, ultra-high reliability with an almost guaranteed availability of 99.999 percent, H2H/M2M coexistence, integration of data-centric technologies with a particular focus on WiFi, and security [5].

1.1.2 Recent Progress

The realization of Tactile Internet applications based on real-time H2R communications will not be realized without addressing several system design challenges. Real-time H2R based applications are sensitive to end-to-end latency, which comprises various delay components (e.g., channel access, queuing, transmission, and propagation delay) experienced during the communication process between human operators and remotely controlled robots/machines. If the end-to-end latency exceeds the human reaction time (100 ms for auditory, 10 ms for visual, 1 ms for haptic), the experience becomes less realistic due to the large gap between stimulation and response [3]. A service can be defined as real-time, when the communication response time is faster than the time constants of the application. Humans have the ability to react to sudden environmental changes by using their muscles, e.g., reacting to a sudden unforeseen incident by hitting the brakes in a car or quickly pulling back a hand after touching a hot platter on a stove. Note that there are two different time scales of human reaction, depending on being prepared or unprepared for the situation. If unprepared, the sensing-to-muscular reaction time is in the range of 500 ms to 1 s. Translating this to comparable situations in technical applications sets the targets for specifications and design requirements. Clearly, if humans are prepared for a situation, faster reaction times are needed, such as when driving a formula-1 car in a race [2]. To obtain a low round-trip latency, the authors of [7] emphasized that the concept of locally available edge-cloud servers/cloudlets will enable us to realize the vision of the Tactile Internet. Even at the speed of light (e.g., in optical fiber access networks), a round-trip propagation delay of 1 ms requires a computing/processing server within 150 km distance from the point of tactile interaction. This computing/processing server (e.g., cloudlet) at the edge of the mobile radio access network (WiFi/LTE-A base station) is a central part of the *mobile-edge cloud* computing concept. Cloudlets may be viewed as decentralized proxy cloud servers with processing and storage capabilities just one or more wireless hops away from the mobile user. Cloudlet research has tended to focus on WiFi in the past, though recently there has been a growing interest among cellular network operators. The importance of cloudlets can be witnessed in many end-to-end latency-sensitive applications such as augmented reality, real-time cognitive assistance, or face recognition on mobile devices. Recently, to manage and offload high volumes of data, Akamai developed the *Edge Redirector Cloudlet*, which is an early example of commercial applications of the cloudlet concept. In September 2014, the so-called *mobile-edge computing (MEC)* industry initiative introduced a reference architecture in order to identify challenges that need to be overcome to facilitate the implementation of cloudlet servers [8]. MEC provides IT and cloud computing capabilities in the radio access network (RAN) in close proximity to mobile subscribers. Moreover, MEC aims at transforming mobile base stations into intelligent service hubs by exploiting proximity, context, agility, and speed

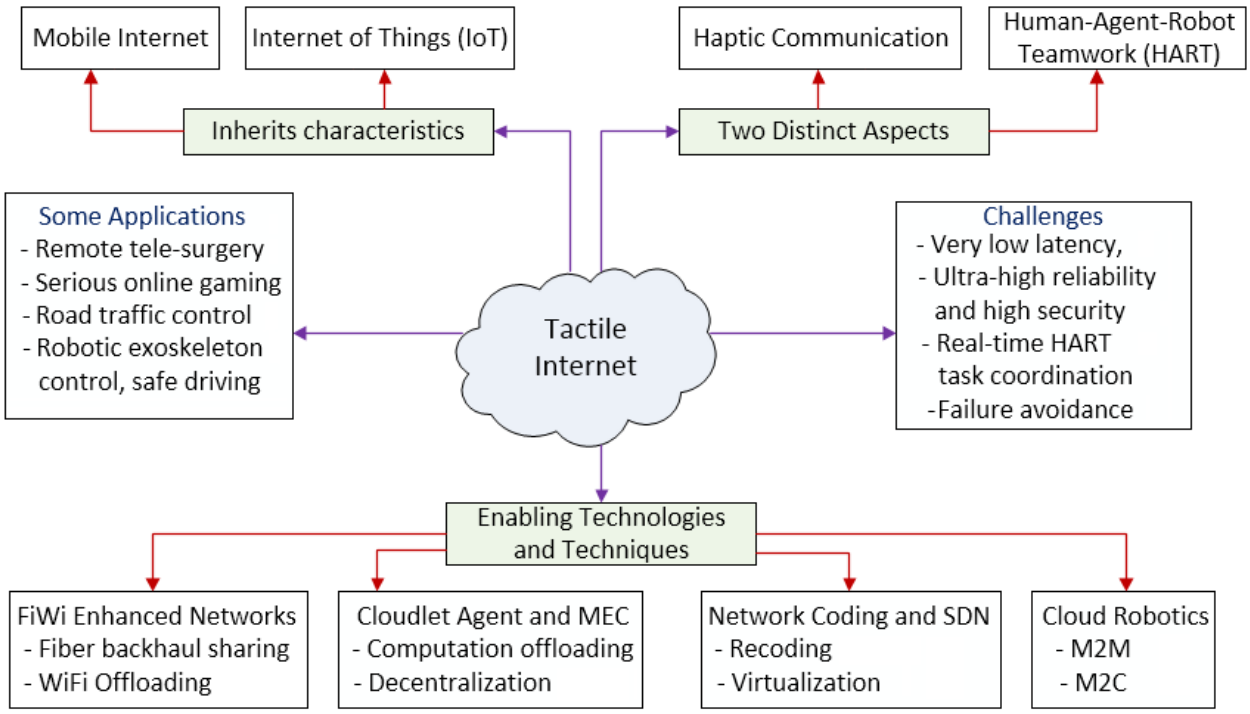


Figure 1.2: The Tactile Internet: applications, challenges, and enabling technologies.

in order to create a new value chain and stimulate revenue generation. It is expected that advanced caching, computation offloading, and user-oriented traffic management at the edge of wireless networks will not only reduce backhaul traffic loads but also improve latency of Tactile Internet applications.

The Tactile Internet application sets demanding requirements for future access networks in terms of latency, reliability, and capacity. To achieve the 5G and Tactile Internet key requirements of very low latency and ultra-high reliability, in [9], the authors proposed the concept of so-called FiWi enhanced LTE-A HetNets that unifies coverage-centric 4G mobile networks and capacity-centric fiber-wireless (FiWi) broadband access networks based on data-centric optical fiber and wireless Ethernet technologies. By means of probabilistic analysis and verifying simulations based on recent and comprehensive smartphone traces the authors showed that an average end-to-end latency of 1-10 ms and almost guaranteed availability can be achieved via fiber backhaul sharing and WiFi offloading capabilities. Note, however, that only conventional H2H communications was considered in [9] without any coexistent H2R or M2M communications. To realize low-latency H2R communications in the Tactile Internet, in [5] the authors discussed the role of several key enabling technologies, including FiWi enhanced LTE-A HetNets, cloudlets, cloud robotics, network coding, and software-defined networking (SDN), among others, as shown in Fig. 1.2. To extend the capabilities of both tele-operated and multi-robot based networked robotics for different Tactile Internet applications, this work

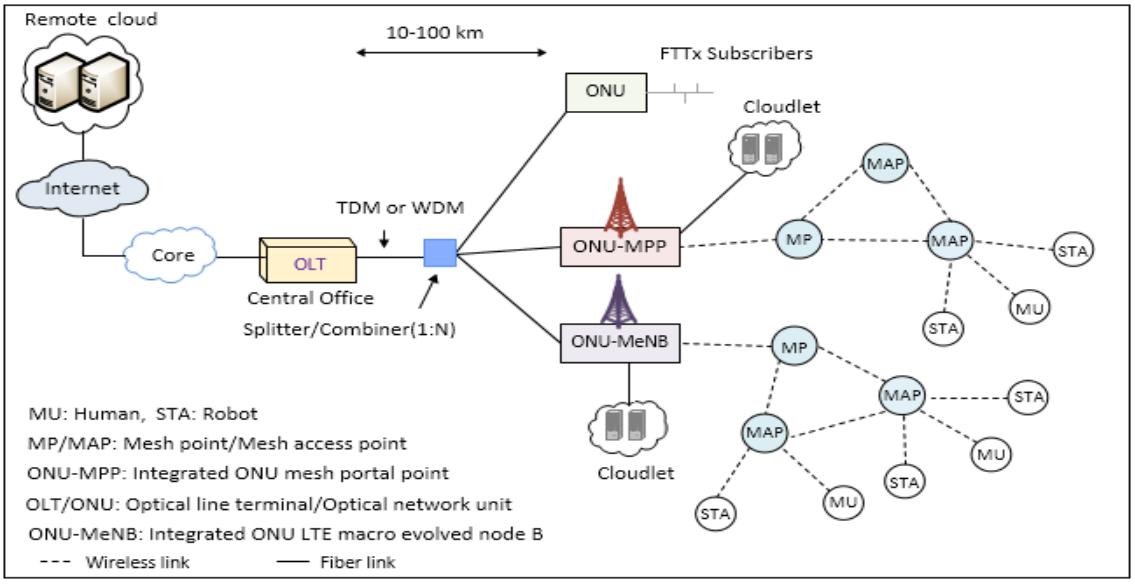
also elaborated on the importance of a cloud robotic system architecture that leverages the combination of an ad-hoc cloud formed by M2M communications among participating robots and an infrastructure cloud enabled by machine-to-cloud (M2C) communications between the robots and the remote cloud. M2M communications was used to enable a team of networked robots to complete tasks cooperatively in a distributed fashion by sharing computation/storage resources and exchanging information via a wireless communication network. M2C communications makes it feasible to learn from the shared history of all cloud-enabled robots. Moreover, by leveraging on the high throughput, reliability, and in particular delay performance of FiWi enhanced LTE-A HetNets, the authors reported that integrated FiWi multi-robot infrastructures based on decentralized cloudlets will be essential for the coordination of Tactile Internet applications based on H2R communications. For the cost-effective deployment of Tactile Internet applications, the authors also identified several important research challenges such as the design of adaptive bandwidth resource management techniques for the support of both H2H and H2R traffic over FiWi enhanced networks with proper service coordination, H2R task allocation strategies (optimal online/offline scheduling), failure handling, and mobility management, among others.

To speed up the execution of Tactile Internet applications, in [38], the authors claimed that the extensive use of a flexible network coding mechanism such as random linear network coding (RLNC) throughout the network can improve the latency performance and reduce the frequency of required packet retransmissions. RLNC is the most general form of network coding, whose main characteristics are recoding and a sliding window based operation. Although network coding and SDN hold promise to reduce end-to-end latency, further investigations are needed to explore the use of the sliding window approach in multi-path SDN based networks to improve their throughput and resilience performance. Additionally, to provide high tracking performance between the master (human user) and slave (robot) domains, the authors of [39] demonstrated that predictive resource allocation is necessary in both upstream and downstream data transmission. Recently, in [40], the author studied the uplink radio resource allocation problem for haptic communications, whereby the queueing delay and queueing delay violation probability were taken into account. Recently, the authors of [41] optimized the number of subchannels, the bandwidth of each subchannel, and threshold for each device to minimize the total bandwidth required by the system for ensuring the reliability of H2R communications. Further, in [42] and [43], a time division duplex (TDD) based energy-efficient resource allocation scheme was presented for Tactile Internet users. More recently, in [44] and [45], the authors investigated the feasibility of IEEE 802.11 hybrid coordination function controlled channel access (HCCA) for delay-sensitive Tactile Internet applications under different system settings. Note that none of the aforementioned studies considered the co-existence of

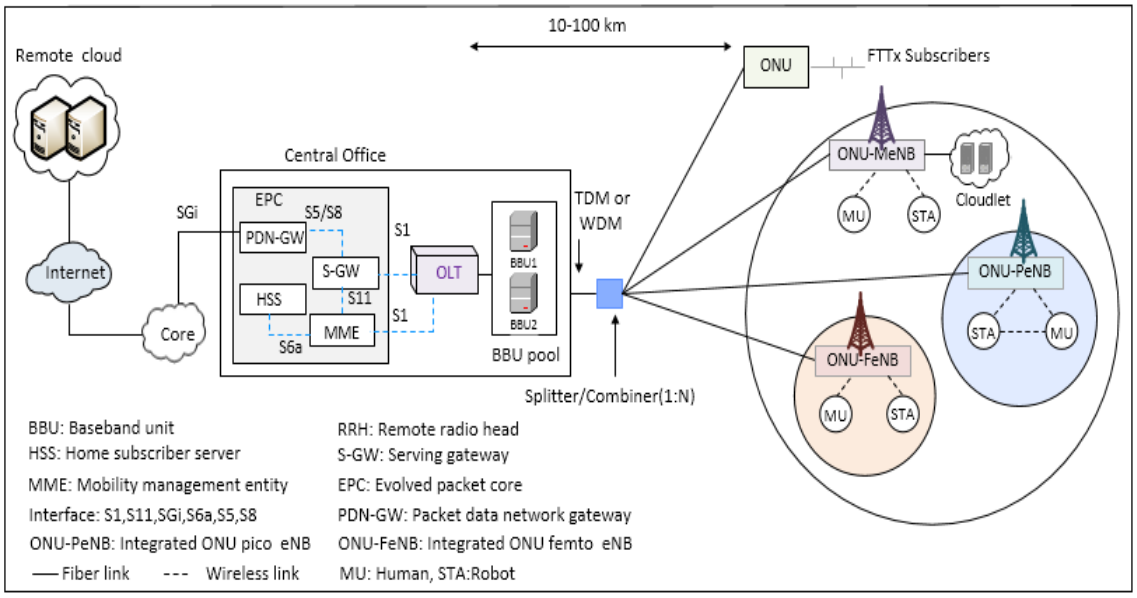
latency-sensitive Tactile Internet (haptic and non-haptic) applications and other bandwidth-intensive H2H applications nor the impact of user mobility. Furthermore, a comprehensive end-to-end H2R communication delay analysis including queuing delay, task processing delay, uplink, and downlink transmission delay analysis is missing in the existing literature.

1.1.3 FiWi Enhanced Network Infrastructures

To enable cost-effective solutions for real-time Tactile Internet applications, network operators have been looking for reliable, fast, low-cost, and future-proof communication infrastructures. To tackle this challenge, integrated fiber-wireless (FiWi) access networks that combine the high capacity and reliability of optical fiber networks with the ubiquity and mobility of wireless networks represent a promising communication platform [46], [47]. According to the IEEE Technical Sub-committee on Fiber-Wireless (Sub-TC FiWi) integration, the role of FiWi integration is defined as follows: *"The Sub-TC on Fiber-Wireless integration addresses architectures, techniques, and interfaces for the integration of fiber and wireless network segments in a unified wired-wireless infrastructure. It does not address architectures or techniques specific to individual optical or wireless networks."* Note that in our work FiWi networks are based on optical fiber (Ethernet passive optical network or EPON) and wireless (wireless local area network or WLAN) Ethernet technologies [48], which are then integrated with their cellular counterparts, namely, 4G Long Term Evolution Advanced (LTE-A), to give rise to FiWi enhanced LTE-A heterogeneous networks (HetNets) [9]. For illustration, Fig. 1.3(a) depicts the generic architecture of cloudlet empowered Ethernet-based FiWi enhanced networks, which are based on the integration of IEEE 802.3ah/av time division multiplexing (TDM)/wave division multiplexing (WDM) EPON in the optical backhaul and IEEE 802.11ac WLAN and 4G LTE-A technologies in the wireless front-end. The optical fiber backhaul consists of a passive optical network (PON) with a fiber range of 10-100 km between optical line terminal (OLT) and optical network units (ONUs). The PON may comprise multiple stages, each stage separated by a wavelength-broadcasting splitter/combiner (or alternatively a wavelength multiplexer/demultiplexer). The PON comes in two flavors: (i) TDM PON and (ii) WDM PON. The OLT is located at the central office to serve three different subsets of ONUs through a 1:N optical splitter/combiner. To provide FTTx services (e.g., fiber-to-the-home/business), the first subset of ONUs serves a single or multiple attached fixed (non-mobile) wired subscribers, called fixed wired users, that may be located at the premises of residential or business subscribers. The second subset of ONUs connects to a cellular network base station (BS), i.e., LTE macro evolved node B (MeNB), giving rise to a so-called ONU-MeNB. In the third subset, ONUs have a mesh portal point (MPP) to interface with the WiFi mesh network, whereby



(a)



(b)

Figure 1.3: (a) Cloudlet Empowered Ethernet based FiWi Enhanced 4G LTE-A HetNets; (b) Coexistence of Cloudlet (D-RAN) and C-RAN over FiWi Enhanced 4G LTE-A HetNets.

mesh points (MPs) act as intermediate relay nodes and mesh access points (MAPs) serve mobile users (MUs) within their coverage area. The integration of ONU and MPP (referred to as ONU-MPP) is realized by using so-called radio-and-fiber (R&F) technologies with medium access control (MAC) protocol translation taking place at the optical-wireless interface, as explained in more detail shortly. To provide cloud computing services at the network edge, cloudlet servers are connected to ONU-MPPs/ONU-MeNBs through point-to-point optical

fiber links.

FiWi networks can be categorized into two different types: (*i*) traditional radio-over-fiber networks (RoF) and (*ii*) radio-and-fiber networks (R&F) [49]. While RoF networks use optical fiber as an analog transmission medium between a central office and one or more remote antenna units (RAUs) with the central office being in charge of controlling access to both optical and wireless media, in decentralized R&F networks access to the optical and wireless media is controlled separately from each other by using two different MAC protocols in the optical and wireless media, with protocol translation taking place at their interface. An example of traditional RoF based FiWi networks is China Mobile's cloud RAN (C-RAN), which relies on a centralized cloud infrastructure and baseband units (BBUs) separated from remote radio heads (RRHs), rendering the latter ones intentionally as simple as possible without any processing and storage capabilities [50]. In C-RAN, BBUs that connect a number of macro BSs or small cells (i.e., femto- and picocells) are centralized via pool baseband processing (i.e., BBU pool), while radio frequency (RF) signaling is digitized and transmitted over optical fiber for fronthauling (i.e., between RRHs and BBUs). Further, the digitized RF signal received by the RRH is converted into an analog signal before being transmitted to its associated edge devices in downstream transmissions.

An example of R&F based FiWi networks is the cloudlet enhanced distributed RAN (D-RAN) [51]. In the cloudlet enhanced D-RAN, the functionalities of RRHs and BBUs are split, whereby RRHs and BBUs are linked via an Ethernet interface and the baseband processing is done at a cloudlet server [52]. R&F based FiWi networks may become the choice of emerging Tactile Internet networks, benefiting from decentralization based on cloudlets and intelligent base stations. As shown in Figure 1.3(b), note that both the C-RAN and cloudlet enhanced D-RAN may coexist in FiWi enhanced LTE-A HetNets, whereby the collocated ONU-FeNB (integration of ONU with femtocell base station) and ONU-PeNB (integration of ONU with picocell base station) may rely on a WDM-based C-RAN, while an ONU-MeNB (integration of ONU with macrocell base station) may rely on a cloudlet enhanced D-RAN. Cloudlet servers are connected to the ONU-MeNB and the scheduling and bandwidth allocation are typically handled by the ONU-MeNB with the support of cloudlet enhanced D-RAN. In the coordination of BBUs, the OLT is fully responsible for scheduling transmissions and allocating bandwidth to each ONU-FeNB and ONU-PeNB in a centralized fashion. To reduce capital expenditures, C-RAN and cloudlet enhanced D-RAN may use different wavelength channels for baseband and RF transmissions.

1.1.4 Human-Agent-Robot Teamwork (HART)

Beside lowering latency and jitter for real-time H2M communications, one of the key aspects of the Tactile Internet is how we can make sure that its potential be unleashed for a race with (rather than against) machines. By building on the areas where machines are strong and humans are weak, Tactile Internet H2M/H2R communication leverages on their *"cooperative" and "collaborative" autonomy* such that humans and robots complement each other [5]. The goal here is to take advantage of collaborative teams of humans working alongside machines to create new roles and opportunities for humans. Companies can achieve significant boosts in performance, when machines and humans work together as allies, not adversaries, to capitalize on each other's complementary strengths. For instance, processing and analyzing copious amounts of data from myriad sources in real time, performing routine tasks, working in dangerous life-critical conditions, and detecting hidden patterns in an image can be easy for machines, whereas dealing with unsatisfied customers can be easy for human workers.

In the future, co-working with machines (e.g., robots) will favor geographical clusters of local production ("inshoring") and require human expertise in the coordination of the human-machine symbiosis for the sake of inventing new jobs humans can hardly imagine and did not even know they wanted done [6]. In fact, Stanford University's recently launched One Hundred Year Study on Artificial Intelligence (AI100) released its inaugural report *"Intelligence and Life in 2030,"* in which an increasing focus on developing systems that are human-aware is expected over the next 10-15 years. Unlike the IoT that relies on its underlying M2M communications without any human involvement, the Tactile Internet involves the inherent *human-in-the-loop (HITL)* nature of haptic interaction and thus allows for a human-machine cooperative design approach towards creating and consuming novel immersive experiences via the Internet [10]. In *"Deep Thinking: Where Machine Intelligence Ends and Human Creativity Begins,"* Garry Kasparov elaborates on the importance of a superior process in human-machine collaboration, showing that *weak human + machine + better process* is superior to *strong human + machine + inferior process*. Thus, a clever process beats superior knowledge and superior technology. His observation received interest by Google and other Silicon Valley companies and shifted the research focus from using artificial intelligence (AI) as an automation tool to an augmentation tool for enhancing human decisions (e.g., IBM's Watson) instead of replacing them with autonomous systems. According to Kasparov, this is not just user experience (UX), but entirely new ways of bringing human-machine coordination into diverse fields (e.g., business and manufacturing processes) and creating the new tools we need in order to do so. Interestingly, this approach is fully in line with the original vision of early Internet pioneers. Back in 1962, Douglas C. Engelbart developed a detailed, though rudimentary, conceptual framework with process hierarchies for augmenting the human intellect by increasing via on-line assistance

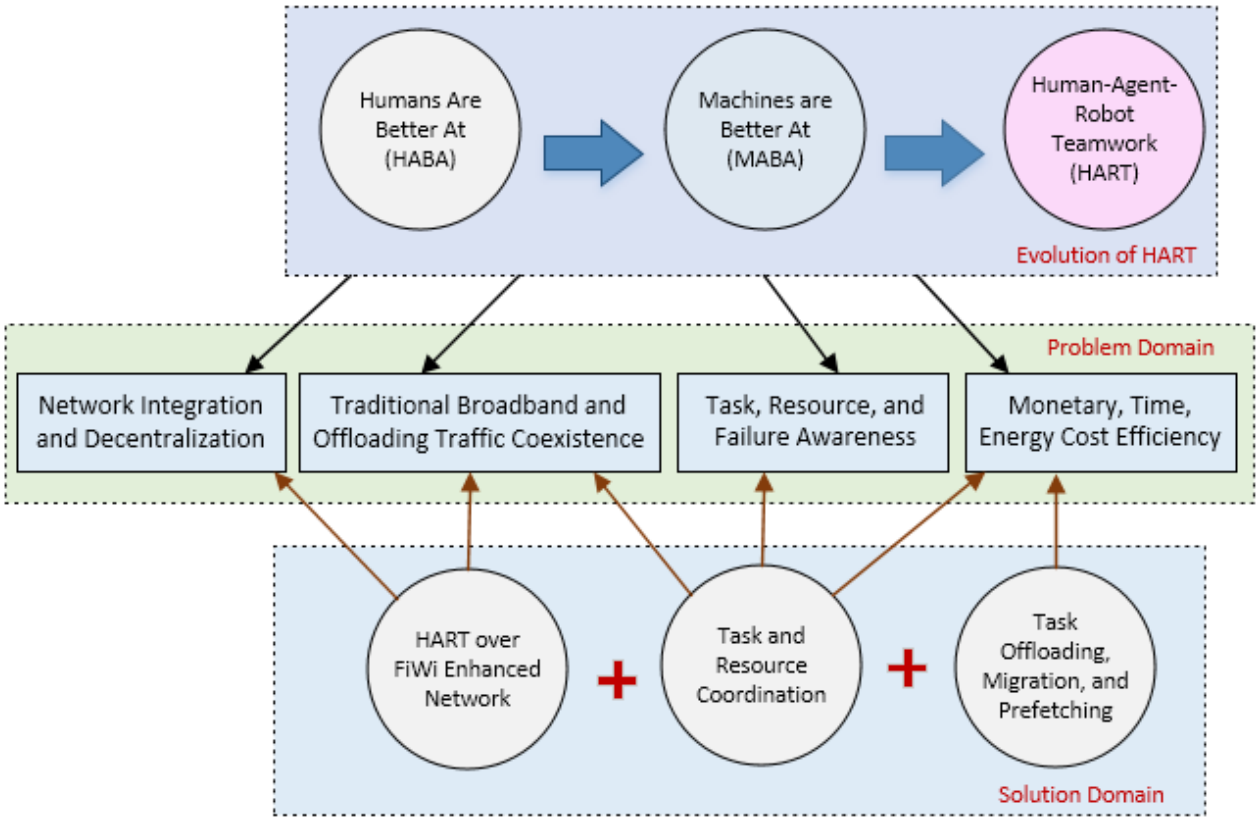


Figure 1.4: Overview of key challenges in HART over the Tactile Internet

the capability of a man to derive solutions to complex problems that before seemed insoluble [53]. Earlier, in 1960, Joseph C. R. Licklider envisioned *man-computer symbiosis*, a subclass of man-machine systems, to enable close interaction between man and computer in mutually beneficial cooperation [54].

A promising approach toward achieving advanced human-machine coordination by means of a superior process for fluidly orchestrating human and machine co-activity may be found in the still young field of *HART* research, whose specific design goal is to keep humans in rather than out of the loop [11]. Historically, HART extends the so-called *humans-are-better-at/machines-are-better-at (HABA/MABA)* approach, which assigns tasks to either humans or machines, whereas HART focuses on how humans and machines could work together (see Fig. 1.4). This collaboration allows weak workers to "punch above their weight" by offloading tedious tasks to powerful workers for processing. Some of the interesting HART-centric applications that harness human-machine interactions are autism therapy, elderly people care, operation on the battlefield, where robots can work alongside doctors as surgical teams, complex search-and-rescue activities in natural disaster/dangerous environments, real-time data-to-decision problems, remote monitoring, disaster warning, gesture, and face recognition, among others [55]. Unlike in early-day HART research, which put emphasis only on making machines

self-sufficient or relying only on autonomous systems, much of the research interest focuses on the autonomy and interdependence of HART members [56]. Such capabilities could enable intelligent systems not merely to do things *for* humans, but also to work together *with* humans and other systems. With regard to underlying human-machine interaction in HART-centric Tactile Internet applications, the main challenge is to orchestrate how tasks can be best executed in concert. Collaboration and communication among HART members are essential to cope with dynamic changes in the task environment, thereby improving the task execution latency. Note that the interdependent activities of HART members may lead to an increased complexity and resource consumption. To facilitate resource efficient HART task execution, research in the area of centralized/decentralized network coordination, adaptive bandwidth resource management for traditional broadband and offloading traffic co-existence, task requirement and failure-aware task assignment, monetary, time, and energy cost-aware task offloading design policies, as well as task and resource coordination become mandatory.

1.1.5 Collaborative Computation and Communication Techniques

For the successful deployment of HART-centric Tactile Internet applications (e.g., human assistive work such as remote monitoring, face detection), efficient task allocation among robots is essential, which has emerged as an interesting research topic by taking into account a wide variety of task and robot types, task location, robot availability, capability, and failure during task execution [6]. Hence, the suitable robot selection may not always be sufficient to satisfy the real-time requirements (e.g., deadline) of different computation intensive tasks due to their limited resources (e.g., computation processing speed, storage) [57]. In response to the aforementioned challenges, mobile devices/robots increasingly seek assistance from powerful collaborative cloud servers via mobile cloud computing[1] and device-to-device communications technology in order to execute their computation-intensive tasks, a technique also known as *collaborative computing* [16].

Importantly, collaborative computing allows resource-constrained mobile devices/robots to migrate full/part of their computation-intensive tasks to collaborative cloud nodes (remote cloud or decentralized cloudlet[2]) for execution by means of *computation offloading* [59]. By enabling collaborative computing among mobile devices and cloud servers, the executable task load on each mobile device/robot is reduced, the overall task processing time is minimized, and the lifetime of the mobile device/robot is extended. However, the interaction between the

[1]Mobile cloud computing (MCC) is a technology that integrates both cloud computing and mobile computing, where cloud services (e.g., computing, storage) are utilized by mobile devices/robots to speed up the running of mobile computation and data-intensive applications.

[2]Cloudlet is a resource-rich computer or cluster of computers that is connected to the Internet and offers cloud services (processing and storage) at the network edge in close proximity to mobile users [30], [58].

cloud server and the mobile device/robot incurs an additional communication latency during offloading the computation-intensive task to the cloud server [17]. Thus, for the cost-effective deployment of HART-centric Tactile Internet applications, collaborative computing needs to tackle several challenges such as minimization of task offloading latency by selecting proper collaborative cloud nodes, selection of non-disruptive network connections for computation task and data offloading activities, among others. To overcome the aforementioned challenges, we discuss the involved collaborative computing and communication techniques in greater detail in the following.

- **Data offloading:** To cope with the network congestion due to the extraordinary growth in data traffic on cellular networks, mobile data offloading represents a potential solution. Mobile data offloading refers to the use of complementary network technologies (e.g., IEEE 802.11 WiFi, femtocell) for delivering data originally destined for cellular networks [60], [61]. Among other alternatives, the most widely used data offloading approach is WiFi offloading due to several benefits such as lower power consumption and reduced operational expenditures (OPEX) than macrocell base stations. Depending on who initiates the offloading process, data offloading can be divided into two groups: a) user-initiated offloading, where the mobile user is responsible for deciding when and how to offload the data; b) network-initiated offloading, where mobile operators are responsible for the data offloading decision [62], [63], [64]. Further, based on whether WiFi technology is used or not, data offloading can be classified into the following two categories: *on-the-spot* offloading, where WiFi technology is used for data offloading only if WiFi connectivity is available, or *delayed* WiFi offloading, where data offloading is delayed for an acceptable time period for future WiFi connections [65]. To ensure seamless data offloading service during handover in both the horizontal and vertical direction, cooperation between cellular and WiFi network service providers is mandatory. Towards this end, different communication standards and protocols were developed to cope with network coordination, frequency management, and traffic rerouting in different data offloading services. For example, the third generation partnership project (3GPP) developed the access network discovery and selection function (ANDSF) [66], local IP access (LIPA) [67], selected IP traffic offload (SIPTO), and IP flow mobility (IFOM) protocols to offer seamless handovers between different access technologies during offloading. For a detailed discussion of ANDSF, IFOM, SIPTO, and LIPA, the interested readers may refer to [68]. Importantly, to enable an efficient mobile data offloading solution for Tactile Internet applications, suitable data offloading techniques need to be developed by taking into account different task, network, and network element prop-

erties such as data offloading deadline, availability of WiFi connectivity, mobile device energy consumption, and amount of data traffic.

- **Task Offloading:** In MCC, task offloading is a technique used to alleviate the burden of resource limited mobile devices by transferring full/part of a task (e.g., computation workload processing) corresponding to given application requirements to more capable devices/surrogates for processing. This capability can take the form of computing power, memory, system load, as well as battery life [69]. Hence, in MCC the decision of whether or not to offload the task to a cloud server depends on several important factors such as why offloading is required (e.g., improve task execution time or save energy), who wants to offload the task (e.g., robot, mobile device, laptop computer), offloading environment (e.g., static, dynamic), different task properties (e.g., workload), and offloading infrastructure (remote/edge cloud computing), among others [70]. For better task execution performance, the appropriate task offloading decision is made by analyzing different important parameters such as available bandwidth, cloud server task processing speed, storage capacity, remaining task processing burden, offload task workload, data size, and communication latencies, among others. Different from data offloading, the computation task offloading life cycle involves the following activities: task input data uploading from mobile device to surrogate/cloud server, offloaded task processing (computation workload) by surrogate/cloud server, and task result downloaded by the mobile device from the surrogate. Hence, in a broad sense, the task offloading process contains the following four parts: application modeling, profiling, optimization, and implementation. For application modeling, a graph or tree-based method can be used to highlight the relation between different sets of tasks corresponding to a given application [71], [72]. Profiling denotes the collection of network device and component information. The collected profiling information (e.g., remaining energy, bandwidth, task processing speed) can be used for making proper task offloading decisions. The optimization part helps achieve different design goals (e.g., minimize task execution delay, maximize energy consumption) for given application and system settings [73]. Finally, the implementation part assigns resources for task offloading and monitors the task execution process.

  At present, existing task offloading frameworks can be divided into three categories: (*i*) system-level, (*ii*) method-level, and (*iii*) optimization level offloading. System-level offloading mainly focuses on the usage of either infrastructure-based cloud resources for offloading (e.g., remote cloud [74], [75], cloudlet [76], [77]) or infrastructure-less cloud resources for task offloading (e.g., mobile ad-hoc cloud [78]). Conversely, method-level offloading emphasizes task/code partitioning, migration techniques, and prediction of

application behavior for parameter variations. The optimization part focuses on achieving different objectives such as maximizing throughput, minimizing energy consumption [71], task execution time [73], or makespan for data streaming applications [79]. However, note that task offloading to remote cloud servers may not always satisfy given task execution requirements of many delay-sensitive tasks (e.g., face detection, gesture recognition, real-time video analytics) due to low bandwidth and high wide-area network (WAN) latencies. To overcome the limitations of distant cloud offloading, there is a growing interest among industries in setting up cloud services (e.g., cloudlet) at the edge of mobile networks, e.g., Nokia's Radio Applications Cloud Server (RACS) that is connected to a 4G LTE base station (eNB) [80], ETSI's MEC server [81], and Cisco's IOx that combines IoT applications with cloudlet servers [82]. Hence, research in the area of developing suitable task offloading platforms for real-time HART-centric Tactile Internet applications is an important open challenge in the existing literature, including several issues such as user-preference aware offloading (e.g., delay and energy awareness), reducing task offloading latency, priority based bandwidth assignment policy, mobility awareness as well as resource and failure awareness, among others.

- **Task Migration:** Contemporary mobile devices offload large amounts of computationally intensive tasks to resource-rich cloud servers for processing. MCC enables mobile devices by offloading tedious tasks to powerful cloud servers [28]. Hence, one of the fundamental challenges for task offloading is to minimize the task processing and communication delay of an MU's offloaded task, whose location changes due to mobility. Further, ensuring high quality-of-service for an MU's task execution is particularly challenging in the dynamic mobile cloud computing environment due to time-varying bandwidth resource availability, dynamic resource availabilities of cloud servers, and time-varying task requests at cloud servers, among others [83].

  Task migration broadens the scope of conventional computation task offloading by not only transferring the task from an MU to cloud servers/surrogates but also from one cloud server/surrogate to another one for execution. In general, task migration between cloud servers/surrogates is considered beneficial only if the anticipated task execution time at the secondary cloud server/surrogate is smaller than that at the primary one [20]. However, task migration incurs an additional migration delay that requires time for: (*i*) stopping task execution at the old server (primarily selected), (*ii*) transferring the remaining task data to the newly selected server (secondary), and (*iii*) starting execution at the newly selected server. Note that task migration provides a more fine-tuned means of balancing the load throughout the system, since migration may take place at any time during the lifetime of a task due to cloud server availability or failure. Hence, there

exists an inherent trade-off in selecting the optimal strategy for task migration. On the one hand, it has to offer high quality-of-experience (QoE) to its customers, which for a particular task indicates lower task execution time. However, due to the residual processing burden of cloud servers and waiting delay for bandwidth availability, task migration services may not always improve the task execution time of an MU's requested task. On the other hand, cloud providers aim to fully exploit task consolidation in order to reduce their operating costs (e.g., electricity cost by turning-off underutilized servers) [19]. Thus, based on the above observations, one of the fundamental research questions that naturally arises is when and where should an MU's task migrate [21]. To answer this question, several aspects need to be investigated for the development of a suitable task migration scheme, e.g., information about the state of the current host and the tentative destination server, number of tasks running on each server, user mobility, task properties, cloud server properties, task migration gain, and latency overhead, among others.

- **Task Prefetching:** Currently, a vast majority of existing cloud computing studies apply the conventional fetching technique for task offloading, where a given mobile user's next computation task input data can be transferred to the cloud server for processing only after the completion of the previously offloaded task [84]. As a result, for multi-task offloading, all other remaining tasks except the first one suffers from higher task offloading waiting times. Thus, conventional fetching based task offloading may not always be sufficient to meet the very low-latency requirements of different HART-centric tasks [85]. To overcome the shortcomings of conventional fetching, the task prefetching concept has recently been proposed in the context of task offloading, where the full or a portion of the MU's next task input data is transferred to the cloud server during the computation processing period of the previously offloaded task [86]. Note that a suitable task prefetching technique has the potential to not only reduce the mobile-device energy consumption by avoiding traditional fetching but also to shorten the program runtime by employing intelligent prediction techniques for parallel task data transfer and processing schedule. Hence, due to the lack of proper task prefetching-aware actor (e.g., cloud agent) selection schemes for task offloading, providing high QoS guarantees is still a major challenging issue for different HART tasks considering both strict task requirements (e.g., deadline) and dynamic cloud environments. Moreover, to ensure collision-free and low-latency HART task execution, research in the area of prefetching-aware dynamic bandwidth allocation is mandatory by taking different task properties and availabilities of cloud servers/surrogates resources into account [87].

1.2 Objectives

The objectives of this thesis are as follows:

- The crucial roadblock toward successful deployment of local and non-local Tactile Internet applications is the lack of proper task allocation strategies among robots. Most existing multi-robot task allocation studies focus on only one or a few parameters for robot selection, e.g., a robot's energy or distance to task location. Clearly, real-time H2R communications based Tactile Internet applications demand advanced robot selection schemes, in which additional parameters need to be considered such as heterogeneous robots and task properties (e.g., robot and task location, robots' energy consumption, task workload and deadline). Moreover, the lack of proper robot failure monitoring strategies during task execution and resource allocation strategies might result in an increased task execution delay and energy consumption of robots. A number of research questions such as (*i*) how human task requests arrive at robot network and (*ii*) how robots are aware of all task requests have largely been neglected in previous studies. Thus, the first objective of this work is to design an efficient failure-aware local and non-local H2R task allocation mechanism and a unified resource management scheme that minimizes the task execution time and energy consumption of robots in FiWi based Tactile Internet infrastructures. To reduce task execution latency overhead, the main focus of this study is to investigate and compare the performance of different robot selection strategies.

- Having a suitable robot selection for satisfying mobile users' task execution requests may not be sufficient to avoid task execution failures due to given resource constraints (e.g., task processing capabilities, storage, or remaining energy) of the selected robot. Note that mobile devices/robots may overcome their resource shortage problem by utilizing the resources of collaborative cloud server (agents). This type of task execution is also known as collaborative computing, where a resource-constrained robot transfers its computation-intensive task to another more powerful cloud agent or nearby robot for execution. At present, research in the area of both infrastructure-cloud and infrastructure-less HART-centric collaborative task execution over FiWi infrastructures is missing in the existing literature. Thus, to improve the task execution time and energy consumption efficiency of resource-constrained robots/mobile devices, the second objective of this thesis is to propose a collaborative computing scheme that jointly selects a suitable host robot and collaborative cloud agent node for executing different HART tasks. Another major aim of this part of the thesis is to investigate a unified bandwidth

allocation scheme to handle coexisting conventional broadband and computation task offloading data traffic over FiWi based Tactile Internet infrastructures.

- Collaborative cloud computing services allow resource-limited mobile devices to offload their computation-intensive tasks onto more powerful cloud servers/surrogates for processing. Hence, one of the major challenges for cloud computing is to minimize the task execution latency of mobile users. Further, due to time-varying resources and higher waiting times in a cloud server (agent), the initially selected cloud server may not always satisfy given offload task execution requirements (e.g., deadline). Thus, to meet the offload task execution requirements, an MU's offloaded task needs to be migrated from one cloud server to another for execution. Note that by taking into account cloud server load, task requirements, task migration latencies, and user mobility, one of the fundamental research questions for HART-centric task execution is whether a task migrates along with the MU or not. Thus, by taking user mobility, different task, and collaborative node properties into account, the third aim of this thesis is to propose a context-aware task migration strategy for the collaborative task execution in FiWi based Tactile Internet infrastructures.

- The benefits of task prefetching and community-cluster resource awareness have not been explicitly studied for offloading multiple HART tasks over FiWi enhanced infrastructures. Moreover, the optimal task-to-resource scheduling order and failure-avoidance service selection for both task onloading and offloading based HART task execution are missing in the existing literature. Thus, by taking both task prefetching and fault tolerance capabilities along with community-cluster resource awareness into account, the fourth aim of this thesis is to design a suitable community and latency-aware multi-task scheduling scheme for task on- and offloading based HART task execution.

- The fifth aim of this thesis is to develop a user preference-aware HART task coordination scheme. Note that research in the area of user preference-aware HART task execution is still in its infancy. At present, no existing study deals with the problem of delay-sensitive and delay-tolerant caching and computing HART task execution considering both dedicated and non-dedicated robot/agent resource awareness and preemptive bandwidth allocation. By taking into account different preferences of MUs such as delay and/or monetary cost saving for different delay-sensitive and delay-tolerant HART task execution, the final goal of this thesis is to develop an appropriate HART task coordination strategy and proactive resource allocation scheme.

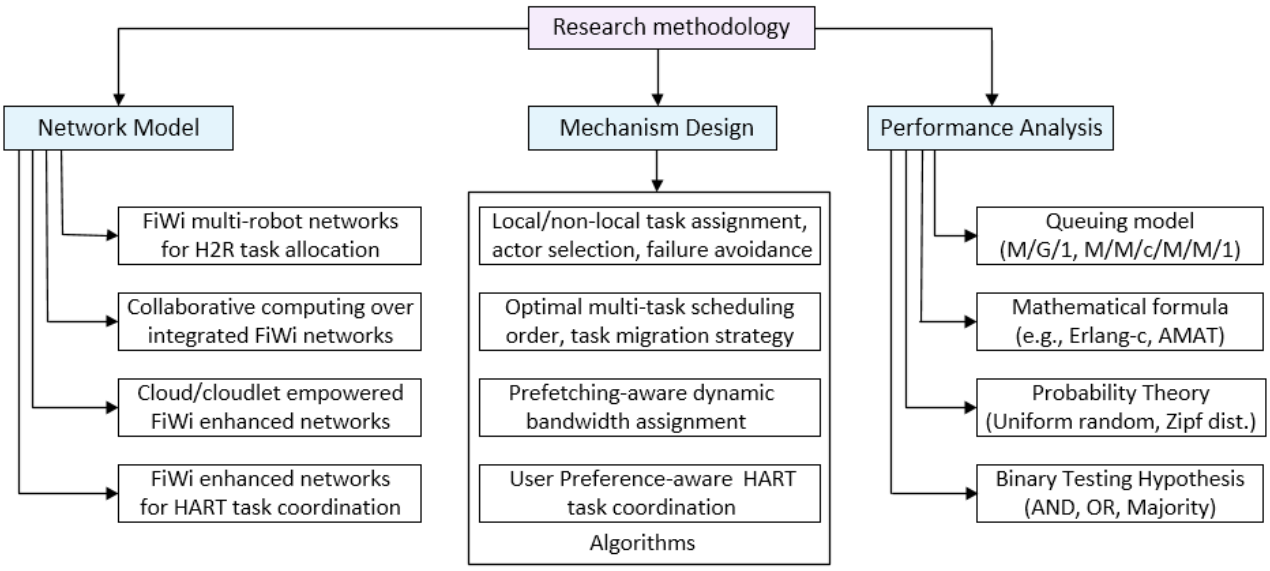


Figure 1.5: Research methodology

1.3 Research Methodology

The research methodology applied in this thesis includes network modeling, coordination mechanism design as well as analytical modeling and performance analysis (see Fig. 1.5) and is described below in greater detail:

- **Network Architecture:** In this thesis, multiple novel network architectures are developed for different HART-centric task coordination schemes. A *top-down approach* is considered, where different task requirements are investigated first and then the network infrastructure is designed to support the service requirements. Importantly, the functionalities of communication networks, task and resource allocation procedures, technologies, and protocols are investigated. Both tree and mesh based topologies are considered in the design of integrated FiWi enhanced Tactile Internet infrastructures.

- **Mechanism Design:** To achieve the optimal performance, different novel algorithms are developed for collaborative HART task execution in FiWi enhanced infrastructures. Most notably, the proposed mechanisms include a unified resource allocation strategy, robot and cloud agent selection for executing different HART tasks, optimal multi-task scheduling order, failure reporting scheme as well as a prefetching-aware dynamic bandwidth allocation (DBA) algorithm.

- **Performance Analysis:** In this work, a performance analysis is conducted based on different queuing models (e.g., M/G/1, M/M/1, M/M/c), analytical tools, mathematical formulas (e.g., Erlang-C, Euclidian distance, average memory access time or AMAT

formula), probability distributions (e.g., uniform random, Zipf), and hypotheses (e.g., binary testing hypotheses). To evaluate the system performance under different scenarios, the analytical and verifying simulation performance is examined for a wide range of performance metrics and varying system settings.

1.4 Contributions of the Thesis

This thesis is compiled based on a total of eight manuscripts ([J1]-[J7] and [B1]), all of them are listed in Section 1.5. The key contributions of this thesis made in [J2-J6] are discussed in the following.

1.4.1 Failure-Aware Local and Non-local H2R Task Allocation in FiWi Multi-Robot Infrastructures

The outcome of this research has been published in the following Journal and the main contributions of this work are summarized below:

[J2] M. Chowdhury and M. Maier, "Local and Nonlocal Human-to-Robot Task Allocation in Fiber-Wireless Multi-Robot Networks," *IEEE Systems Journal,* vol. 12, no. 3, pp. 2250-2260, Sep. 2018.

- **A Novel Local and Non-local H2R Task Allocation Scheme:** To minimize the human users' requested task execution latency, in this work we develop a suitable robot selection mechanism for local and non-local H2R task allocation in FiWi based multi-robot networks. For suitable robot selection, we consider several key performance metrics such as the availability, skill set, distance to task location, and remaining energy of robots. We tackle existing research issues for H2R task assignment such as (*i*) how humans' task requests arrive at the robot network, (*ii*) how to ensure that robots are aware of all arriving task requests, and (*iii*) how to assign bandwidth resources to task requests and result transmission activities.

- **Failure Monitoring Scheme:** To reduce robotic failures during task execution, we introduce a neighboring-robot assisted failure reporting mechanism.

- **Comprehensive Analysis of System Performance:** We develop an analytical model to evaluate the proposed scheme's performance in terms of throughput, task allocation delay, time complexity, sensing error rate, task execution time, and residual energy. In addition, we analyze the end-to-end delay performance for both local and non-local task allocation in integrated FiWi multi-robot networks based on M/G/1 queuing analysis.

1.4.2 Collaborative Computing over FiWi Based Tactile Internet Infrastructures

The outcome of this research has been published in the following journal and the main contributions of this work are summarized below:

[J3] M. Chowdhury and M. Maier, "Collaborative Computing For Advanced Tactile Internet Human-to-Robot (H2R) Communications in Integrated FiWi Multi-Robot Infrastructures," *IEEE Internet of Things Journal,* vol. 4, no. 6, pp. 2142-2158, Dec. 2017.

- **A Novel Collaborative Computing Based Task Assignment and Offloading Scheme:** To improve the energy efficiency of the selected host robot while satisfying given task deadlines, we investigate a proper collaborative task assignment strategy that combines both suitable host robot selection for sensing task execution and collaborative node selection for computation task offloading. We exploit a conventional cloud, decentralized cloudlets, and neighboring robots as collaborative nodes for computation offloading in support of a host robot's requested task execution.

- **Cloud, Cloudlet, and Collaborative Robot Enhanced Integrated FiWi Network Infrastructure:** From an architectural viewpoint, this work introduces an integrated three-level cloud-cloudlet-robot enhanced FiWi network architecture for enabling the execution of Tactile Internet applications.

- **Resource Management Scheme:** In order to handle both conventional broadband traffic and computation offloading traffic at the same time over FiWi network infrastructures, we introduce a unified TDMA-based resource management scheme.

- **Comprehensive Analytical Framework:** We develop an analytical framework to evaluate the performance of our proposed non-collaborative and collaborative/joint task execution schemes in terms of task response time efficiency, energy consumption efficiency, task allocation delay, and task offloading delay.

1.4.3 HART-centric Task Migration Scheme over FiWi Based Tactile Internet Infrastructures

The outcome of this research has been published in the following journal and the key contributions of this work are summarized below:

[J4] M. Chowdhury, E. Steinbach, W. Kellerer, and M. Maier, "Context-Aware Task Migration for HART-Centric Collaboration over FiWi Based Tactile Internet Infrastructures," *IEEE Transactions on Parallel and Distributed Systems,* vol. 29, no. 6, pp. 1231-1246, June 2018.

- **Demonstrate Different Task, Agent, and Robot Properties:** In this work, before describing our task migration scheme, we briefly elaborate on main characteristics of collaborative robots (cobots) in comparison with traditional robots, agents, and shed some light on the different types of task properties (e.g., cognitive and physical tasks). Moreover, the mobility models of both cobots and MUs are described in greater detail.

- **A Novel Context-Aware Task Migration Scheme:** To render HART-centric task migration beneficial for MUs, this part of the thesis analyzes and compares the performance of different task migration schemes: (*i*) cobot-to-cobot (c2c), (*ii*) cobot-to-agent (c2a) migration, (*iii*) inter-agent (remote cloud to cloudlet and vice versa) and (*iv*) intra-agent (remote cloud to remote cloud and cloudlet to cloudlet) task migration. For optimal task migration policy selection, we take into account different context information such as task properties (e.g., task size, deadline), agent (e.g., availability, capability) properties, user mobility, and migration latency, among others. We also develop an appropriate bandwidth resource management scheme for the coordination of the proposed task migration scheme over FiWi based Tactile Internet infrastructures.

- **Analytical Model and Performance Evaluation:** This part of the thesis develops an analytical framework for quantifying the different task migration schemes' performance in terms of a variety of key performance metrics, including task migration gain-overhead ratio, deadline-miss ratio, end-to-end task execution delay, task blocking probability, task response time, and energy consumption efficiency.

1.4.4 Community- and Latency-Aware Multi-Task Scheduling in FiWi Enhanced Networks

The outcome of this work is currently in revision for the following journal and the key contributions of this work are summarized below:

[J5] M. Chowdhury and M. Maier, "Community- and Latency-Aware Multi-Task Scheduling for HART Collaboration in FiWi Enhanced Networks," *IEEE Transactions on Cloud Computing,* November 2018 (submitted).

- **Optimal Multi-Task Scheduling Order Selection:** In this chapter, we investigate a community- and latency-aware multiple HART task scheduling scheme by taking real-time information about arriving task requests, isolated cluster, and community cluster robot/agent resources into account. To reduce the task migration overhead, we incorporate batch based task scheduling into our online task scheduling scheme. In addition, we investigate the performance of both task onloading and task offloading based HART task

execution by taking task prefetching, fault tolerance, and failure avoidance capabilities into account.

- **A Novel Prefetching-Aware DBA Scheme:** In this part of the thesis, we develop a novel prefetching-aware dynamic bandwidth allocation (DBA) scheme for task on- and offloading based HART task execution over FiWi enhanced networks.

- **Analytical Modeling and Performance Evaluation:** The performance of proposed community- and latency-aware multi-task scheduling scheme is evaluated by means of numerical simulations, which is compared with alternative baseline schemes (e.g., communication-aware and random task offloading scheme) in terms of delay and power saving ratio, task prefetching time efficiency, processing-to-service time ratio, speed up, and satisfactory ratio. Moreover, a comprehensive analysis of the mean task service time is presented based on M/M/c queuing model.

1.4.5 User Preference Aware Task Coordination and Resource Allocation in a FiWi Network Infrastructures

This work is currently in revision for the following journal and the key contributions of this work are summarized below:

[J6] M. Chowdhury and M. Maier, "User Preference Aware Task Coordination and Proactive Bandwidth Allocation in a FiWi Based Human-Agent-Robot Teamwork Ecosystem," *IEEE Transactions on Network and Service Management,* Oct. 2018 (in revision).

- **FiWi Enhanced Tactile Internet Infrastructure for Preference Aware HART:** In this work, we investigate an adaptive FiWi enhanced Tactile Internet infrastructure for preference aware HART task coordination by considering the presence of both local and non-local dedicated and non-dedicated robots and cloud agents along with different task arrival numbers. To avoid additional delay and monetary costs while mitigating MUs' different task requests, we propose a user preference aware actor (cloud agent and robot) selection scheme.

- **Proactive Resource Allocation:** A proactive resource allocation model is presented for both delay-sensitive and delay-tolerant caching and computing HART task execution.

- **Analytical Framework and Performance Evaluation:** A comprehensive performance analysis model is developed to asses the performance trade-off between delay cost saving (DCS) and monetary cost saving (MCS) based task execution schemes in terms of average task execution time, monetary and energy cost, time and monetary cost

saving ratio, communication to computation ratio (CCR), and offloading gain overhead ratio. To achieve the cost-effective performance, we compare the following three different DCS and MCS multi-task offloading schemes: (i) maximum throughput and minimum delay (MTMD), (ii) maximum throughput (MT), and (iii) minimum delay (MD) based schemes.

1.5 List of Publications

[J1] M. Maier, M. Chowdhury, B. P. Rimal, and D. Pham Van, "The Tactile Internet: Vision, Recent Progress, and Open Challenges," *IEEE Communications Magazine,* vol. 54, no. 5, pp. 138-145, May 2016.

[J2] M. Chowdhury and M. Maier, "Local and Nonlocal Human-to-Robot Task Allocation in Fiber-Wireless Multi-Robot Networks," *IEEE Systems Journal,* vol. 12, no. 3, pp. 2250-2260, Sep. 2018.

[J3] M. Chowdhury and M. Maier, "Collaborative Computing For Advanced Tactile Internet Human-to-Robot (H2R) Communications in Integrated FiWi Multi-Robot Infrastructures," *IEEE Internet of Things Journal,* vol. 4, no. 6, pp. 2142-2158, Dec. 2017.

[J4] M. Chowdhury, E. Steinbach, W. Kellerer, and M. Maier, "Context-Aware Task Migration for HART-Centric Collaboration over FiWi Based Tactile Internet Infrastructures," *IEEE Transactions on Parallel and Distributed Systems,* vol. 29, no. 6, pp. 1231-1246, June 2018.

[J5] M. Chowdhury and M. Maier, "Community- and Latency-Aware Multi-Task Scheduling for HART Collaboration in FiWi Enhanced Networks," *IEEE Transactions on Cloud Computing,* November 2018 (submitted only).

[J6] M. Chowdhury and M. Maier, "User Preference Aware Task Coordination and Proactive Bandwidth Allocation in a FiWi Based Human-Agent-Robot Teamwork Ecosystem," *IEEE Transactions on Network and Service Management,* Oct. 2018 (in revision).

[J7] M. Maier, A. Ebrahimzadeh, and M. Chowdhury, "The Tactile Internet: Automation or Augmentation of the Human?," *IEEE Access,* vol. 6, no. 1, pp. 41607-41618, Dec. 2018.

[B1] M. Chowdhury and M. Maier, "Human-Agent-Robot Teamwork (HART) over FiWi Based Tactile Internet Infrastructures," *Encyclopedia of Organizational Knowledge, Administration, and Technologies,* Mehdi Khosrow-Pour (Editor), IGI Global (publisher), USA, Aug. 2018 (submitted).

1.6 Thesis Organization

The thesis is organized into seven chapters to provide a consistent overview of the entire research work conducted during the doctoral studies. The rest of the thesis is organized as follows:

Chapter 1 contains the background, motivation, objectives, research methodology, and contributions of the thesis. Moreover, this chapter presents the list of publications and a short outline of the thesis.

Chapter 2 presents a failure-aware local and non-local human-to-robot (H2R) task allocation mechanism in FiWi based multi-robot networks. This chapter presents an analytical framework that models throughput, task execution time, failure sensing error rate, the residual energy of robots, time complexity, and end-to-end delay performance. The obtained results and findings are discussed in great detail.

Chapter 3 explores suitable collaborative computing techniques for executing task requests of humans over integrated FiWi network infrastructures. This chapter investigates the performance of both host robot-based non-collaborative task execution and host robot-collaborative node (e.g., cloudlet) based joint task execution schemes. It also presents a novel resource management scheme for collaborative/joint and non-collaborative task execution over FiWi based Tactile Internet infrastructures. Afterwards, both analytical framework and evaluation results of the proposed schemes are presented.

Chapter 4 presents a context-aware task migration scheme for efficiently orchestrating the real-time collaboration among human users, agents, and collaborative robots (cobots) across converged FiWi communications infrastructures. After describing the key features of physical vs. cognitive task, cobot vs. stand-alone robot, this chapter investigates the problem of whether and, if so, when and where a HART-centric task should be best migrated to. The chapter then evaluates the performance of different task migration scheme, e.g., cobot-to-agent, cobot-to-cobot, intra-agent, and inter-agent task migration schemes in terms of deadline miss ratio, task blocking probability, migration gain-overhead, response time, and energy cost, among others. Finally, simulation results and findings are presented.

Chapter 5 presents a community- and latency-aware task-to-resource mapping scheme for HART task onloading and offloading. This chapter explores both optimal multi-task scheduling order and suitable task processing node selection for different HART tasks. Importantly, this chapter presents a novel prefetching-aware bandwidth allocation scheme and evaluates the performance trade-off between fault tolerance and failure avoidance based HART task execution. It presents an analytical framework to study the performance of our proposed scheme in terms of mean task service time, task prefetching time efficiency, speed up, processing-to-service time ratio (PSR), and satisfactory ratio, among others.

Chapter 6 explores a user preference-aware delay-sensitive and delay-tolerant HART task coordination scheme over FiWi enhanced infrastructures. Specifically, to cope with an MU's delay (DCS) and monetary-cost saving (MCS) preference, this chapter investigates preemptive and non-preemptive resource allocation for different caching and computing HART tasks. It reports on our analytical model and obtained results.

Finally, chapter 7 concludes the dissertation with a brief review of the key findings. This chapter also highlights some future research areas that may build upon our work described in the dissertation.

Chapter 2

Failure-Aware Local and Non-local H2R Task Allocation in FiWi Multi-Robot Infrastructures

2.1 Preamble

This chapter contains material extracted from the following paper:
[J2] M. Chowdhury and M. Maier, "Local and Nonlocal Human-to-Robot Task Allocation in Fiber-Wireless Multi-Robot Networks," *IEEE Systems Journal,* vol. 12, no. 3, pp. 2250-2260, Sep. 2018.

2.2 Introduction

With the advent of highly skilled and remotely controlled robots, we are moving towards a world where human tasks of our everyday life will be increasingly done by robots. To facilitate real-time task execution via remotely controlled robots, the so-called *Tactile Internet* has recently emerged to steer/control elements of our surroundings [4]. The Tactile Internet represents a paradigm shift from traditional wired and mobile Internet based content-delivery to labor-delivery networks via service robots, which will add a new dimension to the human-to-machine interaction by delivering tactile/haptic sensations [35]. To realize the Tactile Internet, recently in [5], we elaborated on the role of several key enabling technologies (e.g., cloudlets, mobile-edge computing, and cloud robotics) and reported on our recent results on the very low latency and ultra-high reliability performance of integrated fiber-wireless (FiWi) communication infrastructures based on data-centric Ethernet technologies. Note that, the ultimate end goal of the Tactile Internet should be the production of new goods and services by means of empowering rather than automating machines that complement humans rather than substitute for them. By taking into account different areas in which humans are more weaker

than machines/robots, one of the major challenge for Tactile Internet is that it should amplify the differences between machines and humans by capitalizing on each other complementary strengths. A promising approach toward achieving advanced human-machine coordination by means of a superior process for fluidly orchestrating human and machine coactivity may be found in the still young field of *human-agent-robot teamwork (HART)* research, whose specific design goal is to keep humans in rather than out of the loop [55].

In recent years, many human-machine togetherness based HART applications have been deployed in industrial plant, natural disaster rescue, and remote surgery operations. Highly reliable and secure communications infrastructures along with intelligent coordination and task allocation strategies need to be developed to minimize the latency and real-time complexity of such applications. To this end, due to their superior latency and reliability performance integrated FiWi networks provide an efficient communication solution to support both human-robot togetherness and quality of service (QoS) for real-time HART applications. The use of FiWi communication infrastructures in HART applications has several benefits. First and foremost, a local loop controller placed at the optical-wireless interface may act as agent, which assigns a given human task to the suitable robot. Second, given the wired/wireless network integration and decentralization principles of future 5G networks, FiWi communication infrastructures provide the necessary support of both local and non-local task allocation requests. Third, the agent at the optical-wireless interface of FiWi networks can dynamically allocate resources to team members and perform task reallocation in the event of failures.

To tackle the challenges in multi-robot task allocation (MRTA), e.g., suitable robot selection, proper coordination among robots, and failure avoidance, various solutions have been proposed [57]. The authors of [88] introduced a role based task allocation mechanism, in which the appropriate robot selection depends on the matching of task type and robots skill. In [14], a decentralized framework was proposed, where tasks of higher priority are allocated before lower-priority tasks. Most of such priority based assignment schemes mainly allocate a task to a predefined robot based on either the matching of the task identification number and robot address or only on the robot's particular task execution capabilities without waiting for the best eligible candidate robot [89], [15]. The authors of [90] used both distance and target qualities for their robot selection. None of these models [88]-[90] considered any task reallocation mechanism nor any failure avoidance procedure. An inter-robot communication-aware task allocation mechanism without any centralized controller was presented in [12]. In [91], a comparative study of market- and distance threshold-based task allocation scheme was investigated. A dynamic task reallocation process was presented in [13], considering robots' distance to the actual task location for robot selection. In [92], a distributed market based algorithm was investigated, whereby all robots are able to announce and bid for a task. Note

that none of these models [12]-[92] consider other important parameters, such as execution time and robot skill, for robot selection. We note that the research on task allocation and coordination in multi-robot networks is still in its infancy. To address the above issues, this chapter proposes an efficient robotic failure aware local and non-local task allocation strategy for integrated FiWi multi-robot networks. The presented analytical results show that our proposed mechanism outperforms traditional distance and priority based robot selection schemes in terms of task execution time and average residual energy.

The remainder of this chapter is structured as follows. In Section 2.3, we first elaborate on the motivation of our work and highlight our major contributions. Section 2.4 describes our proposed FiWi network architecture, resource management scheme, and task allocation mechanism, whose performance is analyzed in Section 2.5. Section 2.6 presents our obtained results and findings. Finally, Section 2.7 concludes the chapter.

2.3 Motivation and Contributions

From the above discussion it is clear that current MRTA solutions typically suffer from several inefficiencies during task execution (e.g., high task completion time and energy wastage of robots) due to their lack of suitable robot selection and control mechanisms. Heterogeneous robot and task types make the task allocation problem even more challenging. To speed up the real-time robotic task execution process and reduce the energy consumption of resource limited robots, the utilization of robotic services for human tasks needs to be done in a more resource-efficient fashion. Most previous studies consider only one parameter for robot selection, e.g., distance and residual energy. There are a number of additional parameters such as robot skill, availability, and task execution time that should be taken into account as well. In the past, a number of important resource allocation and networking aspects of MRTA related to key design questions, including but not limited to (*i*) how human task requests arrive at robot networks, (*ii*) how to recover from robot failures, or (*iii*) how to ensure that robots are aware of all task requests, have been largely neglected in previous studies.

In this work, we develop a FiWi empowered human-to-robot (H2R) task allocation mechanism and make the following novel contributions. First, to efficiently allocate a given human task to a suitable robot, we propose a robot selection algorithm based on distance, residual energy, as well as the robot's ability and availability. Second, we introduce a neighboring robot assisted failure reporting mechanism to avoid task execution failures. Further, to facilitate both local and non-local H2R task allocation at the same time over our proposed FiWi network infrastructure we propose a unified resource allocation scheme and evaluate its performance analytically in terms of delay and throughput. Beside robot selection delay, we analyze the upper end-to-end delay bounds of both local and non-local task allocation. To

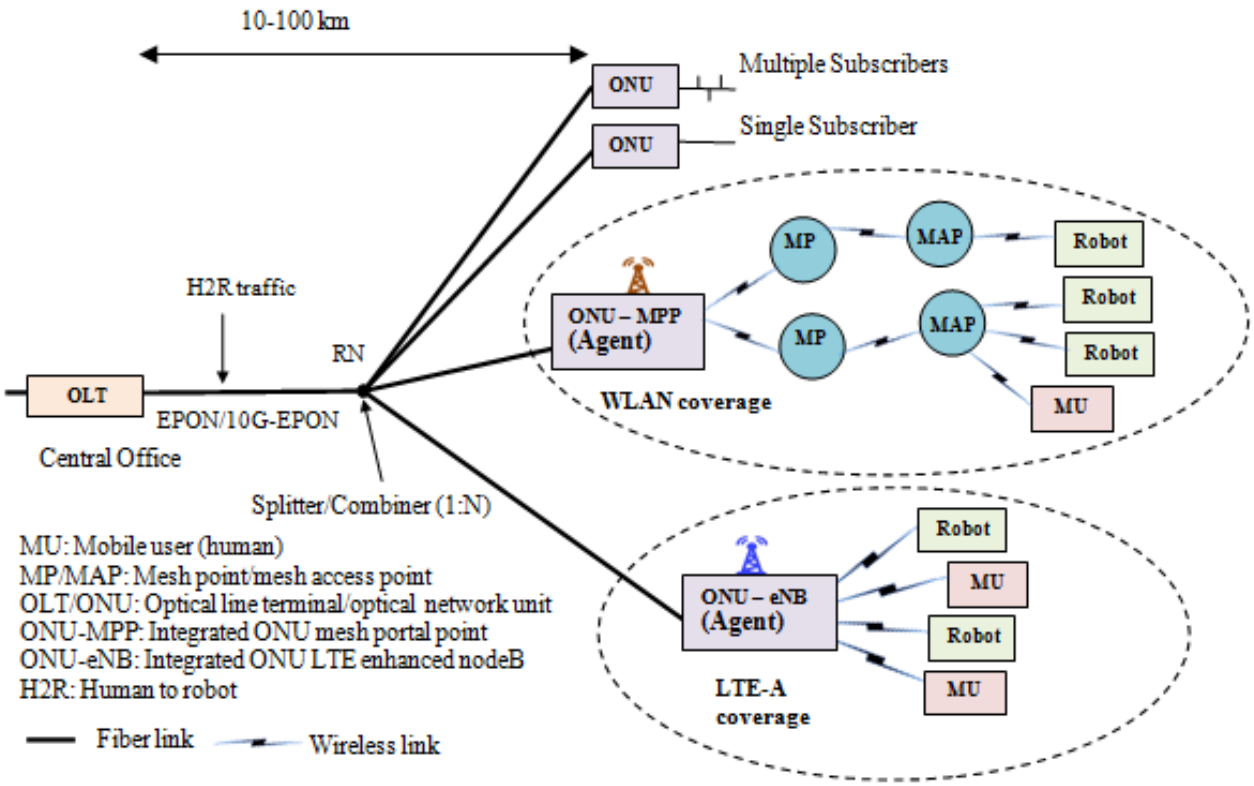


Figure 2.1: FiWi communication infrastructure for H2R task allocation.

the best of the authors' knowledge, no existing study deals with both the local and non-local H2R task allocation in integrated FiWi multi-robots networks.

2.4 FiWi Infrastructure for H2R task allocation

A. Network Architecture: The FiWi based network architecture used for our studied H2R task allocation among humans, robots, and agents is shown in Fig. 2.1. The optical fiber backhaul consists of an IEEE 802.3av 10G-EPON or IEEE 802.3ah 1G-EPON with an optical fiber range of 10-100 km between the central optical line terminal (OLT) and the remote optical network units (ONUs). The OLT connects to the ONUs through a 1:N optical splitter/combiner at the remote node (RN). We consider three different subsets of ONUs. The first subset of ONUs serves a single or multiple fixed (non-mobile) wired subscribers. To interface with the wireless mesh network (WMN), the second subset of ONUs is equipped with a mesh portal point (MPP). The WMN consists of relaying mesh points (MPs) and mesh access points (MAPs). Each MAP serves mobile users (MUs) within its wireless coverage zone, whereby intermediate MPs are used to relay traffic between MPPs and MAPs. The integration of ONU and MPP into one unit is done by using so-called radio-and-fiber (R&F)

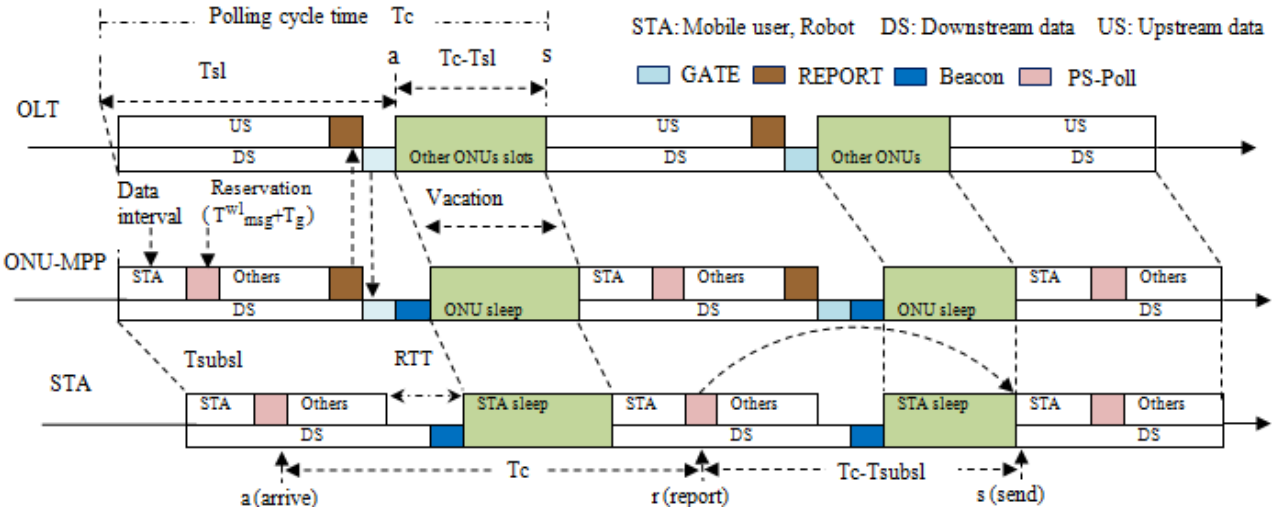


Figure 2.2: Resource management scheme for H2R task allocation in FiWi multi-robot networks.

technologies with protocol translation at the optical-wireless interface [5]. In the wireless front-end, different readily available WLAN technologies are used to meet given design requirements. For instance, IEEE 802.11ac WLAN may be used to achieve physical data rates of up to 6900 Mb/s in conjunction with IEEE 802.11e for QoS support. To provide 4G cellular services to MUs, the third subset of ONUs hosts an LTE enhanced nodeB (eNB), giving rise to an integrated ONU-eNB [93]. In this work, we focus on WLAN coverage only and leave cellular coverage for future work.

B. Resource Management Scheme: Fig. 2.2 depicts the resource management scheme based on TDMA in both the optical fiber (between OLT and ONUs) and wireless subnetworks (between ONU-MPPs and robots/MUs), which is described in greater detail in the following:

- Similar to [93],[94], in the optical part, the OLT dynamically allocates an upstream (US) time slot and sends downstream (DS) frames to each ONU via IEEE 802.3ah multi-point control protocol (MPCP) messages (`REPORT` and `GATE`). The `GATE` message is used by the OLT to inform ONUs about their US transmission window after ONUs have sent their individual bandwidth requests to the OLT via `REPORT` messages. During its assigned time slot, an ONU sends its US data frames to the OLT and receives DS data frames from the OLT, which are subsequently forwarded to its associated users (see also Algorithm 1 below).

- In the wireless part, our resource allocation mechanism differs from [93],[94] in several ways. First, an ONU-MPP allocates bandwidth in sub-slots and schedules transmission opportunities only for MUs and robots associated with the task allocation process by means of `Beacon` and `PS-Poll` frames, as specified in IEEE 802.11e. Second, for control

Algorithm 1 Resource allocation algorithm

Consideration: Total number of ONU-MPP (N) and each ONUs attached STAs (M), polling cycle time T_c, ONUs requested (T_{sl}^{req}) and maximum allocated timeslot ($T_{sl}^{max} = \frac{T_c}{N}$), MPCP message duration (T_{msg}^{pon}), vacation period $V = (T_{msg}^{wl} + T_g)$

Process executed at the OLT:
if Receive `REPORT` message from the ONU-MPP **then**
  Determine the polling cycle time $T_c = N(T_{sl}^{onu} + RTT + 2T_{msg}^{pon})$ and each ONU-MPPs allocated timeslot time $T_{sl}^{onu} = min(T_{sl}^{req}, T_{sl}^{max})$
  Transmit a `GATE` message to all ONU-MPPs indicating their timeslot start time and duration (T_{sl}^{onu})
  Send/receive DS/US data to/from the ONU-MPPs
end if
Process executed at the ONU-MPP:
if Receive `GATE` message from the OLT **then**
  Determine associated STAs subslot duration T_{subsl}^{sta} by using $T_{sl}^{onu} = M(T_{subsl}^{sta} + V)$
  Send a `Beacon` message to all STAs indicating their timeslot duration (T_{subsl}^{sta}) and the ONU-MPPs sleep period ($T_{sleep}^{onu} = T_c - T_{sl}^{onu} - 2T_{msg}^{pon} - RTT$)
  Receive/send US data from/to the STAs/OLT
  Collect their own DS data frame from the OLT and broadcast them to its associated STAs
  if Receive `T-REQ` message from the OLT (non-local task) or STAs (local task) **then**
    Select suitable robot for that task via `RTS-CTS-ACK` message exchange with robots
  end if
  if Receive `PS-Poll` message from the STAs **then**
    The ONU-MPP update their requested timeslot duration ($T_{sl}^{req} = T_{sl}^{req} + T_{subsl}^{req} + V$) and send a `REPORT` message to the OLT indicating next timeslot request
  end if
end if
Process executed at the STA:
if Receive a `Beacon` message from the ONU-MPP **then**
  Extract and update their transmission subslot time and the STA sleep duration ($T_{sleep}^{sta} = T_{sleep}^{onu}$)
  Transmit/receive their US/DS data frames to/from the ONU-MPP
  if Receive a `T-REQ` message from the ONU-MPP **then**
    Participate suitable robot selection process via `RTS-CTS-ACK` message exchange with the ONU-MPP
  end if
  Each STA send their US subslot request (T_{subsl}^{req}) to the ONU-MPP via `PS-Poll` message
end if

signaling during the task allocation and failure reporting process, we assume that all robots associated with a given ONU-MPP use a dedicated common control channel (e.g., separate frequency channel in the unlicensed ISM band). Each user is equipped with two transceivers. One of the transceivers operates on a dedicated control channel, while

the other one works on the data channel. Each user sends her bandwidth request to the ONU-MPP via a `PS-Poll` frame. The broadcast `Beacon` frame is used to inform users about their respective US subslot duration. Subsequently, MUs send their task allocation request frames (similar to the cloud offload/H2H data frames in [94]), which contain the respective task type and task location information, to their associated ONU-MPP during their assigned subslots. An additional flag is piggybacked in the task request frame by MUs to indicate to the ONU-MPP whether the task request is local or non-local.

- For non-local task allocation, the MU and robot associated with that task reside under different ONU-MPPs. After receiving the non-local task request frame from an MU, the ONU-MPP sends it to the OLT during its transmission period. Once the OLT receives the frame, it broadcasts it to all ONU-MPPs. After the intended ONU-MPP receives the frame, it selects a suitable robot for that task by using our proposed task allocation procedure (to be described shortly in Section 2.4.C). For local task allocation, after receiving the task request frame from an MU, the ONU-MPP starts the task allocation procedure immediately without forwarding the frame to the OLT. Network synchronization is achieved by using the timestamp mechanism specified in IEEE 802.3ah, whereby ONUs and their associated users update their local clock to the global clock by receiving DS control messages.

C. Task Allocation Mechanism: The robot selection for each task is performed by a coordinating agent located at the ONU-MPP (see Algorithm 2 and Fig. 2.1) according to the following steps:

- After receiving the task request (`T-REQ`) message, which includes task location, task type, remaining energy threshold, and task instruction, the ONU-MPP sends it to all robots within its coverage. For transmission, the `T-REQ` message represents a modified request-to-send (`RTS`) frame, whose format is shown in Fig. 2.3(b).

- Next, all eligible robots send their task response messages (`T-RES`) to the ONU-MPP, which contains information about the remaining energy, robot location, robot type, availability, moving, and processing speed. The `T-RES` message is encapsulated in a modified clear-to-send (`CTS`) frame, as illustrated in Fig. 2.3(b).

- After receiving the robots' response, the ONU-MPP selects a suitable robot for each task by checking robots residual energy, busy time, and predicted minimum execution time (see Section 2.5.C for cost calculation) and notifies the selected robot by using a winner notification message (`T-WNT`), which includes both task instruction and time slot

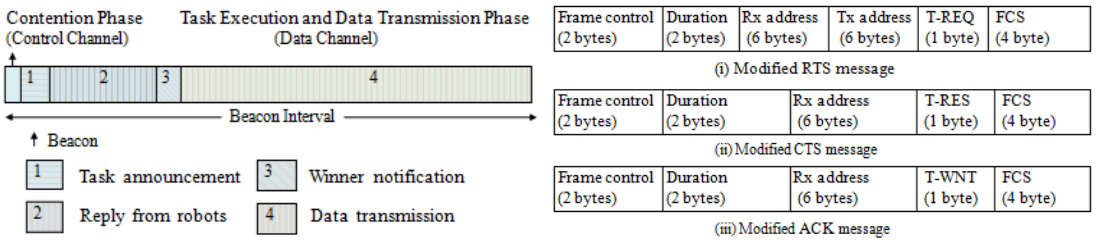
Algorithm 2 Task allocation algorithm

Consideration: Number of task arrives (N_{task}), number of robots under any ONU-MPP (m), task type (t^i_{type}), task load (L_i), service type of robot (r^j_{type}), energy threshold for each task (e_{th}), robots distance to task location ($\bar{d}^j_i$), processing speed (s^j_p), moving speed (s^j_m), robot status before task allocation ($s_{j,busy}$), initial energy (e^j_i), and selected robot status during task execution ($s^i_{j,busy}$)

1: **for** $i = 1$ to N_{task} **do**
2: **for** $j = 1$ to m **do**
3: The ONU-MPP will check the availability and service type of all participating robots via exchange of `T-REQ` and `T-RES` messages
4: **if** ($s_{j,busy}$==$false$) && (t^i_{type}==r^j_{type}) **then**
5: Check residual energy (e^j_r) of participating robots
6: **if** ($e^j_r \geq e_{th}$) **then**
7: Check task execution time $t^j_{ex,i}$ of (see eq. 9) all robots j
8: Allocate task i to robot j with minimum $t^j_{ex,i}$ (MET approach)
9: All neighbor robots monitor the selected robot status ($s^i_{j,busy}$) by using carrier sensing process and inform the ONU-MPP about robots status
10: **end if**
11: **end if**
12: **if** ($s^i_{j,busy}$==$false$) during task execution phase **then**
13: ONU-MPP will re-announce the task i
14: Repeat step 1 to 11 to select robot for task i
15: **else**
16: No failure occurred during task execution
17: **end if**
18: Each robot (j) calculates its own e^j_r by using eq. (8)
19: **end for**
20: **end for**

information. The `T-WNT` message is sent in a modified acknowledgment (`ACK`) frame, as shown in Fig. 2.3(b).

- Each robot is able to monitor the status of its neighboring robots via periodic carrier sensing. In the event that a robot fails, the neighboring robot informs the ONU-MPP about the failing robot such that the ONU-MPP can select another robot for the task. The failure reporting process consists of the following three steps:
  - *Step 1:* Each robot performs local carrier sensing using an energy detection process (see Section 2.5.F) with a 1-bit local decision (D_j) whether the neighboring robot operates or not.
  - *Step 2:* Each robot sends the 1-bit local decision to its associated ONU-MPP.



(a) Timing structure model (b) Modified control frame

Figure 2.3: Proposed task allocation mechanism: (a) timing structure and (b) modified control frames.

– *Step 3:* After receiving the local decisions from the robots, the ONU-MPP makes a final decision (D_f) whether the reported robot is fault-free or not.

D. Time Complexity of Task Allocation Algorithm: In this subsection, we analyze the time complexity of our proposed task allocation algorithm. As shown above in Algorithm 2, at the beginning of each iteration, the agent node (located at the ONU-MPP) considers the number of task requests (N_{task} or $\hat{n}$) arrived at any given time. Therefore, the initial for-loop of our algorithm runs $O(\hat{n})$ times. Next, for each task, the agent node checks the information consisting of availability, service type, residual energy, and task execution time of each robot (m). Thus, the second for-loop of our algorithm requires $O(m)$ time to select a suitable robot for each given task ($\hat{n}$). The information checking and robot selection part (steps 3 to 11) of our algorithm takes constant $O(1)$ time for each given robot. Similarly, the decision on the working robot status (whether fault-free or not) also requires $O(1)$ time for each selected robot. Each robot's residual energy computation is done by the corresponding robot itself requiring $O(1)$ time. It does not affect the agent node's own task allocation operation. Hence, for a total number of tasks ($\hat{n}$) and participating robots (m), our task allocation algorithm takes $O(\hat{n}m)$ time. For a given $\hat{n}$ and m, the minimum distance and priority based robot selection (step 8) requires $O(\hat{n}m)$ time. As the considered model is distributed, any robot can announce the task. The task auctioneer robot has the information about $\hat{n}$ tasks and requires the information about m robots in that network in order to select a suitable robot, resulting in an average computation complexity of $O(\hat{n}m)$. Therefore, our proposed task allocation mechanism has the same time complexity as the minimum distance and priority based selection.

2.5 Analytical Model

In this section, we develop an analytical model of the MRTA problem for the allocation of different tasks to suitable robots, whereby each robot is able to execute only one task at any

given time. We make the following assumptions:

- *Task and robot types*: We assume that an H2R application is composed of a set of sensing and computing tasks N_{task}, similar to [94],[95]. The task types are independent (e.g., face detection, environment monitoring) and robot types are heterogeneous due to their varying task processing skills. The task execution consists of task load processing and transmission of processed data, whereby task i has its own task location (x_i, y_i) and task load (L_i) that depends on the number of instructions needed to be executed. All users under a given ONU-MPP are static in nature, while robots are able to move and change their position towards the task location with very low mobility, depending on their respective moving capabilities (e.g., pedestrian speed). Robot j is located at (x_j, y_j) in a simple 2D space, whereby its energy (e_t^j), task processing speed (s_p^j), and moving speed (s_m^j) are assigned a scalar value (default values are listed in Table 2.1 below).

- *Robot sensing and coverage*: We assume that m robots are randomly distributed in an area a with radius r_{max} $(a = \pi r_{max}^2)$ according to a homogeneous Poisson point process. Hence, the average number of robots per area unit is given by $\lambda_m = \frac{m}{a}$ and the probability that a robot has at least k neighbors equals $\frac{(e^{-\lambda_m a})(\lambda_m a)^k}{k!}$, where $k = 1, 2, 3, ..m$.

The cost calculation and our proposed task allocation algorithm (see Algorithm 2) are described in greater detail next.

A. Distance Calculation Model: The distance between a given pair of task i and robot j $(\bar{d}_i^j)$ is obtained as follows:

$$\bar{d}_i^j = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}, \tag{2.1}$$

where (x_i, y_i) and (x_j, y_j) represent the task and robot location, respectively.

B. Energy Consumption Model: The energy consumption model in multi-robot networks consists of three parts. The first part is the energy required to process the task $(e_p(i, j))$, which is given by [96]:

$$e_p(i, j) = p_{avg}^c \cdot \frac{L_i}{s_p^j} = p_{avg}^c \cdot t_p, \tag{2.2}$$

where p_{avg}^c, L_i, s_p^j, and t_p denote the average energy consumption during processing (per second), computation load for task i, processing speed of robot j, and task processing time of robot, respectively. The second part is the energy required to reach the task location $(e_r(i, j))$:

$$e_r(i, j) = p_{avg}^s \cdot \frac{\bar{d}_i^j}{s_m^j} = p_{avg}^s \cdot t_r, \tag{2.3}$$

where $\bar{d}_i^j$, s_m^j, p_{avg}^s, and t_r denote the distance between task i and robot location j, moving speed of robot j, average energy consumption at moving speed s (per second), and time to reach the task location, respectively. The third part of energy consumption is associated with the communication overhead ($e_c(i,j)$). This is the energy consumption related to transmitting ($e_{tx,l}(l,d)$) and receiving ($c_{rx}(l)$) a packet (l bit) over a distance (d), which are computed as follows:

$$e_{tx}(l,d) = (\varepsilon_{elec} + \varepsilon_{amp} \cdot d^2) \cdot l, \tag{2.4}$$

$$c_{rx}(l) = \varepsilon_{elec} \cdot l, \tag{2.5}$$

$$e_c(i,j) = e_{tx}(l,d) + c_{rx}(l), \tag{2.6}$$

where ε_{elec} and ε_{amp} are the energy dissipation of radio electronics and transmit amplifier, respectively. Thus, the total energy consumption of robot j (e_t^j) is given by:

$$e_t^j = \sum_{i \epsilon \tilde{n}} e_c(i,j) + \sum_{i \epsilon \tilde{n}} e_p(i,j) + \sum_{i \epsilon \tilde{n}} e_r(i,j), \tag{2.7}$$

where $\tilde{n}$ denotes the number of tasks processed by robot j. By using e_t^j and the robot's initial energy e_i^j, the residual energy of the robot (e_r^j) is determined as follows:

$$e_r^j = e_i^j - e_t^j. \tag{2.8}$$

Note that the average residual energy (e_a^j) of robot j is equal to $e_a^j = \frac{\sum_{j \epsilon m} e_r^j}{m}$. If $\hat{e}_o$ and $\hat{e}_c$ are expressed as the average residual energy (e_a^j) of our proposed method (minimum execution time based selection) and the other compared approach (minimum distance/priority based selection), respectively, then the average residual energy efficiency ratio ($\hat{\eta}^o$) of our proposed method over other compared method can be expressed as follows: $\hat{\eta}^o = \frac{\hat{e}_o - \hat{e}_c}{\hat{e}_c}$, where subscript c can be either minimum distance or priority based selection, respectively. Details on both approach are described in performance comparison sub-section (see Section 2.6).

C. Task Execution Time: The task execution time for robot j comprises the time to reach the task location t_r and the task processing time t_p. Thus, by using Eqs. (2.2) and (2.3), the task execution time ($t_{ex,i}^j$) is calculated as follows:

$$t_{ex,i}^j = t_r + t_p = \frac{\bar{d}_i^j}{s_m^j} + \frac{L_i}{s_p^j}. \tag{2.9}$$

For a given number of tasks (N_{task}), the total task execution time (t_i^j) is equal to $t_i^j = \sum_{i \epsilon N_{task}} t_{ex,i}^j$. If $\hat{t}_o$ and $\hat{t}_c$ are defined as the execution time (t_i^j) of our proposed method (minimum execution time based selection) and the other compared approach (minimum distance/priority based selection), respectively, then the task execution time efficiency ratio ($\hat{\gamma}^o$) of our proposed method over other compared method can be expressed as follows: $\hat{\gamma}^o = \frac{\hat{t}_c - \hat{t}_o}{\hat{t}_c}$,

where subscript c indicates either the minimum distance or priority based selection approach, respectively.

D. Throughput vs. Task Allocation Delay: Saturation throughput is one of the important performance metrics to evaluate network performance. It can be defined as the fraction of slot time that is utilized for data transmission by robots for their assigned task. When the number of suitable robots is sufficiently large, the number of successfully executed tasks is equal to the number of task arrivals in the system. In our analysis, we assume that the number of the available channels (slots) is equal to the number of tasks successfully assigned to suitable robots. Note that the average number of task executions during a certain time period depends on the number of suitable robots available at that time. Robot j may be either free ($s_{j,busy} = false$) due to the lack of task assignments (probability $P_{j,busy} = 0$) or busy ($s_{j,busy} = true$), if one task is already assigned to that robot (probability $P_{j,busy} = 1$). For proper robot selection, the task type needs to match the corresponding robot's capability ($Pr^j_{type} = 1$). Moreover, if any robots have sufficient remaining energy (e^j_r) greater or equal to the minimum energy threshold value (e_{th}) then that robot is eligible ($Pe^j_r = 1$) for selection, otherwise it's ineligibile ($Pe^j_r = 0$). With $P_{j,busy}$, Pe^j_r, and Pr^j_{type} given, the probability that a suitable robot is available ($P^s_{j,free}$) is approximated as follows:

$$P^s_{j,free} = (1 - P_{j,busy}) \cdot Pe^j_r \cdot Pr^j_{type}. \tag{2.10}$$

For comparing throughput vs. task allocation/contention delay for robots, note that the WLAN beacon interval is divided into two phases: contention and task execution, as shown in Fig. 2.3(a). The task allocation delay during the contention phase includes task request, reply from robots, and winning robot notification. Let t_a, t_r, and t_w denote the duration of the modified request `RTS`, reply `CTS`, and winner notification `ACK` frame, respectively. Thus, the average required time for robot selection (t_{si}) is given by:

$$t_{si} = \frac{N_{task}t_a + mt_r + N_{task}t_w}{N_{task}}, \tag{2.11}$$

where N_{task} and m are the total number of tasks and robots that participate in the contention phase, respectively. Hence, the duration of the task allocation (t_{alloc}) during the contention phase is computed as follows:

$$t_{alloc} = t_{si}(N_{task} \cdot P^s_{j,free}). \tag{2.12}$$

Similarly, the total number of successful robot selections (n_s) during the contention phase is obtained as $n_s = \frac{t_{alloc}}{t_{si}}$. If the total number of tasks to be completed (N_{task}) and the probability of suitable robot availability ($p^s_{j,free}$) are known, the average number of successful

task allocations (N_{avg}) during the contention phase is given by:

$$N_{avg} = \sum_{i=1}^{N_{task}} min(i, n_s) \binom{N_{task}}{i = 1} (P^s_{j,free})^i (1 - P^s_{j,free})^{N_{task}-i}. \tag{2.13}$$

The saturation throughput (S_{avg}) for different numbers of allocated tasks (N_{avg}) without execution time overhead is obtained as follows:

$$S_{avg} = \frac{N_{avg}(T - t_{alloc})}{T}, \tag{2.14}$$

where T is the WLAN beacon interval. Considering the extra execution time delay overhead ($t_{overhead}$), the saturation throughput is measured as follows: $S_{avg} = \frac{N_{avg}(T - t_{alloc} - t_{overhead})}{T}$. Note that for the calculation of S_{avg} we assume that periodic messages are small and therefore their affect is negligible.

E. FiWi End-to-end Delay Analysis: In this section, we extend the maximum US (ONU-MPP to OLT) and DS (OLT to ONU-MPP) frame transmission delay model proposed in [93] in order to calculate both local and non-local H2R task allocation delay. As illustrated in Fig. 2.2, an ONU-MPP schedules transmission opportunities for its associated users based on an M/G/1 queuing model with reservations and vacations. We define the aggregated traffic load as $\rho^{h2r}=\overline{\lambda X}$, where $\overline{\lambda}$ is the traffic arrival rate and the random packet service time is denoted by its first moment ($\overline{X}$). During a cycle time (T_c), the ONU-MPP timeline consists of data, reservation, and vacation intervals. The H2R traffic time period during transmission is $(1 - \rho^{h2r})$, which equals $N(MV + RTT)$. Thus, T_c is equal to $T_c = \frac{N(MV+RTT)}{1-\rho^{h2r}}$, where M is the total number of users (e.g., humans, robots), reservation duration V is equal to $T^{msg}_{wl}+T_g$, and $RTT = 2T_{prop}$ is the round-trip time between the OLT and ONU-MPP.

If a task allocation request frame arrives after a `PS-Poll` message, it has to wait for a polling cycle time T_c for reporting. If the frame transmitted in the next cycle sub-slot $T_{subsl} \geq T^{msg}_{wl}$ holds, it takes another queuing delay of $T_c - T_{subsl}$. Thus, the total waiting time of the frame is $2T_c - T_{subsl}$. Both the propagation time T_{prop} and US frame transmission time $\overline{X}_u$ are accounted for the total waiting time in our US frame delay calculation. By replacing T_{subsl} with T^{msg}_{wl} and $\overline{X}_u$ with X_{max} for $\overline{X}_u \leq X_{max}$, the maximum US delay of the task request frame (D_u) is given by $D_u = 2T_c - T^{msg}_{wl} + T_{prop} + X_{max}$. Similarly, we calculate the maximum downstream frame delay when an H2R task allocation request frame arrives at the OLT immediately after its transmission of a `GATE` message to the ONU-MPP. Thus, the frame has to wait for $T_c - T_{sl}$. By adding the maximum DS frame transmission time X_{max} for $\overline{X}_d \leq X_{max}$, T_{prop} and frame queuing delay $T_c - T_{sl}$ while considering $T_{sl} \geq RTT + 2T^{msg}_{pon}$, the maximum DS frame delay (D_d) is approximated by $D_d = T_c - 2T^{msg}_{pon} - T_{prop} + X_{max}$.

For a non-local task allocation request, the H2R task request frame is first transmitted to the OLT in the US direction. Next, the OLT broadcasts it to all ONU-MPPs. After

receiving the task request frame, the corresponding ONU-MPP selects a suitable robot for the task. Hence, three delay components are part of the non-local task allocation end-to-end delay ($D_{non-local}$) calculation: US frame transmission delay (D_u), frame DS transmission delay (D_d), and robot selection delay (t_{alloc}). Thus, $D_{non-local}$ is given by:

$$D_{non-local} = D_u + D_d + t_{alloc}. \tag{2.15}$$

Conversely, the local task allocation end-to-end delay (D_{local}) consists of only two components: US frame delay (D_u) required to transmit the frame to the ONU-MPP and robot selection delay (t_{alloc}). Thus, D_{local} is computed as follows:

$$D_{local} = D_u + t_{alloc}. \tag{2.16}$$

F. Cooperative Sensing for Robot Failure Detection: To make a decision on the selected robot status, we use the binary testing hypothesis H_0 (absence of robot transmission in the assigned timeslot) and H_1 (presence of robot transmission) [97]. Hypothesis H_1 states that the neighboring robots observed a signal $x(\bar{n})$ that contains white Gaussian noise $v(\bar{n})$ and the corresponding robot's signal $S_r(\bar{n})$ when the robot operates properly, whereby $\bar{n} = 1, 2, 3..\bar{N}$ and $\bar{N}$ denotes the maximum number of samples. Conversely, hypothesis H_0 states that $x(\bar{n})$ contains only white Gaussian noise $v(\bar{n})$ whenever the robot experiences a failure. The noise is assumed to be additive white Gaussian with zero mean and variance σ_v^2. Hence, the decision statistic t_j for energy detector that is employed by neighboring robot j to detect the working robot's periodic signal energy is given by [98]: $t_j = \frac{1}{\bar{N}} \sum_{\bar{n}=1}^{\bar{N}} |x(\bar{n})|^2$, where λ is a predefined threshold that tests the decision statistic. Subsequently, each neighboring robot makes a local sensing decision (D_j) on the working robot's status such that $D_j = 1$ if $t_j > \lambda$, or 0 otherwise. Based on all local decisions sent to the agent node (ONU-MPP), the final decision (D_f) on robot's status is determined by applying the *n out of k* logic rule:

$$D_f = \sum_{j=1}^{k} D_j \begin{cases} \geq n, & H_1 \\ < n, & H_0, \end{cases} \tag{2.17}$$

where k is the total number of neighboring robots. Further, the false alarm ($\bar{P}_F = P(t_j > \lambda \mid H_0)$) and detection probability ($\bar{P}_D = P(t_j > \lambda \mid H_1)$) of neighboring robot j are given by:

$$\bar{P}_F = Q\left(SNR\sqrt{l_s f_s} + Q^{-1}(\bar{P}_D)\sqrt{1 + 2SNR} \right) \tag{2.18}$$

$$\bar{P}_D = Q\left(\frac{Q^{-1}(\bar{P}_F) - SNR\sqrt{l_s f_s}}{\sqrt{1 + 2SNR}} \right), \tag{2.19}$$

Table 2.1: Parameters and default values for local and non-local H2R task allocation

Parameter	**Value**
Number of robot (m)	1-50
Number of task (N_{task})	1-20
Beacon interval (T)	1-100 ms
Area (a)	9 km^2
Communication radius (r_{max})	56 m
Initial energy of robot (e_i)	2 J
Residual energy threshold (e_{th})	0.5 J
t_a, t_r, t_w	0.17 μs, 0.12 μs, 0.12 μs
$L_i, \bar{d}_i^j, s_p^j, s_m^j$(weight)	0.1 to 1
p_{avg}^c and p_{avg}^s	4 mJ/sec
ε_{elec}	50 nJ/bit
ε_{amp}	10 pJ/bit/m^2

where f_s and l_s denote the sampling frequency and sensing time, respectively. Hence, we have signal-to-noise ratio (SNR) given by $\frac{P_s}{\sigma_v^2}$ and $\bar{N} = l_s f_s$. Then, by using $\bar{P}_F$ and $\bar{P}_D$, the joint probability of false alarm (Q_F) and miss-detection (Q_{MD}) can be calculated as follows:

$$Q_F = \sum_{n}^{k} \binom{k}{n} (\bar{P}_F)^n (1 - \bar{P}_F)^{k-n} \tag{2.20}$$

$$Q_{MD} = 1 - Q_D = 1 - \sum_{n}^{k} \binom{k}{n} (\bar{P}_D)^n (1 - \bar{P}_D)^{k-n}. \tag{2.21}$$

2.6 Results

In the following, we present results on the performance of our proposed task allocation mechanism. Table 2.1 summarizes the key design parameters and their assigned default values in compliance with [95].

System settings, requirements, and configurations: In this work, the ONUs are located at a distance of 20 Km from the central office (OLT). Further, the MAP radius, ONU-MPP

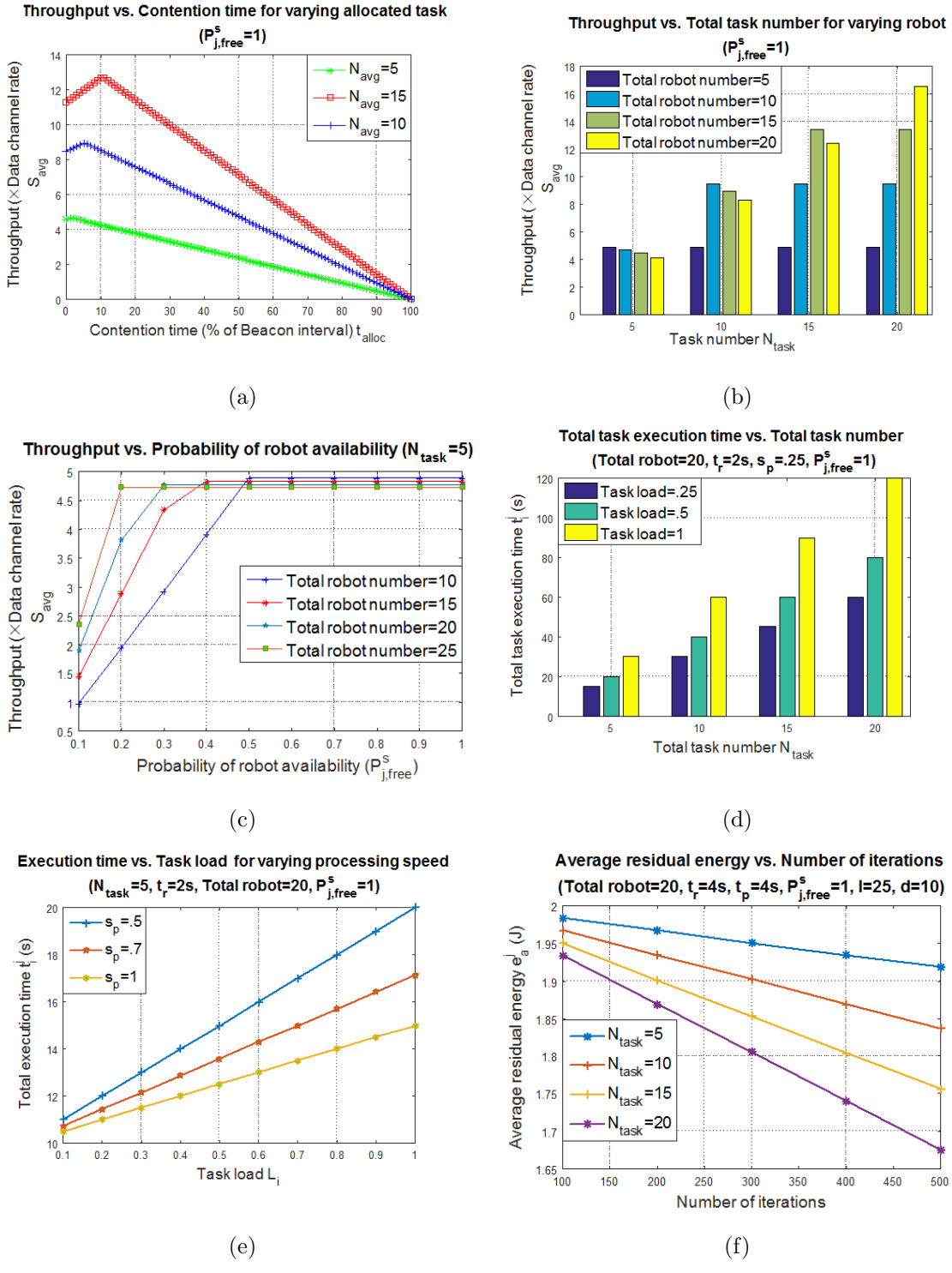


Figure 2.4: Throughput, task execution time, and average residual energy variation.

coverage area, and density of MAPs within each ONU-MPP coverage area is set to 100 m, 9 km^2, and 3, respectively. The FiWi traffic load is varied within the range between 0.05 to 0.95. For evaluation, the task workload (L_i), robot processing (s_p^j), and moving speed (s_m^j) weight is assigned randomly from the range between 0.1 to 1. The task request, response, and winner notification message duration are set to .17μs, .12 μs, and .12μs, respectively. The maximum frame service time X_{max} (at ONUs and OLTs) is assumed to be equal to 12.14 μs. The MPCP messages (REPORT and GATE) duration is set to 64 bytes ($T_{pon}^{msg} = .512\mu s$), whereas the wireless messages (Ps-Poll) duration is assumed to be $T_{wl}^{msg} = .512$ μs. The maximum data rate at the wired (EPON) and wireless (WLAN) link are set to 10 Gb/s and 6900 Mb/s, respectively. Note that, the main requirements of the low-latency H2R application execution (capturing image by using camera at a task location and object detection from captured image) is availability of robots for task execution and availability of bandwidth resources to transfer the task request to robot, hardware/software interface to transfer the task request, connectivity of both mobile users device and robots with networks (WLAN at front-end), and robots failure avoidance, among others.

A. Throughput Performance Analysis: As shown in Fig. 2.4(a), for a given average task allocation number (N_{avg}) and probability of robot availability ($P_{j,free}^s$), the saturation throughput (S_{avg}) increases as the contention time (t_{alloc}) increases until it reaches the peak value given by Eq. (2.14), whereby the contention time is long enough to allocate all tasks to suitable robots. Further increasing the contention time reduces the task processing and data transmission time and thereby negatively affects the throughput performance. Fig. 2.4(b) shows the throughput variation for different numbers of tasks (N_{task}) and robots (m). Provided that sufficient robots are available ($P_{j,free}^s = 1$), the network is able to achieve high throughput only when the number of robots participating in the contention phase is equal to the number of different tasks. This is because higher robot participation results in a long contention time, which in turn leads to a reduced data transmission time. For increasing task arrivals, enough robots are needed to process a large number of task, whereby task load and robot properties (e.g., distance, processing speed, moving speed) are assumed to be independent. We observe from Fig. 2.4(c) that initially a larger number of robots result in an increased throughput for a fixed number of tasks due to the lower probability of robot availability ($P_{j,free}^s$). For sufficiently available robots, a small number of robots achieves higher throughput compared to a large number of robots.

Figs. 2.4(d) and 2.4(e) depict the variation of total task execution time (t_i^j) for different task load (L_i) and processing speed (s_p) values, respectively. From Fig. 2.4(d) we observe that the total task execution time increases for an increasing task number and task load value. Fig. 2.4(e) indicates that a robot with high processing speed (s_p) results in a lower

task execution time. Fig. 2.4(f) shows the average residual energy (e_a^j) for different task and iteration numbers (see Eq. (2.8)). A large number of tasks (per iteration) results in a lower average residual energy for each robot than for a small number of tasks.

B. Performance Comparison: In this subsection, we compare the performance of our proposed scheme with two generalized task allocation schemes: (i) minimum distance based robot selection (MD) [12]-[13] and (ii) priority based robot selection (PS) [14]-[15]. More specifically, their performance is compared in terms of total task execution time, average residual energy, and average throughput. The MD approach selects a suitable robot for each task based on the lowest distance to a given task location. Conversely, the PS scheme allocates a given task to the robot with the lowest ID or by using robot and task ID matching. Our proposed MET task allocation approach selects a robot based on the pre-calculation of the minimum execution time $t_{ex,i}^j$ (see Eq. (2.9)). For fair comparison, we relate both generalized PS and MD mechanism to our analytical model and test their performance based on the same system settings as listed in Table 2.1. Moreover, to evaluate the performance of the MD, PS, and our proposed MET selection schemes, we randomly assign different robot properties, including a robot's moving speed, processing speed, distance associated with each task and robot, as well as task load. Next, we select an eligible robot for each task using the MD, PS, and MET robot selection processes. As shown in Fig. 2.5(a), for a small number of tasks and highly available robots, MET results in a smaller total task execution time than the MD and PS approaches. This is because both MD and PS approaches neglect additional selection criteria other than the minimum distance and the robots' own priority. Note that the execution time difference between the three methods decreases as the number of tasks increases. For the same number of tasks and robots, the execution time difference between the three approaches equals zero. In Fig. 2.5(b), we vary the task load to show its impact on the task execution time. Clearly, the task execution time increases with the task load for all three methods. We observe that the PS approach performs better than the MD selection scheme under high task loads. This is because the robot selected by the MD scheme exhibits a lower task processing speed and moving speed than that robot selected by the PS scheme. Fig. 2.5(b) also indicates that our MET approach outperforms both MD and PS approaches. Fig. 2.5(c) depicts the average residual energy variation of the robots for the three methods under consideration. The average residual energy of the robots decreases as the number of iterations increases. Due to its smaller execution time our proposed MET approach is able to achieve a superior average residual energy performance than the alternative two methods.

Fig. 2.5(d) compares the throughput performance of our proposed MET method with that of the MD and PS approaches for different robot availability probabilities. Clearly, the throughput of all three approaches increases with growing robot availability probability.

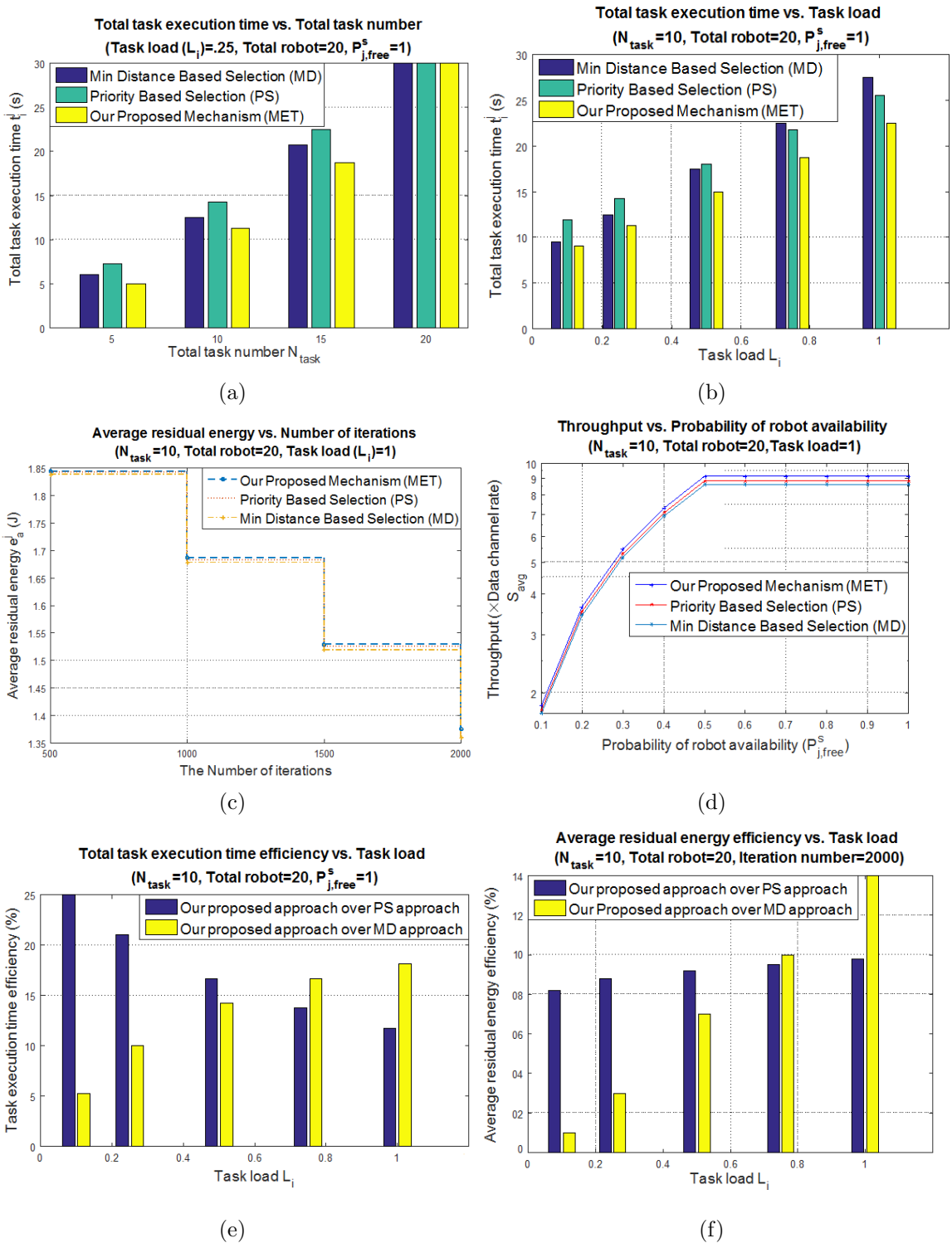
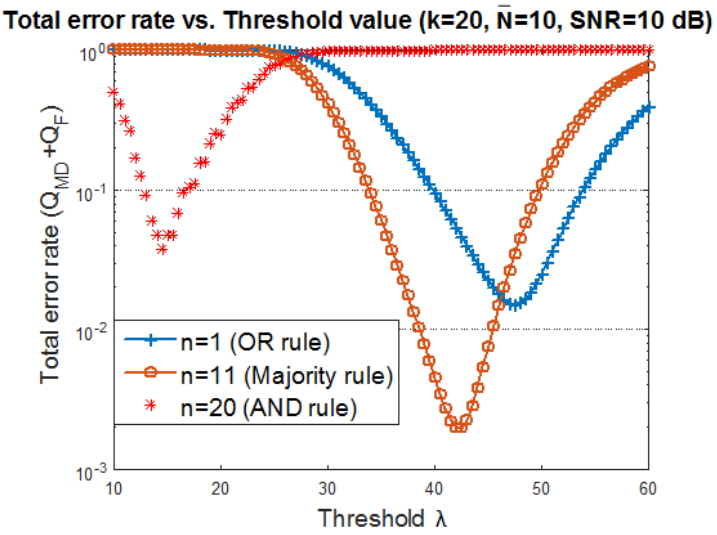


Figure 2.5: Performance comparison of our proposed method with MD and PS based approach.

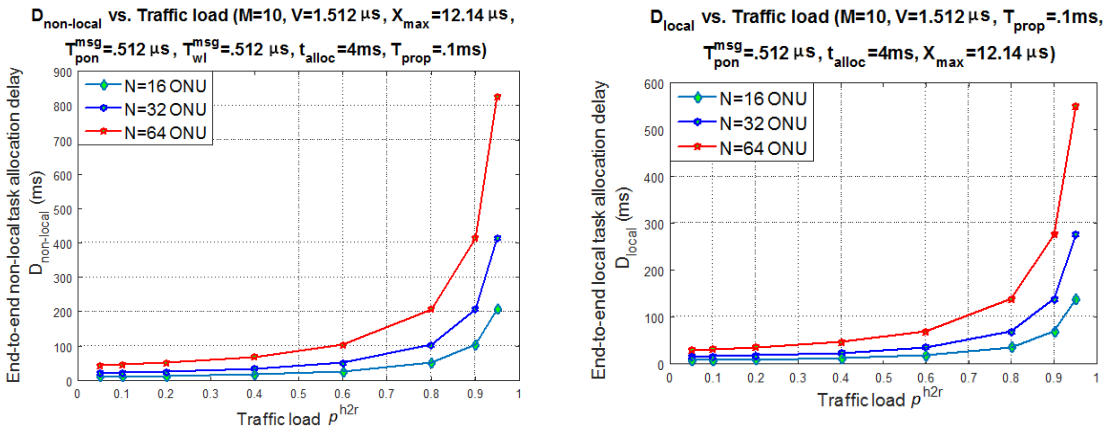


(a)

Figure 2.6: Total failure sensing error rate evaluation.

However, both MD and PS schemes achieve a smaller throughput than our MET approach due to their higher task execution time overhead. Fig. 2.5(e) and (f) illustrate the overall task execution time and average residual energy efficiency of our proposed method in comparison to that of the MD and PS selection schemes. Fig. 2.5(e) shows that for high task loads our proposed method achieves a superior execution time efficiency than the PS approach. For instance, at a task load of 1, the task execution time efficiency of our proposed method outperforms the PS and MD selection scheme by 11% and 18%, respectively. Similarly, we observe from Fig. 2.5(f) that the average residual energy efficiency of our proposed method outperforms both MD and PS approaches for increasing task loads. For smaller task loads, our proposed method shows a higher residual energy efficiency than both PS and MD approaches. Note that under a high task load our proposed method achieves a significantly better residual energy efficiency than the PS and in particular the MD approach.

C. Failure Sensing Error Rate and End-to-End delay Analysis: Fig. 2.6(a) depicts the variation of the robots' total failure sensing error rate for different energy thresholds (λ) (see Eqs. (2.20) and (2.21)). We compare three different fusion rules (n out of k sensing neighbors) to identify their respective effectiveness. We observe that the AND ($n = k$), Majority ($n \geq \frac{k}{2}$), and OR rule ($n = 1$) achieve minimal error rate for a low, medium, and large threshold, respectively. Thus, the AND, Majority, and OR rules are optimal for a low, medium, and large threshold, respectively. Next, the non-local ($D_{non-local}$) and local (D_{local}) H2R task allocation end-to-end delay for different numbers of users (M) and traffic loads (ρ^{h2r}) are shown in Fig. 2.7(a) and Fig. 2.7(b), respectively. Both delays (D_{local} and $D_{non-local}$)



(a) (b)

Figure 2.7: Local and non-local task allocation end-to-end delay evaluation.

increase with the traffic load (ρ^{h2r}) and different number of ONUs (N) in our considered FiWi multi-robot networks (see Eqs. (2.15) and (2.16)). We observe from these figures that $D_{non-local}$ experiences a higher task allocation delay than D_{local}. This is because that beside the robot selection delay (t_{alloc}), $D_{non-local}$ depends on both US (D_u) and DS (D_d) frame transmission delays, while D_{local} depends on the US frame transmission delay (D_u) only.

2.7 Conclusions

In this chapter, we proposed a suitable robot selection mechanism for H2R task allocation in integrated FiWi multi-robot networks. Unlike existing solutions, we investigated both local and non-local task allocation. To reduce robotic failure during task execution, we introduced a neighboring robot monitoring based failure sensing scheme, whereby both human users and robots are synchronized and incorporated into a TDMA-based resource management process. We developed a comprehensive performance analysis model to evaluate system throughput, task allocation delay, task execution time, and robots' average residual energy. Importantly, we showed that there exists a trade-off between task allocation delay and system throughput. Our results indicate that the minimum execution time based robot selection achieves a superior total task execution time and average robot residual energy performance than minimum distance and priority based selection when the number of tasks is smaller than the total number of available robots. Further, we investigated the total sensing error rate of three fusion rules and examined their respective effectiveness during the failure sensing process. By taking frame transmission delay (US and DS) and robot selection delay into account, we also analyzed both local and non-local task allocation end-to-end delay. Our obtained results show that the

proposed H2R task allocation represents an important stepping stone towards ensuring QoS for future HART-centric Tactile Internet applications [4].

Chapter 3

Collaborative Computing over FiWi Based Tactile Internet Infrastructures

3.1 Preamble

This chapter contains material extracted from the following paper:
[J3] M. Chowdhury and M. Maier,"Collaborative Computing For Advanced Tactile Internet Human-to-Robot (H2R) Communications in Integrated FiWi Multi-Robot Infrastructures," *IEEE Internet of Things Journal,* vol. 4, no. 6, pp. 2142-2158, Dec. 2017.

3.2 Introduction

With the emergence of 5G mobile networks and rapid development of smart devices, it is expected that a wide variety of real-time machine-centric applications are finding their way into our life. To unleash their full potential, some of those 5G applications (e.g., cognitive assistance) require low-latency communications with ultra-reliable, ultra-responsive, and intelligent network connectivity. To meet the aforementioned requirements, we recently evaluated the performance gains obtained from unifying coverage-centric LTE-Advanced (LTE-A) heterogeneous networks (HetNets) with capacity-centric fiber-wireless (FiWi) access networks based on data-centric Ethernet passive optical network (PON) and Gigabit-class WLAN technologies [9]. We showed that very low latency on the order of 1 ms and ultra-high reliability with almost guaranteed network connectivity can be achieved in FiWi enhanced LTE-A HetNets. Note that both very low latency and ultra-high reliability are not only crucial for future 5G networks but are also essential design goals of the emerging *Tactile Internet* to remotely steer/control virtual and/or physical objects (e.g., remote-controlled robots) [35].

In order to realize Tactile Internet robotic applications, the feasibility of several key enabling technologies such as FiWi enhanced 4G mobile networks, cloudlets, mobile-edge computing (MEC), and cloud robotics were identified in [5]. Moreover, the subtle differences

between 5G, Internet of Things (IoT), and Tactile Internet were outlined based on their different underlying communications paradigms. IoT and 5G leverage on machine-to-machine (M2M) and human-to-human (H2H) communications, respectively. Conversely, the Tactile Internet relies on human-to-robot (H2R) communications to facilitate the interaction between human operators and tele-operated robots. Note that, in [5], we introduced the concept of FiWi enabled H2R communications for Tactile Internet applications and identified specific H2R communications requirements apart from low latency and high reliability, most notably efficient H2R task allocation. However, we did not elaborate on how H2R communications may be realized in technically sufficient detail.

Taking the respective areas where robots are strong and humans are weak into account, FiWi enabled H2R communications aims at leveraging on their "cooperative" and "collaborative" autonomy such that humans and robots may complement each other. This design approach is also known as *Human-Agent-Robot Teamwork (HART)* [55]. The potential benefits of real-time HART applications are immense, ranging from industrial applications (e.g., coal mining) to emergency response operations (e.g., food supply and human rescue). However, highly reliable and secure communication platforms along with intelligent task allocation and service coordination strategies are needed to meet their stringent quality-of-service (QoS) requirements. To ensure proper coordination in both local and non-local HART task allocation processes, integrated FiWi multi-robot communication infrastructures play a crucial role, whereby humans (H), agents (A), and robots (R) perform the following respective functions. A human user delegates her task request to a robot through a nearby agent. The agent in turn coordinates the task allocation/robot selection process, while the selected robot executes the human's task. Note that typically the agent is placed at the optical-wireless interface of integrated FiWi multi-robot networks.

For the efficient utilization of robotic resources, proper task allocation among robots is crucial by taking the different capabilities of robots and the specific task requirements such as task execution deadline and energy consumption of robots into account [57]. Most existing *multi-robot task allocation (MRTA)* studies focus on only one or a few parameters for robot selection, e.g., the robot's energy [99], distance to task location [12], or task priority [89]. Clearly, in more advanced robot selection schemes, additional parameters need to be considered such as the robot's ability, availability, and task execution time. More importantly, note that the aforementioned MRTA schemes suffer from several shortcomings, most notably, the lack of control and coordination as well as task re-allocation mechanisms, thus resulting in an increased task execution delay and energy consumption of selected robots. Further, some H2R tasks may involve location dependent sensing sub-tasks (e.g., image capturing) and/or location independent computation sub-tasks (e.g., processing of sensed data). For these types

of task, different challenging scenarios may arise, where the available robots may fail to execute either the sensing or computation sub-tasks or both due to their limited computing and storage resources.

To address these shortcomings, mobile devices increasingly seek assistance from collaborative nodes (e.g., mobile-cloud computing, mobile device-to-device communications) for executing their computation tasks, a trend also known as *cyber-foraging* [16] or *collaborative computing* [18], [17]. Despite recent progress on MRTA, the impact of collaborative/joint task execution schemes that consider both the host robot and collaborative nodes (e.g., central cloud or cloudlet) for the H2R task execution process has not been examined in sufficient detail previously. For clarification, note that the H2R task execution process typically consists of two sub-parts. The first one comprises the initial processing or sensing sub-task (e.g., capturing an image), which can be executed only by the selected host robot located in the given task area. The second sub-part of the task involves the location independent computation/processing of the sensed data (e.g., image/face detection), which can be done by the host robot itself or alternatively be offloaded to collaborative nodes. Computation task offloading to collaborative nodes comes in three flavors: System-based, method-based, and optimization-based offloading. System-based offloading is used to decide whether to offload the computation task to an infrastructure-based cloud (e.g., central cloud [74],[75], cloudlet [76], [77]) or an ad-hoc virtual cloud [78], [30]. Method-based offloading [71] involves application partitioning and code migration to an infrastructure-based cloud. Optimization-based task offloading, on the other hand, aims at achieving objectives related to minimizing energy consumption [75] and task response time [77] of mobile devices, though computation task offloading might not always be beneficial for mobile devices. Several studies showed that computation task offloading can help save energy of mobile devices only if they consume less energy during offloading the computation task to a central cloud than executing the task by themselves [74],[75].

Most previous studies considered either full task (i.e., both sensing and computation sub-parts) allocation to a robot or only computation offloading (i.e., sub-part of the full task) onto cloud nodes (central cloud and local cloudlet) for execution (also see Table 3.1 for details). Hence, the question of how to assign a local/non-local H2R task to a host robot considering different tasks and robot types with their respective energy consumption, availability, distance to task location, processing, and moving speed remains an open research challenge. Moreover, how to coordinate both task allocation among robots and computation sub-task offloading onto collaborative nodes represents another unaddressed research challenge. Further, different challenging situations need to be investigated, where both collaborative cloud/cloudlet nodes may satisfy or fail to satisfy given computation offloading requirements.

Table 3.1: Comparison of proposed scheme with existing task allocation schemes

Scheme	**Main Features**	**Robot Selection and Computation Offloading**	**Resource Management**	**Task Allocation and Offloading Coordination**	**Other Comments**
Minimum Distance based robot selection [12]	Used only robots distance value for task allocation, not considered task re-allocation	Used only for robot selection based task execution, not considered task offloading	Not addressed	Not considered	May suffer from higher task execution latency
Fixed task assignment scheme [89]	Used only robot identification number for task allocation	Not considered task offloading	Not addressed	Not considered	May suffer from higher task execution latency
Remote cloud based system [75]	Used min energy consumption of mobile device for cloud offloading	Used only for location independent offloading task	Not addressed	Not considered	Not considered location dependent task
Delay optimal computing [77]	Minimized computation task execution time, considered cloudlet for offloading	Used only for computation offloading task, did not consider any robot selection scheme	Not addressed	Not considered	Not considered location dependent H2R task execution
Hybrid cloud based system [76], [78]	Minimized energy consumption of host mobile device, considered remote cloud and cloudlet for offloading	Only offload computation task to suitable cloud server, did not consider any robot selection scheme	Not addressed	Not considered	Not considered location dependent H2R task execution
Our proposed scheme	Host robot selection for sensing task based on different parameters (e.g., distance, energy, skill)	Considered both suitable host robot selection and computation sub-task offloading	Addressed	Considered	H2R task includes both sensing and computation task
	Considered multiple collaborative nodes for computation offloading	Offload computation task to collaborative node that satisfies resource availability, deadline, and energy consumption criteria			

Towards this end, in this chapter we develop an efficient H2R task allocation strategy that includes both suitable host robot and collaborative node selection in integrated FiWi multi-robot networks. We propose to use not only the central cloud and local cloudlets as collaborative nodes but also available neighboring robots for computation sub-task offloading. To achieve energy savings of the robots and accomplish H2R tasks within their required time, the main objective of this chapter is to select the proper policy for H2R task execution by evaluating the performance of a non-collaborative task execution scheme, in which the selected host robot executes the full H2R task, and the collaborative/joint H2R task execution, in which the selected host robot performs only the sensing sub-task while the selected collaborative node executes the computation sub-task via computation offloading.

The main contributions of this chapter are threefold. First, we present a suitable host robot selection policy for sensing sub-task allocation by taking different parameters such as robot availability, remaining energy, and task execution time into account. Second, to improve the task response time and energy consumption of robots, we propose an efficient collaborative node selection scheme for computation offloading. Third, we investigate a unified resource management scheme that is able to handle coexisting conventional broadband traffic and computation offloaded data traffic.

The remainder of this chapter is structured as follows. Section 3.3 describes our proposed FiWi multi-robot network architecture, unified resource management, and task allocation scheme in technically greater detail. Section 3.4 presents our developed analytical model. Section 3.5 reports on our obtained results and findings. Finally, Section 3.6 concludes the chapter.

3.3 FiWi Multi-Robot Network infrastructure for H2R Task Allocation

3.3.1 Network Architecture

In this section, we re-design the generic FiWi network architecture introduced in [9] for coordinating the local and non-local allocation of H2R task that consist of both sensing and computation sub-parts, whereby humans, robots, and agents actively participate in the task allocation process, as shown in Fig. 3.1. We exploit the central cloud, cloudlets, and neighboring robots as collaborative nodes for computation sub-task offloading. Note that the generic FiWi network architecture proposed in [9] was designed only for H2H communications and did not examine any H2R task allocation and computation offloading capabilities.

In our proposed collaborative computing based FiWi multi-robot network architecture, the optical fiber backhaul consists of an IEEE 802.3av 10 Gb/s Ethernet Passive Optical Network

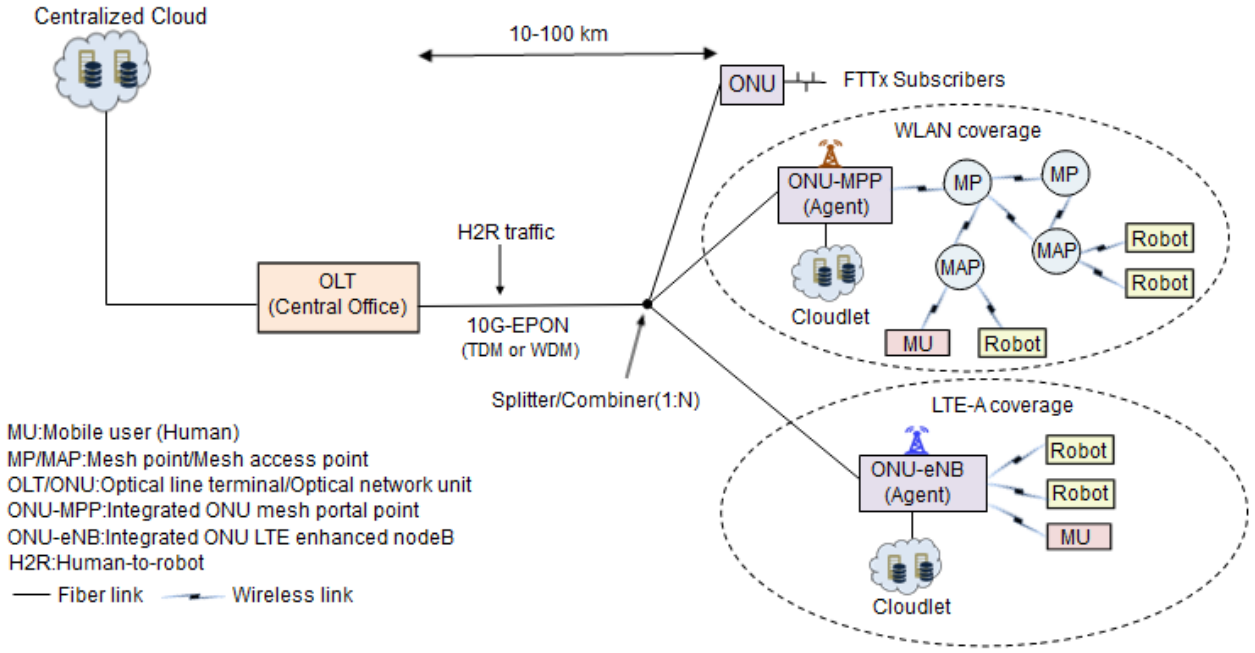


Figure 3.1: Integrated FiWi multi-robot network architecture for coordinating local and non-local H2R task allocation.

(10G-EPON) with an optical fiber range of 10-100 km between the central optical line terminal (OLT) and remote optical network units (ONUs). The OLT is located at the central office and connects to three different subsets of ONUs via a typical tree-and-branch topology. The first subset of ONUs provides services to a single or multiple attached fixed (i.e., non-mobile) wired subscribers via FTTx access, e.g., fiber-to-the-home/business (FTTH/B). To provide an interface with the wireless mesh network (WMN), the second subset of ONUs is attached to an IEEE 802.11s mesh portal point (MPP), referred to as ONU-MPP. The WMN consists of wireless mesh points (MPs) and mesh access points (MAPs), whereby intermediate MPs relay traffic between MPPs and MAPs. Each MAP serves associated mobile users (MUs) and robots within its wireless coverage area. Note that the integrated ONU-MPP is realized by using so-called radio-and-fiber (R&F) technologies with medium access control (MAC) protocol translation taking place at the optical-wireless interface [94]. The third subset of ONUs, each connecting to an LTE enhanced nodeB (eNB) base station, giving rise to ONU-eNB, in order to provide 4G cellular services to MUs [93]. The central cloud servers are connected to the OLT via dedicated fiber links. In addition, to provide mobile-edge computing (MEC) services to MUs and robots at the edge of our integrated FiWi multi-robot network, cloudlet servers are placed at the optical-wireless network edge and connected to separate ONU-MPPs or ONU-eNBs via dedicated fiber links.

In the following, we focus on WLAN coverage only, whereby MUs and robots connect to the ONU-MPPs via IEEE 802.11ac WLAN, and leave cellular coverage for future work.

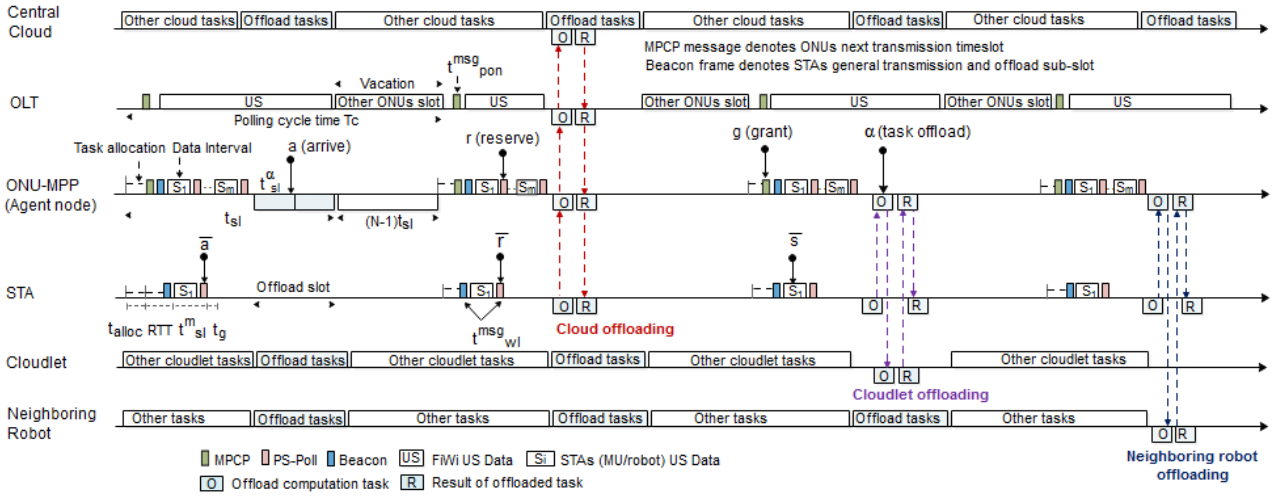


Figure 3.2: Proposed resource management scheme.

3.3.2 Resource Management Scheme

3.3.2.1 A. General Operation

Similar to [94], our proposed resource management scheme uses a two-layer time division multiple access (TDMA) based operation in both optical and wireless sub-networks, as illustrated in Fig. 3.2. However, our proposed resource management scheme differs from [94] in several important ways. First, in the wireless part, we split the overall timing structure into three parts: (*i*) initial robot selection for task allocation, (*ii*) assigned time slot for associated users' conventional broadband traffic, and (*iii*) computation offloaded data transmissions. Second, apart from the central cloud and local cloudlets, we consider available neighboring robots for computation sub-task offloading. Third, our resource management scheme flexibly accommodates H2R task execution both with and without computation offloading onto a collaborative node.

To better understand the basic operation during a polling cycle, Fig. 3.3(a) illustrates the proposed resource allocation and control signal exchange process, which operates as follows:

- During the initial task allocation phase, the agent located at the ONU-MPP exchanges three control messages (`RTS`, `CTS`, and `ACK`) with its associated robots to select one suitable robot for each H2R task (to be described in greater detail in Section 3.3.3).

- The resource management operation in the optical fiber backhaul uses IEEE 802.3av multipoint control protocol (MPCP) messages (`REPORT` and `GATE`), whereby the `REPORT` message is used by ONU-MPPs to report their upstream (US) transmission demands to the OLT and the `GATE` message is used by the OLT to inform ONU-MPPs about their US transmission windows (i.e., start time and duration) for the next polling cycle.

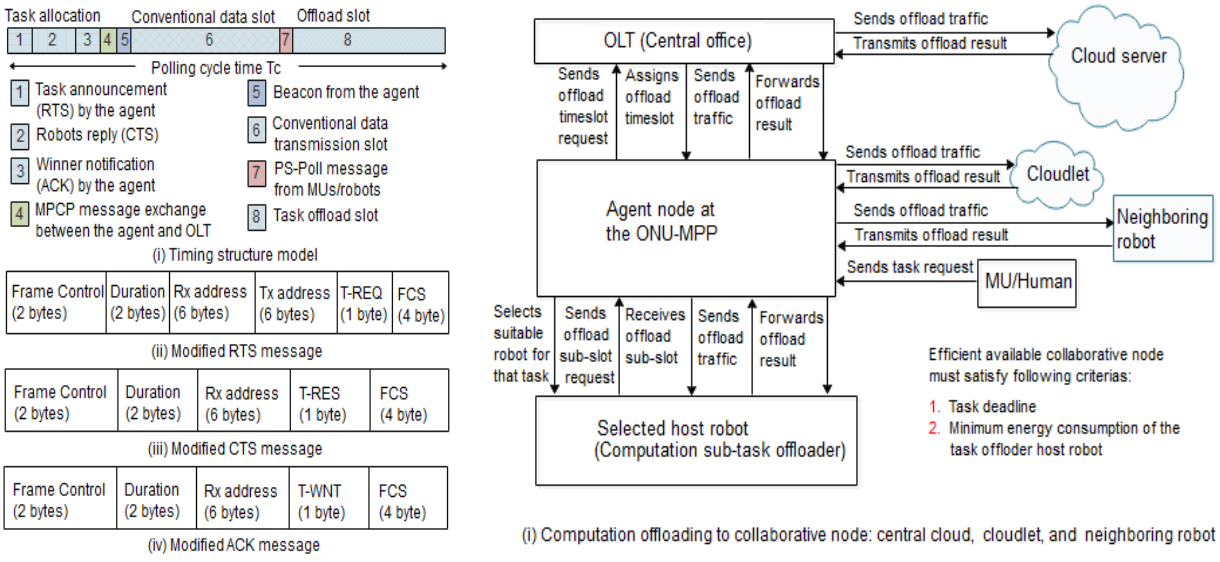


Figure 3.3: (a) Timing structure and control frame format, and (b) operational steps of computation offloading process.

The `REPORT` message contains the ONU-MPPs' US bandwidth requests in terms of buffer backlogs expressed in time units. In this work, the traditional `REPORT` message is extended by using its reserved bits (32 bits) for carrying the computation offloading time slot request information, which contains the time instant up to which a given ONU-MPP can schedule its associated wireless users' computation offloaded data transmissions. The computation offloading bandwidth request information embedded in the `REPORT` message is used by the OLT to assign ONU-MPPs computation offloading transmission time slots in the next polling cycle.

In the wireless part, the resource management operation is realized via the exchange of IEEE 802.11ac WLAN frames (i.e., `Beacon` and `PS-Poll`), whereby associated users (MUs/robots) report their bandwidth requests by sending an extended `PS-Poll` message to their corresponding ONU-MPP. The `PS-Poll` frame is extended by using its pad/reserved bits to include an offload flag bit. The offload flag bit is embedded in the `PS-Poll` frame to inform the ONU-MPP about the MUs/robots' computation offloading time slot requests. After receiving the `GATE` message from the OLT, the ONU-MPP assigns conventional broadband US traffic and computation offloaded data transmission opportunities to its associated users, resets its clock time, and then sends the broadcast `Beacon` frame to inform all associated MUs/robots about their US transmission time slots (i.e., start time and duration).

- Subsequently the ONU-MPP receives the US transmissions from its associated users and forwards them to the OLT. At the same time, the ONU-MPP receives its intended downstream (DS) frames from the OLT and forwards them to its associated users. Note that each MU/robot sends/receives its US/DS data traffic to/from the ONU-MPP during its assigned time slot.

3.3.2.2 B. Computation Offloading Operation

- In this work, the transmission opportunity for computation offloading is kept separate from conventional broadband transmissions to permit both broadband and computation offloading operations within a polling cycle. Fig. 3.3(b) depicts the computation offloading process to a collaborative node, which might be a remote cloud server, decentralized cloudlet, or neighboring robot. Initially, the MU sends her H2R task execution request to the agent located at the ONU-MPP. The agent then selects a suitable host robot for the H2R task that contains both sensing and computation sub-parts via our task allocation algorithm (to be described in greater detail in Section 3.3.3). Next, after completing the sensing sub-task (e.g., image capturing), provided that the selected host robot requires assistance from a collaborative node (central cloud, cloudlet, or neighboring robot) to process the remaining computation sub-task (e.g., image detection), it sends a computation offloading request to the ONU-MPP in an extended `PS-Poll` message.

- After receiving the computation offloading request from a given host robot, the ONU-MPP selects where (i.e., cloud node or neighboring robot) to offload the computation sub-task onto subject to given computation sub-task offloading requirements (see Section 3.3.3) and sends an extended `REPORT` message to the OLT, which embeds the computation offloading request. Once the ONU-MPP receives the `GATE` message from the OLT that contains the ONU-MPPs' conventional broadband and computation offloading time slot map, it immediately schedules the host robots' computation offloading opportunities. The ONU-MPP then notifies all host robots about their computation offloading time slot information via a broadcast `Beacon` message (i.e., computation offloading time slot start time and duration). A given host robot then transmits the computation sub-task data frame (see Section 3.4.1 for details) to the ONU-MPP via its assigned offloading time slot. After receiving the computation sub-task input data frame from the task offloading host robot, the ONU-MPP forwards them to the selected collaborative node (central cloud/cloudlet/neighboring robot) for further processing.

- In case of cloudlet/neighboring robot offloading, the ONU-MPP sends the computation sub-task input data frame to a cloudlet/neighboring robot via the fiber/wireless link for processing. Once the ONU-MPP receives the results of the computation sub-task from the cloudlet/neighboring robot, it immediately sends them to the task offloading host robot. For central cloud offloading, the ONU-MPP sends the computation sub-task input data frame to the OLT. Then, after receiving the computation sub-task input data, the OLT transfers them to the central cloud. The OLT receives the computation sub-task results from the central cloud after processing and sends them to the ONU-MPP. Once the ONU-MPP receives the computation sub-task results from the OLT, it immediately forwards them to the corresponding host robot.

3.3.3 Proposed Task Allocation Algorithm

We consider the following two different task execution schemes: (*i*) a non-collaborative scheme, where the suitable host robot executes the full task, and (*ii*) a collaborative scheme, where the suitable host robot and collaborative node (central cloud, cloudlet, or neighboring robot) conduct the sensing and computation sub-task, respectively. Our proposed task allocation algorithm, which performs both the suitable host robot and collaborative node selection, comprises the following four steps:

- *Step 1:* Initially, an MU sends her H2R task request message (`T-REQ`) to the agent node during her assigned US transmission time slot containing the following information: task location, task type, remaining energy threshold, and task deadline.

- *Step 2:* When the agent at the ONU-MPP receives the task request (`T-REQ`) from the MU, it first checks which robots in its wireless coverage area satisfy the given availability, energy threshold to conduct the task, and task execution deadline requirements. Towards this end, the agent first broadcasts a task announcement message (`RTS`) to all nearby robots. The `RTS` frame is extended by using its pad/reserved bits in order to include additional task request information (`T-REQ`), similar to [100]. After the reception of a `RTS` frame, the available robots in that network send reply messages (`T-RES`) embedded in `CTS` frames to the agent containing the following information: remaining energy, location, moving and processing speed, and pre-calculation of task execution time of each robot. After checking each robot's reply, the agent selects a suitable host robot according to the following criteria: robot availability, remaining energy, and minimum task execution time. The selected host robot is notified by the agent via a winner notification (`T-WNT`) embedded in the `ACK` message. The modified `RTS`, `CTS`, and `ACK` frames are depicted in Fig. 3.3(a).

Algorithm 3 Task allocation algorithm

Considerations: Number of task arrives (n), number of available robots before task allocation (m), residual energy of robots (e_r^j), required energy threshold to process the task (e_{th}), required CPU cycles to process the sensing (s_{cpu}) and computation (c_{cpu}) sub-task, full task execution time of robot (t_j), computation sub-task response time of central cloud (t_c^{cl}), cloudlet (t_c^{ct}), neighboring robot (t_c^o), and deadline (t_c^d), energy consumption of host robot considering own (e_c^j), cloud (e_c^{cl}), cloudlet (e_c^{ct}), and neighboring robot computation sub-task execution (e_c^o).

1: **for** $i = 1$ to n **do**
2: **for** $j = 1$ to m **do**
3: The agent node at the ONU-MPP will check the residual energy (e_r^j) and pre-calculation of task execution time (t_j) of participating robots in that network via exchange of modified `RTS` and `CTS` messages
4: **if** $e_r^j \geq e_{th}$ **then**
5: Allocate task i to host robot j with minimum task execution time t_j
6: The selected host robot executes the sensing sub-task (l_s) of the full task and sends a `PS-Poll` message to agent for computation offloading
7: The agent checks the availability and resource type of central cloud (cl), cloudlet (ct), and neighboring robot (o) for computation sub-task (s_{cpu}) execution
8: **if** $e_c^{cl} < e_c^{ct} \leq e_c^j$ & $e_c^{cl} < e_c^o \leq e_c^j$ & $t_c^{cl} \leq t_c^d$ **then**
9: Offload the computation sub-task to central cloud
10: **else if** $e_c^{ct} < e_c^{cl} \leq e_c^j$ & $e_c^{ct} < e_c^o \leq e_c^j$ & $t_c^{ct} \leq t_c^d$ **then**
11: Offload the computation sub-task to cloudlet
12: **else if** $t_c^o \leq t_c^d$ **then**
13: Offload the computation sub-task to a suitable neighboring robot in that network
14: **else if** $t_c^{cl} \geq t_c^d$ & $t_c^{ct} \geq t_c^d$ & $t_c^o \geq t_c^d$ **then**
15: Execute the computation sub-task (c_{cpu}) locally in the selected host robot (see step 5)
16: **end if**
17: **else**
18: Go to step 1
19: **end if**
20: **end for**
21: **end for**

- *Step 3:* The selected host robot first executes the sensing sub-part of the task. If the host robot sends a computation sub-task offloading request to the agent, the agent selects a suitable collaborative node for the computation sub-task execution (remaining sub-part).

- *Step 4:* The agent checks the computation sub-task response time, resource availability, and energy consumption of all collaborative nodes (i.e., central cloud, cloudlet, and neighboring robots with minimum execution time and energy consumption value). The agent then selects the most suitable collaborative node for computation sub-task execution based on the following criteria: (*i*) computation sub-task response time of the collaborative node is less than or equal to the computation sub-task deadline, (*ii*) sufficient resource availability, and (*iii*) minimum energy consumption of task offloading host robot among all collaborative nodes.

The pseudocode (see **Algorithm 3**) of our proposed task allocation algorithm executed by the ONU-MPP (agent node) is described below in a more formal way by defining the various parameters used in our analytical model. First, the agent node broadcasts the H2R task request message `RTS` (total number of arrived tasks is n and available robots is m) to all available robots under the ONU-MPP and receives the robots' response `CTS` messages from all available robots (lines 1-2). Next, the agent node extracts the robots' residual energy information and pre-calculation of task execution time information from each robot's response message and allocates the H2R task to the robot that has sufficient residual energy to process the task ($e_r^j \geq e_{th}$) and satisfies the minimum task execution time requirements (lines 3-5). Then, if the selected host robot executes the sensing sub-task of the full task and sends a `PS-Poll` message to agent node for collaborative node selection to perform the computation sub-task execution, the agent checks the available resource type of all collaborative nodes (lines 6-7). If only one collaborative node (central cloud, cloudlet, or neighboring robot) satisfies the given task execution deadline, the agent at the ONU-MPP offloads the computation sub-task onto that collaborative node. If more than one collaborative node is able to meet the given computation sub-task execution deadline, the agent offloads the computation sub-task onto the most suitable collaborative node based on its minimum energy consumption (lines 8-13). Conversely, if all available collaborative nodes fail to satisfy the given computation sub-task deadline, the computation sub-task is executed locally at the initially selected host robot (lines 14-15). Further, if the selected host robot fails to execute the computation sub-task, a new host robot needs to be selected for the computation sub-task execution (lines 16-18) and the process is repeated starting at step 1.

3.4 Analytical Model

In this section, we first briefly discuss the assumptions of our analytical model. We then evaluate the performance of our suitable host robot selection scheme in terms of task allocation delay and task execution time. Afterwards, we analyze the end-to-end local/non-local task allocation and computation sub-task offloading delay. Finally, we assess the performance of our proposed collaborative and non-collaborative task execution schemes in terms of task response time and energy consumption efficiency.

3.4.1 Assumptions

The H2R application is assumed to consist of one fine-grained task that includes both the sensing (e.g., image capturing) and computation sub-task (e.g., image/face detection), similar to [94] and [95]. Each robot is able to execute a single task at any given time. Unlike MUs, robots are assumed to be static, though they are able to move and change their position towards task location at low speed (e.g., pedestrian speed). Furthermore, robots are heterogeneous with regard to their remaining energy e_r^j (given in Joule), moving speed v_j (m/s), CPU clock speed μ_j (MHz), and position (x_j, y_j). The request from a given MU for a particular task i includes the task location (x_i, y_i), total or full task input size $(l_i = l_s + l_u)$ that contains both the sensing (l_s) and computing (l_u) sub-task input data size (kBytes), total CPU clock cycles (Megacycles) required to process the full task input data size, including both the sensing and computation sub-task CPU cycles $(t_{cpu} = s_{cpu} + c_{cpu})$, required robot energy threshold for each task (e_{th}), and deadline (t_d) to complete the full H2R task. We also assume that the output of the sensing sub-task is the input of the computation sub-task. The computation sub-task considered for offloading is defined as follows: $c_i \triangleq (c_{cpu}, t_c^d, m_r, l_u, l_r)$, where c_{cpu}, t_c^d, m_r, l_u, and l_r indicate the number of required CPU cycles to process the computation sub-task, deadline, required memory, input, and output size of the computation sub-task, respectively.

3.4.2 Task Allocation Delay

The calculation of the average task allocation delay involves the transmission times of several control messages that are exchanged between the ONU-MPP (agent) and robots, namely, task announcement via an extended `RTS` (t_{rts}), robot response via an extended `CTS` (t_{cts}), and winner notification via an extended `ACK` (t_{ack}) message, similar to the IEEE 802.11 DCF analysis in [101]. Hence, the probability that a given robot sends a response message in a random time slot is given by $p_k = \frac{2}{k+1}$, where k denotes the constant backoff window size. For a total number m of robots participating in the task allocation, the probability that at least one robot transmits a response in a random time slot equals $p_{tr} = 1 - (1 - p_k)^m$. Thus,

the probability that a single robot transmits a response in a time slot is then obtained as $p_{cs} = \frac{mp_k(1-p_k)^{m-1}}{p_{tr}}$.

For computing the average robot selection delay per task (t_{si}), we assume that the agent sends an `RTS` message to robots after waiting for DCF interframe space (t_{DIFS}), while robots send a `CTS` message after waiting for short interframe space (t_{SIFS}). After t_{SIFS}, the agent sends an `ACK` message to the selected robot. By using the values of t_{DIFS}, t_{SIFS}, t_{rts}, t_{cts}, and t_{ack}, both successful ($t_{sa} = t_{DIFS} + t_{rts} + mt_{SIFS} + mt_{cts} + t_{SIFS} + t_{ack}$) and unsuccessful task allocation time periods ($t_{ua} = t_{DIFS}+t_{rts}+mt_{SIFS}+mt_{cts}$) can be evaluated. Consequently, the average robot selection delay per task (t_{si}) is equal to $t_{si} = \frac{p_e\sigma+p_st_{sa}+p_ut_{ua}}{p_s}$, where $p_e = (1-p_{tr})$, $p_s = p_{tr}p_{cs}$, and $p_u = p_{tr}(1-p_{cs})$ denote the empty, successful, and unsuccessful transmission probabilities in a time slot, respectively, and σ denotes the length of an idle time slot.

Next, we calculate the saturation throughput of our network. Assuming that n is the total number of task requests arriving at a given agent and m is the number of available robots in the network with sufficient remaining energy, the robot availability probability per task p_{avg} can be approximated by $p_{avg} = min(1, \frac{m}{n})$. Further, by taking into account t_{si} and p_{avg}, the total task allocation delay (t_{alloc}) can be calculated as $t_{alloc} = t_{si}(n \cdot p_{avg})$. Additionally, if the total task allocation duration (t_{alloc}) is known, the total number of suitable robots (n_r) selected during task allocation is given by $n_r = \frac{t_{alloc}}{t_{si}}$. Similarly, by using n, p_{avg}, and n_r, the total number of allocated tasks (n_a) during task allocation is given by $n_a = \sum_{i=1}^{n} min(i, n_r) \cdot \binom{n}{i=1}(p_{avg})^i(1-p_{avg})^{n-i}$. Thus, given the total data transmission duration (t_{total}), number of allocated tasks (n_a), and task allocation duration (t_{alloc}), the saturation throughput (s_{th}) is obtained as follows:

$$s_{th} = \frac{n_a(t_{total} - t_{alloc})}{t_{total}}. \tag{3.1}$$

3.4.3 Task Execution Time Without Offloading

The task execution/response time for different robots (t_j) is a crucial performance metric in the suitable host robot selection process that consists of the following four delay components: task allocation delay (t_{alloc}), time delay to reach the task location (t_r^j), time delay to process the sensing sub-task (t_s^j), and time delay to process the computation sub-task (t_c^j). Thus, t_j is given by

$$t_j = t_{alloc} + t_r^j + t_s^j + t_c^j = t_{alloc} + \frac{d_{ij}}{v_j} + \frac{s_{cpu}}{\mu_j} + \frac{c_{cpu}}{\mu_j}, \tag{3.2}$$

where d_{ij}, v_j, μ_j, s_{cpu}, and c_{cpu} denote the distance between task i and location of robot j, moving and processing speed of robot j, required CPU cycles for sensing and computation sub-task, respectively. The distance between task and robot location (d_{ij}) is calculated via

the Euclidean distance as follows: $d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$, where (x_i, y_i) and (x_j, y_j) denote task i's and robot j's position in the two dimensional space, respectively. For local (i.e., MUs and robots are associated with the same ONU-MPP) and non-local task (i.e., MUs and robots are associated with different ONU-MPPs) allocation, the calculation of the total task execution time considers the end-to-end local (t^l_{alloc}) and non-local task allocation delay (t^{nl}_{alloc}) instead of t_{alloc}. By accounting for both t^l_{alloc} and t^{nl}_{alloc} (see Section 3.4.4), the end-to-end local (t^j_{local}) and non-local ($t^j_{non-local}$) task execution time for selected host robots are given by $t^j_{local} = t^l_{alloc} + t^j_r + t^j_s + t^j_c$ and $t^j_{non-local} = t^{nl}_{alloc} + t^j_r + t^j_s + t^j_c$, respectively.

3.4.4 FiWi End-to-End Task Allocation and Offloading Delay

In this section, we present the end-to-end delay analysis for both local and non-local task allocation based on the US (from ONU-MPP to OLT) and DS (from OLT to ONU-MPP) frame transmission delay model in [93]. Note, however, that we had to modify the analytical model in [93] in order to accommodate our proposed resource allocation process, as explained in the following.

As shown in Fig. 3.2, the ONU-MPP time cycle (T_c) consists of the task allocation delay (t_{alloc}), MPCP message duration (t^{msg}_{pon}), conventional broadband traffic transmission (t^m_{sl}), computation offloading data transmission (t^{α}_{sl}), guard band (t_g), reservation ($V = t^{msg}_{wl} + t_g$), and vacation period. Recall that the considered FiWi network serves N ONUs and M users (humans and robots) at each ONU-MPP. We assume that H2R packets arrive at the ONU-MPP with an aggregate arrival rate of $\overline{\lambda}$ according to a Poisson process and M/G/1 queue. The random packet service time is given by the first moment ($\overline{X}$). Further, the aggregate H2R traffic load is equal to ρ^{h2r}=$\overline{\lambda X}$. During each cycle time T_c, the vacation duration equals $(N-1)t_{sl}$ and $RTT = 2t_{prop}$ denotes the round-trip time between OLT and ONU-MPPs. The non-data traffic transmission time during T_c is equal to $N(MV + t_{alloc} + t^{msg}_{pon})$. Thus, T_c is calculated as follows:

$$T_c = \frac{N(MV + RTT + t_{alloc} + t^{msg}_{pon})}{1 - \rho^{h2r}}. \tag{3.3}$$

Recall from above that an MU sends her US bandwidth request via a `PS-Poll` message. If the H2R task request is generated at the MU after the current `PS-Poll` message, it waits during the first delay component T_c (polling cycle time) for the next `PS-Poll` message to report to the ONU-MPP (agent) about her US sub-slot request (see Fig. 3.2). After the MU sends the task request frame to the ONU-MPP during the next time cycle sub-slot ($t^m_{sl} \geq t^{msg}_{wl}$), the total queuing delay time is equal to $2T_c - t^{msg}_{wl}$. The other two delay components associated with the US frame transmission are the US frame service time at the ONU-MPP ($\overline{X}_u \leq x_{max}$) and the propagation delay (t_{prop}). Thus, the maximum US frame transmission delay (t_u) is given by $t_u = 2T_c - t^{msg}_{wl} + t_{prop} + x_{max}$. Next, we compute the DS frame delay. If the OLT

receives the MU's task request frame after the `GATE` message, the frame has to wait for $T_c - t_{sl}$. The other delay components are the DS frame service time delay at the OLT ($\overline{X}_d \leq x_{max}$) and the associated propagation delay (t_{prop}). Given that $t_{sl} \geq 2t_{prop} + 2t_{pon}^{msg}$, the DS frame transmission delay (t_d) is equal to $t_d = T_c - 2t_{pon}^{msg} - t_{prop} + x_{max}$.

By using US and DS frame delays, we are able to measure the end-to-end local and non-local task allocation delay as follows. If the MU requests a non-local task allocation, the associated ONU-MPP first sends the US task request frame to the OLT. The OLT then broadcasts the task request frame to all ONU-MPPs, whereby only the intended ONU-MPP processes the task request frame and initiates the non-local task allocation process. For the calculation of the non-local task allocation delay (t_{alloc}^{nl}), both US and DS frame transmission delays are required along with the task allocation delay (t_{alloc}). Thus, we have $t_{alloc}^{nl} = t_u + t_d + t_{alloc}$. Conversely, if the task request is local, the agent at the associated ONU-MPP receives the task request frame from the MU and starts the task allocation process immediately. Thus, for the calculation of the local task allocation delay (t_{alloc}^{l}), only the US frame transmission delay (t_u) is required along with the t_{alloc}. As a result, t_{alloc}^{l} is obtained as $t_{alloc}^{l} = t_u + t_{alloc}$.

Next, we analyze the maximum offload packet delay that incurs during the computation sub-task offloading process. By assuming that a computation offload sub-slot is assigned to all devices M during each time cycle T_c and the collaborative nodes' traffic load is equal to $(\rho^{cl}, \rho^{ct}, \rho^{o})$, the computation offload sub-slot duration (t_{sl}^{α}) can be expressed as $t_{sl}^{\alpha} = \frac{\rho^{cn} \cdot T_c}{M}$, where subscript cn can be either the central cloud (cl), a local cloudlet (ct), or a neighboring robot (o), respectively. As shown in Fig. 3.2, if the computation offload request (a) is generated at the host robot after a `PS-Poll` message, it experiences the maximum computation offload packet buffering delay. For an upcoming transmission opportunity, the computation sub-task offload packet needs to wait.

Note that the end-to-end maximum computation sub-task offload packet delay consists of four delay components, as shown in Fig. 3.2. The first delay component is the time difference between the computation offload request arrival (a) and the transmission of offload reservation request (r) via a `PS-Poll` frame ($\overline{d}_1 = (M-1)t_{sl}^{\alpha} + (N-1)t_{sl} + t_{sl}^{m} + t_{alloc} + t_{pon}^{msg} + t_{wl}^{msg}$). The second delay component is the time gap between the transmission of offload reservation request (r) and the reception of offload grant (g) or `MPCP` frame ($\overline{d}_2 = (M-1)t_{sl}^{m} + Mt_{sl}^{\alpha} + (N-1)t_{sl} + t_{alloc} + t_{pon}^{msg}$). The third delay component is the time gap between the last grant (g) message and the start time of the offload (α) slot ($\overline{d}_3 = t_{wl}^{msg} + Mt_{sl}^{m}$). By summing up the first three delay components ($\overline{d}_1, \overline{d}_2, \overline{d}_3$), we observe from Fig. 3.2 that the maximum computation offload packet delay for cloudlet ($\overline{d}_{ct} \leq 2T_c$) and neighboring robot offloading ($\overline{d}_o \leq 2T_c$) is equal to $2T_c$. Whereas for central cloud offloading, the fourth delay component ($\overline{d}_4 = 2t_{prop}$) along with the first three

components of the maximum offload packet delay calculation is given by $\overline{d}_{cl} \leq 2T_c + 2t_{prop}$, whereby ($t_{prop}$) denotes the one-way propagation delay between OLT and central cloud.

3.4.5 Performance Analysis of Computation Offloading and Collaborative Task Execution

In this subsection, we analyze the performance of non-collaborative and collaborative task execution schemes in terms of task response time and energy consumption efficiency. The first part of this subsection presents the calculation of computation sub-task response time and energy consumption of a host robot for different offloading schemes to decide whether or not computation sub-task offloading to a collaborative node is useful for the task offloading robot.

The computation sub-task response time for central cloud offloading (t_c^{cl}) consists of two parts: offloading delay (t_{ofl}^{cl}) and computation sub-task processing delay at the central cloud ($t_{ex}^{cl} = \frac{c_{cpu}}{\mu_{cl}}$). The central cloud offloading delay considers the uplink communication delay ($t_{cl}^{u} = \frac{l_u}{b_{wl}} + \frac{l_u}{b_{cl}} + \frac{l_u}{b_{cl}}$), downlink communication delay ($t_{cl}^{r} = \frac{l_r}{b_{wl}} + \frac{l_r}{b_{cl}} + \frac{l_r}{b_{cl}}$), and total propagation delay, including both air (wireless part) and fiber (optical part) propagation delays. The uplink delay (t_{cl}^{u}) consists of the time required to transfer computation sub-task offloading packets from the host robot to the agent across the wireless link and further from the agent (ONU-MPP) to the OLT and to the central cloud. The downlink delay (t_{cl}^{r}) measures the time period required to transfer the resultant offloading packet from the central cloud to the host robot via OLT and agent, respectively. Thus, t_c^{cl} is computed as follows:

$$t_c^{cl} = t_{ofl}^{cl} + t_{ex}^{cl} = \frac{2(l_u + l_r)}{b_{cl}} + \frac{(l_u + l_r)}{b_{wl}} + t_{prop}^{cl} + \frac{c_{cpu}}{\mu_{cl}}. \tag{3.4}$$

Similarly, the computation sub-task response time for cloudlet offloading (t_c^{ct}) comprises the offloading delay (t_{ofl}^{ct}) and computation sub-task processing delay at the cloudlet ($t_{ex}^{ct} = \frac{c_{cpu}}{\mu_{ct}}$). The cloudlet offloading delay involves the uplink ($t_{ct}^{u} = \frac{l_u}{b_{wl}} + \frac{l_u}{b_{ct}}$) and downlink ($t_{ct}^{r} = \frac{l_r}{b_{wl}} + \frac{l_r}{b_{ct}}$) communication delay along with the total propagation delay (t_{prop}^{ct}) that incurs during the offloading process. The uplink delay (t_{ct}^{u}) includes the offload packet transmission time from the host robot to the agent and from the agent to the cloudlet by using the wireless and fiber link, respectively. The downlink delay (t_{ct}^{r}) captures the resultant offload packet transfer time period from the cloudlet to the host robot via the intermediate agent. Thus, t_c^{ct} is given by

$$t_c^{ct} = t_{ofl}^{ct} + t_{ex}^{ct} = \frac{(l_u + l_r)}{b_{ct}} + \frac{(l_u + l_r)}{b_{wl}} + t_{prop}^{ct} + \frac{c_{cpu}}{\mu_{ct}}. \tag{3.5}$$

The computation sub-task response time for neighboring robot offloading (t_c^{o}) also accounts for both offloading delay (t_{ofl}^{o}) and computation sub-task processing delay at the neighboring robot ($t_{ex}^{o} = \frac{c_{cpu}}{\mu_{o}}$). The neighboring robot offloading delay (t_{ofl}^{o}) includes the suitable neighboring robot selection delay (t_{alloc}), uplink delay ($t_{o}^{u} = \frac{2l_u}{b_{wl}}$), downlink delay ($t_{o}^{r} = \frac{2l_r}{b_{wl}}$), and

total propagation delay (t^o_{prop}) that occurs during the offloading process. The uplink delay (t^u_o) comprises the offload packet transfer time from the task offloading host robot to the agent and from the agent to the selected neighboring robot by using the wireless link. The downlink delay (t^r_o) consists of the resultant packet transfer time from the neighboring robot to the task-offloading host robot across the agent. Thus, t^o_c is obtained as follows:

$$t^o_c = t^o_{ofl} + t^o_{ex} = t_{alloc} + \frac{2(l_u + l_r)}{b_{wl}} + t^o_{prop} + \frac{c_{cpu}}{\mu_o}. \tag{3.6}$$

Next, we analyze the total task response time of the following three collaborative task execution schemes: (*i*) host robot-central cloud, (*ii*) host robot-cloudlet, and (*iii*) host robot-neighboring robot. If the sensing sub-task is executed by the selected host robot and the computation sub-task is offloaded onto the central cloud, by using Eqs. (3.2) and (3.4) the total task response time ($t_{j,cl}$) for the host robot-central cloud based joint task execution is given by

$$t_{j,cl} = t_{alloc} + t^j_r + t^j_s + t^{cl}_c. \tag{3.7}$$

If the sensing sub-task is executed by the host robot and the computation sub-task is offloaded onto a nearby cloudlet for execution, by using Eqs. (3.2) and (3.5) the total task response time for the host robot-cloudlet ($t_{j,ct}$) based joint execution is obtained as follows:

$$t_{j,ct} = t_{alloc} + t^j_r + t^j_s + t^{ct}_c. \tag{3.8}$$

Further, if the sensing sub-task is performed by the host robot and the computation sub-task is offloaded onto a neighboring robot, then by using Eqs. (3.2) and (3.6) the total task response time ($t_{j,o}$) for the host robot-neighboring robot based joint execution is equal to

$$t_{j,o} = 2t_{alloc} + t^j_r + t^j_s + t^o_c \tag{3.9}$$

By taking the local and non-local task allocation delays (see Section 3.4.4) into account, the end-to-end non-local ($t^{j,cn}_{non-local} = t_u + t_d + t_{j,cn}$) and local task response time ($t^{j,cn}_{local} = t_u + t_{j,cn}$) for the case of collaborative task execution are obtained, whereby the subscript *cn* can stand for either *cl*, *ct*, or *o*, respectively. Using both collaborative total task response time ($t_{j,cn}$) and maximum offload packet buffering delay ($\overline{d}_{cn}$), we are able to assess the maximum total task response time for the host robot-central cloud ($\overline{t}_{j,cl} = \overline{d}_{cl} + t_{j,cl}$), host robot-cloudlet ($\overline{t}_{j,ct} = \overline{d}_{ct} + t_{j,ct}$), and host robot-neighboring robot ($\overline{t}_{j,o} = \overline{d}_o + t_{j,o}$) based joint execution schemes. Note that in the non-collaborative host robot based task execution the maximum response time ($\overline{t}_j$) is equal to t_j since there is no offloading delay.

Energy consumption of host robot for computation sub-task offloading onto the collaborative node (central cloud, cloudlet, or neighbor robot) consists both energy consumption for

offloading communication delay and computation sub-task processing delay. Next, by using Eqs. (3.4)-(3.6) and Table 3.2, the energy consumption of the host robot for computation sub-task offloading onto the central cloud ($e_c^{cl} = e_{ofl}^{cl} + e_{ex}^{cl}$), cloudlet ($e_c^{ct} = e_{ofl}^{ct} + e_{ex}^{ct}$), and neighboring robot ($e_c^{o} = e_{ofl}^{o} + e_{ex}^{o}$) is obtained as follows:

$$e_c^{cl} = p_u\left(\frac{2l_u}{b_{cl}} + \frac{l_u}{b_{wl}}\right) + p_r\left(\frac{2l_r}{b_{cl}} + \frac{l_r}{b_{wl}}\right) + p_{idle}\left(\frac{c_{cpu}}{\mu_{cl}} + t_{prop}^{cl}\right) \tag{3.10}$$

$$e_c^{ct} = p_u\left(\frac{l_u}{b_{ct}} + \frac{l_u}{b_{wl}}\right) + p_r\left(\frac{l_r}{b_{ct}} + \frac{l_r}{b_{wl}}\right) + p_{idle}\left(\frac{c_{cpu}}{\mu_{ct}} + t_{prop}^{ct}\right) \tag{3.11}$$

$$e_c^{o} = p_u\left(\frac{2l_u}{b_{wl}}\right) + p_r\left(\frac{2l_r}{b_{wl}}\right) + p_{idle}\left(\frac{c_{cpu}}{\mu_{o}} + t_{prop}^{o}\right) + e_a^j, \tag{3.12}$$

where e_a^j denotes the energy consumed during the task allocation process. From Eqs. (3.10)-(3.12), we can calculate energy consumption of host robot for computation sub-task processing delay at central cloud ($e_{ex}^{cl} = p_{idle} \cdot t_{ex}^{cl}$), cloudlet ($e_{ex}^{ct} = p_{idle} \cdot t_{ex}^{ct}$), and neighbor robot ($e_{ex}^{o} = p_{idle} \cdot t_{ex}^{o}$), where p_{idle} is the average idle energy consumption of host robot for computations sub-task processing at collaborative node. Similarly, from Eqs. (3.10)-(3.12), we can obtain energy consumption of host robot for central cloud ($e_{ofl}^{cl} = e_c^{cl} - e_{ex}^{cl}$), cloudlet ($e_{ofl}^{ct} = e_c^{ct} - e_{ex}^{ct}$), and neighbor robot ($e_{ofl}^{o} = e_c^{o} - e_{ex}^{o}$) offloading communication delay.

For the power consumption (p_c) and time (t_c^j) to process the computation sub-task given by Eq. (3.2) and Table 3.2, the energy consumption of the host robot for executing its own computation sub-task (e_c^j) is equal to $e_c^j = p_c \cdot t_c^j = p_c \cdot \frac{c_{cpu}}{\mu_j}$.

In the following, we analyze the total energy consumption of the host robot for both collaborative and non-collaborative full-task execution schemes, which involves four parts. The first part of the task allocation process (e_a^j) corresponds to transmitting ($e_{tx}(l,d) = (\varepsilon_{elec} + \varepsilon_{amp} \cdot d^2) \cdot l$) and receiving ($e_{rx}(l) = \varepsilon_{elec} \cdot l$) the robot selection control packet (l bit) over a distance d, whereby ε_{elec} and ε_{amp} represent the energy dissipation of the radio electronics ($\varepsilon_{elec} = 50$ nJ/bit) and transmit amplifier ($\varepsilon_{amp} = .0013$ pJ/bit), respectively [95]. Hence, e_a^j is given by $e_a^j = e_{tx}(l,d) + e_{rx}(l)$. The second part is the energy consumption (e_d^j) of selected host robot to reach the task location. For the average power consumption (p_v) and time (t_r^j) to reach the task location given in Eq. (3.2) and Table 3.2, e_d^j is obtained as $e_d^j = p_v \cdot t_r^j = p_v \cdot \frac{d_{ij}}{v_j}$, where d_{ij} and v_j denote the distance between robot and task location and the speed of the moving robot, respectively. The third type of energy consumption is related to the sensing sub-task. For a given average power consumption (p_s) and time (t_s^j) to process the sensing sub-task, the energy consumption of the host robot for executing the sensing sub-task (e_s^j) is given by $e_s^j = p_s \cdot t_s^j = p_s \cdot \frac{s_{cpu}}{\mu_j}$. Finally, the fourth type of energy consumption related to the computation sub-task was computed above in Eqs. (3.10-3.12).

By summing up the above four energy consumption parts, the total energy consumption of the robot (e_w^j) during the full-task execution is obtained as follows:

$$e_w^j = \sum_{i\epsilon n} e_a^j + \sum_{i\epsilon n} e_s^j + \sum_{i\epsilon n} e_d^j + \sum_{i\epsilon n} e_c^j + \sum_{i\epsilon n} e_c^{cn}, \tag{3.13}$$

where n is the number of processed tasks and the subscript cn stands for cl, ct, and o, respectively. Taking the host robot's initial energy (e_j^i) and total consumed energy (e_w^j) into account, the residual energy of a robot (e_r^j) equals $e_r^j = e_j^i - e_w^j$. Note that if the full task is executed by the host robot itself, the total energy consumption of the host robot for the resultant non-collaborative task execution is given by $e_j = e_a^j + e_s^j + e_d^j + e_c^j$. Hence, if the sensing sub-task is performed by host robot itself while the computation sub-task is offloaded onto a collaborative node, the energy consumption of the host robot for such a collaborative/joint task execution is equal to $e_{j,cn} = e_a^j + e_s^j + e_d^j + e_c^{cn}$.

3.4.6 Task Response Time and Energy Consumption Efficiency

As the total task response time and energy efficiency ratio are key performance metrics, we analyze both of them in this subsection. In our calculation of the total task response time efficiency (t_{eff}), we use the total task response time of the collaborative execution ($t_{j,cn}$) and non-collaborative host robot execution (t_j). The energy efficiency of the full-task execution (e_{eff}) is obtained from the energy consumption of the host robot for collaborative ($e_{j,cn}$) and non-collaborative task execution (e_j). The total task response time (t_{eff}) and energy consumption (e_{eff}) efficiency ratio taking both collaborative and non-collaborative task execution into account are given by $t_{eff} = \frac{t_j - t_{j,cn}}{t_j}$ and $e_{eff} = \frac{e_j - e_{j,cn}}{e_j}$, where subscript cn may stand for the central cloud (cl), cloudlet (ct), or neighboring robot (o), respectively.

Hence, the offload gain-overhead ratio (γ_{cn}) for computation sub-task offloading to a collaborative node is given by the ratio of computation sub-task offload gain ($t_c^j - t_c^{cn}$) and overhead ($\overline{d}_{cn} + t_{ofl}^{cn}$):

$$\gamma_{cn} = \frac{t_c^j - t_c^{cn}}{\overline{d}_{cn} + t_{ofl}^{cn}}. \tag{3.14}$$

3.5 Results

In this section, we investigate the performance of our proposed collaborative and non-collaborative task execution schemes leveraging on the different capabilities of the central cloud, cloudlets, and robots. Table 3.2 summarizes the key system parameters and their assigned default values in compliance with previous studies [17], [30], [94], [95], [101].

Table 3.2: Notation and default values for evaluation of collaborative computing based task execution scheme

Symbol	Definition	Value/Unit
d_{ij}, e_j^i, e_{th}	Distance (robot to task location), initial energy of robots, and required energy threshold	1-10 m, 200 KJ, 50 J
p_c, p_v, p_s	Robot average power consumption for processing, moving, and sensing sub-task	.5 W, .5 W, .5 W
l_i, l_s, l_u, l_r	Total task input (full), sensing sub-task input, computation sub-task input, and output data size	KB (vary)
v_j, t_{cpu}, s_{cpu}, c_{cpu}	Robot moving speed, total task, sensing, and computation sub-task CPU cycles (workload)	.1-1 m/s, Mega cycles (vary)
m_{ct}, m_o, m_r	Available memory space of cloudlet, neighboring robot, and required size for offloading	MB (vary)
$\mu_j, \mu_{cl}, \mu_{ct}, \mu_o$	CPU clock frequency of host robot, central cloud, cloudlet, and neighboring robot	MHz (vary)
$t_{prop}^{cl}, t_{prop}^{ct}, t_{prop}^{o}$	Total propagation delay for cloud, cloudlet, and neighboring robot offloading	ms (vary)
p_{idle}, p_u, p_r	Average energy consumed by robot during idle, data transmission, and reception	.001 W, .1 W, .05 W
b_{wl}, b_{cl}, b_{ct},	Transmission capacity of wireless and fiber link for cloud, cloudlet offloading	6900 Mb/s, 10 Gb/s, 10 Gb/s

System settings, requirements, and configurations: In this work, we assume that robot can only perform location dependent sensing sub-task (capturing image at a task location) due to their movement and workload processing capabilities, whereas both cloud agent and robot can execute location independent computation sub-task (object detection from captured image). The output of sensing sub-task (e.g., capturing image) is the input data of computation sub-task (object detection from captured image). The main requirements of the low-latency H2R applications (move to task location, capturing image, and face detection from captured image) is the availability of robot and cloud server resources for the requested task processing, satisfaction of the task execution deadline and lower energy consumption criteria, priority based cloud/robot and bandwidth resource assignment to HART applications users and worker nodes, transmission at the speed of light, placement of edge cloud server within 20 Km distance from the decentralized ONUs to achieve very low round-trip latency (i.e., 1 ms), connectivity of robot, MUs, and cloudlet server with ONUs, hardware/software interface to transfer the task request and task result, among others. The decentralized task coordinator is located at the ONU-MPP to assign the MU's task to suitable robot or cloud server for processing. Further, note that the robots are connected with the task coordinator (ONU) by using WLAN connectivity, whereas the cloudlets are connected to ONUs through dedicated point-to-point fiber links. In addition, the EPON based fiber backhaul (20 Km) is used for

ONU to OLT connectivity. Moreover, the remote cloud server is connected with the OLT through dedicated point-to-point fiber links. Each task request is served based on first-come-first-served (FCFS) manner. For polling cycle time and queuing delay analysis, the ONU-MPP incorporates M/G/1 queue with reservations and vacations. The maximum clock speed (CPU speed) of cloudlet and central cloud server is varied within the range between 1800 MHz to 5000 MHz. The CPU speed of robots is varied within the range between 500 MHz to 1600 MHz. The EPON transmission capacity between the link of ONU-MPP and cloudlet, OLT and central cloud, ONU to OLT is set to 10 Gb/s, whereas maximum line rate at the wireless medium is set to 6900 Mb/s. The optical fiber length between the ONU and OLT, the ONU and cloudlet server, and the OLT and central cloud server is set to 20 Km, 1 Km, and 10 Km, respectively. The total STA number per ONU-MPP coverage, total ONUs, MAP radius, ONU-MPP coverage area, and density of MAPs within each ONU-MPP coverage area, maximum transmission capacity at fiber and wireless link is set to 10, 16, 100 m, 10 km^2, 4, 10 Gb/s, and 6900 Mb/s, respectively. The FiWi traffic load is varied within the range between .05 to .95. The MPCP (REPORT and GATE) and WLAN messages (Ps-Poll) duration is set to 64 bytes and 20 bytes, respectively. Average energy consumption cost (per second) of robots/MUs during idle (p_{idle}) state, data transmission (p_u), workload processing, and task result reception (p_r) activities are set to .001W, .1W, .5W, and .05W, respectively. Both sensing (s_{cpu}) and offloaded sub-task (c_{cpu}) workload is varied within the range between 40 to 960 Mcycles. The offloaded sub-task input data size (l_u) and output data size (l_r) is varied within the range between 40-800 KB and 16-400 KB, respectively.

3.5.1 Collaborative vs. Non-collaborative Task Execution

In this sub-section, we compare the performance of the non-collaborative (i.e., without offloading) and collaborative/joint task execution schemes, whereby the sensing sub-task is conducted by the selected host robot and the computation sub-task is offloaded onto a collaborative node. To examine the impact of our proposed collaborative computing based task execution scheme, we studied different evaluation scenarios based on different H2R task input and output data sizes, required workload (in terms of CPU cycles) to process the task, and collaborative nodes' resource conditions (i.e., processing power, available memory size, availability), similar to [17], [30], [94]. The parameter settings related to each particular scenario are given in Figs. 3.4-3.7. Moreover, for a particular H2R task that includes both sensing and computation sub-parts, four different types of task execution schemes were considered: selected host robot based full-task execution without offloading, host robot (sensing sub-task) with central cloud execution (computation sub-task), host robot (sensing sub-task) with cloudlet execution (computation sub-task), and host robot (sensing sub-task) with neighboring robot execution (computation

sub-task). The total task response time for the three collaborative and one non-collaborative schemes are calculated by using Eqs. (3.7)-(3.9) and Eq. (3.2), respectively. The energy consumption of the host robot for the collaborative ($e_{j,cn} = e_a^j + e_s^j + e_d^j + e_c^{cn}$) and non-collaborative ($e_j = e_a^j + e_s^j + e_d^j + e_c^j$) total task execution is calculated by using Eq. (3.13) (see Section 3.4.5).

Figs. 3.4(a) and (b) illustrate the total task response time and host robot energy consumption of the different task execution schemes for scenario 1. In this scenario 1, both the central cloud and cloudlet are assumed to have the same computation capability/CPU power. The figures show that the task response time and energy consumption of host robot increase for increasing task input data size in all proposed task execution schemes. We notice that the host robot-neighboring robot based joint task execution scheme shows a higher task response time than the host robot-central cloud scheme and fails to meet the task deadline requirement. The reason for this observation is the fact that the neighboring robot CPU power (500MHz) is lower than the central cloud CPU power (3200MHz). Thus, the computation sub-task processing delay is much higher in the neighbor robot than that of the central cloud execution, which additionally results in a longer total task response time for the host robot-neighbor robot scheme. For instance, for a typical total task input size of 240KB, the total task response time in the host robot-neighbor robot and host robot-central cloud scheme equals 4.56 and 2.95 seconds, respectively, whereas the computation sub-task processing delay of the neighbor robot (t_{ex}^o) and central cloud (t_{ex}^{cl}) equals 1.92 and 0.3 seconds, respectively. Hence, the computation sub-task offloading delay of the neighbor robot (t_{ofl}^o) and central cloud (t_{ofl}^{cl}) are equal to 0.049 and .056 second, respectively. However, the energy efficiency gain of the host robot-neighbor robot compared to the host robot-central cloud is negligible, less than 1%. This is because the difference between the energy consumption of the host robot for the central cloud ($e_{ex}^{cl} = p_{idle} \cdot t_{ex}^{cl}$) and neighbor robot ($e_{ex}^o = p_{idle} \cdot t_{ex}^o$) computation sub-task processing delay is very small. The average energy consumption of the host robot (per second) is very low during the neighbor robot (in host robot-neighbor robot scheme) and central cloud (in host robot-central cloud scheme) computation sub-task processing, e.g., p_{idle}=.001W (see Table 3.2), as the host robot is idle at that time. Therefore, the difference between the host robot energy consumption for the host robot-neighbor robot and host robot-central cloud total task execution is also very low. Due to the lower energy consumption of the host robot for central cloud computation sub-task processing in Fig. 3.4(b), the host robot-central cloud execution shows 1% higher energy efficiency gain compared to the host robot-neighbor robot scheme. For instance, for a typical total task input size of 240KB, the host robot energy consumption in the host robot-neighbor robot and host robot-central cloud scheme equals 1.72J and 1.71J, respectively, whereby the host robot energy consumption for computation sub-task processing

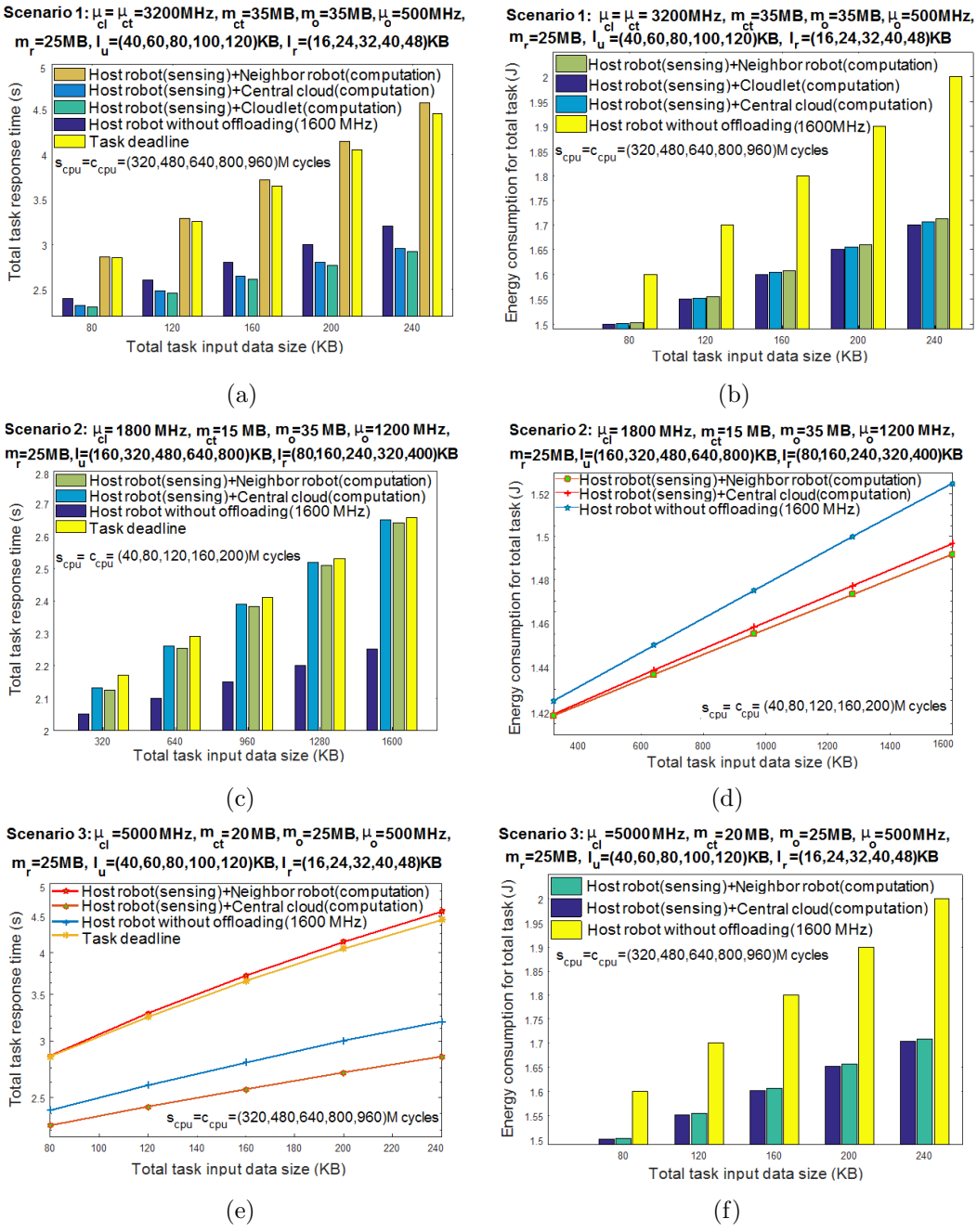


Figure 3.4: Task response time and energy consumption variation of collaborative and non-collaborative task execution schemes versus total task input data size for three different scenarios.

at the neighbor robot (e^o_{ex}) and central cloud (e^{cl}_{ex}) is equal 0.00192J and 0.0003J, respectively. Note, however, that the host robot energy consumption for neighbor robot (e^o_{ofl}) and central cloud (e^{cl}_{ofl}) offloading equals 0.00418J and 0.0049J, respectively.

Furthermore, we observe from both figure 3.4(a) and (b) that the host robot-cloudlet based joint task execution scheme outperforms the host robot-central cloud based joint scheme in terms of task response time and energy consumption of host robot. This is mainly due to the fact that the cloudlet implies a smaller computation offloading delay than the central cloud. The host robot-cloudlet based scheme shows a 36%, 8%, 2% increase of task response time efficiency and a 3%, 15%, 2% higher energy efficiency than the host robot-neighbor robot, host robot without offloading, and host robot-cloud based scheme, respectively. Thus, the host robot-cloudlet based joint task execution scheme is optimal for scenario 1.

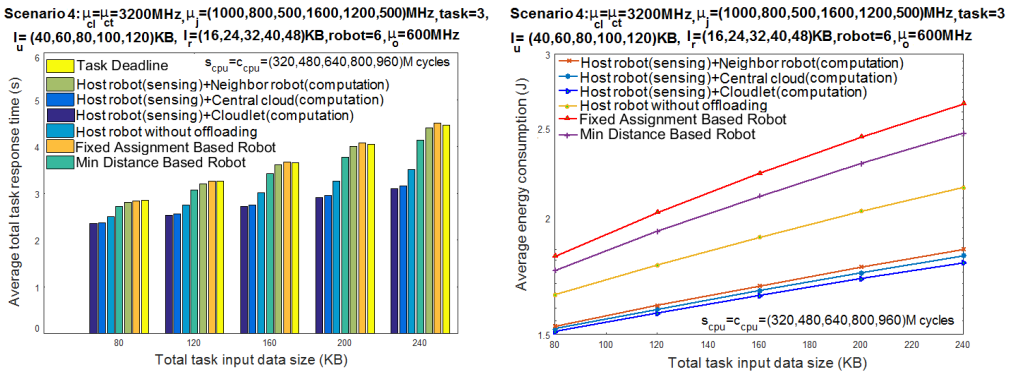
Interestingly, Figs. 3.4(c) and (d) indicate that the neighboring robot can also be selected as a collaborative node for computation sub-task offloading since the sensing sub-task is restricted to the initially selected host robot. In scenario 2, the central cloud CPU power and task workload (required CPU cycles to process the task) are smaller than in scenario 1. Hence, the cloudlet is unsuitable in this task execution scenario due to its insufficient available memory size. further, from Fig. 3.4(c) we notice that, in comparison with the host robot without offloading scheme, the host robot-neighbor robot scheme experiences a longer response time (15% less gain than host robot without offloading scheme for 1600 KB total task input data size). This is because the host robot CPU power (μ_j=1600MHz) is higher than that of the neighbor robot CPU power (μ_o=1200MHz), which eventually causes a longer total task response time in the host robot-neighbor robot scheme compared to the host robot without offloading scheme. However, the host robot-neighbor robot scheme achieves a higher energy efficiency gain than host robot without offloading scheme by offloading the computation sub-task to neighbor robot. Host robot consumes very little average idle energy consumption (p_{idle}=.001W per second) during neighbor robot computation sub-task execution in the host robot-neighbor robot scheme, which is smaller than the host robot average energy consumption (p_c=.5W per second) for its own computation sub-task processing in the host robot without offloading scheme. By contrast, Fig. 3.4(d) shows that the energy savings of the host robot for the host robot-neighbor robot scheme compared with host robot without offloading is not that significant for the following two reasons. First, the longer computation sub-task processing time at the neighbor robot causes an increased idle energy consumption that reduces the host robot's energy savings in the host robot-neighbor robot scheme. Second, due to the smaller computation sub-task response time during the host robot's own execution, the energy consumption of the host robot in the host robot without offloading scheme is less. Thus, in Fig. 3.4(d), the energy efficiency gain achieved by the host robot-neighbor robot

scheme compared with host robot without offloading is very low. For instance, in Fig. 3.4(d), for a typical total task input size of 1600KB, the host robot energy consumption for its own computation sub-task execution in the host robot without offloading scheme equals 0.0625J, while that of the neighbor robot computation sub-task execution in the host robot-neighbor robot scheme is equal to 0.0292J. Hence, the host robot energy consumption of the total task (sensing and computation) execution equals 1.53J in the host robot without offloading and 1.49J in the host robot-neighbor robot scheme, respectively.

Further, from both Figs. 3.4(c) and (d) we observe that the host robot-neighboring robot based joint task execution scheme exhibits an improved task response time compared to the host robot-central cloud based joint scheme (1%) due to its lower computation offloading delay. The host robot-neighboring robot scheme achieves a 1% and 3% higher energy efficiency than the host robot-cloud and host robot without offloading scheme, respectively. Thus, the host robot-neighboring robot based joint task execution scheme is the most suitable one for this scenario by providing the lowest energy consumption while satisfying the task deadline.

Figs. 3.4(e) and (f) depict the task response time and energy consumption of our task execution schemes for a different scenario 3. In this scenario, the total task workload (CPU cycles to process the task) is higher than in the previously considered scenario 2. More specifically, the central cloud is assumed to be more powerful than the collaborative node (i.e., neighboring robot). We observe that the host robot-neighboring robot based joint task execution is unable to meet the task deadline requirement. In addition, the cloudlet is unable to execute the task due to insufficient available memory. By contrast, the host robot-central cloud based joint task execution scheme is the best choice for this scenario as it offers a smaller task response time and energy consumption of host robot than its counterparts. The host robot-cloud based scheme shows a 38% and 11% higher task response time efficiency than the host robot-neighbor robot and host robot without offloading scheme and a 1% and 15% higher energy efficiency than the host robot-neighbor robot and host robot without offloading scheme, respectively.

In Figs. 3.5(a) and (b), we compare the performance of our collaborative and non-collaborative schemes with previously proposed minimum distance [12] and fixed assignment [89] based robot task execution schemes in a setting referred to as scenario 4. Note that these task offloading schemes examined only computation task for execution, while location dependent sensing sub-task was considered out of their scope. In this work, the considered H2R task consists of both sensing and computation sub-tasks, whereby location-dependent sensing sub-tasks are restricted to robots and location-independent computation sub-tasks can either be done by robots or offloaded to a collaborative node (central cloud/cloudlet/neighbor robot) for execution. For fair comparison, we consider only existing robot based full task (sensing and



(a) (b)

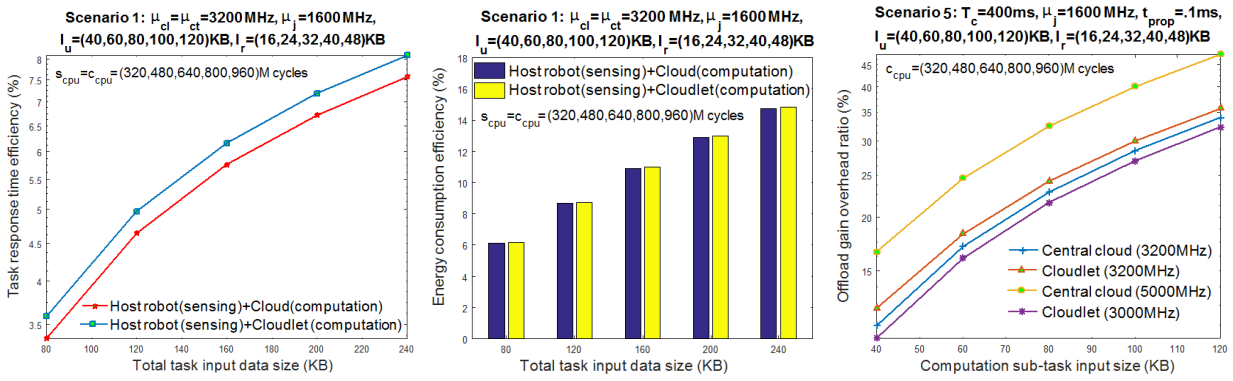
Figure 3.5: Average task response time and energy consumption comparison of our collaborative (host robot-central cloud, host robot-cloudlet, host robot-neighbor robot) and non-collaborative (host robot without offloading) schemes with existing schemes.

computation sub-task) execution schemes. In scenario 4, both central cloud and cloudlet are assumed to have the same CPU power. However, the neighbor robot CPU power is assumed to be smaller than that of the central cloud and cloudlet. Other parameters settings are shown in Figs. 3.5(a) and (b).

Figs. 3.5(a) and (b) clearly show that for scenario 4 the host robot-cloudlet based joint task execution achieves a significantly improved average task response time and energy consumption efficiency than the other schemes. For instance, for a typical task input size of 240KB, the host robot-cloudlet based joint execution shows a 30%, 12%, 1%, 25%, and 31% improved task response time with regard to the host robot-neighbor robot, host robot without offloading, host robot-cloud, minimum distance [12], and fixed assignment [89] based scheme, respectively. Moreover, for the assumed task input size of 240 KB, the host robot-cloudlet based joint scheme achieves a 2%, 18%, 1%, 28%, and 33% higher energy efficiency than the host robot-neighbor robot, host robot without offloading, host robot-cloud, minimum distance [12], and fixed assignment [89] based scheme, respectively.

Next, we investigate the total task response time and energy efficiency of collaborative schemes that satisfies the task deadline for scenario 1. The task response time efficiency of our collaborative full task execution scheme is defined as the ratio of the collaborative task response time gain $(t_j - t_{j,cn})$ and the task response time of non-collaborative (t_j) execution. The energy consumption efficiency of our collaborative full task execution scheme is defined as the ratio of the energy consumption gain of the host robot for collaborative execution $(e_j - e_{j,cn})$ and the energy consumption of the host robot for non-collaborative (e_j) scheme.

The host robot-central cloud and host robot-cloudlet based joint task execution gain over non-collaborative host robot task execution scheme are depicted in Figs. 3.6(a) and (b),



(a) (b) (c)

Figure 3.6: (a) Task response time efficiency versus total task input data size; (b) energy consumption efficiency versus total task input data size; (c) offload gain overhead ratio versus computation sub-task input data size.

respectively. Both figures clearly indicate that for scenario 1, the host robot-cloudlet based joint task execution achieves a superior task response time and energy efficiency than the host robot-central cloud based joint scheme. For instance, for a typical total task input size of 240 KB, the host robot-cloudlet based joint task execution shows an 8.75% improvement of task response time and a 14.98% improvement of energy efficiency than the host robot based non-collaborative task execution scheme. Hence, the host robot-central cloud based joint execution achieves a 7.81% decrease of task response time and a 14.72% increase of energy efficiency in comparison with the non-collaborative scheme. Further, in Fig. 3.6(c), the computation sub-task offload gain-overhead ratio of both central cloud and cloudlet execution are shown. The computation sub-task offload gain-overhead is defined as the ratio of the offload gain for collaborative node based computation sub-task execution ($t_c^j - t_c^{cn}$) and the offload overhead ($\overline{d}_{cn} + t_{ofl}^{cn}$) incurred by the communication protocols (see Eq. 3.14).

Importantly, we observe from the figure 3.6(c) that under the assumption that both the central cloud and cloudlet have same computation power, the cloudlet based computation sub-task execution achieves a higher offload gain than the central cloud. For instance, for a typical computation sub-task input data size of 120 KB, the offload gain overhead ratio of central cloud (3200 MHz) and cloudlet (3200 MHz) is 34% and 36%, respectively. However, if the computation power of the central cloud (5000 MHz) is assumed to be higher than that of the cloudlet (3000 MHz), the central cloud shows a much better offload gain (48%) than the cloudlet execution (32%).

Finally, we evaluate the end-to-end local (t_{local}) and non-local ($t_{non-local}$) task response time under different FiWi traffic loads, as shown in Figs. 3.7(a) and (b). Local (MU and robot for task execution are located under the same ONU-MPP) task response calculation

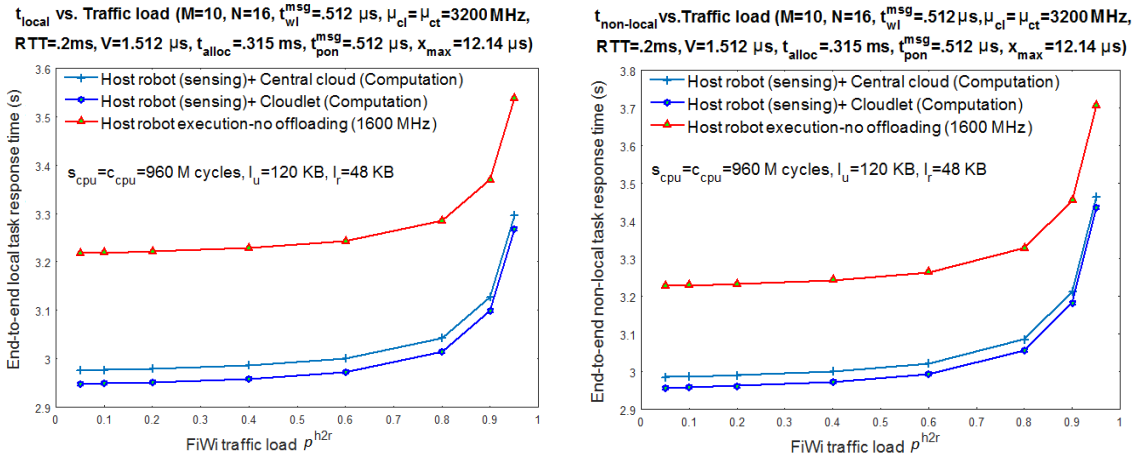


Figure 3.7: End-to-end local and non-local task response time variation versus FiWi traffic load.

accounts for the US frame transmission delay of an MU's task request transmission (see Section 3.4.4), task allocation delay for robot selection, time to reach the task location, sensing, and computation sub-task execution. Conversely, non-local (MU and robot for task execution are located under different ONU-MPPs) task response time calculation takes into account both US and DS frame transmission delay of an MU's task request transmission (see Section 3.4.4), robot selection delay for task allocation, required time for selected robots to reach the task location, sensing, and computation sub-task execution. Recall from Section 3.4.5 that for the collaborative task execution scheme, the non-local and local task response time is equal to $t_{non-local} = t_u + t_d + t_{j,cn}$ and $t_{local} = t_u + t_{j,cn}$, respectively. by contrast, for the non-collaborative host robot task execution scheme, the non-local and local task response time is equal to $t_{non-local} = t_u + t_d + t_j$ and $t_{local} = t_u + t_j$, respectively.

Note that, both local and non-local task response times increase for increasing traffic loads in our considered FiWi network scenario in Fig. 3.7(a) and (b). Both figures clearly indicate that the host robot-cloudlet based scheme provides an improved task response time than host robot-central cloud (2%) and host robot without offloading (10%) schemes. This is because the host robot-central cloud based scheme incurs a higher computation offloading delay than the host robot-cloudlet based execution. Moreover, the host robot based non-collaborative task execution experiences a much higher task response time than the alternate collaborative schemes. This result is expected given that the host robot is less powerful than the central cloud and cloudlet. We also note that the end-to-end local task response of all compared schemes (collaborative and non-collaborative) are lower than their non-local task response time. The reason behind this is that beside task execution delay, the calculation of $t_{non-local}$

involves both US and DS frame transmission delays for end-to-end task allocation, while t_{local} involves only the one-way US frame transmission delay.

3.6 Conclusions

Efficient task allocation among robots, computation offloading onto collaborative nodes, and adaptive resource allocation schemes represent key design challenges for reducing the end-to-end latency in advanced Tactile Internet H2R communications. In this chapter, we presented a collaborative computing enhanced task allocation mechanism that combines suitable host robot and collaborative node selection in integrated FiWi multi-robot networks.

To improve the energy efficiency of the selected host robot while satisfying a given task deadline, we investigated both host robot based non-collaborative and joint task execution schemes, in which the sensing sub-task is conducted by a suitable host robot and the computation sub-task is offloaded onto one of the collaborative nodes consisting of central cloud, cloudlets, and neighboring robots. In order to handle both conventional broadband and computation offloading traffic at the same time, we introduced a unified TDMA-based resource management scheme. Moreover, we developed an analytical framework to evaluate the performance of our proposed non-collaborative and collaborative task execution schemes in terms of task response time efficiency and energy efficiency of host robots. Unlike previous studies, we also analyzed the end-to-end local/non-local task response time for both collaborative and non-collaborative task execution schemes.

Our results provide insight into finding the optimal task execution scheme for a variety of use case scenarios with different task, robot, and collaborative node availability characteristics. The results of both collaborative/joint and non-collaborative task execution schemes demonstrate that for a typical task input size of 240 KB, the collaborative task execution scheme decreases the task response time by up to 8.75% and the energy consumption by up to 14.98% compared to the non-collaborative task execution scheme. The introduced collaborative computing based task allocation and resource management scheme represents a promising solution for enabling low-latency Tactile Internet applications.

Chapter 4

HART-centric Task Migration Scheme over FiWi Based Tactile Internet Infrastructures

4.1 Preamble

This chapter contains material extracted from the following paper:
[J4] M. Chowdhury, E. Steinbach, W. Kellerer, and M. Maier, "Context-Aware Task Migration for HART-Centric Collaboration over FiWi Based Tactile Internet Infrastructures," *IEEE Transactions on Parallel and Distributed Systems,* vol. 29, no. 6, pp. 1231-1246, June 2018.

4.2 Introduction

Mobile cloud computing (MCC) has emerged as a promising technology that allows mobile devices to offload part or all of their computation tasks onto resource-rich surrogates through a process known as *computation task offloading* [102]. However, computation task offloading onto remote cloud servers may not always improve the task execution latency and energy consumption of mobile devices due to the involved communication overhead. Alternatively, computation-intensive tasks may be offloaded onto so-called cloudlets located at the network edge in close proximity to mobile users (MUs). Recent studies on task offloading mainly focused on whether to offload computation-intensive tasks onto an infrastructure-based cloud (remote cloud [17], [74] or cloudlet [77]) or mobile ad-hoc cloud formed by nearby mobile devices [30] in order to reduce both task response time and energy consumption of mobile devices.

Taking the idea of task offloading a step further, *task migration* has emerged as a promising approach to improve the quality of experience (QoE) of MUs by minimizing their task execution time [19]. Task migration broadens the scope of conventional computation task

offloading by not only transferring the task from an MU onto the cloud, but also from one cloud server to another one for execution. In general, task migration between cloud servers is considered beneficial only if the anticipated task execution time at the secondary cloud server is smaller than that at the primary one [20]. Note, however, that task migration incurs an additional migration delay. Hence, for a given task migration gain and latency overhead, the question of how and where a task should migrate to is key. To answer this question, several migration decision criteria need to be considered such as the state of the current and tentative destination cloud servers, task properties, and task migration latency, among others.

Task migration has the potential to speed up the execution of tasks running not only on hand-held devices, e.g., smartphones, but also on commercially available remote-presence robots, which allow humans to see, hear, touch, and manipulate objects in places where they are not physically present. These remote-presence robots may be the precursor of an age of technological convergence, where important human tasks will be increasingly done by low-latency networked robots. This vision of real-time human-to-robot (H2R) interaction-centric applications gives rise to the so-called *Tactile Internet*, which has recently emerged as a new paradigm to remotely steer/control virtual and/or physical objects such as robots via the Internet [35]. Recently, we explored the performance gains obtained from unifying coverage-centric 4G LTE-Advanced (LTE-A) heterogeneous networks (HetNets) and capacity-centric fiber-wireless (FiWi) access networks based on data-centric Ethernet technologies with resulting fiber backhaul sharing and WiFi offloading capabilities for enabling the future Tactile Internet [9]. Importantly, we showed that a very low latency on the order of 1 ms and ultra-high reliability with an almost guaranteed FiWi network connectivity of MUs can be obtained in FiWi enhanced LTE-A HetNets. More recently, we advocated that multi-robot FiWi network infrastructures leveraging central cloud and decentralized cloudlet resources will be instrumental for ushering in low-latency Tactile Internet applications [5].

In this chapter, we build on our previous studies and extend their scope by investigating task migration for different types of task and cobot/agent in technically greater detail. Note that depending on the context-awareness of future Tactile Internet applications, tasks may be classified into two different categories. Specifically, a task may be either a *location-dependent physical* task (e.g., image capturing at a given physical location), a *location-independent cognitive* task (e.g., face recognition from a captured image, which might be offloaded for computation at a remote cloud or nearby cloudlet), or it may include both types of tasks (e.g., face recognition where the image was captured).

Another crucial aspect of the Tactile Internet we pay particular attention to in this chapter is the overarching goal that cobots should complement humans rather than substitute for them, giving rise to a cooperative and collaborative design approach known as *human-agent-robot*

teamwork (HART) [55]. HART differs from the traditional humans-are-better-at/machines-are-better-at (HABA/MABA) approach, which only divides up work between humans and machines without driving any symbiotic human-robot development in search for synergies. Conversely, with a HART-centric Tactile Internet design approach, humans and cobots with the support of central cloud and decentralized cloudlet resources together acting as intelligent multi-agent systems exploit the different characteristics of physical and cognitive tasks and jointly execute them by means of smart orchestration techniques. To render HART-centric task migration beneficial to MUs, however, context information about the task (e.g., task size, deadline, type), collaborative agent/cobot (e.g., availability, capability), user mobility, and migration latency needs to be taken into account properly.

The contributions of this chapter are as follows. We first introduce an integrated two-level cloud-cloudlet FiWi based Tactile Internet architecture for HART task execution. After describing the key features of physical vs. cognitive task and cobot vs. stand-alone robot types, we present a suitable HART-centric task migration scheme, taking different task (deadline, workload, data size) and collaborative node (availability, task processing speed, remaining energy) characteristics into account. Next, we develop a unified FiWi resource management scheme that is able to handle both traditional broadband and task migration data traffic at the same time. Finally, we present an analytical model to evaluate the performance of our proposed scheme in terms of end-to-end task execution delay, migration gain-overhead, deadline-miss ratio, task response time, and energy consumption efficiency, while paying close attention to its performance comparison for both with and without task migration. Note that the focus of this chapter is on the performance evaluation of the different task migration schemes for the execution of a single full HART task that includes both physical and cognitive sub-tasks. The problem of optimizing the performance of simultaneously executing multiple full HART tasks in a resource and time efficient manner is outside the scope of this chapter.

The remainder of the chapter is structured as follows. The state of the art and open challenges of task migration are discussed in Section 4.3. Section 4.4 describes FiWi based Tactile Internet infrastructures for HART-centric task migration in greater detail. In Section 4.5, we elaborate on the specific characteristics and key parameters of cobots and tasks. Section 4.6 describes our proposed context-aware HART-centric task migration scheme, whose performance is analyzed in Section 4.7. In Section 4.8, we present our obtained numerical results and findings. Section 4.9 concludes the chapter.

4.3 Task Migration: State of the Art and Open Challenges

Research in the area of task migration in the context of HART-centric Tactile Internet applications is still in its infancy. The authors of [57] showed that suitable cobot selection for task execution requests by humans is essential in order to achieve good performance by reducing the various latency components of a given task, e.g., task execution delay. The authors also emphasized that most of the existing work on suitable cobot selection focused on the involved cobots' task processing power or remaining energy for task migration. To render the HART-centric task migration process more effective, additional task properties (e.g., task deadline or type) and cobot properties (e.g., availability, skill, distance to task location, mobility, or minimum energy consumption) have to be taken into account for suitable cobot selection. Importantly, note that suitable cobot selection for task migration may not be sufficient to avoid task execution failures due to the constrained resources (e.g., task processing capabilities, storage, or remaining energy) of the selected cobot. To do so, however, cobots may overcome their limited resources by utilizing the ones of other collaborative HART members, e.g., cloud based agents. The resultant HART-centric task execution approach is also known as *collaborative computing*, where a resource-constrained cobot migrates its assigned task to another more powerful agent or cobot for execution [17], [59]. At present, only a few studies exist on collaborative task migration exploiting cloud based agents, e.g., cloud agent selection for task migration based on load prediction [21], service delay [22], distance [23], resource availability information (i.e., CPU speed and workload) [24]-[25], mobile user task result download location [19], and energy consumption [26]. Note that existing studies on task migration considered only the problem of task migration from an MU either to a suitable robot or to a cloud agent, rather than both. None of the existing studies has focused on the active participation/cooperation of all HART members, namely, MUs (humans), agents (central cloud/cloudlet), and collaborative robots (cobots), which is necessary for the proper HART task execution involving both physical and cognitive sub-tasks. Hence, existing studies cannot be directly applied to HART task execution.

Further, note that most of the aforementioned studies considered either infrastructure-based task migration, e.g., remote cloud and local cloudlet, or infrastructureless task migration onto local ad-hoc clouds comprising nearby cobots. These previous studies did not include important decision variables such as different task types (e.g., physical vs. cognitive task), task properties (e.g., total number of task arrivals, task input and output sizes, or task deadlines), collaborative node properties (e.g., availability, distance, or CPU speed), and human user mobility to reduce HART task execution delay and energy consumption. Another open

question is how to coordinate the HART-centric task migration from MUs to collaborative nodes (cobots and agents) and among collaborative nodes (cobot to agent as well as agent to agent). Similarly, the development of adaptive integrated FiWi resource management schemes in support of coexistent traditional triple-play traffic and data traffic stemming from HART-centric task migration remains an open research challenge. Furthermore, there is also a lack of an analytical framework for evaluating HART task execution performance in terms of task response time and energy consumption, while providing sufficient fault recovery via cellular and/or WLAN networks.

In this chapter, we aim at addressing some of the aforementioned open research challenges in the area of task migration. Specifically, we analyze the performance of our proposed HART-centric task migration scheme, considering both inter-agent (cloud to cloudlet and vice versa) and intra-agent (cloud to cloud and cloudlet to cloudlet) task migration in FiWi based Tactile Internet infrastructures. In addition, we compare the performance of the following three different task migration schemes: (*i*) no migration, (*ii*) cobot-to-cobot (c2c) migration, and (*iii*) cobot-to-agent (c2a) migration. Note that in the non-migration scheme, only the initially selected cobot locally executes an MU's requested task, which in turn may comprise physical and cognitive sub-tasks. Conversely, in the c2c migration scheme, the initially selected cobot executes the physical sub-task and migrates the cognitive sub-task to a suitable nearby cobot for execution. Whereas in the c2a task migration scheme, the initially selected cobot executes the physical sub-task and migrates the cognitive sub-task to a suitable agent (local cloudlet or remote cloud) for execution. Furthermore, to determine the optimal task migration scheme we investigate the following three types of c2a scheme using a number of HART-specific performance metrics: (*i*) cobot at a given task location to cloudlet that is near the task location, (*ii*) cobot to cloudlet that is near the location of the MU downloading the computation result, and finally (*iii*) cobot to remote cloud.

4.4 FiWi Based Tactile Internet Infrastructure for HART-Centric Task Migration

4.4.1 Network Architecture

In this section, we extend the generic FiWi enhanced LTE-A HetNet architecture introduced in [9] for enabling and coordinating HART-centric task migration, whereby humans, cobots, and agents actively participate in the joint task execution process. For convenience, we briefly review the salient features of the FiWi enhanced LTE-A HetNets architecture, which aimed at removing the traditional barriers between coverage-centric 4G mobile networks and capacity-centric FiWi broadband access networks based on low-cost data-centric optical fiber

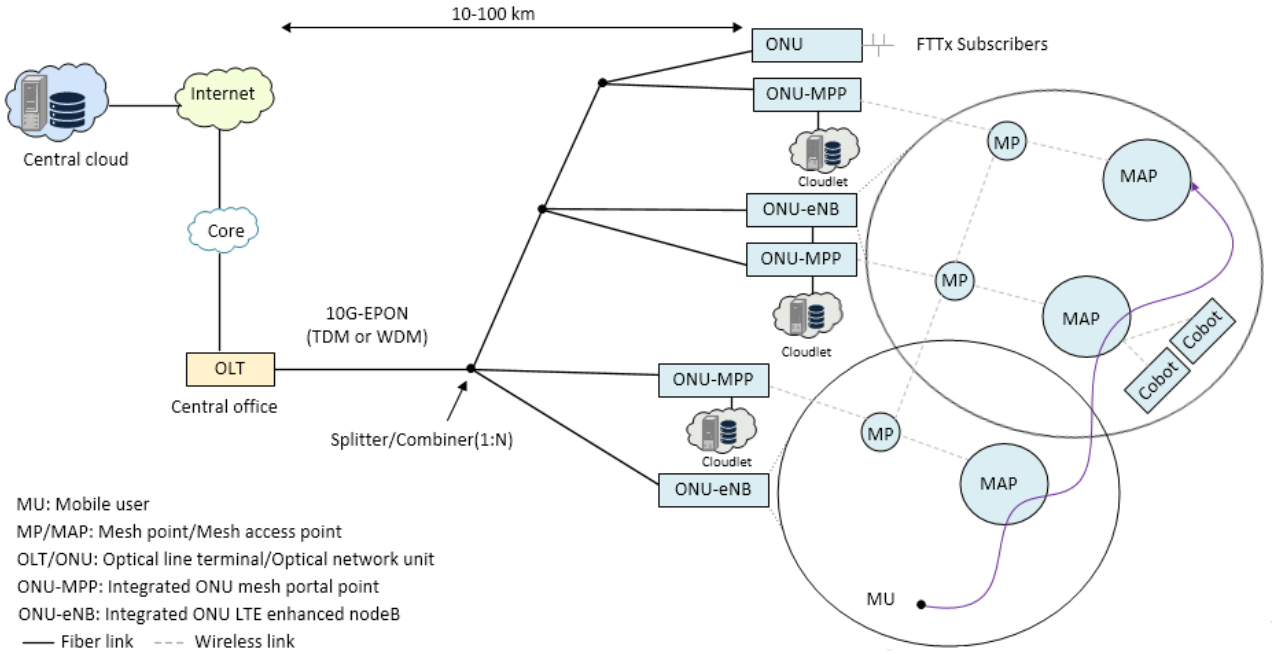


Figure 4.1: FiWi based Tactile Internet infrastructure based on embedded cloudlets, cobots, and human MUs for HART-centric task migration.

and wireless Ethernet technologies. It was shown that a very low latency on the order of 1 millisecond and ultra-high reliability can be achieved in unified FiWi enhanced LTE-A HetNets with resultant fiber backhaul sharing and WiFi offloading capabilities. By complementing fast evolving LTE-A HetNets with FiWi access networks, low-cost high-speed mobile data offloading is achievable in FiWi enhanced LTE-A HetNets using high-capacity fiber backhaul (e.g., IEEE 802.3av 10G-EPON) and Gigabit-class IEEE 802.11ac WLAN technologies. The interested reader is referred to [9] for further details on FiWi enhanced LTE-A HetNets.

Next, we describe our proposed network extensions in greater detail. Note that the generic FiWi enhanced LTE-A HetNets architecture proposed in [9] considered only human MUs. Unlike in our proposed architecture, important HART-centric architectural components such as cobots and agents were not studied in [9]. Furthermore, the authors of [9] concentrated on mobile data offloading rather than task migration, which is the main focus of this work.

As shown in Fig. 4.1, our proposed FiWi based Tactile Internet infrastructure consists of a time division multiplexing (TDM) or wavelength division multiplexing (WDM) IEEE 802.3av 10 Gb/s Ethernet Passive Optical Network (10G-EPON) with a fiber backhaul range of 10-100 km between the central optical line terminal (OLT) and remote optical network units (ONUs). The OLT collocated with the central office serves three different subsets of ONUs, which are connected through a 1:N optical splitter/combiner at the remote node. The first subset of ONUs provide FTTx services, e.g., fiber-to-the-home/business (FTTH/B) to a

single or multiple fixed wired subscribers. To interface with the WiFi mesh network (WMN) at the wireless front-end, the second subset of ONUs are equipped with a mesh portal point (MPP) and are henceforth referred to as ONU-MPPs, whereby mesh points (MPs) act as intermediate relay nodes between MPPs and mesh access points (MAPs). Each MAP serves both MUs and WiFi enabled cobots within its respective wireless coverage area. Note that the integrated ONU-MPP is realized by using so-called radio-and-fiber (R&F) technologies with medium access control (MAC) protocol translation taking place at the optical-wireless interface. To provide 4G cellular services to MUs, the third subset of ONUs are connected to an LTE enhanced nodeB (eNB) base station, giving rise to so-called ONU-eNB. All BSs together are assumed to provide ubiquitous wireless connectivity to MUs. For enabling direct communication between ONU-MPP and ONU-eNB, we also make use of so-called interconnection fiber links between a subset of selected pairs of neighboring ONU-MPP and ONU-eNB. The central cloud servers are connected to the OLT via dedicated fiber links. In addition, local cloudlets are connected via dedicated fiber links to ONU-MPPs at the edge of our FiWi based Tactile Internet infrastructure in order to provide cloud services in close proximity to nearby MUs and/or cobots. In general, we assume that only one cloudlet is attached to an ONU-MPP. However, multiple cloudlets may be connected to an ONU-MPP depending on the given number of arriving task requests, cloudlet capacity, and number of ONU-MAPs, among other network design parameters. For further details on cloudlet network planning and optimal placement of ONU-MPP/ONU-eNBs in cloudlet enhanced FiWi access networks we refer the interested reader to [52].

4.4.2 Mobility of Cobots and Human MUs

We assume that the WiFi mesh access points (MAPs) are randomly distributed according to a Poisson point process with density λ_{MAP} throughout the cellular coverage area (A_{cell}). Similarly, human MUs are assumed to be randomly distributed with density λ_{MU}. Further, we assume that each MAP has a circular coverage area $\hat{a}$ with radius r_{MAP} (i.e., $\hat{a} = \pi r_{MAP}^2$), in which m cobots are randomly distributed according to a homogeneous Poisson point process. Note that the cobots have limited mobility and can move at pedestrian speed only to a given physical task location that resides in the same MAP coverage area. In contrast, a human MU can move from one MAP or BS to another one according to a certain moving pattern based on her current position, speed, and moving direction. In doing so, a given human MU can send her task requests and receive the corresponding task results via the cellular and/or WiFi network along her trajectory, depending on whether she is inside or outside the coverage area of a traversed MAP. For modeling the WiFi connectivity of MUs, we adopt the mobility model based on recent smartphone traces in [9], whereby the complementary cumulative distribution

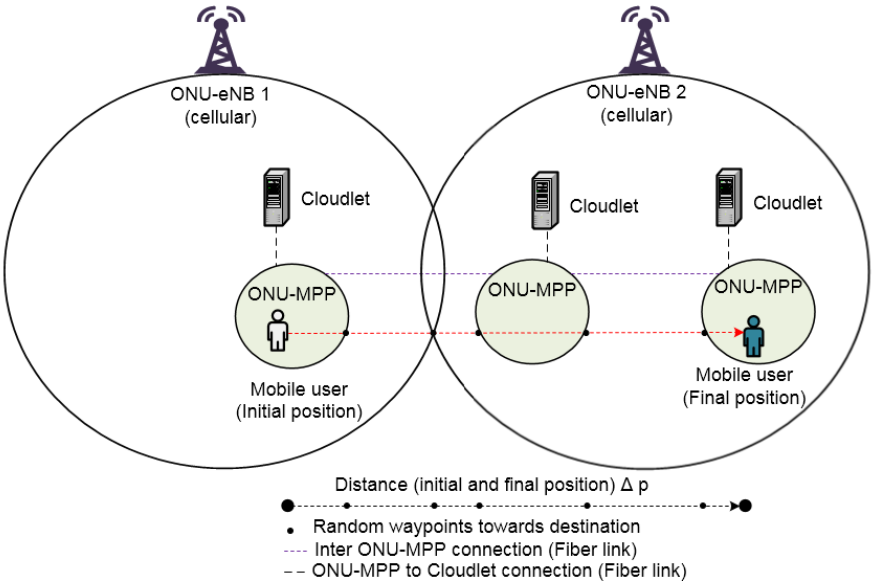


Figure 4.2: Random waypoint (RWP) model of human MUs for predicting location of task request transmission and task result reception.

function (CCDF) of both WiFi connection time (time period an MU stays within a given MAP coverage area) and WiFi interconnection time (time period after an MU leaves an MAP coverage area until she returns or enters another MAP coverage area) was shown to fit a truncated Pareto distribution.

For predicting the location of a given MU's task request transmission and task result reception, we adopt the widely used random waypoint (RWP) model [102]. As shown in Fig. 4.2, in our considered RWP model a given MU traverses several waypoints at different predefined speeds, whereby the initial position of the MU is chosen randomly. For a given set of n different speeds v_i at time instants t_i along the MU's trajectory, the average speed between the randomly selected initial point and the final point (i.e., location of task result reception) equals $v_u = \frac{\sum_{i=1}^{n} v_i t_i}{\sum_{i=1}^{n} t_i}$. Accordingly, the predicted location of the MU's task result reception is given by

$$p_{u+\Delta t} = p_u + v_u \cdot \Delta t, \tag{4.1}$$

where $p_{u+\Delta t}$, p_u, v_u, and Δt denote the MU's final position, initial position, average speed, and required time to travel from the initial to the final position, respectively. Note that the distance traveled by the MU between her initial and final positions is equal to $\Delta p = p_{u+\Delta t}$ - $p_u = v_u \cdot \Delta t$ (see also Fig. 4.2). Then, based on MUs task request transmission time delay (t_{req}), MUs required travel time Δt, and task result reception delay ($t^{rx}_{\theta \to mu}$), the requested task execution deadline (t_d) can be calculated as follows: $t_d = t_{req} + t^{rx}_{\theta \to mu} + \Delta t$, where subscript θ stands for k (initially selected cobot), a (agent), and k^* (nearby cobot), respectively.

Table 4.1: Cobots vs. robots

Criteria	**Cobots**	**Robots**
Definition	Collaborative robots (cobots) can sense the environment around them and have the ability to work with other cobots	Traditional stand-alone robots cannot sense the environment around them
Connectivity	Connected with other cobots and humans through infrastructure (e.g., WiFi)	Not connected with other robots/humans
Context awareness	Avoid task execution failures through task migration to other cobot/agent and machine-learning capability with strict QoS support	Suffer from task execution failures due to lack of collaboration
Flexibility	Programmable and able to learn independently from environment	Traditional robots require manual support
Task type	Accomplish multiple types of task (e.g., household work, teaching, healthcare, and entertainment)	Only execute specific types of manual task (e.g., car manufacturing)
Movement	Cobots can move anywhere with advanced navigation, obstacle avoidance, and path planning capabilities	Restricted to a fixed place without any movement capabilities (e.g., industrial robot enclosed in safety cage)
Repetitive task	Execute both mobile and non-mobile repetitive tasks	Execute only non-mobile repetitive tasks
Task location	Cobots can execute both location-independent cognitive and location-dependent manual task	Traditional robots can only execute location-dependent manual task in industry
Safety	Cobots can be controlled or programmed to protect humans during possible encounter in a shared workspace (e.g., YUMI cobots)	Humans can be injured by traditional robots due to lack of intelligence/control
Key performance indicators	Task migration gain-overhead ratio, energy efficiency, task response time with failure recovery, task processing speed, and mobility	Manual task response time, average utilization, and deadline miss ratio

4.5 Cobots and Tasks: Characteristics and Assumptions

In this section, we first briefly elaborate on the main characteristics of cobots in comparison with traditional robots and shed some light on the different types of cognitive and physical tasks. Subsequently, we introduce several parameters to formally define the various types of cobot and task, which will then be used to describe our proposed context-aware HART-centric

task migration scheme in the next section.

4.5.1 Characteristics

Table 4.1 highlights the major differences between cobots and traditional robots. The comparison between cobots and robots is best made according to their *ability to act* and their *ability to learn*. In the first category, we look at their ability to perform different types of task, i.e., physical vs. cognitive tasks. Typically, a physical task is a manual repetitive task that can be executed at a specific location (e.g., manipulation of a given physical object), whereas a cognitive task involves location-independent decision making or computation that might be offloaded onto remote entities such as cloud servers or nearby cloudlets. We observe from Table 4.1 that traditional industrial robots are typically standalone entities that perform only a single type of stationary manual (physical) task (e.g., manufacturing). Conversely, advanced cobots are in general programmable, mobile, and able to execute multiple types of task, including both physical (e.g., image capturing, delivery service) and cognitive tasks (e.g., intrusion detection from captured image). In the second category, there exist differences between cobots and robots based on their ability to learn from their environments, most notably in terms of their repetitive task execution as well as context awareness and learning capabilities. Unlike advanced cobots, the state of knowledge of traditional standalone robots cannot grow based on new experiences or changing conditions. Thus, traditional stand-alone robots typically can only execute repetitive manual tasks and also suffer from possible task execution failures due to their inability to migrate the interrupted task to other robots. In contrast, cobots are able to execute both repetitive and context-aware tasks (e.g., mobility, machine-learning) by monitoring their performance and making adjustments to what they observe and experimenting with other possibilities that might perform better. This flexibility enables cobots to avoid task execution failures via task migration to collaborative cobots or agents.

Table 4.2 compares the aforementioned cognitive and physical tasks in greater detail. One of the major differences is the fact that location-dependent physical tasks that require movement capability can only be done by cobots, whereas location-independent cognitive tasks that require only computation and/or storage capability rather than physical presence can be done by cobots and in particular agents such as central cloud and cloudlet with typically more powerful computation/storage resources. Another important aspect of cognitive tasks lies in the fact that most of them are rather non-repetitive and therefore much harder to be automated. Furthermore, to provide MUs with strict QoS support for non-repetitive cognitive tasks, collaborative cobots/agents may benefit from advanced artificial intelligence (AI) and machine learning capabilities for task migration. Note that several of the key performance

Table 4.2: Cognitive vs. physical tasks

Criteria	**Cognitive Tasks**	**Physical Tasks**
Definition	Deal with perception, interaction, planning, memory, learning, and reasoning phenomena	Follow specific instructions (manual), do not require sophisticated judgment
Human vs. machine	Humans are better at performing cognitive tasks, e.g., intelligent decision making	Machines are better at executing repetitive and routine tasks, e.g., precise physical movement
General tasks	Analyze numbers, digest words and images, perform digital tasks	Remote operation in specific places (e.g., manipulation of objects in hostile environment)
HART-centric tasks	Both location-dependent and -independent decision making (e.g., face detection)	Only location-dependent manual operation such as heavy machinery transport or image capturing
Cobot vs. agent	HART-centric cognitive task can be done by both cobot and agent (cloud, cloudlet)	HART-centric physical task is restricted to only cobot located in given task area
AI/machine learning capability	Require artificial intelligence (AI)/machine learning capability for non-repetitive task	Do not require AI/machine learning capability for repetitive manual task
Predictability and automation	Non-repetitive cognitive jobs are non-predictable and harder to automate (e.g., financial analysis, intrusion detection)	Physical (manual) repetitive jobs are predictable and easy to automate (e.g., assembly line jobs)
Key performance indicators	Deadline miss ratio, task migration latency, and communication-computation ratio (CCR)	Response time for moving and processing task, task blocking probability

indicators listed at the bottom of Tables 4.1 and 4.2 will be investigated in our analysis below in Section 4.7.

4.5.2 Assumptions

Our HART-centric task considered for migration includes both physical (image capturing at task location) and cognitive sub-tasks (image recognition). More specifically, we describe a HART-centric task δ_i by using the following notation: $\delta_i = (u_i, s_i, s_o, t_d, e_r, tl_i, wl_i, mt)$, where u_i is the requested task type, s_i denotes the total task input data size $s_i = s_i^p + s_i^c$ that includes both physical (s_i^p) and cognitive (s_i^c) sub-task input data (given in megabytes), s_o is the task output data size ($s_o = s_o^p + s_o^c$), e_r is the required energy to process the requested task (in Watt), tl_i represents the two-dimensional location of the task (x_i, y_i), $wl_i = wl_p + wl_c$ is the task workload or amount of CPU cycles (in million instructions) required to process the

physical sub-task (wl_p) and cognitive sub-task (wl_c), $t_d = t_d^p + t_d^c$ is the total task deadline (in seconds), and mt denotes the MU's trajectory given by $mt = (p_u, t_u, v_u)$, whereby p_u is the initial position where the MU pauses for t_u seconds before moving onwards to the next waypoint at speed v_u. Hence, we assume that the selection of a suitable agent/cobot for each HART task request arriving at the ONU-MPP (task location) is done in a first-come-first-served (FCFS) manner. Note that an available cobot/agent is assumed to be able to perform only one HART task at any given time. Furthermore, we assume that the output of a physical sub-task (captured image at task location) is the input of the corresponding cognitive sub-task (image recognition).

Moreover, we assume that a cobot executes both physical and cognitive sub-tasks by taking its movement and computation capability into account. Accordingly, we assume that cobots are heterogeneous using the following notation: $m_k = (\alpha_k, v_m, v_s, e_k, pl_k)$, where α_k, v_m, and v_s denote the cobot type (ability to process a specific task), moving speed, and task processing speed, respectively. Further, e_k is the remaining energy and pl_k is the two-dimensional location of the cobot (x_k, y_k).

Finally, we assume that an agent (remote cloud or local cloudlet) can execute only location-independent cognitive sub-tasks for cobots and human MUs. Let us use the following notation: $cl_a = (v_a, \sigma_a)$, where v_a and σ_a denote the CPU capacity and availability of the agent, respectively.

4.6 Context-Aware Task Migration Scheme

Before describing our task migration algorithm in more detail, let us first consider a couple of illustrative examples to demonstrate our proposed method. Consider an MU who would like to obtain information about a painting currently at display in a museum or temporary exhibition. To do so, the MU sends a HART task request (capturing image of painting and recognizing the authenticity of the captured image) to the ONU-MPP at the corresponding task location. After receiving the HART task request, the ONU-MPP at the task location assigns a suitable cobot to perform the HART task by conducting the physical sub-task (moving to task location and capturing image at task location) and subsequently assigning the cognitive sub-task (recognition of captured image) to a suitable agent (cloudlet/central cloud). The agent then performs the cognitive sub-task and sends the cognitive sub-task result (authenticated image information) back to the MU. Another practical example would be the transport and delivery of a product such as pizza delivery (physical sub-task) and confirming the authenticity of the intended recipient by means of speech and face recognition (cognitive sub-task). To perform the MU requested HART task that includes both physical and cognitive sub-task in an time efficient manner, our proposed context-aware HART-centric task migration algorithm, which

performs the selection of both suitable cobot and agent, comprises the following five steps (see also Algorithm 4):

Step 1: A given MU sends a HART task request message during her assigned upstream (US) transmission subslot. The MU's task request message contains the following information: task location (tl_i), task type (u_i), required energy for task execution (e_r), task workload (wl_i), MUs average speed (v_u), initial location (p_u), task result download location ($p_{u+\Delta t}$) of MU, and task deadline (t_d).

The ONU-MPP/ONU-eNB associated with the MU receives the task request frame and forwards it to the OLT in the upstream direction. Subsequently, the OLT broadcasts the MU's task request message to all ONUs. The ONU-MPP serving the corresponding task location processes the task request message and allocates the task to a suitable cobot for execution.

Step 2: For suitable cobot selection, the ONU-MPP transmits the task request message to all cobots within its coverage area. Upon reception, each associated cobot sends a task response message to the ONU-MPP, which includes the following information about the cobot: availability (α_k), location (pl_k), remaining energy (e_k), and precalculated task response time (t_k). Next, the ONU-MPP selects a suitable cobot for each task by taking the following decision variables into account: task deadline ($t_d \geq t_k$), remaining energy threshold ($e_k \geq e_r$), and minimum task response time (t_k). Note that each task consists of two sub-tasks: a location-dependent physical (e.g., image capturing) sub-task and a location-independent cognitive sub-task (e.g., image recognition from captured image). Clearly, the cognitive sub-task is executed after the physical sub-task. Hence, the cognitive sub-task may be executed by the selected cobot itself or may be migrated to a suitable agent (cloud/cloudlet) for execution. Note that cognitive sub-task migration is done if either the cobot suffers from a failure or the agent is able to reduce the total task response time.

Step 3: To select a suitable agent for executing the cognitive sub-task, the ONU-MPP nearest to the corresponding task location checks the availability (σ_a) and cognitive sub-task response time (t_a^c) of all agents. If more than one agent satisfies the cognitive sub-task deadline (t_d^c) and availability criteria, a suitable agent is selected based on the minimum cognitive sub-task response time (t_a^c). After selecting a suitable agent, the cobot that performs the physical sub-task (wl_p) migrates the cognitive sub-task input data (s_i^c) to the selected agent for processing.

Step 4: Note that in the event of an agent failure or unavailability, a cognitive sub-task migration takes place from the failing/unavailable agent to another intact agent/nearby cobot in order to improve the overall task response time.

Algorithm 4 Task migration algorithm

Considerations: Number of arrived task request (δ) and cobots (m), a set of ONU-MPP (N) and agent (z), cobot type (α_k), task type (u_i), remaining energy cobot (e_k), required energy for full task (e_r) and agent cognitive sub-task (e_a^c), full (t_d) and cognitive sub-task deadline (t_d^c), cobot response time for full task (t_k) and cognitive sub-task (t_k^c), cognitive sub-task response time for cloud (t_{cl}^c), cloudlet near physical sub-task location ($t_{ct,i}^c$) and result download location ($t_{ct,d}^c$), and selected agent (t_a^c)

1: **for** each $n_j \in N$ **do**
2: **for** each arrived task request $i \in \delta$ **do**
3: **for** each cobot $k \in m$ **do**
4: **if** $(\alpha_k == u_i)$ & $(e_k \geq e_r)$ & $(t_k \leq t_d)$ **then**
5: Select available cobot (k) with minimum t_k
6: **if** $t_k^c < t_a^c$ & $e_a^c \geq e_k^c$ & $t_k^c \leq t_d^c$ **then**
7: Selected cobot executes full task (wl_i)
8: **else**
9: Execute only physical sub-task (wl_p)
10: Go to step 11 for cognitive sub-task migration (wl_c) to suitable agent $(a \in z)$
11: **if** $t_{ct,i}^c \geq t_{ct,d}^c > t_{cl}^c$ & $t_{cl}^c \leq t_d^c$ **then**
12: Select available central cloud server (cl) as agent (a) for cognitive sub-task migration
13: **else if** $t_{cl}^c \geq t_{ct,d}^c > t_{ct,i}^c$ & $t_{ct,i}^c \leq t_d^c$ & $a_{ct,i} \neq a_{ct,d}$ **then**
14: Select available cloudlet near physical sub-task location (ct, i) as suitable agent (a)
15: **else if** $t_{cl}^c \geq t_{ct,i}^c > t_{ct,d}^c$ & $t_{ct,d}^c \leq t_d^c$ & $a_{ct,i} \neq a_{ct,d}$ **then**
16: Select available cloudlet near MUs result download location $(a_{ct,d})$ as suitable agent
17: **end if**
18: The selected agent receives other agent information during cognitive sub-task execution
19: **if** the selected agent finds other suitable agent with lower task response time t_a^c **then**
20: Migrates the cognitive sub-task (remaining) to new agent with minimum t_a^c
21: **if** the selected suitable agent (a) can not execute the assigned cognitive sub-task **then**
22: Go to step 10
23: **else**
24: Go to step 1
25: **end if**
26: **end if**
27: **end if**
28: **end if**
29: **end for**
30: **end for**
31: **end for**

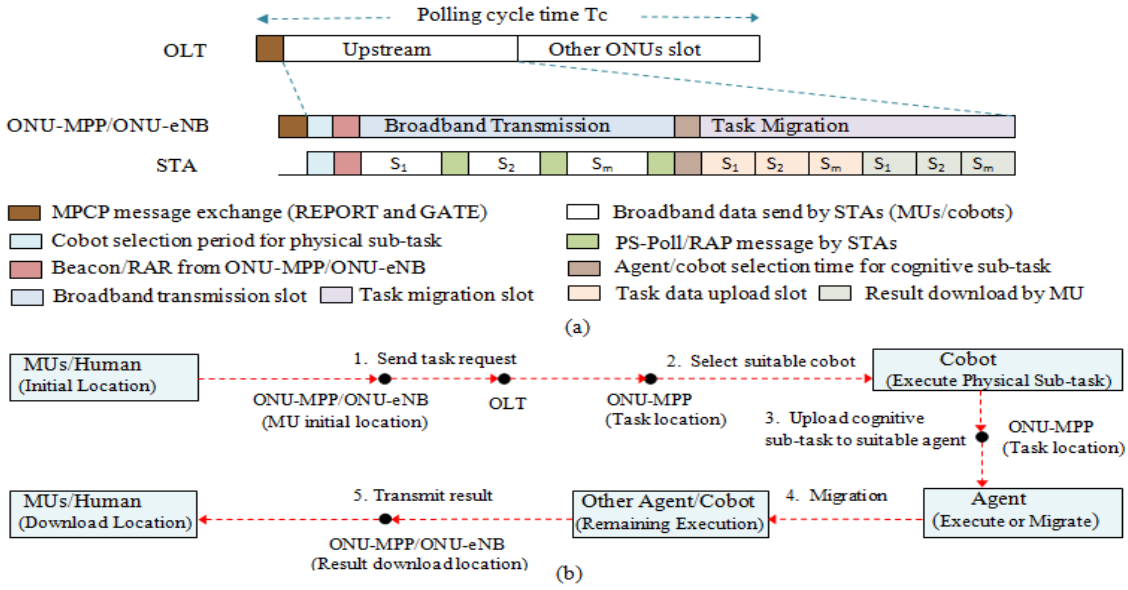


Figure 4.3: HART-centric task migration: (a) timing structure and (b) operational steps.

Step 5: Finally, after executing the cognitive sub-task (wl_c), the selected agent sends the processed data (s_o^c) to the MU's predicted result download location ($p_{u+\Delta t}$), as depicted in Fig. 4.3(b).

Fig. 4.3(a) depicts the signaling and timing structure of our proposed polling-based resource management scheme, which operates as follows. In the optical fiber backhaul, the OLT allocates an US transmission opportunity to each ONU-MPP/ONU-eNB by exchanging IEEE 802.3ah multipoint control protocol (MPCP) messages (`REPORT` and `GATE`) and broadcasts downstream (DS) frames to all ONU-MPPs/ONU-eNBs. The `REPORT` message is used by each ONU-MPP/ONU-eNB to report its current US bandwidth demand to the OLT. Upon reception, the OLT transmits a `GATE` message to inform the ONU-MPP/ONU-eNB about its granted US transmission slot. In the wireless front-end, ONU-MPPs and ONU-eNBs allocate US transmission opportunities to their associated users (MUs/cobots) via IEEE 802.11 WLAN `Beacon`/`PS-Poll` messages and LTE-A random access preamble `(RAP)`/random access response `(RAR)` messages, respectively [103].

After receiving the `GATE` message from the OLT, the corresponding ONU-MPP/ONU-eNB of a given task location first selects a suitable cobot for each task request, divides its allocated US bandwidth into subslots among its associated users (MUs/cobots), and broadcasts a `Beacon`/`RAR` frame to them. The task request message arriving at the ONU-MPP/ONU-eNB is sent by MUs during their previous polling cycle's broadband time subslots. Moreover, the broadcast `Beacon`/`RAR` frame contains the associated users' US transmission map, i.e., subslot start time and duration. Each associated user sends its US transmission subslot request to the ONU-MPP/ONU-eNB by using an extended `PS-Poll`/`RAP` frame, which contains an extra

migration flag bit (0 or 1) to notify the ONU-MPP/ONU-eNB about its task migration subslot request.

Prior to starting task migration, the corresponding ONU-MPP/ONU-eNB at a given task location selects a suitable agent for cognitive sub-task migration via DHCP protocol messages [104] (`Discover-Offer-Request-Ack`) between ONU-MPP and agent node (cloud/cloudlet). The selected cobot that executes the physical sub-task then migrates the cognitive sub-task input data to the selected agent for execution during its task upload subslot. In the opposite direction, after receiving the cognitive sub-task result data from the selected agent, the ONU-MPP/ONU-eNB at the MU's result download location transfers the cognitive sub-task result back to the MU during the corresponding result download subslot.

4.7 Analysis

In this section, we investigate the performance of our context-aware HART-centric task migration scheme in terms of a variety of key performance indicators.

4.7.1 Polling Cycle Time and Task Migration Subslot

We model the polling system of Fig. 4.3 as an M/G/1 queueing system with reservations and vacations. Let N denote the number of ONUs, whereby each ONU provides service to M associated users. More specifically, each ONU-MPP/ONU-eNB serves (i.e., broadband and migration) FiWi traffic of a given user during her assigned timeslot. We assume Poisson distributed FiWi traffic with mean arrival rate $\overline{\lambda}$. Hence, the aggregate FiWi traffic load equals $\rho_t{=}\overline{\lambda}X$, where X denotes the average service time. Furthermore, each ONU-MPP/ONU-eNB divides its polling cycle into data, reservation ($R = t_{wl}^{msg} + t_g$), and vacation intervals, i.e., $V = (N-1)t_{sl}$. Note that the non-data traffic time within T_c is denoted by $(1-\rho_t)$ and is equal to $N(MR + t_{as} + t_{cs} + t_{pon}^{msg} + t_{wl}^{msg})$. Thus, T_c is obtained as

$$T_c = \frac{N(MR + t_{as} + t_{cs} + t_{pon}^{msg} + t_{wl}^{msg})}{1-\rho_t}, \tag{4.2}$$

where t_{as} and t_{cs} represent the agent and cobot selection time, respectively.

For M users with an US transmission opportunity during T_c, each ONU's timeslot duration is equal to $t_{sl} = T_c/(N \cdot M \cdot \rho_t)$, where ρ_t denotes the FiWi traffic load ($\rho_t \leq 1$), including both broadband traffic load (ρ_m) and task migration traffic load (ρ_c). Similarly, timeslot t_{sl} includes both broadband ($t_{m,sl}$) and task migration ($t_{c,sl}$) along with the time needed for cobot (t_{cs}) and agent (t_{as}) selection. Hence, the broadband (t_{sl}^m) and task migration ($t_{sl}^c = t_{sl}^u + t_{sl}^d$) subslot duration of an associated user equals $t_{sl}^m = \frac{\rho_m \cdot t_{m,sl}}{M}$ and $t_{sl}^c = \frac{\rho_c \cdot t_{c,sl}}{M}$, respectively, where t_{sl}^u and t_{sl}^d denote the task upload and result download duration during a polling cycle.

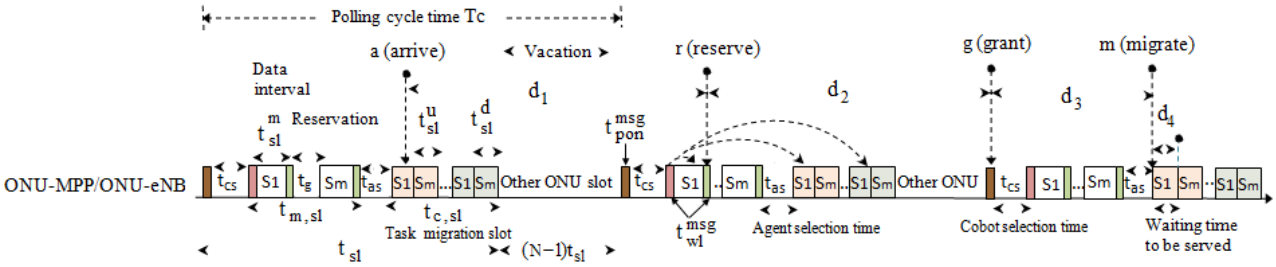


Figure 4.4: Task migration packet delay components.

4.7.2 Task Migration Packet Delay

As shown in Fig. 4.4, an US task migration packet experiences four different delay components during migration. The first delay component d_1 is the time interval between task migration packet arrival (a) and transmission of the task migration subslot reservation (r) request (`PS-Poll`). Thus, we have

$$d_1 = (M-1)t_{sl}^c + (N-1)t_{sl} + t_{pon}^{msg} + t_{cs} + t_{wl}^{msg} + t_{sl}^m, \tag{4.3}$$

where t_{sl}^c denotes the task migration subslot duration.

The second delay component (d_2) is the time interval between transmission of the resource reservation request (r) and reception of a grant (g) message (`Beacon/RAR`) for the task migration subslot. Thus, d_2 is given by

$$d_2 = (M-1)t_{sl}^m + t_{as} + Mt_{sl}^c + (N-1)t_{sl} + t_{pon}^{msg}. \tag{4.4}$$

The third delay component is the time that elapses between the received grant message (g) and transmission time of the corresponding task migration data packet (m). It is given by $d_3 = t_{cs} + t_{wl}^{msg} + Mt_{sl}^m + t_{as}$.

Finally, the fourth delay component (d_4) is the average waiting time of a migrated packet. Note that d_4 includes both queueing (d_{qt}) and service time ($d_{st} = \frac{1}{\mu}$) at the corresponding collaborative node. Hence, we have

$$d_4 = d_{st} + d_{qt} = \frac{1}{\mu} + \frac{C(\omega,\tau)}{\omega\mu - \lambda}, \tag{4.5}$$

where ω, μ, and λ represent the number of servers at the collaborative node, its service rate, and task arrival rate per server, respectively. $C(\omega,\tau)$ denotes the well-known Erlang-C formula given by

$$C(\omega,\tau) = \frac{\left(\frac{(\omega\tau)^\omega}{\omega!}\right)\left(\frac{1}{1-\tau}\right)}{\sum_{l=0}^{\omega-1}\frac{(\omega\tau)^l}{l!} + \left(\frac{(\omega\tau)^\omega}{\omega!}\right)\left(\frac{1}{1-\tau}\right)}. \tag{4.6}$$

Summing up all four delay components and considering that $d_1 + d_2 + d_3 \leq 2T_c$ yields the mean task migration packet delay d_m as follows:

$$d_m = 2T_c + \frac{1}{\mu} + \frac{C(\omega, \tau)}{\omega\mu - \lambda}. \tag{4.7}$$

4.7.3 Task Response Time

Given that in general a task consists of physical and cognitive sub-tasks, we analyze the task response time for execution with and without task migration.

4.7.3.1 Task Execution Without Migration

In this scenario, the initially selected cobot executes the full task (i.e., both physical and cognitive sub-tasks) and then transfers the task result to the MU. Thus, the task response time t_k is given by

$$t_k = t_k^f + t_k^{rx} = t_k^p + t_k^c + t_k^{rx}, \tag{4.8}$$

where t_k^p and t_k^c denote the cobot's physical and cognitive sub-task processing time, respectively, and t_k^{rx} is the required time to transfer the task result from the cobot to the MU. Hence, the physical sub-task response time t_k^p of a cobot, including both the time t_r^p to reach a task location and time t_{ex}^p to process the physical task workload, equals $t_k^p = \frac{d_i^k}{v_m} + \frac{wl_p}{v_s}$, where wl_p, v_m, and v_s represent the physical sub-task work load, moving and processing speed of the cobot, respectively; d_i^k denotes the distance between the cobot (x_k, y_k) and the corresponding task location (x_i, y_i), which is equal to the Euclidean distance $d_i^k = \sqrt{(x_i - x_k)^2 + (y_i - y_k)^2}$. Further, the cognitive sub-task response time of the cobot equals $t_k^c = wl_c/v_s$, where wl_c is the cognitive sub-task workload. Thus, the result transfer delay (t_k^{rx}) is given by

$$t_k^{rx} = \frac{s_o^c}{b_{wl}^k} + h_{o\to\bar{o}}^{rx} \cdot \frac{s_o^c}{b_{o\to\bar{o}}} + \frac{s_o^c}{b_{wl}^{mu}} + t_{k\to mu}^{prop}, \tag{4.9}$$

where s_o^c is the task output data size, $t_{k\to mu}^{prop}$ is the propagation delay of the task result transfer process, and $h_{o\to\bar{o}}^{rx}$ is the hop distance between ONU-MPP/ONU-eNB at the task and result download location. Further, b_{wl}^k, $b_{o\to\bar{o}}$, and b_{wl}^{mu} denote the transmission capacity of the link between cobot and ONU-MPP at the task location (b_{wl}^k=b_{wl}), ONU-MPP at the task location to ONU-MPP/ONU-eNB at the result download location ($b_{o\to\bar{o}}$=b_{fl}), and ONU-MPP/ONU-eNB at the result download location to the involved MU ($b_{wl}^{mu} = max\{b_{wl}, b_{wl}^*\}$), respectively.

4.7.3.2 Task Execution With Migration

Next, let us consider task execution with migration for the following two different cases:

• **Case 1 – $c2a$ and $c2c$ migration:** Recall that with $c2a$ task migration, the selected cobot executes only the physical sub-task while the cognitive sub-task is migrated to a suitable agent, i.e., cloud/cloudlet near the task location or result download location. The full task response time of $c2a$ task migration is equal to $t_{k,a} = t_k^p + t_a^c$, where t_k^p and t_a^c represent the cobot's and agent's physical and cognitive sub-task response time, respectively. Further, note that t_a^c is given by $t_a^c = t_{k\to a}^{tx} + t_a^{ex} + t_{a\to mu}^{rx}$, where $t_{k\to a}^{tx}$, t_a^{ex}, and $t_{a\to mu}^{rx}$ denote the cognitive sub-task upload delay (cobot to agent), processing time at agent ($t_a^{ex} = \frac{wl_c}{v_a}$), and result reception delay (agent to MU), respectively. For migration to local cloudlet ($a = ct$) and remote cloud ($a = cl$) migration, $t_{k\to a}^{tx}$ is computed as follows:

$$t_{k\to a}^{tx} = \begin{cases} \frac{s_i^c}{b_{wl}^k} + h_{o\to\bar{o}}^{tx} \cdot \frac{s_i^c}{b_{o\to\bar{o}}} + \frac{s_i^c}{b_{fl}} + t_{k\to a}^{prop} & \text{if } a = ct, \\ \frac{s_i^c}{b_{wl}^k} + \frac{2s_i^c}{b_{fl}} + t_{k\to a}^{prop} & \text{if } a = cl, \end{cases} \tag{4.10}$$

where subscript $a = ct$ stands for ct, i and ct, d for cloudlet near task location and result download location, respectively, $h_{o\to\bar{o}}^{tx}$ denotes the hop distance between ONU-MPP near the cobot's and agent's locations, and $t_{k\to a}^{prop}$ is the propagation delay incurred during the cognitive sub-task upload process. Hence, the cognitive sub-task result download delay $t_{a\to mu}^{rx}$ for cloudlet/cloud migration is obtained as

$$t_{a\to mu}^{rx} = \begin{cases} \frac{s_o^c}{b_{fl}} + h_{o\to\bar{o}}^{rx} \cdot \frac{s_o^c}{b_{o\to\bar{o}}} + \frac{s_o^c}{b_{wl}^{mu}} + t_{a\to mu}^{prop} & \text{if } a = ct, \\ \frac{2s_o^c}{b_{fl}} + \frac{s_o^c}{b_{wl}^{mu}} + t_{a\to mu}^{prop} & \text{if } a = cl, \end{cases} \tag{4.11}$$

where $t_{a\to mu}^{prop}$ is the propagation delay incurred during the result download process to the involved MU.

Conversely, with $c2c$ migration, the initially selected cobot and a nearby cobot perform the physical and cognitive sub-tasks, respectively. Thus, the total task response time t_{k,k^*} is given by $t_{k,k^*} = t_k^p + t_{k^*}^c$, where t_k^p and $t_{k^*}^c$ denote the primary cobot's (k) physical and nearby cobot's (k^*) cognitive sub-task response time. Note that $t_{k^*}^c$ is given by

$$t_{k^*}^c = t_{k\to k^*}^{tx} + t_{k^*}^{ex} + t_{k^*\to mu}^{rx}, \tag{4.12}$$

where $t_{k\to k^*}^{tx}$, $t_{k^*}^{ex}$, $t_{k^*\to mu}^{rx}$ represent the cognitive sub-task upload delay (primary cobot to nearby cobot), nearby cobot's cognitive sub-task processing time ($t_{k^*}^{ex} = \frac{wl_c}{v_s^*}$), and result download delay (secondary cobot to MU), respectively. For $c2c$ migration, $t_{k\to k^*}^{tx}$ and $t_{k^*\to mu}^{rx}$ are given by

$$t_{k\to k^*}^{tx} = \frac{s_i^c}{b_{wl}^k} + \frac{s_i^c}{b_{wl}^{k^*}} + t_{k\to k^*}^{prop}, \tag{4.13}$$

$$t^{rx}_{k^*\rightarrow mu} = \frac{s^c_o}{b^{k^*}_{wl}} + h^{rx}_{o\rightarrow\bar{o}} \cdot \frac{s^c_o}{b_{o\rightarrow\bar{o}}} + \frac{s^c_o}{b^{mu}_{wl}} + t^{prop}_{k^*\rightarrow mu}, \tag{4.14}$$

where $b^{k^*}_{wl}$, $t^{prop}_{k\rightarrow k^*}$, and $t^{prop}_{k^*\rightarrow mu}$ denote the transmission capacity between the nearby cobot and its associated ONU-MPP ($b^{k^*}_{wl}$=b_{wl}), uplink and downlink propagation delay, respectively.

• **Case 2 – inter-agent and intra-agent migration:** Inter-agent migration transfers a cognitive sub-task from one type of agent to a different one, e.g., cloudlet to cloud or vice versa. Whereas intra-agent migration transfers an uncompleted cognitive sub-task from one agent to another agent of the same type, e.g., cloudlet to cloudlet. In either case, the physical sub-task is done by the cobot. The total task response time for inter-agent ($a^* = \overline{a}$) and intra-agent ($a^* = \widetilde{a}$) migration is equal to $t_{k,a^*} = t_{k,a} + t^c_{a^*} - t^{rx}_{a\rightarrow mu} - \frac{wl_c}{v_a}$, where $t^c_{a^*}$ denotes the cognitive sub-task response time of the newly selected agent (a^*) given by $t^c_{a^*} = t^{tx}_{a\rightarrow a^*} + t^{ex}_{a^*} + t^{rx}_{a^*\rightarrow mu}$ with $t^{tx}_{a\rightarrow a^*}$, $t^{ex}_{a^*}$, $t^{rx}_{a^*\rightarrow mu}$ being the cognitive sub-task upload, execution time at migrated agent a^* ($t^{ex}_{a^*} = \frac{wl_c}{v_{a^*}}$), and result download delay (migrated agent to MU), respectively. Hence, we have

$$t^{tx}_{a\rightarrow a^*} = \begin{cases} \frac{2s^c_i}{b_{fl}} + h^{tx}_{o\rightarrow\bar{o}} \cdot \frac{s^c_i}{b_{o\rightarrow\bar{o}}} + t^{prop}_{a\rightarrow a^*} & \text{if } a^*{=}\widetilde{a}, \\ \frac{3s^c_i}{b_{fl}} + t^{prop}_{a\rightarrow a^*} & \text{if } a^*{=}\overline{a}, \end{cases} \tag{4.15}$$

$$t^{rx}_{a^*\rightarrow mu} = \begin{cases} \frac{s^c_o}{b_{fl}} + h^{rx}_{o\rightarrow\bar{o}} \cdot \frac{s^c_o}{b_{o\rightarrow\bar{o}}} + \frac{s^c_o}{b^{mu}_{wl}} + t^{prop}_{a^*\rightarrow mu} & \text{if } a^*{=}\widetilde{a}, \\ \frac{2s^c_o}{b_{fl}} + \frac{s^c_o}{b^{mu}_{wl}} + t^{prop}_{a^*\rightarrow mu} & \text{if } a^*{=}\overline{a}, \end{cases} \tag{4.16}$$

whereby $t^{prop}_{a\rightarrow a^*}$ and $t^{prop}_{a^*\rightarrow mu}$ are the propagation delay incurred during task upload and result download for intra-agent ($a^* = \widetilde{a}$)/inter-agent ($a^* = \overline{a}$) migration, respectively.

4.7.4 Energy Consumption

In the following, we analyze the energy consumption of MUs and cobots for task execution with and without migration. In the latter case, we account for the cobot's energy consumption for executing the full task and MU's energy consumption for receiving the task result. Thus, their total energy consumption e_k without task migration is given by

$$e_k = e^p_k + e^c_k + e^{rx}_k + e^{rx}_{mu}, \tag{4.17}$$

where e^p_k, e^c_k, and e^{rx}_k represent the cobot's energy consumption for executing the physical sub-task ($e^p_k = p_m \cdot \frac{d^k_i}{v_m} + p_s \cdot \frac{wl_p}{v_s}$) and cognitive sub-task ($e^c_k = p_c \cdot \frac{wl_c}{v_s}$), and transferring the result (e^{rx}_k= $p_u{\cdot}t^{rx}_k$) to the MU, respectively; e^{rx}_{mu} denotes the MU's energy consumption for receiving the task result (e^{rx}_{mu}= $p_d{\cdot}t^{rx}_k$).

With task migration, the energy consumption of MUs and cobots in the case of agent migration ($c2a$) is equal to $e_{k,a} = e^p_k + e^c_a$, where e^p_k and e^c_a denote the energy consumption for executing the cobot's physical and the agent's cognitive sub-task, respectively. With e^c_a = $p_u{\cdot}t^{tx}_{k\rightarrow a}$+$p_{idle}{\cdot}t^{ex}_a$+$p_d{\cdot}t^{rx}_{a\rightarrow mu}$, the energy consumption of MUs and cobots for the case of nearby

cobot migration ($c2c$) equals $e_{k,k^*} = e^p_k + e^c_{k^*}$, where e^p_k and $e^c_{k^*}$ stand for the energy consumption for executing the physical sub-task of the initially selected cobot and the cognitive sub-task of the nearby cobot, respectively. Finally, we have $e^c_{k^*} = p_u \cdot t^{tx}_{k\to k^*} + p_{idle} \cdot t^{ex}_{k^*} + p_d \cdot t^{rx}_{k^*\to mu}$.

4.7.5 Task Response Time and Energy Efficiency

Let us define the task response time efficiency $\bar{\beta}_{k,\theta}$ of the migration based scheme, e.g., agent ($\theta = a$) and nearby cobot migration ($\theta = k^*$), with regard to the non-migration scheme as the ratio of response time gain obtained from task migration and the response time t_k obtained without task migration. Hence, $\bar{\beta}_{k,\theta}$ is given by

$$\bar{\beta}_{k,\theta} = \frac{t_{k,\theta} - t_k}{t_k} \times 100\%, \tag{4.18}$$

whereby $t_{k,\theta}$ is the task response time in the case of task migration. Similarly, the energy consumption efficiency $\bar{\varepsilon}_{k,\theta}$ of task migration with regard to non-migration is obtained as $\bar{\varepsilon}_{k,\theta} = \frac{e_{k,\theta} - e_k}{e_k} \times 100\%$, where $e_{k,\theta}$ and e_k denote the energy consumption of STAs with and without task migration, respectively.

4.7.6 Migration Gain Overhead Ratio

The task migration gain overhead ratio $\overline{\eta}$ of task migration is calculated by taking the ratio of task response time gain with regard to non-migration ($t_{k,\theta} - t_k$) and the communication cost for task migration, which in turn consists of the mean task migration delay d_m, task upload delay $t^{tx}_{k\to\theta}$, and result download delay $t^{rx}_{\theta\to mu}$. In case of agent ($\theta = a$) or nearby cobot ($\theta = k^*$) migration, $\overline{\eta}$ is computed as follows:

$$\overline{\eta} = \frac{t_{k,\theta} - t_k}{d_m + t^{tx}_{k\to\theta} + t^{rx}_{\theta\to mu}}. \tag{4.19}$$

4.7.7 Deadline Miss Ratio

Next, we calculate the task deadline miss ratio MR as the ratio of the number of tasks missing the task deadline according to their corresponding assignment to a cobot/agent and the number of task requests. Hence, we have

$$MR = \frac{\sum_{i=1}^{\delta} NM_i}{\sum_{i=1}^{\delta} NM_i + \sum_{i=1}^{\delta} NS_i}, \tag{4.20}$$

where NM and NS denote the number of tasks that are completed with and without missing the deadline, respectively.

4.7.8 Task Blocking Probability

For the calculation of the task blocking probability p_b, we note that if the number δ of arriving task requests is smaller than the number of available collaborative nodes, a given task can be assigned to an available cobot/agent (N_w=$N_t \cdot p_s$) without blocking. However, if $\delta > N_w$, the first i tasks are assigned to available cobots/agents while the remaining task requests $\delta - i$ are blocked. Thus, p_b is computed as follows:

$$p_b = \begin{cases} 0 & \text{if } N_w \geq \delta, \\ \sum\limits_{i=N_w+1}^{\delta} \binom{\delta}{i} (p_s)^i (1-p_s)^{\delta-i} & \text{otherwise,} \end{cases} \tag{4.21}$$

where N_t and p_s denote the total number of collaborative nodes and the probability that a suitable cobot/agent is available, respectively. The average utilization rate u_w of a collaborative node is obtained as the ratio of number of utilized nodes $N_u = \delta \cdot p_s$ and total number of nodes N_t, translating into $u_w = \frac{N_u}{N_t} \times 100\%$.

4.7.9 Communication-to-Computation Ratio (CCR)

Another important performance metric is the so-called communication-to-computation ratio (CCR). CCR is defined as the ratio of communication latency and task processing time of a selected collaborative node and is given by

$$CCR = \frac{d_m + t^{tx}_{k\to\theta} + t^{rx}_{\theta\to mu}}{t^{p}_{k} + t^{ex}_{\theta}}, \tag{4.22}$$

where d_m, $t^{tx}_{k\to\theta}$, $t^{rx}_{\theta\to mu}$, t^{p}_{k}, and t^{ex}_{θ} denote the mean task migration waiting delay, cognitive sub-task upload delay, result download delay, physical sub-task execution time, and cognitive sub-task execution time, respectively.

4.7.10 End-to-End Task Execution Delay

In this subsection, we analyze the end-to-end task execution delay with and without task migration, taking into account task response time, task migration waiting delay as well as US and DS frame delay. If a given MU generates a task request message after her current bandwidth request message, the MU has to wait for polling cycle T_c to report her task request. After transmitting the task request in the next cycle timeslot t^m_{sl} the MU experiences an additional delay of $T_c - t^{msg}_{wl}$ since $t^m_{sl} \geq t^{msg}_{wl}$. Thus, the total US waiting delay equals $2T_c - t^{msg}_{wl}$. The corresponding ONU-MPP/ONU-eNB receives and forwards the incoming task request to the OLT in the US direction, which leads to the maximum US frame transmission delay t_u.

Table 4.3: Parameters and default values for evaluation of context-aware task migration strategies

Notation	**Description**	**Default value/unit**
T_c, N, M, m	Polling cycle time, number of ONUs, STAs, and cobots	ms, 32, 10, 5
$v_s/v_{s^*}, v_a$	CPU clock speed (cycles/second) of cobot (k)/nearby cobot (K^*) and agent (a)	100-500 MHz, 3.2 GHz
v_m, d_i^k, e_k^i, e_r, λ_{MU}	Cobot's moving speed, distance between cobot and task location, cobot's initial energy, and required energy for full task, density of MU within cellular coverage	1-5 m/s, 1-10 m, 500 KJ, 1-5 Joule, 1-50
p_m, p_s, p_c, v_u	Cobot's average power consumption during moving, physical, and cognitive task processing, MUs average speed	0.7 W, 0.5 W, 0.5 W, 1-10 mph (vary)
$s_i, s_i^p/s_i^c, s_o^p/s_o^c$	Total task input data size, physical/cognitive sub-task input, and output data size	KB
$b_{wl}/b_{wl}^*, b_{fl}$	Transmission capacity of WLAN/cellular link for cobot (b_{wl}^k) and MU (b_{wl}^{mu}), fiber link	6900/300 Mbps, 10 Gb/s
wl_i, wl_p, wl_c, p_s	Full task, physical, cognitive sub-task workload, worker node availability probability	CPU cycles, 0-1
$t_{k\to mu}^{prop}, t_{k^*\to mu}^{prop}$, $t_{a\to a^*}^{prop}$	Total propagation delay between cobot and MU, nearby cobot (k^*) and MU, primary agent (a) and secondary agent (a^*)	2.66 μs, 2.66 μs, 50 ms (inter)/0.02 ms (intra)
$t_{k\to a}^{prop}, t_{k\to k^*}^{prop}$, $t_{a^*\to mu}^{prop}$	Total propagation delay between cobot (k) and agent (a=cloudlet task location/result download location/remote cloud), cobot and nearby cobot, secondary agent and MU	0.010/0.012/50 ms, 0.6 μs, 50 ms (inter)/0.02 ms (intra)
p_u, p_d, p_{idle}	Cobot's average power consumption in upload, download, idle state (per second)	0.1W, 0.05W, 0.001W
$t_{wl}^{msg}, t_{pon}^{msg}, t_{as}$, t_{cs}, t_g	WLAN message length (e.g., `PS-Poll`), MPCP message length (`GATE,REPORT`), cobot, and agent selection delay, guard time between two slots	0.231μs, 0.512μs, ms, 46 μs
$h_{o\to\bar{o}}^{tx}/h_{o\to\bar{o}}^{rx}$	Number of hops between initial and final ONU during data transmission and reception	2
r_{MAP}, A_{cell}, λ_{MAP}	MAP radius, cellular coverage area, density of MAPs within cellular coverage area	100 m, 3·3 km^2, 3
$N_w/N_t, t_{req}/$ t_{res}/t_{ack}	total number of collaborative nodes, task request/cobot response/ack message duration	1-20, 0.17μs/0.12μs/0.12 μs
$d_m, \omega/\mu/\lambda$, δ	Mean task migration packet delay, number of servers in cloudlet or remote cloud/mean task service rate/task arrival rate per server, total arrived task request	ms, 1-10, 1-20

By taking into account the US waiting delay $2T_c - t_{wl}^{msg}$, service time delay X_{max} at the ONU-MPP/ONU-eNB, and US propagation delay t_{prop}, t_u is obtained as $t_u = 2T_c - t_{wl}^{msg} + t_{prop} + X_{max}$.

Next, we calculate the maximum DS frame delay. The DS waiting delay is equal to $T_c - t_{sl}$. The maximum DS delay t_d is calculated by summing up the DS waiting delay, service delay X_{max}, and associated propagation delay t_{prop}, which yields $t_d = T_c - 2t_{pon}^{msg} - t_{prop} + X_{max}$ with $t_{sl} \geq 2t_{prop} + 2t_{pon}^{msg}$. After receiving the task request from the OLT, the corresponding ONU-MPP starts the cobot selection if there is no task migration. Otherwise, the ONU-MPP selects both a cobot (t_{cs}) and an agent (t_{as}) for $c2a$ migration. Thus, the end-to-end task execution delay t_k^{e2e} without task migration equals $t_k^{e2e} = t_u + t_d + t_{cs} + t_k^f + d_m + t_k^{rx}$, where t_k^f, t_k^{rx}, d_m, and t_{cs} denote the cobot's full task processing time, task result transfer time, mean task result buffering delay, and cobot selection delay, respectively.

Similarly, the end-to-end task execution delay for $c2a$ migration ($t_{k,a}^{e2e}$) and $c2c$ migration (t_{k,k^*}^{e2e}) is obtained as $t_{k,a}^{e2e} = t_u + t_d + t_{cs} + t_k^p + d_m + t_{as} + t_a^c$ and $t_{k,k^*}^{e2e} = t_u + t_d + 2t_{cs} + t_k^p + d_m + t_{k^*}^c$, where t_{as}, t_a^c, and $t_{k^*}^c$ denote the agent selection delay and the cognitive task response time of an agent and a nearby cobot, respectively.

4.8 Results

In this section, we investigate the performance of our proposed task migration scheme. For convenience, Table 4.3 summarizes the key parameters and their assigned default values in compliance with [17], [30], [9], and [105]. To examine the performance of our proposed task migration scheme, we consider multiple task execution scenarios with different task workload, input, and output data size values. More specifically, in scenario 1 we consider a lighter full task workload with smaller input and output data sizes than in scenario 2. Note that the specific parameter settings for Scenario 1 is given by: $wl_p = wl_c = 100, 200, 300, 400, 500$ M cycles, s_i^p=s_i^c=100,200,300,400,500 KB, s_o^c=40,80,120,160,200 KB, t_d=1.5,2,2.5,3,3.5s. Whereas, Scenario 2 parameter settings is described as follows: wl_p=wl_c= 160,320,480,640,800 M cycles, s_i^p=s_i^c=120,240,360,480,600 KB, s_o^c=240,480,720,960,1200 KB, t_d=1.8,2.6,3.4,4.2,5s.

Simulation Setup: In this section, we present results by means of Matlab based computer simulations. The physical (move to a task location and image capturing) and cognitive sub-task (face detection form captured image) workload, cognitive sub-task input data size (output of physical sub-task), cognitive sub-task output data size, and full task deadline ranges from 100-800 M cycles, 100-600 KB, 40-1200 KB, and 1.5-5s, respectively. Further, the maximum CPU clock speed of an cloud agent (central cloud/cloudlet server) is set to 3.2 GHz. A cobot's distance to the task location, CPU clock speed, and moving speed is chosen randomly from an interval of 1-10 m, 100-500 MHz, and 1-5 m/s, respectively. Note that, at the wireless front-end both the MU's device and cobot can be attached to both cellular

and WLAN interface via connectivity with ONU-MPP and ONU-eNB, respectively. The fiber backhaul length between ONUs and the central office (OLT) is 20 Km long. Whereas, the optical fiber length between the ONU and cloudlet server is 2 km long. The transmission capacity of a fiber link, WLAN, and cellular link is set to 10 Gb/s, 6900 Mbit/s, and 300 Mbit/s respectively. The MPCP (`GATE,REPORT`) and `PS-Poll` messages are of size 64 and 20 bytes, respectively, i.e., t_{pon}^{msg}=0.512 μs and t_{wl}^{msg}=0.231 μs. The number of ONUs, associated STAs, and hop distance between ONU for MU's trajectory (initial and final location of MUs) is set to 32, 10, and 2, respectively. The FiWi traffic load and polling cycle time is varied in an interval of 0.05-0.95 and 100-800 ms, respectively. Further, the MAP radius, cellular coverage area, and density of MAPs within each cellular coverage area is set to 100 m, $3 \cdot 3$ km^2, and 3, respectively. Note that the total number of task request arrivals and number of collaborative nodes (agents/cobots) is varied in the range of 1-20 in order to investigate their impact on the performance. The remaining default values and parameter settings related to each particular evaluation scenario are provided in Table 4.3 and Figs. 4.5-4.7, respectively. Importantly, for low-latency HART task execution the main requirements are as follows: the availability of cobot and cloud server resources for requested task processing, satisfaction of the task execution deadline criteria, transmission at the speed of light, edge cloud server within 20 Km distance from the decentralized ONUs for processing, connectivity of cobots/MUs with ONUs via a wireless interface, connectivity of cloudlet server with ONU via a fiber link, hardware/software interface to transfer the task request to cobot/cloud agent and task result reception by MUs from cobot/cloud agent, among others. We also note that in the event of multi-task arrivals, we assume that the task requests are served in a first-come-first-served (FCFS) fashion. Furthermore, we assume that an available agent/cobot can execute only one cognitive/physical sub-task at any given time.

We first investigate the task response time and energy consumption, which are the two key performance metrics that determine whether the initially selected cobot should execute the full task (without migration) or migrate the cognitive sub-task to a collaborative agent (cloudlet or cloud in $c2a$ migration) or nearby cobot ($c2c$ migration) for execution. Figs. 4.5(a)-(c) show the task response time and energy consumption evaluation of the different task migration schemes for varying total task input data size (single full HART task input data size that includes both physical and cognitive sub-task). We observe that the task response time and energy consumption of all compared schemes increase for increasing full task input data size. Note, however, that the $c2a$ (cloudlet near task location) and $c2a$ (cloudlet near result download location) schemes achieve the minimum task response time and energy consumption in scenario 1 and 2, respectively. The is due to the fact that in scenario 1 the migrated task input data size (s_i^c) is larger than the migrated task output

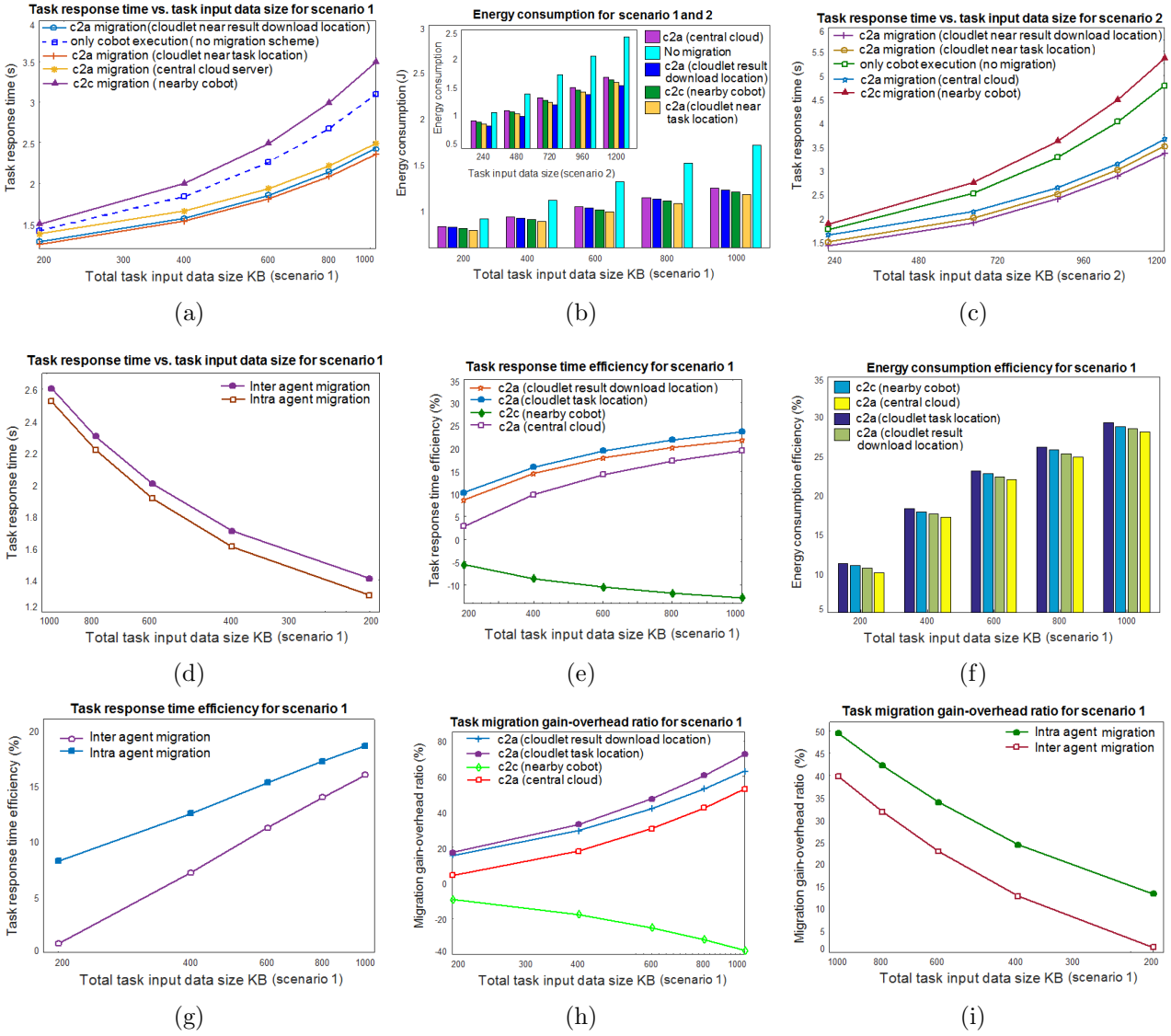


Figure 4.5: Task response time, energy consumption, task response time, energy efficiency, and migration gain-overhead ratio vs. task input data size (s_i) evaluation of different task migration schemes under scenario 1 and 2.

data size (s_o^c), whereas in scenario 2 the relation between migrated task input and output data size is reversed. The opposite relation between total task input and output data size results in the minimum task migration latency (both upload and result download delay) for the $c2a$ scheme (cloudlet near task location) and $c2a$ scheme (cloudlet near result download location) in scenario 1 and 2, respectively. Moreover, we note that the central cloud based $c2a$ migration scheme experiences a higher task response time and energy consumption than in both cloudlet based $c2a$ schemes. This is because the central cloud based task migration scheme suffers from a higher task migration latency (upload and download delay) than both cloudlet task migration schemes, whereby the central cloud and cloudlet servers are assumed to have the same processing capability in terms CPU speed. Further, we observe that the task response time of all considered schemes except the $c2c$ migration (nearby cobot) satisfy the task deadline criteria of both scenarios, whereas the $c2c$ migration can only meet the task deadline criteria of scenario 1. We also note that the task migration to nearby cobot ($c2c$) is able to improve the energy consumption of the initially selected cobot (without task migration). Hence, due to the nearby cobot's lower task processing speed, the $c2c$ scheme results in the worst task response time performance among all compared schemes.

To highlight the impact of task migration between two agents, Fig. 4.5(d) compares the task response time of the two different schemes: inter-agent (cloudlet to central cloud) and intra-agent (cloudlet to another cloudlet) migration. The figure shows that in scenario 1 the intra-agent migration provides a shorter task response time than its inter-agent counterpart. Clearly, this is because intra-agent migration suffers from a lower task migration communication overhead. Thus, intra-agent migration is more preferable when a failure occurs during agent task execution.

To demonstrate the suitability of task migration, Figs. 4.5(e) and (f) depict the task response time and energy consumption efficiency of different task migration schemes in comparison with the non-migration scheme. Clearly, for increasing task input data size, the task response time and energy consumption efficiency rise rapidly for all considered $c2a$ migration schemes. Both figures indicate that the $c2a$ (cloudlet near task location) scheme is the best choice in scenario 1 since it offers the maximum task response time and energy consumption efficiency. For instance, for a typical case of 600 MB in scenario 1, the $c2a$ migration (cloudlet near task location) yields the highest task response time (20%) and energy efficiency (23%), whereas the second best $c2a$ (cloudlet result download location) scheme achieves approximately 17% and 21% improvement of the task response time and energy efficiency compared to the non-migration scheme. Hence, the task response time and energy efficiency of $c2a$ migration (central cloud) is equal to 13% and 20%, respectively. In addition, note that the $c2c$ migration results in a negative task response time and positive energy efficiency gain over the

non-migration scheme. Fig. 4.5(g) shows that the maximum task response time efficiency is achieved with intra-agent migration. For instance, for a task input data size of 600 MB in scenario 1, the task response time efficiency gain over the non-migration scheme is approximately 15% with intra-agent migration, as opposed to only 11% with inter-agent migration.

In Fig. 4.5(h), we examine the task migration gain-overhead ratio of the different task migration schemes for varying task input data size. Note that a higher task migration gain-overhead ratio indicates the suitability of a particular task migration scheme over other compared schemes. From Fig. 4.5(h) we observe that for an increasing task input data size the task migration gain-overhead ratio increases in all considered $c2a$ migration schemes and decreases in the $c2c$ migration scheme. For instance, for a task input data size of 600 MB in scenario 1, $c2a$ migration (cloudlet task location) shows the highest task migration gain-overhead ratio, which is approximately 48%, 5%, 18%, 73% higher than that of the non-migration, $c2a$ migration (cloudlet result download location), $c2a$ migration (central cloud), and $c2c$ migration schemes, respectively. Further, Fig. 4.5(i) illustrates that for varying task input data size the task migration gain-overhead ratio is smaller with inter-agent rather than intra-agent migration. For instance, for a typical task input data size of 400 MB in scenario 1, intra-agent migration provides a 12% higher task migration gain than inter-agent migration.

Next, Fig. 4.6(a) investigates the impact of task input data size on CCR for different task migration schemes. We notice that for increasing task input data size CCR decreases in all considered schemes. This is because the migrated task execution time is inversely proportional to CCR (see Eq. (4.22)). Also note that a task migration scheme with a lower CCR involves a lower communication overhead for a given migrated task execution. In scenario 1, we observe that the $c2a$ (cloudlet near task location) and intra-agent migration schemes offer a smaller CCR than their counterparts. Furthermore, note that CCR is the highest with $c2a$ migration (central cloud) and inter-agent migration.

Fig. 4.6(b) sheds light on the average collaborative node utilization ratio by varying the total number of collaborative nodes (cobots and agents) for different collaborative node availability probability p_s. The figure clearly shows that for a fixed number δ of arriving tasks, the average utilization ratio increases with the total number of available collaborative nodes $N_w = N_t \cdot p_s$ until it levels off at $\delta = N_w$. The average utilization ratio then decreases for $N_w > \delta$. Further, we note that the average utilization ratio is higher for large p_s. Fig. 4.6(c) shows how the average task blocking probability p_b varies with the total number N_t of collaborative nodes and collaborative node availability probability p_s. Interestingly, note that p_b is higher at smaller numbers of available collaborative nodes, i.e., $N_w = N_t \cdot p_s$. Furthermore, we observe that for fixed non-zero values of p_s and increasing N_t the blocking probability decreases and reaches zero for $N_w = \delta$. Fig. 4.6(d) depicts the deadline-miss ratio

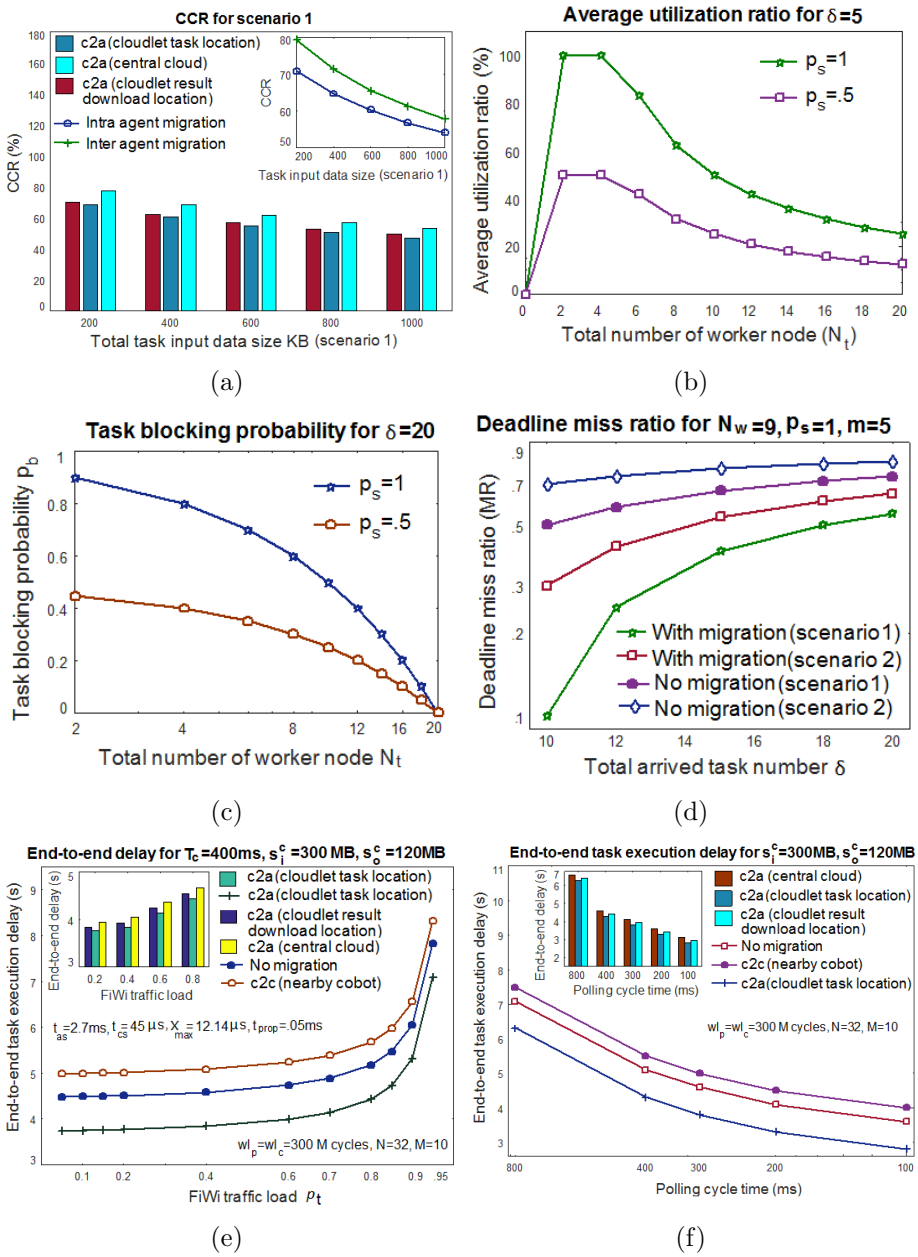


Figure 4.6: Communication-to-computation ratio (CCR), average utilization of collaborative node, task blocking probability, deadline-miss ratio, and end-to-end task execution delay evaluation of different task migration schemes.

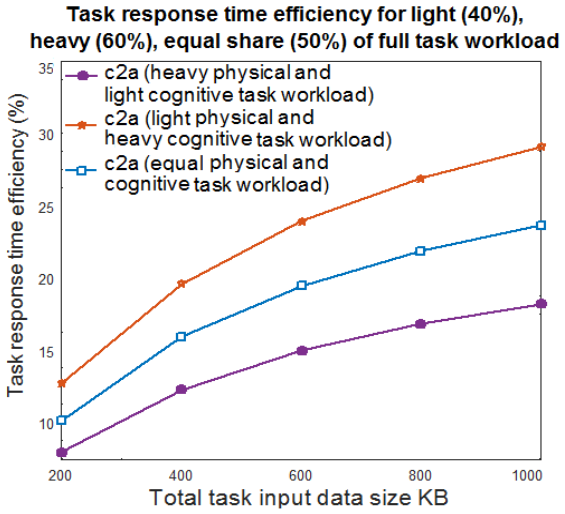


Figure 4.7: Task response time efficiency of proposed task migration scheme over non-migration scheme for Scenario 3 task workload settings.

with and without task migration for varying total number δ of arriving tasks, with N_w fixed. Fig. 4.6(d) reveals that for increasing δ, the deadline-miss ratio increases rapidly in both schemes with and without task migration. Note, however, that the non-migration scheme results in a higher task deadline-miss ratio than the migration scheme in both scenarios. The reason behind this is that the non-migration scheme considers only cobots for task execution, whereas the migration scheme utilizes both cobots and agents as collaborative nodes for task execution. As a result, more task requests can be served by the migration scheme. Fig. 4.6(e) shows that the end-to-end task execution delay of the different task migration schemes remains low at light FiWi traffic load ρ_t, but rapidly increases for high ρ_t. Note that the end-to-end task execution delay is minimum in the $c2a$ migration (cloudlet near task location) scheme. For instance, for a given task input data size of 300 MB and FiWi traffic load of 0.8, $c2a$ migration (cloudlet near task location) offers a 15% and 21% lower end-to-end task execution delay than the non-migration and $c2c$ migration scheme, respectively. Fig. 4.6(f) depicts the end-to-end task execution delay performance of the different task migration schemes for varying polling cycle time T_c. We observe that for a large T_c, the task execution delay of the different task migration schemes remains high, but rapidly decreases for smaller T_c. Notably, $c2a$ migration (cloudlet near task location) outperforms its counterparts in terms of end-to-end task execution delay.

Finally, Fig. 4.7 sheds some light on the task response time efficiency evaluation of our proposed $c2a$ task migration (cloudlet near task location) scheme over non-migration scheme under different physical and cognitive sub-task workloads for a full HART task. We notice that the highest task response time efficiency is achieved in our proposed task migration scheme

for a full HART task with light physical and heavy cognitive sub-task workload in Scenario 3 (wl_i=200, 400, 600, 800, 1000 M cycles, s_i^p=s_i^c=100,200,300,400,500 KB, s_o^c=40,80,120,160,200 KB, t_d=1.5,2,2.5,3,3.5s). Thus, a full HART task with light physical sub-task and heavy cognitive sub-task is most suitable for our proposed scheme. Further note that among all three different task workload settings the achievable task response time efficiency in our proposed scheme is significantly lower for a full HART task with heavy physical and light cognitive sub-task workload. This is because in our proposed *c2a* migration scheme, the physical sub-task is performed by a cobot that is computationally less powerful, whereas the cognitive sub-task is executed by a more powerful agent (cloudlet). By contrast, in the non-migration scheme, the full task (involving both physical and cognitive sub-tasks) is performed by the cobot. Thus, the observed task response time difference between our proposed scheme and the non-migration scheme is lower for a HART task with heavy physical and light cognitive sub-task workload. Whereas the task response time difference is higher for a HART task with light physical and heavy cognitive sub-task workload. Furthermore, Fig. 4.7 illustrates that for increasing task input data size, the task response time efficiency increases in our proposed scheme for a full HART task under all three different task workloads. For instance, for a typical case of 1000 MB, our proposed scheme achieves up to 18%, 23%, 29% higher task response time efficiency than the non-migration scheme for a full HART task with heavy physical and light cognitive sub-task workload, equal physical and cognitive sub-task workload, light physical and heavy cognitive sub-task workload, respectively.

4.9 Conclusions

In this chapter, we examined the performance of our proposed context-aware task migration scheme for HART-centric task execution in FiWi based Tactile Internet infrastructures. To improve the task execution latency, our proposed scheme not only selects a suitable cobot and collaborative node for HART-centric task execution, but also migrates a task from one collaborative node to another. Further, we presented an adaptive resource allocation scheme to handle both traditional broadband and task migration data traffic at the same time. We developed an analytical model to investigate the performance of our proposed task migration scheme in terms of task response time, energy efficiency, communication-to-computation ratio, and task migration gain-overhead ratio, among others. The presented results help determine the optimal task migration schemes under a variety of use case scenarios with different task, agent, and cobot characteristics. Our obtained result show that for a typical task input data size of 600 MB, the *c2a* task migration (cloudlet near task location) scheme exhibits up to 20% task response time and 23% energy efficiency improvement over task execution without migration. The results also indicate that in the case of an agent node failure, intra-agent task

migration offers a higher task response time gain than inter-agent migration. Our proposed task migration scheme is thus well suited to provide low-latency performance for emerging Tactile Internet applications.

Chapter 5

Community- and Latency-Aware Multi-Task Scheduling and Prefetching-Aware DBA in FiWi Enhanced Networks

5.1 Preamble

This chapter contains material extracted from the following paper:
[J5] M. Chowdhury and M. Maier, "Community- and Latency-Aware Multi-Task Scheduling for HART Collaboration in FiWi Enhanced Networks," *IEEE Transactions on Cloud Computing,* November, 2018 (submitted) [106].

5.2 Introduction

With the proliferation of smart mobile devices, a wide variety of emerging mobile applications such as 3-D interactive games and gesture/object recognition are increasingly turning into indispensable assets. Despite their ongoing development, many latency-sensitive mobile applications (e.g., face detection) cannot be efficiently run on mobile devices due to their limited computing and storage resources. In response to this challenge, mobile cloud computing (MCC) has been a promising solution, where resource-limited mobile devices transfer their latency-sensitive and computation-intensive tasks to resource-rich cloud servers for remote execution through a process known as *computation offloading* [17].

Most existing MCC based task offloading solutions are based on the following platforms: centralized cloud [17], decentralized cloudlet [28], or mobile ad-hoc cloud [30]. Remote cloud based task offloading may not always satisfy the QoS requirements of many real-time mobile applications due to the higher propagation delay. To lower the latency of centralized cloud

offloading, cloudlets may placed at the network edge to offer cloud services in close proximity to mobile subscribers. The importance of decentralized cloudlets can be witnessed in many real-time human-machine interaction based applications (e.g., remote surgery), where an extremely low round-trip latency of 1-10 ms is required to match humans' interaction with their environment. This vision of very low-latency communications gives rise to the so-called Tactile Internet, where the remote control of virtual/physical objects via the Internet allows humans to accomplish their tasks in places in which they don't have to be physically present [5].

The overarching goal of the Tactile Internet is the production of new goods/services that complement humans rather than substitute for them. This collaborative human-machine co-activity approach is part of the still young field of *human-agent-robot teamwork (HART)* research, where humans, robots, and intelligent agents play complementary roles in accomplishing different latency-sensitive tasks, e.g., crowd sensing [55]. HART tasks may be either a physical task, a digital task, or may include both. Physical tasks are location dependent and follow specific manual instructions (e.g., capturing image at a given task location). Conversely, digital tasks are location independent and require sophisticated judgement capability for making intelligent decisions (e.g., intrusion detection from captured image). In general, the task assignment to unsuitable robots/agents may lead to unnecessarily increased delays and resource consumption. To avoid these problems, only mobile robots near a given task location can be selected to execute physical task, whereas digital tasks may be executed by either the robot or a nearby agent. To the best of authors knowledge, no existing works deals with the problem of minimizing the real-time execution overhead of multiple HART tasks by means of multi-task scheduling, community cluster resource awareness, task prefetching, and failure avoidance.

The majority of existing task scheduling policies focus on offloading computation-intensive digital tasks either onto cloud servers or mobile devices rather than both, whereby a significant number of these policies apply *offline scheduling*. This in turn requires that a-priori information about future tasks (e.g., arrival time, deadline) is available to the task scheduler. Offline scheduling schemes are well suited for periodic tasks, but become less suitable for executing aperiodic tasks in real-time [25],[27]. Due to their uncertain cloud resource requirements, the execution real-time aperiodic tasks demands a suitable online (dynamic) task scheduling scheme to maintain QoS assurances. In contrast, online task scheduling may cause significant task migration latency [28]. Further, the lack of a proper bandwidth assignment strategy may cause higher waiting times for data transmission and result reception during the task offloading process. Thus, one of the key challenges for online task scheduling is to minimize the total task execution latency, including both task processing and offloading communication delay,

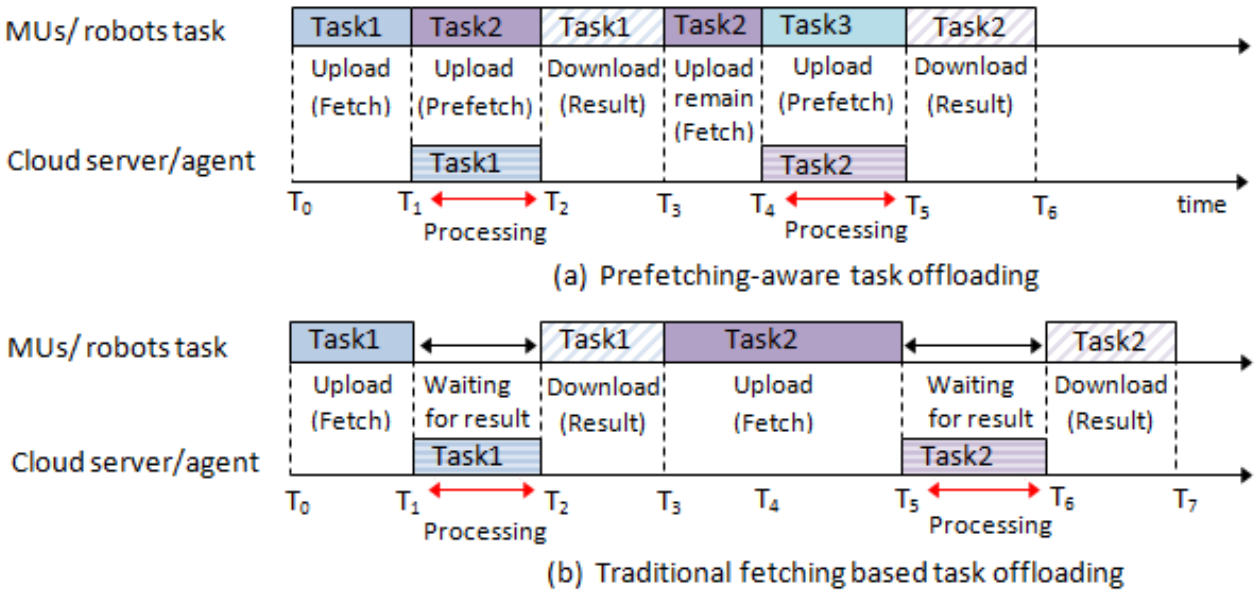


Figure 5.1: Resource block allocation in task offloading: (a) prefetching vs. (b) conventional fetching.

by mitigating the uncertainty of cloud/bandwidth resource management and failure avoidance [29].

Importantly, most of the existing computation task offloading studies [17],[30],[25],[29] apply the *conventional fetching* technique, where a given mobile user's (MU) next computation task can be transferred to the cloud server only upon the completion of the previously offloaded task, as shown in Fig. 5.1(b). As a result, conventional fetching suffers from an increased multi-task offloading latency. To overcome this shortcoming of conventional fetching, *task prefetching* has been recently applied in the context of task offloading, as illustrated in Fig. 5.1(a), where the full or a portion of the MU's next task input data is transferred to the cloud server already during the computation period of the previously offloaded task [85],[86]. Note, however, that both studies [85] and [86] considered only remote cloud (agent) based computation task offloading rather than multi-cloud agent (e.g., cloudlet, robot) based heterogeneous HART task offloading in an online setting. Furthermore, most existing studies considered only dedicated remote clouds or isolated host cluster resources (mobile devices/cloudlets within the coverage zone of a single wireless access point) for the MUs' task assignment. Clearly, relying only on isolated cluster resources may not always satisfy different given task execution requirements when arriving task requests exceed available isolated cluster resources. To overcome these shortcomings, a promising solution is to utilize available nearby community cluster resources (mobile devices/cloud servers under multiple nearby wireless access point coverage areas) for low-latency HART task execution. Notwithstanding, the task assignment

to available community cluster resources may sometimes experience extra task execution overhead due to the higher task location traverse time and task offloading communication latency. Thus, to improve task execution performance by selecting the optimal task processing node, further investigation of adaptive HART task scheduling schemes with isolated and community cluster resource awareness is mandatory.

In this chapter, we propose a community- and latency-aware task scheduling scheme that selects a suitable robot and cloud agent for HART task execution by collecting time-varying robot/cloud resource information in an online manner. Unlike previous work, our goal is to minimize both task execution latency and power consumption for multiple HART tasks. To reduce the task migration overhead, we incorporate batch based HART task scheduling into our online scheme. In addition, we investigate the performance of both task onloading and task offloading based HART task execution while benefitting from task prefetching and failure avoidance. Further, we develop a prefetching-aware dynamic bandwidth allocation scheme for multiple HART task execution. We present an analytical framework to evaluate the performance of our proposed schemes in terms of mean task service time, delay and power saving ratio, task prefetching time efficiency, processing-to-service time ratio, speed up, and satisfactory ratio.

The remainder of the chapter is structured as follows. Section 5.3 reviews related work and outlines open challenges. In Section 5.4, we introduce our considered fiber-wireless (FiWi) enhanced Tactile Internet infrastructure and describe our proposed adaptive task scheduling scheme in greater detail. Sections 5.5 and 5.6 present our analytical model and obtained results, respectively. Section 5.7 concludes the chapter.

5.3 Related Work and Open Challenges

In recent years, a significant amount of research efforts on *online* task scheduling aimed at scheduling real-time aperiodic tasks of MUs. In [18], the authors outlined a heterogeneity-aware task scheduling scheme that selects only suitable peer mobile devices for computation task offloading based on their respective skills and task processing times. In [107], the authors devised a location-aware task scheduling algorithm that leverages on infrastructure-based cloud resources for computation task offloading, trying to keep a balance between user equipment energy consumption and load balancing. The study in [108] and [109] investigated how to minimize bandwidth usage and queuing delay of different cloud tasks, respectively. In [110], the authors proposed an online resource management scheme that jointly optimizes cloud task offloading profits and user equipment battery lifetime. The authors of [111] presented a real-time task scheduling mechanism, where a suitable cloud server is selected for offloaded tasks according to their minimal energy consumption cost while meeting given task deadlines. The

problem of parallel task scheduling in a remote cloud with the objective to reduce task workload processing time was addressed in [112]. In [113], a suitable cloud server selection scheme for offloaded task assignment was developed to lower transmission delay. Note that [112] and [113] minimized either task offloading communication delay or task processing delay, but not both.

All aforementioned studies focused on the decision whether or not to offload a single user's computation-intensive task to a dedicated/isolated host cluster cloud node (cloudlet/peer mobile device). Furthermore, they assumed that task offloading can be performed immediately without considering the availability of cloud servers and bandwidth resources. An important issue little addressed in previous work is how to collect information about different cloud/robot resources for online task scheduling. More specifically, none of the previous studies investigated the problem of minimizing task service/execution time and user equipment energy consumption jointly for real-time HART task scheduling. There is also a lack of task prefetching-aware bandwidth allocation policies in order to minimize HART task offloading latency. Moreover, previous work has not explicitly studied failure-avoidance service selection and optimal resource scheduling order for the execution of different HART tasks. Furthermore, the question of how to coordinate the execution of multiple HART tasks using infrastructure-based and infrastructure-less isolated/community cluster resources remains an open research issue. Clearly, to analyze the task scheduling performance a proper HART task service delay calculation model needs to be developed by taking the different delay components, including task workload processing, transmission, and waiting time, into account.

In this chapter, we aim at addressing open research challenges related to the real-time multi-task scheduling for HART collaboration across FiWi enhanced Tactile Internet infrastructures. Our proposed task scheduling scheme considers not only isolated host and community cluster resource awareness but also both prefetching-aware task offloading and suitable failure avoidance for HART task execution. To determine the optimal task scheduling order, we compare the following schemes: First Come First Serves (FCFS), Earliest Deadline First (EDF), and Concurrent Policy (CP). We compare the performance of our proposed task offloading schemes with task onloading, random, and communication-aware task offloading schemes in terms of a variety of HART-specific performance metrics.

5.4 FiWi Enhanced Tactile Internet Infrastructure for HART Task Scheduling

In this section, we describe the considered FiWi enhanced Tactile Internet infrastructure and our proposed adaptive HART task scheduling and bandwidth allocation scheme.

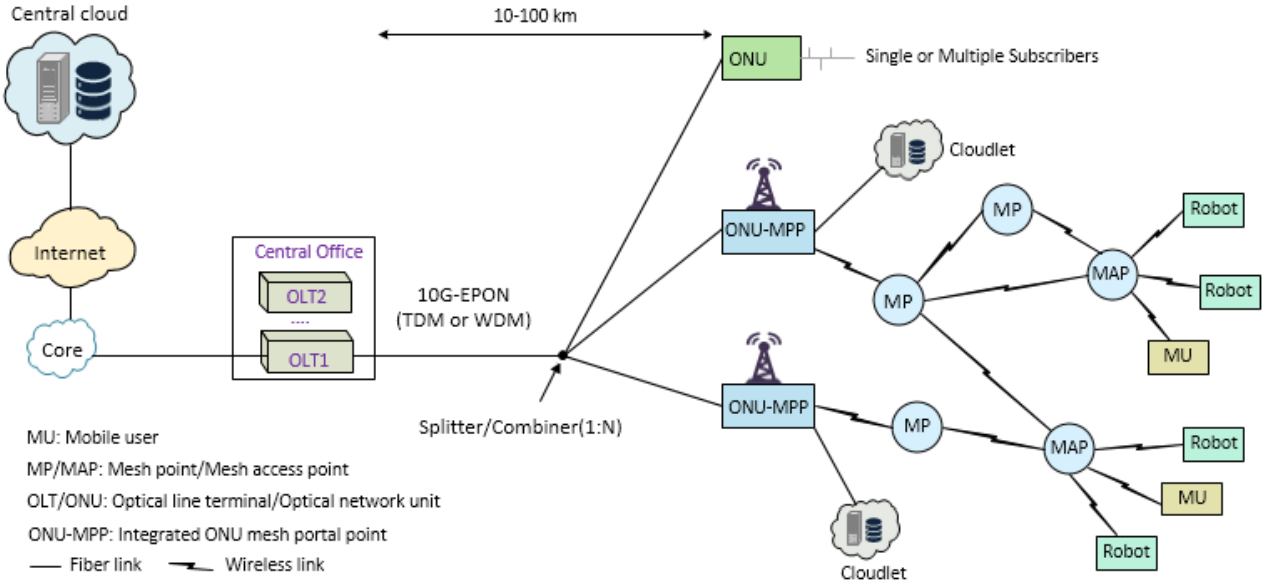


Figure 5.2: FiWi enhanced Tactile Internet infrastructure for adaptive HART task scheduling.

5.4.1 Network Architecture

Fig. 5.2 depicts the FiWi enhanced Tactile Internet architecture, which is based on low-cost, data-centric Ethernet passive optical network (EPON) and WLAN technologies. Specifically, the fiber backhaul is based on an IEEE 802.3av 10 Gb/s EPON (10G-EPON), which consists of the central optical line terminal (OLT) and remote optical network units (ONUs). The OLT connects to the ONUs at the customer premises via a tree-and-branch topology. The distance between the central OLT and remote ONUs ranges between 10-100 km. A subset of ONUs are located at residential or business subscriber premises, offering FTTx services, e.g., fiber-to-the-home/business (FTTH/B) to a single or multiple subscribers. To interface with front-end WiFi mesh network (WMN), another subset of ONUs are equipped with a mesh portal point (MPP), giving rise to a so-called ONU-MPP. Intermediate mesh points (MPs) serve as relay nodes between MPPs and mesh access points (MAPs), each being associated with one or more MUs and robots.

The integration of ONU and MPP is done by using decentralized radio-and-fiber (R&F) technologies, where medium access control (MAC) protocol translation is performed at the optical-wireless interface. A subset of selected ONU-MPPs are connected to each other via inter-connection fiber (IF) links for redundancy. Further, to provide low-latency cloud services to MUs at the network edge, cloudlets are attached to ONU-MPPs via dedicated fiber links [94].

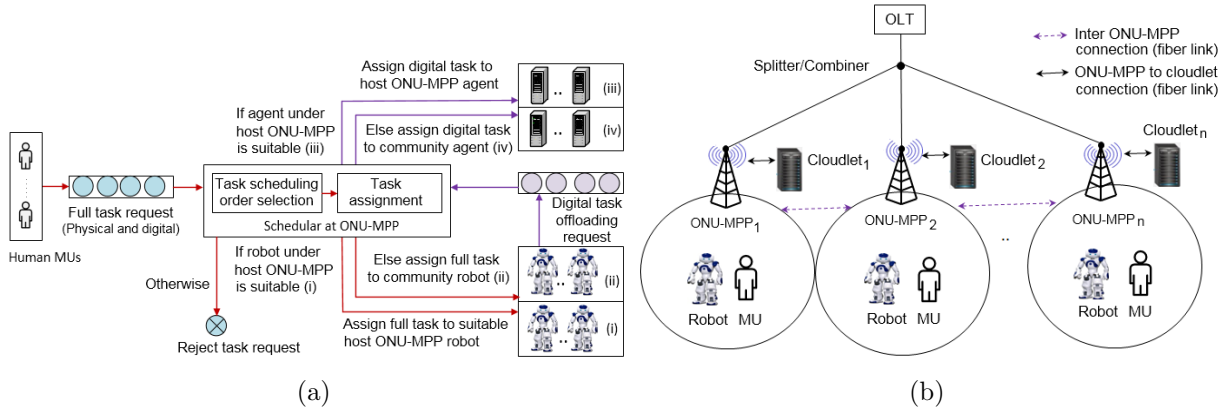


Figure 5.3: (a) Adaptive batch model for proposed HART task scheduling and (b) community-cluster architecture.

5.4.2 Adaptive HART Task Scheduling Scheme

5.4.2.1 Community/Latency-Aware Task Assignment

Given that task request and robot/agent availability information isn't known a priori, we apply a distributed approach where the task scheduler at the ONU-MPP makes task scheduling decisions by exchanging control messages with each robot/agent. The process comprises the following three steps:

1) Task Request Collection: The first step of our proposed scheme collects a given MU's task request that arrives during a FiWi polling cycle, as shown in Fig. 5.3. Each MU sends a full HART task request message to the associated ONU-MPP during the assigned broadband sub-slot. The task scheduler at the ONU-MPP may receive multiple task requests from MUs arriving during a FiWi time cycle and stores all task requests in a batch queue. Note that each HART task request includes information about the full HART task workload (i.e., physical and digital sub-tasks) in terms of required CPU cycles to process the task, task location, task input and output data size. Next, the task scheduler selects a suitable robot/agent for each arriving task request. For enhanced performance, we select suitable task execution scheme with both community- and latency-aware task offloading and onloading, as explained next.

(*i*) *Community- and Latency-Aware Task Offloading:* In this scheme, multiple nearby ONU-MPPs form a *community cluster* and utilize their available robot/agent resources for each HART task execution (see also Fig. 5.3). The physical and digital sub-tasks are assigned to the most suitable robot and agent, respectively, by evaluating all robots/agents within the coverage area of the host and community ONU-MPP. For selecting a suitable robot/agent, our *community- and latency-aware* offloading scheme distinguishes the following cases. If the number of arriving task requests is smaller than/equal to the number of available host ONU-MPP resources (e.g., robots, agents), then a suitable robot/agent can only be selected from

Algorithm 5 Adaptive multiple HART task scheduling

Notation: Number of full task request ($k = n_p + n_\delta$), physical (n_p)/ digital sub-task (n_δ), total robots ($q = h_r + c_r$)/agents ($z = h_a + c_a$), scheduling policy P =EDF, FCFS, and CP, robot(h_r)/agent(h_a) under host ONU-MPP, robot(c_r)/agent(c_a) under community ONU-MPP

1: **while** arrived task request $k \neq \varnothing$ **do**
2: **if** there exist multiple scheduling policy **then**
3: **for** each policy $p_i \in P$ **do**
4: go to step 10 to 25 for $t^k_{s,tot}$ calculation
5: **end for**
6: Compare $t^k_{s,tot}$ of all policy $\forall p_i \in P$
7: select optimal policy with minimum $t^k_{s,tot}$
8: Go to step 26
9: **end if**
10: **for** each arrived full task $\forall \gamma_i \in k$ **do**
11: **if** $\eta_i \leq 0$ **then**
12: select suitable robot $r \in h_r$ with min $(t^i_r \cap p^i_r)$
13: $t^i_{on} = t^i_r$,
14: **else if** $\eta_i > 0$ **then**
15: **for** each physical sub-task $l_i \in n_p$ **do**
16: select suitable robot $r \in q$ with min $(t^l_r \cap p^l_r)$
17: **for** each digital sub-task $v_i \in n_\delta$ **do**
18: select suitable agent $a \in z$ with min $(t_a \cap p_a)$
19: $t^i_{off} = t^l_r + t_a$,
20: **end for**
21: **end for**
22: $t^i_s = \min\{t^i_{on}, t^i_{off}\}$, where $i \in 1, 2, ...k$
23: $t^k_{s,tot} = \max\{t^1_s, t^2_s, ..., t^k_s\}$
24: **end if**
25: **end for**
26: **end while**

the available host ONU-MPP resources. If the number of arriving task requests is greater than than the number of available host ONU-MPP resources, then a suitable task processing node can be selected from both host and nearby ONU-MPP resources.

(*ii*) *Task Onloading:* In our task onloading scheme, a suitable host ONU-MPP robot is selected for full HART task execution without involving any digital sub-task offloading onto the agent (see Algorithm 5, lines 10-13). Note that in this case task onloading doesn't save power of the initially selected robot ($\eta_i \leq 0$).

2) Optimal Multi-Task Scheduling Order Selection: To optimize multi-task scheduling performance, the task scheduler decides which task is executed first. To determine the optimal multi-task scheduling order, the task scheduler compares the overall task service time ($t^k_{s,tot}$) of

the task offloading and onloading schemes using the following three scheduling policies (Alg. 5, lines 1-9):

(i) First Come First Served (FCFS): The task scheduler assigns resources to HART tasks based on their arrival order.

(ii) Earliest Deadline First (EDF): The task with the earliest deadline is scheduled first.

(iii) Concurrent Policy (CP): The task scheduler assigns resources randomly to all arriving task requests.

3) Task Processing Node Selection: For selecting a task processing node in our task offloading scheme, the task scheduler at the host ONU-MPP initially sends the full task request message to all robots. Their response messages to the task scheduler contain each robot's respective remaining energy, location, moving and processing speed information. After collecting all robots' response messages, the task scheduler first computes the physical sub-task service time (t_r^l) and power consumption (p_r^l) for each robot. Subsequently, it selects the most suitable robot with minimum t_r^l and p_r^l required for the given physical sub-task execution (Alg. 5, lines 14-16) in out task offloading scheme (η_i>0). To do so, we extend the DHCP protocol messages `Discover`, `Offer`, and `Ack` using their pad/reserved bits to include the aforementioned task request, robot/agent response, and robot/agent selection messages [104]. For assigning the digital sub-task to an agent, the task scheduler broadcasts a task offloading request message to all agents, which comprises the digital sub-task input and output data size as well as workload information. After receiving the task offloading request, each agent sends a response message back to the task scheduler, containing information about the agent's availability and task processing speed. The task scheduler calculates each agent's digital sub-

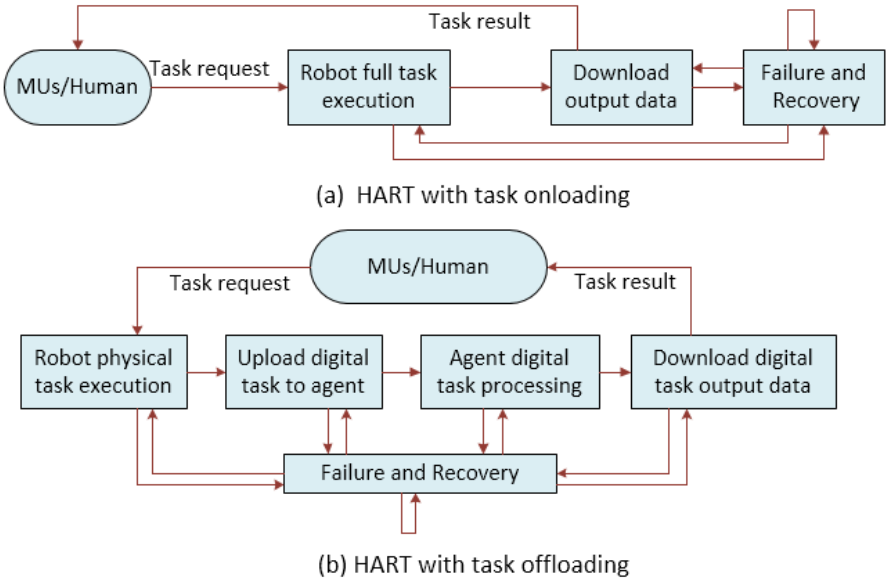


Figure 5.4: HART task execution model with failure avoidance.

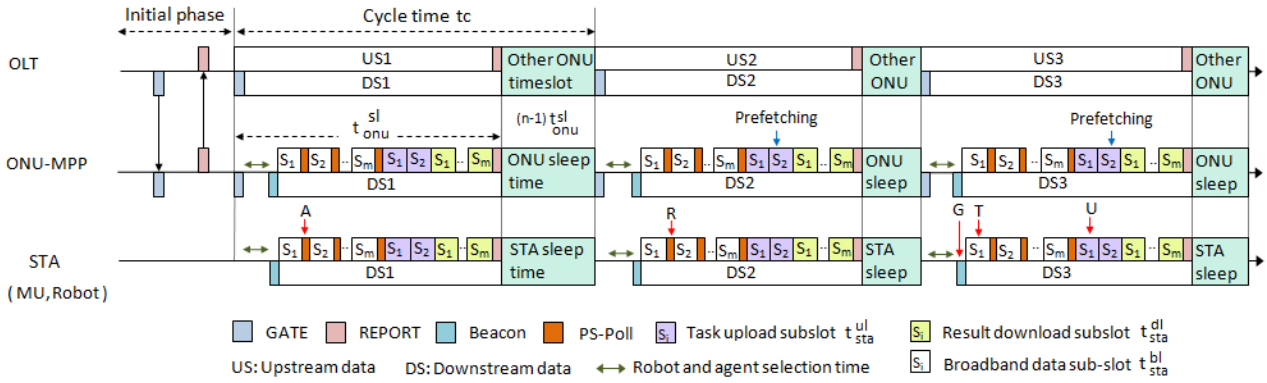


Figure 5.5: Space-time diagram of prefetching-aware dynamic bandwidth allocation (DBA).

task service time (t_a) and power consumption (p_a) and then selects the optimal agent with minimum t_a and p_a (Alg. 5, lines 17-21).

5.4.2.2 Minimization of Processing Overhead

Unlike previous work, our proposed task scheduling scheme performs robot/agent selection and bandwidth assignment simultaneously, thus reducing multi-task execution latency. Further, in this chapter, full HART task service time is calculated taking task workload processing, transmission, and waiting delay into account. To avoid extra task processing overhead due to failure, we design an *optimal failure avoidance* selection scheme, whereby task execution failures may occur due to unreachability of robots/agents. More specifically, in our task on-loading scheme, task execution failures may occur during a robot's full task processing and result transfer process. Conversely, in our task offloading scheme, task execution failures may happen during a robot's physical and an agent's digital sub-task processing or during the digital sub-task upload and result download process, as depicted in Fig. 5.4. To detect failures during task execution, the task scheduler at the ONU-MPP periodically broadcasts heartbeat messages to all robots/agents and waits for their responses at pre-defined checkpoints. The task scheduler is able to detect un-reachability failures of a robot/agent when heartbeat responses are absent at several subsequent checkpoints. For failure recovery, we apply a failure avoidance scheme that selects the minimum task service time t_s^i, which can be either a *failure recovery* or a *fault tolerance* scheme. In the failure recovery scheme, the faulty task execution restarts from beginning after recovering from a connection failure. In contrast, in the fault tolerance scheme, the faulty task execution resumes from the latest checkpoint rather than beginning (to be explained in further detail below in Sections 5.5.4 and 5.5.5).

Algorithm 6 Dynamic bandwidth allocation (DBA) at OLT

1: **for** each ONU-MPP $i \in n$ **do**
2: **if** the OLT receives `REPORT` message **then**
3: Extract ONU-MPP's bandwidth demand (t_{sl}^{req}) from `REPORT` and assign t_{onu}^{sl}=t_{sl}^{req}+$2t_{msg}^{pon} + RTT$
4: **else**
5: Assign minimum bandwidth $t_{onu}^{sl} = 2t_{msg}^{pon} + RTT$
6: **end if**
7: Generate and transmit a `GATE` message to ONU-MPP including their timeslot start time/duration (t_{onu}^{s}, t_{onu}^{sl}), broadband (t_{onu}^{bs},t_{onu}^{bl})/offload (t_{onu}^{os},t_{onu}^{ol}) slot start time/duration, t_{olt}^{clk}, polling cycle start time t_{cycle}^{s}
8: **if** ($t_{onu}^{bs} \leq t_{olt}^{clk} \leq t_{onu}^{bs} + t_{onu}^{bl}$) **then**
9: Receive US broadband frames from ONU-MPP
10: **else if** ($t_{onu}^{os} \leq t_{olt}^{clk} \leq t_{onu}^{os} + t_{onu}^{ol}$) **then**
11: Receive offload input/output data frames from ONU-MPP/agent, send to agent/ONU-MPP
12: **end if**
13: **end for**

5.4.3 Prefetching-Aware Dynamic Bandwidth Allocation

Fig. 5.5 depicts the space-time diagram of our prefetching-aware dynamic bandwidth allocation (DBA) algorithm. In both optical and wireless domains, bandwidth is allocated in a TDMA manner. Specifically, in the fiber backhaul, the OLT allocates bandwidth to each ONU-MPP via exchange of the IEEE 802.3ah MPCP messages `REPORT` and `GATE`. In the wireless front-end, each ONU-MPP allocates bandwidth to its associated stations (STAs), which comprise MUs and robots, by exchanging IEEE 802.11 WLAN `Beacon` and `PS-Poll` messages.

1. DBA operation at OLT: Initially, the OLT polls all ONU-MPPs and assigns minimal bandwidth for enabling each ONU-MPP's `REPORT` transmission (see Algorithm 6). In every recurring time cycle, the OLT receives the `REPORT` message from each ONU-MPP, indicating its current bandwidth demand (t_{sl}^{req}). Next, the OLT allocates bandwidth to each ONU-MPP based on its demand (t_{sl}^{req}) by sending a `GATE` to the ONU-MPP. The `GATE` message contains the ONU-MPP's broadband ($t_{onu}^{bs}, t_{onu}^{bl}$) and offload timeslot map ($t_{onu}^{os}, t_{onu}^{ol}$) for the next time cycle. During the ONU-MPP's broadband timeslot, the OLT receives/sends upstream/downstream (US/DS) broadband data frames from/to the ONU-MPP. Conversely, during the ONU-MPP's offload timeslot, the OLT receives/sends the offloaded task input/output data frames from and to the ONU-MPP/agent.

2. DBA operation at ONU-MPP: A given ONU-MPP starts its upstream transmission after receiving the `GATE` message from the OLT (see Algorithm 7). More precisely, the ONU-MPP extracts its broadband/offload transmission window information from thes `GATE`

Algorithm 7 DBA operation at ONU-MPP

1: **if** the ONU-MPP receives `GATE` message from OLT **then**
2: Extract ONU-MPP's timeslot start time/length (t^{s}_{onu},t^{sl}_{onu}), broadband (t^{bs}_{onu},t^{bl}_{onu}) and offload (t^{os}_{onu}, t^{ol}_{onu}) transmission start time/length, and t^{clk}_{olt}, synchronize ONU-MPP clock time with OLT (t^{clk}_{onu}=t^{clk}_{olt})
3: Determine ONU-MPP's sleep time (t^{ss}_{onu}=t^{s}_{onu}+t^{sl}_{onu}, t^{sd}_{onu}=t^{s}_{cycle}-t^{sl}_{onu}), wake-up time (t^{ws}_{onu}=t^{ss}_{onu}+t^{sd}_{onu}), robot/agent selection time/length (t^{s}_{onu}+t^{pon}_{msg}, t_{ra}), `REPORT` transmission time (t^{s}_{report}=t^{s}_{onu}+t^{sl}_{onu}-t^{pon}_{msg})
4: Assign broadband and offload sub-slot to STA's evaluating their last `PS-Poll`, t^{sl}_{onu}, task scheduling policy, and agents offload task processing start time (t^{s}_{a}), processing duration (t^{v}_{a}), and finish time (t^{f}_{a})
5: $t^{bs}_{sta_1} = t^{bs}_{onu}$, $t^{bl}_{sta_1} = t^{br}_{sta_1}$, $t^{us}_{sta_1} = t^{os}_{onu}$, $t^{ul}_{sta_1} = t^{ur}_{sta_1}$
6: $t^{ds}_{sta_1} = t^{us}_{sta_1} + t^{ul}_{sta_1} + t^{v}_{a_1}$, $t^{dl}_{sta_1} = t^{dr}_{sta_1}$
7: **for** i=2 to m **do**
8: $t^{bs}_{sta_i} = t^{bs}_{sta_{i-1}} + t^{bl}_{sta_{i-1}}$, $t^{bl}_{sta_i} = t^{br}_{sta_i}$
9: Go to step 10 for task offload sub-slot assignment
10: **if** $t^{ur}_{sta_i} > t^{v}_{a_{i-1}}$ **then**
11: Assign both prefetching based task upload sub-slot ($t^{\overline{us}}_{sta_i}, t^{\overline{ul}}_{sta_i}$) and fetching based remaining task data upload sub-slot ($t^{\widehat{us}}_{sta_i}, t^{\widehat{ul}}_{sta_i}$) with download sub-slot start time/duration ($t^{ds}_{sta_i}, t^{dl}_{sta_i}$)
12: $t^{\overline{us}}_{sta_i}$=$t^{s}_{a_{i-1}}$,$t^{\overline{ul}}_{sta_i}$=$t^{v}_{a_{i-1}}$, $t^{\widehat{us}}_{sta_i}$=$t^{ds}_{sta_{i-1}}$+$t^{dl}_{sta_{i-1}}$,
13: $t^{\widehat{ul}}_{sta_i}$=$t^{ur}_{sta_i} - t^{\overline{ul}}_{sta_i}$,$t^{ds}_{sta_i}$=$t^{f}_{a_i}$, $t^{dl}_{sta_i}$=$t^{dr}_{sta_i}$
14: **else if** $t^{ur}_{sta_i} \leq t^{v}_{a_{i-1}}$ **then**
15: Assign prefetching based task upload ($t^{\overline{us}}_{sta_i}, t^{\overline{ul}}_{sta_i}$) sub-slot and result download sub-slot ($t^{ds}_{sta_i}, t^{dl}_{sta_i}$) $t^{\overline{us}}_{sta_i}$=$t^{s}_{a_{i-1}}$, $t^{\overline{ul}}_{sta_i}$=$t^{ur}_{sta_i}$, $t^{ds}_{sta_i}$=$t^{f}_{a_i}$, $t^{dl}_{sta_i}$=$t^{dr}_{sta_i}$
16: **end if**
17: **end for**
18: Generate a `Beacon` message with STA's (MUs/robots) broadband sub-slot {($t^{bs}_{sta_1}$, $t^{bl}_{sta_1}$)...($t^{bs}_{sta_m}, t^{bl}_{sta_m}$)}, task upload {($t^{us}_{sta_1}, t^{ul}_{sta_1}$)..($t^{us}_{sta_m}, t^{ul}_{sta_m}$)}, and download sub-slot map {($t^{ds}_{sta_1}, t^{dl}_{sta_1}$)...($t^{ds}_{sta_m}, t^{dl}_{sta_m}$)}, $t^{\widehat{ul}}_{sta_i}, t^{\overline{ul}}_{sta_i} \in t^{ul}_{sta_i}$, t^{clk}_{onu}, t^{s}_{onu}, t^{sl}_{onu}, and send to STA's
19: **else if** ($t^{bs}_{onu} \leq t^{clk}_{onu} < t^{bs}_{onu} + t^{bl}_{onu}$) **then**
20: Receive US/DS broadband data frames from STAs/OLT and sends them to OLT/STAs
21: Extract STA's broadband ($t^{br}_{sta_i}$) and task offload bandwidth ($t^{ur}_{sta_i}, t^{dr}_{sta_i}$) request from `PS-Poll` message, update $t^{req}_{sl} \rightarrow t^{req}_{sl} + t^{br}_{sta_i} + t^{ur}_{sta_i} + t^{dr}_{sta_i} + t_{ra} + t^{msg}_{wl}$
22: **else if** ($t^{os}_{onu} \leq t^{clk}_{onu} < t^{os}_{onu} + t^{ol}_{onu}$) **then**
23: Receive offload task input/output frames from STAs/agents and transfers them to agents/STAs
24: **else if** ($t^{clk}_{onu} == t^{s}_{report}$) **then**
25: Send a `REPORT` message to OLT with t^{req}_{sl}
26: **end if**

message (lines 1-2, Alg. 7). Next, the ONU-MPP determines its next sleep start time/duration ($t^{ss}_{onu}, t^{sd}_{onu}$), wake-up time ($t^{ws}_{onu}$), and `REPORT` transmission (t^{s}_{report}) times (line 3). The ONU-

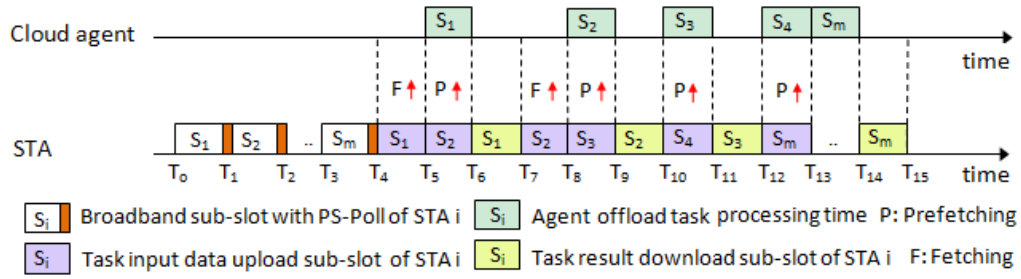


Figure 5.6: Illustration of prefetching-aware task offload sub-slot assignment.

MPP in turn assigns broadband and offload sub-slots to its associated STAs based on each STA's bandwidth request (`PS-Poll`), t^{sl}_{onu}, and agent offload task processing schedule ($t^{s}_{a}, t^{v}_{a}, t^{f}_{a}$). Next, STA_1's broadband sub-slot start time ($t^{bs}_{sta_1}$) and duration ($t^{bl}_{sta_1}$) are calculated according to the ONU-MPP's broadband slot start time ($t^{bs}_{sta_1}$=t^{bs}_{onu}) and STA_1's requested broadband sub-slot ($t^{bl}_{sta_1}$=$t^{br}_{sta_1}$), respectively (lines 5-6). STA_1's assigned task upload ($t^{us}_{sta_1}$) and result download sub-slot ($t^{ds}_{sta_1}$) start time, and task upload ($t^{ul}_{sta_1}$) and result download sub-slot ($t^{dl}_{sta_1}$) duration correspond to the ONU-MPP's offload slot start time (t^{os}_{onu}), STA_1, offload task processing completion time at agent ($t^{us}_{sta_1}$+$t^{ul}_{sta_1}$+$t^{v}_{a_1}$), STA_1's requested upload ($t^{ur}_{sta_1}$), and download ($t^{dr}_{sta_1}$) sub-slot length, respectively. Next, the ONU-MPP assigns a broadband sub-slot start time to the remaining STAs ($i = 2$ to m) ($t^{bs}_{sta_i}$) and length ($t^{bl}_{sta_i}$) according to the previous STA's broadband sub-slot ($t^{bs}_{sta_i}$=$t^{bs}_{sta_{i-1}}$+$t^{bl}_{sta_{i-1}}$) and requested broadband sub-slot ($t^{bl}_{sta_i}$=$t^{br}_{sta_i}$), respectively (lines 7-8). The ONU-MPP assigns the next STA's task offload sub-slot based on the earlier STA's offload task processing start time ($t^{s}_{a_{i-1}}$), agent offload task processing duration ($t^{v}_{a_{i-1}}$), and STA_i required task upload ($t^{ur}_{sta_i}$) duration (lines 9-17). If the next STA's required task upload sub-slot duration ($t^{ur}_{sta_i}$) is smaller than or equal to the previous STA's offload task processing (at agent) duration ($t^{v}_{a_{i-1}}$), then the ONU-MPP assigns a prefetching-based task upload sub-slot to the next STA (lines 14-15). If the next STA's required task upload sub-slot duration ($t^{ur}_{sta_i}$) is greater than the previous STA's offload task processing duration ($t^{v}_{a_{i-1}}$), the ONU-MPP assigns both task upload (prefetching) and task upload (fetching) sub-slots to the next STA (lines 10-13). The next STA's task upload (prefetching) sub-slot start time ($t^{\overline{us}}_{sta_i}$) and length ($t^{\overline{ul}}_{sta_i}$) are equal to the previous STA's offload task processing start time ($t^{\overline{us}}_{sta_i}$=$t^{s}_{a_{i-1}}$) and offload task processing duration (at agent) ($t^{\overline{ul}}_{sta_i}$=$t^{v}_{a_{i-1}}$), respectively, e.g., STA_2's prefetching subslot duration lasts from T_5 to T_6, as illustrated in Fig. 5.6. Note that the next STA's task upload (fetching) sub-slot start time ($t^{\widehat{us}}_{sta_i}$) and duration ($t^{\widehat{ul}}_{sta_i}$) are equal to the previous STA's task result download completion ($t^{\widehat{us}}_{sta_i}$=$t^{ds}_{sta_{i-1}}$+$t^{dl}_{sta_{i-1}}$) and next STA's remaining task data upload time ($t^{\widehat{ul}}_{sta_i}$=$t^{ur}_{sta_i} - t^{\overline{ul}}_{sta_i}$), respectively, e.g., STA_2's fetching subslot duration lasts from T_7 to T_8 in Fig. 5.6. The next STA's assigned task result download sub-slot start time ($t^{ds}_{sta_i}$) and duration correspond to the

next STA's offload task processing completion time (at agent) ($t^f_{a_i}$) and required task result download time ($t^{dr}_{sta_i}$), respectively.

The ONU-MPP sends a `Beacon` message to the STA in oder to inform it about its broadband and offload sub-slot information in the next cycle (line 18). The ONU-MPP receives US/DS broadband data frames from the STA/OLT and forwards them immediately during its broadband timeslot (lines 19-21). During its assigned offload timeslot, the ONU-MPP receives offloaded task input/output data frames from the STAs/agents and transfers them to the agents/STAs (lines 22-23). Finally, the ONU-MPP transmits a `REPORT` message to the OLT at end of its task offload sub-slot (lines 24-26).

5.5 Performance Analysis

In this section, we present our analytical model to evaluate the performance of the task offloading and task onloading schemes. The performance metrics of interest include mean task service delay, speed up, task pre-fetching efficiency, power saving, and satisfactory ratio, among others.

5.5.1 HART Task Service Time Analysis

In this subsection, we compute the multiple full HART task service delay for both proposed task offloading and task onloading schemes. The task service delay is the total time duration required for full HART task (physical and digital sub-tasks) workload processing by the assigned robot/agent and task result reception by the MU. Specifically, the full HART task service time calculation comprises the following three delay components: (*i*) physical and digital sub-task workload processing delay, (*ii*) transmission delay (Section 5.5.2 below), and (*iii*) waiting delay at robot/agent (Section 5.5.3 below). In the task onloading scheme, the selected host robot executes the full task and transfers the task result to the MU. Thus, the task service delay in the task onloading scheme (t^i_{on}=t^i_r) is given by

$$t^i_{on} = \hat{t}_r + t^r_w + t^d_{on} = t^l_r + t^v_r + t^r_w + t^{on}_b + t^{on}_{r\rightarrow mu}, \tag{5.1}$$

where t^r_w, $\hat{t}_r$, and t^d_{on} denote the waiting delay at the robot, the robot's full task workload processing delay, and the task result transmission delay (from robot to MU), respectively. Further, we have $\hat{t}_r$=t^l_r+t^v_r, where t^l_r and t^v_r represent the robot's physical sub-task (e.g., capturing image at task location) and digital sub-task (e.g., face recognition from captured image) processing time, respectively. Whereas t^l_r includes both the robot's task location traverse time (t^l_{tra}) and physical sub-task workload processing time (t^l_{ex}) and is given by t^l_r=t^l_{tra}+t^l_{ex}=$\frac{d_l}{s^r_m}$+$\frac{w_l}{s^r_p}$, where w_l, s^r_m, and s^r_p are the physical sub-task workload, the robot's

moving speed, and its computation processing speed, respectively; d_l denotes the Euclidean distance between the selected robot (x_p^r, y_p^r) and task location (x_p^l, y_p^l). The robot's digital sub-task processing time is denoted by $t_r^v{=}\frac{w_v}{s_p^r}$, where w_v and s_p^r stand for the digital sub-task workload and the robot's computation processing speed, respectively.

In the task offloading scheme, the selected robot executes only the physical sub-task and offloads the digital sub-task to a suitable agent for processing. Thus, the full HART task service time (t_{off}^i) in the task offloading scheme is given by

$$t_{off}^i = t_{r,a} + t_{\hat{w}} + t_{off}^d = t_r^l + t_a^v + t_{\hat{w}} + t_b^{off} + t_{r\to mu}^{off}, \tag{5.2}$$

where $t_{r,a}$, $t_{\hat{w}}$, and t_{off}^d denote the full HART task processing delay, the robot/agent's waiting delay ($t_{\hat{w}}{=}t_w^r{+}t_w^a$), and the transmission delay for task offloading. We have $t_{r,a}{=}t_r^l{+}t_a^v$, where t_r^l and t_a^v denote the robot's physical and the agent's digital sub-task processing time, respectively; t_a^v is given by $t_a^v{=}\frac{w_v}{s_p^a}$, where w_v and s_p^a are the digital sub-task workload and the agent's computation processing speed, respectively. Note that the main difference between the task onloading and task offloading schemes is their digital sub-task execution time. Given the digital sub-task execution time in task onloading ($t_{\tilde{r}}{=}t_{on}^i$-t_r^l-t_w^r) and task offloading ($t_a{=}t_{off}^i$-t_r^l-t_w^r), the digital sub-task execution time efficiency in the task offloading scheme is computed as $\beta_i{=}\frac{t_{\tilde{r}}-t_a}{t_{\tilde{r}}} \times 100\%$. The optimal task execution policy is determined by using the minimum full task service time of the task onloading (t_{on}^i) and offloading (t_{off}^i) schemes: $t_s^i{=}\min\{t_{on}^i, t_{off}^i\}$. The overall task service time for processing task k is then given by $t_{s,tot}^k{=}\max\{t_s^1, t_s^2.., t_s^k\}$. Next, given t_{on}^i and t_{off}^i, the mean task service time for multiple HART task service is calculated for the task onloading ($t_{s,a}^k{=}t_{on,a}^k$) and task offloading ($t_{s,a}^k{=}t_{off,a}^k$) schemes as follows: $t_{s,a}^k = \sum_{i=1}^k t_s^i \cdot \frac{1}{k}$. Finally, if both $t_{on,a}^k$ and $t_{off,a}^k$ are known, the mean task service time delay saving ratio ($\hat{\gamma}{=}\frac{t_{on,a}^k - t_{off,a}^k}{t_{on,a}^k} \times 100\%$) between task offloading and task onloading can be obtained.

5.5.2 Transmission Time Delay

In addition to the task workload processing delay, the full HART task execution involves transmission delay. In the task onloading scheme, the transmission delay ($t_{on}^d{=}t_b^{on}{+}t_{r\to mu}^{on}$) includes the task result buffering delay (t_b^{on}) and task result transfer ($t_{r\to mu}^{on}$) delay (from robot to MU), whereby t_{on}^d is given by $t_{on}^d = t_b^{on} + t_{r\to mu}^{on} = t_b^{on} + \frac{d_v^o}{c_{r\to\bar{o}}^w} + \frac{d_v^o}{c_{\bar{o}\to mu}^w} + t_{r\to mu}^p$, where d_v^o, $c_{r\to\bar{o}}^w$, $c_{\bar{o}\to mu}^w$, and $t_{r\to mu}^p$ denote digital task output data size, transmission capacity at link between robot and host ONU-MPP, host ONU-MPP and MU, and propagation delay for task result transfer, respectively. In task offloading scheme, transmission delay ($t_{off}^d{=}t_b^{off}{+}t_{r\to mu}^{off}$) consists offload packet buffering (t_b^{off}) and communication delay ($t_{r\to mu}^{off}{=}t_{r\to a}^u{+}t_{a\to mu}^d$), where

$t^u_{r\to a}$ and $t^d_{a\to mu}$ denote the digital task upload (from robot to agent) and result download delay (from agent to MU), respectively; $t^u_{r\to a}$ and $t^d_{a\to mu}$ are computed as follows:

$$t^u_{r\to a} = \begin{cases} \frac{d^i_v}{c^w_{r\to\bar{o}}} + \frac{d^i_v}{c^f_{\bar{o}\to a}} + t^p_{r\to a} & \text{if } a = h_{ct}, \\ \frac{d^i_v}{c^w_{r\to\bar{o}}} + \frac{hd^i_v}{c^f_{\bar{o}\to o}} + \frac{d^i_v}{c^f_{o\to a}} + t^p_{r\to a} & \text{if } a = c_{ct}, \\ \frac{d^i_v}{c^w_{r\to\bar{o}}} + \frac{d^i_v}{c^w_{\bar{o}\to a}} + t^p_{r\to a} & \text{if } a = h_{pr}, \\ \frac{d^i_v}{c^w_{r\to\bar{o}}} + \frac{hd^i_v}{c^f_{\bar{o}\to o}} + \frac{d^i_v}{c^w_{o\to a}} + t^p_{r\to a} & \text{if } a = c_{pr}, \end{cases} \tag{5.3}$$

$$t^d_{a\to mu} = \begin{cases} \frac{d^o_v}{c^f_{a\to\bar{o}}} + \frac{d^o_v}{c^w_{\bar{o}\to mu}} + t^p_{a\to mu} & \text{if } a = h_{ct}, \\ \frac{d^o_v}{c^f_{a\to o}} + \frac{hd^o_v}{c^f_{o\to\bar{o}}} + \frac{d^o_v}{c^w_{\bar{o}\to mu}} + t^p_{a\to mu} & \text{if } a = c_{ct}, \\ \frac{d^o_v}{c^w_{a\to\bar{o}}} + \frac{d^o_v}{c^w_{\bar{o}\to mu}} + t^p_{a\to mu} & \text{if } a = h_{pr}, \\ \frac{d^o_v}{c^w_{a\to o}} + \frac{hd^o_v}{c^f_{o\to\bar{o}}} + \frac{d^o_v}{c^w_{\bar{o}\to mu}} + t^p_{a\to mu} & \text{if } a = c_{pr}, \end{cases} \tag{5.4}$$

where h_{ct}, c_{ct}, h_{pr}, and c_{pr} stand for cloudlet (host), cloudlet (community), peer robot (host), and peer robot (community), respectively. Further, d^i_v, d^o_v, $c^f_{\bar{o}\to a}/c^w_{\bar{o}\to a}$, $c^f_{\bar{o}\to o}/c^f_{o\to\bar{o}}$, $c^f_{a\to\bar{o}}/c^w_{a\to\bar{o}}$, h, $t^p_{r\to a}$, and $t^p_{a\to mu}$ denote the digital task input and output data size, transmission capacity of the link between host ONU-MPP and agent (fiber/wireless link), host and community ONU-MPP (fiber link), agent and host ONU-MPP (fiber/wireless link), hop distance between host and community ONU-MPP, and propagation delay during digital task up- and downloading, respectively.

Next, we analyze the transmission packet buffering delay (t_b) that consists of three delay components in task offloading (t^{off}_b) and onloading (t^{on}_b) schemes. The first delay component (t_{d1}) is the time interval between the arrival (A) of transmission packet and transmission of bandwidth reservation (R) request (see also Fig. 5.5). If the transmission packet arrives after STA's broadband sub-slot, t_{d1} is equal to polling cycle time t_c. The second delay component (t_{d2}) is the time interval between the bandwidth request (R) and grant message (G). For STA_1, t_{d2} is equal to $t_c - t^{bl}_{sta_i}$. In the task onloading scheme, the third delay component (t^b_{d3}) is equal to the time interval between the bandwidth grant message (G) and broadband transmission (task result transfer) start time (T). Hence, in the task offloading scheme, the third delay component (t^o_{d3}) equals the time interval between the bandwidth grant message (G) and offload start time (U). For STA_1, t^o_{d3} is equal to $\sum_{i=1}^{m} t^{bl}_{sta_i}$. Summing up all three delay components, the packet buffering delay in the task onloading and offloading schemes is given by $t^{off}_b = t_{d1}+t_{d2}+t^o_{d3}$ and $t^{on}_b = t_{d1}+t_{d2}+t^b_{d3}$, respectively.

5.5.3 Waiting Time Delay

Beside task processing and transmission delay, each task execution may experience a certain waiting time delay before getting access and being served by the selected agent/robot [22]. To

calculate the average waiting delay, we model the robot/agent is as a M/M/c queue, where c denotes the number of processors available at each robot/agent for processing the assigned task request. In our M/M/c queue model, the occupancy rate of a robot/agent server is given by $\tau{=}\frac{\omega}{c\cdot\mu}$, where ω is the task arrival rate and μ is the service rate of the robot/agent. With c and τ, the waiting probability at a robot/agent (λ) is obtained as follows:

$$\lambda = \frac{(c\cdot\tau)^c}{c!}\cdot\left((1-\tau)\sum_{i=0}^{c-1}\frac{(c\cdot\tau)^i}{i!}+\frac{(c\cdot\tau)^c}{c!}\right)^{-1}. \tag{5.5}$$

By using λ, τ, c, and μ, the average waiting delay (t_w^ϵ) for task processing at a robot ($\epsilon = r$)/agent ($\epsilon = a$) is then given by $t_w^\epsilon = \lambda\cdot(1-\tau)^{-1}\cdot(c\cdot\mu)^{-1}+\frac{1}{\mu}$.

5.5.4 Task Service Time With Failure Recovery

For failure recovery during the HART task processing procedure, we assume that a connectivity failure (robot/agent) is detected by the task scheduler as soon as it has occurred and the task execution process needs to be relaunched [114]. Hence, the full HART task service time ($t_s^i{=}\hat{t}_{on}$) with task onloading is given by

$$\hat{t}_{on}^i = \begin{cases} t_{on}^i, & \text{if } F \geq t_{on}^i, \\ F + d^* + R + \bar{t}_r + t_b^{on} + t_{r\to mu}^{on}, & \text{otherwise,} \end{cases} \tag{5.6}$$

where F denotes the failure duration, d^* is the disconnection duration, $\bar{t}_r$ is the HART task workload processing time with waiting delay ($\bar{t}_r{=}\hat{t}_r{+}t_w^r$), and R is the failure recovery time for nested failure $R_i{=}F{+}R_{i-1}$. If $F \geq t_{on}^i$, the full task is executed without any failure and the service time equals t_{on}^i. If $F < t_{on}^i$, the robot connection failure occurs before the full task is executed. In this case, the task needs to be started from beginning at another robot and the service time equals $F{+}d^*{+}R{+}\bar{t}_r{+}t_b^{on}{+}t_{r\to mu}^{on}$. Hence, the task service time ($t_s^i{=}\hat{t}_{off}^i$) with failure recovery in the task offloading scheme is given by

$$\hat{t}_{off}^i = \begin{cases} t_{off}^i, & \text{if } F \geq t_{off}^i \\ F + d^* + R + t_{r^*}^l + t_{a^*}^v + t_{off}^d, & \text{otherwise,} \end{cases} \tag{5.7}$$

where $t_{r^*}^l$ and $t_{a^*}^v$ represent the robot's physical sub-task ($t_r^l{+}t_w^r$) and the agent's digital sub-task ($t_a^v{+}t_w^a$) processing time. If $F \geq t_{off}^i$, the full HART task completely executed without any failure and the task service time equals t_{off}^i. Conversely, if $F < t_{off}^i$, the failure occurs before full task execution is complete. The recovered full task execution again starts from beginning in the task offloading scheme ($\hat{t}_{off}^i = F + d^* + R + t_{r^*}^l + t_{a^*}^v + t_{off}^d$).

5.5.5 Task Service Time With Fault-Tolerance

With our fault-tolerance policy, after recovering from a connection failure recovered the HART task execution restarts from the latest checkpoint, rather than from beginning as done in [114]. Thus, the task service time (t_s^i=$\bar{t}_{on}^i$) with fault tolerance in the task onloading scheme is given by

$$\bar{t}_{on}^i = \begin{cases} t_{on}^i, & \text{if } F \geq t_{on}^i \\ \omega_{on}^i, & \text{if } F < \bar{t}_r \\ \phi_{on}^i, & \text{if } \bar{t}_r + t_b^{on} \leq F < \bar{t}_r + t_b^{on} + t_{r\to\bar{o}}^{on} \\ \psi_{on}^i, & \text{if } \bar{t}_r + t_b^{on} + t_{r\to\bar{o}}^{on} \leq F < t_{on}^i, \end{cases} \tag{5.8}$$

where $t_{r\to\bar{o}}^{on}$ is robot's task result transfer time to the associated ONU-MPP. If $F \geq t_{on}^i$, full task execution is completed without failure ($\bar{t}_{on}^i = t_{on}^i$). If $F < \bar{t}_r$, a connectivity failure occurs before full task processing is done by the robot. For failure avoidance, the full task is executed by an initially selected robot after connection has been regained, translating into a task service time equal to $\omega_{on}^i = F + d^* + R + \bar{t}_r - \bar{t}_r^* + t_b^{on} + t_{r\to mu}^{on}$, where $\bar{t}_r^*$ denotes the portion of task that has been already processed before the failure occurs. If a robot connection failure happens during the task result transfer, the full task service time with fault tolerance is given by $\phi_{on}^i = \bar{t}_r + t_b^{on} + \frac{\bar{d}_v^o}{c_{r\to\bar{o}}} + d^* + R + \frac{d_v^o - \bar{d}_v^o}{c_{r\to\bar{o}}^*} + t_{\bar{o}\to mu}^{on}$, where d^*, R, $\bar{d}_v^o$, d_v^o, $c_{r\to\bar{o}}$, and $c_{r\to\bar{o}}^*$ denote the disconnection duration, failure recovery time (process re-start), already transferred task output before failure, total output data size, and transmission capacity of the link between robot (r) and ONU-MPP ($\bar{o}$) before and after failure, respectively. If a connection failure occurs during task result transfer from ONU-MPP ($\bar{o}$) to MU, the task service time with fault tolerance is given by $\psi_{on}^i = \bar{t}_r + t_b^{on} + t_{r\to\bar{o}}^{on} + \frac{\bar{d}_v^o}{c_{\bar{o}\to mu}} + d^* + R + \frac{d_v^o - \bar{d}_v^o}{c_{\bar{o}\to mu}^*}$, where $c_{\bar{o}\to mu}$ and $c_{\bar{o}\to mu}^*$ represent the transmission capacity of the link between ONU-MPP and MU before and after failure, respectively.

The HART task service time (t_s^i=$\bar{t}_{off}^i$) in the task offloading scheme with fault tolerance is computed as follows:

$$\bar{t}_{off}^i = \begin{cases} t_{off}^i, & \text{if } F \geq t_{off}^i \\ \omega_{off}^i, & \text{if } F < t_{r^*}^l \\ \phi_{off}^i, & \text{if } t_{r^*}^l + t_b^{off} \leq F < t_{r^*}^l + t_b^{off} + t_{r\to a}^u \\ \chi_{off}^i, & \text{if } t_{r^*}^l + t_b^{off} + t_{r\to a}^u \leq F < t_{off}^i - t_{a\to mu}^d \\ \psi_{off}^i, & \text{if } t_{r^*}^l + t_b^{off} + t_{r\to a}^u + t_{a^*}^v \leq F < t_{off}^i. \end{cases} \tag{5.9}$$

If $F \geq t_{off}^i$, the full task is completed without failure ($\bar{t}_{off}^i = t_{off}^i$). If the selected robot's connection fails before the physical sub-task is completed, task execution is restarted by the initially selected robot after connection recovery. In this case, the task service time equals $\omega_{off}^i = t_{r^*}^l - \hat{t}_{r^*}^l + F + d^* + R + t_{r\to a}^u + t_{a^*}^v + t_{a\to mu}^d$, where $\hat{t}_{r^*}^l$ is the already processed portion of

the physical sub-task before failure. If the connection failure occurs during the digital sub-task input data transfer to the agent, the process restarts after re-establishing the connection. Thus, the task service time (ϕ^i_{off}) is given by $\phi^i_{off} = t^l_{r^*} + t^{off}_b + \frac{\bar{d}^i_v}{c_{r\to a}} + d^* + R + \frac{d^i_v - \bar{d}^i_v}{c_{r\to a^*}} + t^v_{a^*} + t^d_{a\to mu}$, where $\bar{d}^i_v$, d^i_v, $c_{r\to a}$, and $c^*_{r\to a}$ denote the already uploaded and full digital sub-task input data size, transmission capacity of the link between robot and agent before and after failure, respectively. For recovery from a connection failure occurring during the digital sub-task execution at an agent, the digital sub-task processing again restarts at the selected agent after the connection has been re-established ($\chi^i_{off} = t^l_{r^*} + t^{off}_b + t^u_{r\to a} + d^* + R + t^v_{a^*} + t^d_{a\to mu}$). If a connection failure occurs during digital sub-task output data transfer (agent to MU), the full task service time is equal to ψ^i_{off}. $\psi^i_{off} = t^l_{r^*} + t^{off}_b + t^u_{r\to a} + t^v_{a^*} + \frac{\bar{d}^o_v}{c_{a\to mu}} + d^* + R + \frac{d^o_v - \bar{d}^o_v}{c^*_{a\to mu}}$, where $\bar{d}^o_v$, d^o_v, $c_{a\to mu}$, and $c^*_{a\to mu}$ denote the already downloaded and total output data size, transmission capacity of the link between agent and MU before and after failure, respectively.

5.5.6 Task Completion Time

The task completion delay is the time interval between the generation of a given MU's task request to the reception of the task result. Thus, both task request arrival time (from MU to task scheduler at host ONU-MPP) and task service time in the task onloading/offloading schemes need to be considered for computing the task completion delay. Depending on the MU's requested task location, the full HART task can be remote or nearby. For nearby tasks, both MU and task location are within same ONU-MPP coverage area. For remote tasks, both MU and task location are associated with different ONU-MPPs. Thus, the remote task request packet arrival time ($t^i_{rat} = t_b + t_{up} + t_{dn}$) comprises three delay components: (i) task request packet buffering delay at MU (t_b), (ii) upstream (MU to OLT), and (iii) downstream communication delay (OLT to ONU-MPP). The task request packet buffering delay ($t_b = t_{d1} + t_{d2} + t^b_{d3}$) refers to the time period between generation of the MU task request packet (A) and its transmission time (T) (see also Fig. 5.5 and Section 5.5.2 above). Note that the remote task request packet experiences a US packet delay ($t_{up} = t^u_{mu\to olt} + t^p_{mu\to olt}$) during task request transfer from MU to OLT and a DS packet delay (t_{dn}) during task request transfer from OLT to ONU-MPP ($t_{dn} = t^{dn}_d + t^d_{olt\to\bar{o}} + t^p_{olt\to\bar{o}}$), where $t^u_{mu\to olt}$, $t^d_{olt\to\bar{o}}$, $t^p_{mu\to olt}$, and $t^p_{olt\to\bar{o}}$ denote US/DS packet transmission and US/DS propagation delay, respectively, and t^{dn}_d is the waiting delay experienced at the OLT. If the task request packet arrives at OLT immediately after its GATE transmission, t^{dn}_d is given by $t^{dn}_d = t_c - t^{sl}_{onu}$, where t_c is the polling cycle time and t^{sl}_{onu} is the ONU-MPP's time-slot length. Given a non-data traffic duration of $1 - u_d$, t_c is given by $t_c = \frac{n\cdot(t_{ra} + t_{ba} + t_{mpcp} + m\cdot t_{rd})}{1 - u_d}$, where n, m, u_d, $\overline{\lambda}$, $\overline{X}$, t_{mpcp}, RTT, t_{ra}, t_{rd}, and t_{ba} denote the number of ONU-MPPs in our FiWi network, STAs associated with each ONU-MPP, traffic load ($u_d = \overline{\lambda X}$), traffic arrival rate and first moment of the packet service time at the

ONU-MPP, MPCP message exchange time ($2t^{pon}_{msg}$+RTT), round-trip time (RTT=$2t^{p}_{olt\rightarrow\bar{o}}$), robot/agent selection (t_{ra}=t_{rs}+t_{as}), STA's reservation (t_{rd}=t^{wl}_{msg}+t_g), and bandwidth grant duration (t_{ba}=t^{wl}_{msg}), respectively. Similarly, we calculate the nearby task request packet arrival time (t^i_{nat}=t^*_{up}+t_b), consisting of both transmission packet buffering (t_b) delay and US (MU to ONU-MPP) communication delay (t^*_{up}=$t^u_{mu\rightarrow onu}$+$t^p_{mu\rightarrow onu}$). Using task request arrival and task service time (t^i_s), we are able to compute the remote ($t^i_{tct,r}$=t^i_{rat}+t^i_s) and nearby ($t^i_{tct,n}$=t^i_{nat}+t^i_s) task completion delay. For multiple HART tasks ($i = 1, 2, .., k$), we also compute the overall remote ($\bar{t}^k_{tct,r}$=max$\{t^1_{tct,r}, .., t^k_{tct,r}\}$) and nearby task completion delay ($\bar{t}^k_{tct,n}$=max$\{t^1_{tct,n}, .., t^k_{tct,n}\}$).

5.5.7 Power Saving Ratio

The power consumption of STAs (i.e., MUs and robots) in the task onloading (p^i_{on}) and offloading (p^i_{off}) schemes can be computed as follows:

$$p^i_s = \begin{cases} p^l_{tra} + p^l_{ex} + p^v_r + p^{on}_{od} + p^{on}_{r\rightarrow mu}, & \text{if } s = on \\ p^l_r + p^{off}_{od} + p^v_a + p^{off}_{r\rightarrow mu}, & \text{if } s = off. \end{cases} \tag{5.10}$$

In the task onloading scheme, the selected robot executes the full HART task. Thus, a given STA's power consumption (p^i_{on}=p^i_r) is given by p^i_{on}=p^l_{tra}+p^l_{ex}+p^v_r+p^{on}_{od}+$p^{on}_{r\rightarrow mu}$, where p^l_{tra}, p^l_{ex}, p^v_r, p^{on}_{od}, and $p^{on}_{r\rightarrow mu}$ denote the power consumption for a given robot's task location traversing (p^l_{tra}=e_m·t^l_{tra}), physical (p^l_{ex}=e_l·t^l_{ex}) and digital sub-task processing (p^v_r=e_r· t^v_r), overhead delay (p^{on}_{od}=e_{idle}·t^{on}_b), and task result reception by the MU ($p^{on}_{r\rightarrow mu}$=e_{rx}· $t^{on}_{r\rightarrow mu}$), respectively. The STA's power consumption in the task offloading scheme is given by p^i_{off}=p^l_r+p^{off}_{od}+p^v_a+$p^{off}_{r\rightarrow mu}$, where p^l_r, p^v_a, p^{off}_{od},$p^{off}_{r\rightarrow mu}$ denotes the power consumption for a given robot's physical (p^l_r=p^l_{tra}+p^l_{ex}) and an agent's digital sub-task processing (p^v_a=e_{idle}· t^v_a), offloading buffering (p^{off}_{od}=e_{idle}·t^{off}_b), and communication delay ($p^{off}_{r\rightarrow mu}$=$e_{tx}$· $t^u_{r\rightarrow a}$+e_{rx}·$t^d_{a\rightarrow mu}$), respectively. Further, the power consumption efficiency (η_i) of digital sub-task offloading over task onloading is equal to $\eta_i = \frac{p_r - p_a}{p_r} \times 100\%$, where p_r and p_a represent the power consumption of the digital sub-task in task onloading (p_r=$p^i_{on} - p^l_r$) and offloading (p_a=p^i_{off}-p^l_r) schemes, respectively. The total power consumption (p^k_s) of task k in the task onloading and offloading schemes equals p^k_{on}=$\sum_{i=1}^{k} p^i_{on}$ and p^k_{off}=$\sum_{i=1}^{k} p^i_{off}$, respectively. Finally, the power saving ratio (σ) for executing task k via task offloading over onloading is given by $\sigma = \frac{p^k_{on} - p^k_{off}}{p^k_{on}} \times 100\%$.

5.5.8 Task Prefetching Time Efficiency and TSO Ratio

Next, let us analyze the task prefetching time efficiency and task service time gain to overhead ratio (TSO). The task prefetching time efficiency (p_g) is defined as the ratio of the digital sub-task execution time gain obtained from our proposed prefetching-aware offloading scheme

($t_{\hat{a}}$ =max$\{t_a^1, t_a^2 ... t_a^k\}$) and the digital sub-task execution time of the conventional fetching ($t_{a^*} = \sum_{i=1}^{k} t_a^i$) based offloading scheme: $p_g = \frac{t_{a^*} - t_{\hat{a}}}{t_{a^*}} \times 100\%$. Hence, the TSO ratio ($\delta_i$) is obtained by comparing the HART task service time gain in our proposed task offloading scheme ($t_{on}^i - t_{off}^i$) to the task onloading scheme and communication overhead cost in the task offloading scheme ($t_b^{off} + t_{r \to mu}^{off}$):

$$\delta_i = \frac{t_{on}^i - t_{off}^i}{t_b^{off} + t_{r \to mu}^{off}}. \tag{5.11}$$

5.5.9 Satisfactory Ratio, Speed Up, and PSR

We conclude our analysis by introducing three more important performance metrics: satisfactory ratio, speed up, and processing-to-service time ratio (PSR). The satisfactory ratio s_f is defined as the ratio of number of full HART tasks that meet their task execution deadlines in the task onloading/offloading scheme and the total number of arriving HART task requests: $s_f = (1 - \frac{n_f}{n_s + n_f}) \times 100\%$, where n_s and n_f denote the number of tasks that meet and miss their task deadlines, respectively. The speed up ratio is defined as the ratio of sequential task service time (t_s^*) and parallel (t_p^*) task service time and is given by $s_u = \frac{t_s^*}{t_p^*} = \frac{\sum_{i=1}^{k} t_s^i}{t_{s,tot}^k}$. Finally, the PSR is calculated by taking the ratio of HART task workload processing time and its service time. If the task workload processing (t_p^i) and service time (t_{on}^i) are known, the mean PSR for multiple tasks (k) is calculated in the task onloading scheme ($\hat{\mu}_s$=$\hat{\mu}_{on}$) as $\hat{\mu}_{on} = \frac{1}{k} \cdot \sum_{i=1}^{k} \frac{t_p^i}{t_{on}^i} = \frac{1}{k} \cdot \sum_{i=1}^{k} \frac{t_{on}^i - t_w^r - t_{on}^d}{\hat{t}_r + t_w^r + t_{on}^d}$. Similarly, the mean PSR in the task offloading scheme ($\hat{\mu}_{off}$) is obtained as $\hat{\mu}_{off} = \frac{1}{k} \cdot \sum_{i=1}^{k} \frac{t_{p^*}^i}{t_{off}^i} = \frac{1}{k} \cdot \sum_{i=1}^{k} \frac{t_{off}^i - t_{\bar{w}} - t_{off}^d}{t_{r,a} + t_{\bar{w}} + t_{off}^d}$.

5.6 Results

In this section we present results of our proposed community- and latency-aware task offloading scheme and compare it with the following three baseline schemes: (i) task onloading, (ii) random and (iii) communication-aware task offloading [25]. Table 5.1 lists the parameters and their default values taken from [17], [28], [30], [94].

System settings and configurations: In our work, each full HART task consists of a physical (location dependent image capturing) and a digital (location independent face detection from captured image) sub-task. For evaluating our proposed scheme, multiple full HART task workload, the offload (digital) sub-task input/output data size, and full task deadline values are chosen randomly and can be found in the caption of Fig. 5.7. The CPU clock speed of robots and cloudlets are set to 500 and 3200 MHz, respectively. The IEEE 802.3av 10Gb/s EPON based optical backhual (20 Km long) connects central office (OLT) with ONU's. The optical fiber range (distance) between the ONU-MPP and cloudlet server

Table 5.1: Parameters and default values for community- and latency-aware multi-task onloading and offloading scheme evaluation

Notation	**Description of Parameters**	**Default values/units**
w_l, w_v, d_v^i, d_v^o	Physical and digital sub-task workload, digital sub-task input and output data size	M cycles (vary), K-B (vary)
$e_{tx}, e_{rx}, e_{idle}, e_m, e_l, e_r$	STA's (MU/robot) average power consumption for task upload, result download, idle state, task location traversing, physical and digital sub-task workload processing	0.1W, 0.05W, 0.001W, .7W, .5W, .5W
$n, m, k, d_l, s_m^r, s_p^r, s_p^a$	Total ONU number, total STA in each ONU, total arrived task number, robot's distance to task location and moving speed, robot and agent computation processing speed	1-64, 1-100, 1-12, 1-500m, 1-100m/s, 500/3200 MHz
$t_c, c_a/c_r, h_a/h_r, u_d$	Polling cycle time, number of host and community agent/robot, FiWi traffic load	50-800 ms, 1-50, 1-50, 0-1
$c_{x\to y}^w, c_{x\to y}^f$	Transmission capacity of WLAN and fiber link (x,y=agent,robot,mu,OLT,ONU-MPP)	6900 Mbps, 10 G-b/s
$p_s^k, d^*, R, c_{x\to y}, c_{x\to y}^*$	Total power consumption (STA's), disconnection duration and connection failure recovery time, transmission capacity link between sender and receiver before and after failure	J (vary), s (vary), Mbps (vary)
$t_{msg}^{wl}, t_{msg}^{pon}, t_{ra}, t_g, h$	WLAN (e.g., `PS-Poll`), and MPCP message length(`GATE,REPORT`), robot/agent selection delay, guard time, hop distance (host and community ONU-MPP)	0.512μs, 0.231μs, μs, 46 μs, 1-4 (vary)
$t_{olt\to\bar{o}}^p, t_{\bar{o}\to o}^p, t_{mu\to onu}^p$	Propagation delay between OLT and ONU-MPP, inter ONU-MPP, MU and ONU-MPP	0.02 ms, .001 ms, .00033 ms
$\delta_i, \hat{\mu}_s, a_{onu}^h, a_{onu}^c$	TSO ratio and PSR ratio, coverage area of host and community cluster ONU-MPP	% (vary), 100 m, 500 m
$\omega/\mu/c, t_{rq}, t_{res}, t_{ack}, t_{r\to a}^p, t_{a\to mu}^p$	Task request arrival rate/service rate/number of processor per agent/robot, task request/response/ack message duration, Propagation delay (robot→agent, agent→MU)	1-12 (vary), 0.17/0.12/0.12, μs (vary)

is 1 Km, whereas the fiber range between the remote cloud server and OLT is 10 Km. Note that, in this work ONU's are inter-connected by interconnected fiber links. The requirements of the requested HART task execution is to satisfy the task execution deadline requirement and improve the end-to-end latency. The total number of ONU-MPPs, STAs under each ONU-MPP, and generated task request are varied within the range of 1-64, 1-100, and 1-12, respectively. The length of `PS-Poll` and MPCP messages (`GATE,REPORT`) are set to 20 bytes (t_{msg}^{pon}=.23μs) and 64 bytes (t_{msg}^{wl}=.51μs), respectively. FiWi traffic load and polling cy-

cle time are varied from 0.3-0.8 and 50-800 ms, respectively. The transmission capacity of WLAN and fiber links are set to 6900 Mbps and 10 Gb/s, respectively. Other parameters and default values are given in Figs. 5.7 and 5.8. The specific system settings for scenario 1 (h_r=20,c_r=10,h_a=20,c_a=10) and scenario 2 (h_r=4,c_r=8,h_a=4,c_a=8) with multiple arrived task properties is given by: w_l=50, 25, 100, 75, 150, 125, 175, 200, 250, 225, 300, 275M cycles, w_v=200, 100, 400, 300, 600, 500, 700, 800, 1000, 900, 1200, 1100M cycles, t_d=4, 3.5, 5, 4.5, 6, 5.5, 6.5, 7, 8, 7.5, 9, 8.5s, d_v^i=200, 100, 400, 300, 600, 500, 700, 800, 1000, 900, 1200, 1100KB, d_v^o=80, 40, 160, 120, 240, 200, 280, 320, 400, 360, 480, 440KB.

Performance evaluation: First, let us evaluate the mean task service time and total power consumption performance for scenario 1 described in Fig. 5.7(a) and (b). Note that in scenario 1, the number of robots (h_r) and agents (h_a) available at the host ONU-MPP is higher than the number of arriving task requests (k). Both figures clearly indicate that both mean task service time ($t_{s,a}^k$) and total power consumption (p_s^k) increase with increasing task arrival number (k) in all compared schemes. We observe that the minimum mean task service time and power consumption are achieved in our proposed community- and latency-aware task offloading scheme, as opposed to the alternative schemes (i.e., traditional random and communication-aware task offloading schemes). This is due to the fact that in our proposed task offloading scheme, each physical and digital sub-task of a full task is processed by a suitable robot and agent (e.g., cloudlet), respectively. Further, each task is assigned to a suitable task processing node (host and community cluster robot/agent) by taking into account not only lower task workload processing time but also both incurred transmission and waiting delay. Furthermore, from Fig. 5.7(a) we observe that the task onloading scheme shows the second lowest mean task service time. This is because in the task onloading scheme, the full HART task is processed only by the selected host robot. Note that the host robot requires additional digital sub-task processing time due to its lower computation processing speed than the selected agents in the proposed task offloading scheme. Fig. 5.7(a) also shows that both traditional random and communication-aware task offloading schemes cannot improve the mean task service time of our proposed task offloading scheme due to their higher offload task processing overhead that includes offloading communication delay, digital sub-task processing time, and waiting delay. In the communication-aware task offloading scheme, the selected host robot performs the physical sub-task, while the digital sub-task is offloaded onto the nearest peer robot that resides within the host ONU-MPP coverage area. Conversely, in the random task offloading scheme, the physical sub-task is assigned to the selected host robot, whereas the digital sub-task is offloaded onto a randomly selected agent (host cloudlet or peer robot). Moreover, Fig. 5.7(b) shows that the task onloading scheme suffers from a higher STA power consumption than its counterparts. This is because in the task onloading scheme, the full

task is processed by the initially selected robot without any digital sub-task migration to the agent, which is the case in the alternative schemes.

Fig. 5.7(c) investigates the suitable multi-task scheduling order in our proposed task offloading scheme by comparing its mean task service time performance using three scheduling policies, namely, EDF, FCFS, and concurrent policies. Fig. 5.7(c) demonstrates that for a varying task number k, the deadline delay priority (EDF) based task scheduling policy outperforms both FCFS and concurrent scheduling policies in terms of mean service time delay and is thus suitable for our proposed task offloading scheme. Fig. 5.7(d) and (e) compare the mean task service time delay saving ($\hat{\gamma}$) and power saving ratio (σ) of the different task execution schemes. The figure reveals that for a varying task number k, our proposed task offloading scheme offers the highest delay and power saving ratio. For instance, for a typical task number k=12, our proposed task offloading scheme yields approximately a 19%, 59%, and 45% higher delay saving ratio than the task onloading, communication-aware, and random task offloading schemes, respectively. Hence, for k=12, the achieved power saving ratio of our proposed task offloading, communication-aware, random task offloading schemes and task onloading scheme is approximately 50%, 41%, and 45%, respectively.

Fig. 5.7(f) examines the satisfactory ratio of our proposed task offloading scheme in comparison with the alternative task execution schemes. The figure depicts that generally the satisfactory ratio (s_f) is higher for larger values of the task deadline (t_d). We notice that for both small and large t_d, our proposed task offloading scheme exhibits a satisfactory ratio superior to that of the compared schemes. Next, to demonstrate the impact of our task prefetching-aware bandwidth assignment scheme, Fig. 5.7(g) and (h) compares the task prefetching time efficiency (p_g) of our proposed task offloading scheme with prefetching with the alternative schemes for varying task number (k) and offload task input data size (d_v^i), respectively. Both figures clearly indicate that for both higher and lower task number and offload task input data size values, a task prefetching time efficiency is obtained in our proposed task offloading scheme with prefetching that is superior to that of the other schemes, including our task offloading scheme with conventional fetching, random, and communication-aware offloading. For instance, in Fig. 5.7(g) for task number set to $k = 10$, our proposed task offloading scheme with prefetching yields an approximately 25%, 65%, and 77% higher task prefetching time efficiency than the conventional fetching, random, and communication-aware offloading scheme, respectively. This is because unlike our proposed scheme, all alternative schemes rely on conventional fetching for offloading, thus suffering from a higher multi-task offloading latency.

Fig. 5.7(i) shows that more available host ONU-MPP robots can improve the mean task service time ($t_{s,a}^k$) performance of both proposed task offloading and task onloading schemes.

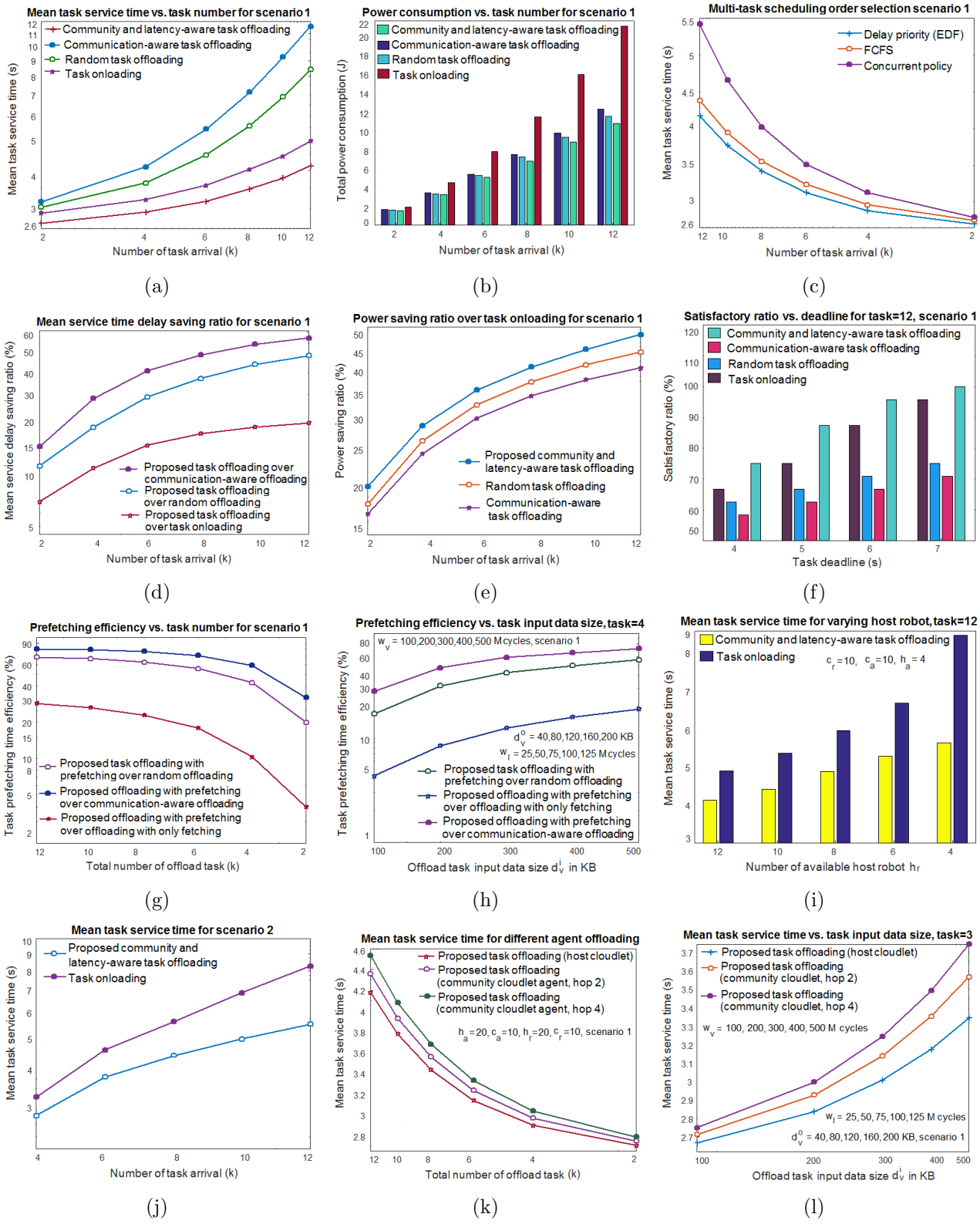


Figure 5.7: Mean task service time, power consumption, delay and power saving ratio, prefetching time efficiency, and satisfactory ratio evaluation for scenario 1 and scenario 2.

The figure indicates that the mean task service time delay difference between the proposed task offloading and onloading schemes increases rapidly with a decreasing host ONU-MPP robot number (h_r). We note that the proposed task offloading scheme that relies on the available host and community robots/agents significantly reduces the mean task service time, as opposed to the task onloading scheme that relies only on host ONU-MPP robots. Next, Fig. 5.7(j) sheds light on the mean task service time performance of both proposed task offloading and onloading schemes for scenario 2. In this scenario, the number of available host ONU-MPP robots (h_r) is fixed, being equal to or less than the varying task number (k). The figure depicts that with increasing task number (k), the task onloading scheme leads to a higher mean task service time than the proposed task offloading scheme. We observe that the mean task service time delay gain of task offloading over task onloading becomes lower when h_r is equal to arriving task requests (k). Note that the maximum mean task service time delay gain in our proposed task offloading scheme is achieved when the number of task arrivals (k) is much larger than the number of available host and community robots (c_r) to tackle the additional task request ($k - h_r$). Fig. 5.7(k) and (l) quantify the impact of host and community cloudlet (agent) selection on our proposed task offloading scheme. Both figures clearly indicate that the mean task service time ($t^k_{s,a}$) increases in all schemes for increasing number of offloaded tasks (k) and offload task input data size (d^i_v). We observe that when both host and community cloudlets are available, our proposed task offloading scheme results in a lower mean task service time for host cloudlet based digital sub-task execution compared to community cloudlets. For instance, in Fig. 5.7(l), for d^i_v set to 500 MB, task offloading with host cloudlet achieves a 6.17% and 10.69% higher mean task service delay saving than task offloading with community cloudlet (hop 2) and community cloudlet (hop 4), respectively. This is because the community cloudlet causes additional task offloading communication delay.

Fig. 5.8(a) examines the task service time gain to overhead (TSO) ratio (δ_i) of our proposed task offloading scheme with prefetching and task onloading scheme. We observe that with an increasing offload task input data size (d^i_v), the TSO ratio grows more rapidly in our proposed task offloading with prefetching scheme than its counterpart. For instance, for an offload task input data size of 500 MB, the TSO ratio for task offloading with prefetching and fetching scheme is approximately 32% and 21%, respectively. In Fig. 5.8(b), we investigate the optimal failure avoidance service selection for our proposed task offloading scheme. The figure shows that the mean task service time increases for increasing service connection recovery time (R) in all considered schemes. We also observe that our proposed task offloading scheme with fault tolerance achieves the lowest mean task service time delay. Note that both task offloading and task onloading schemes with failure recovery exhibit an inferior mean task service time than the schemes with fault tolerance. This is due to the fact that in the fault tolerance scheme,

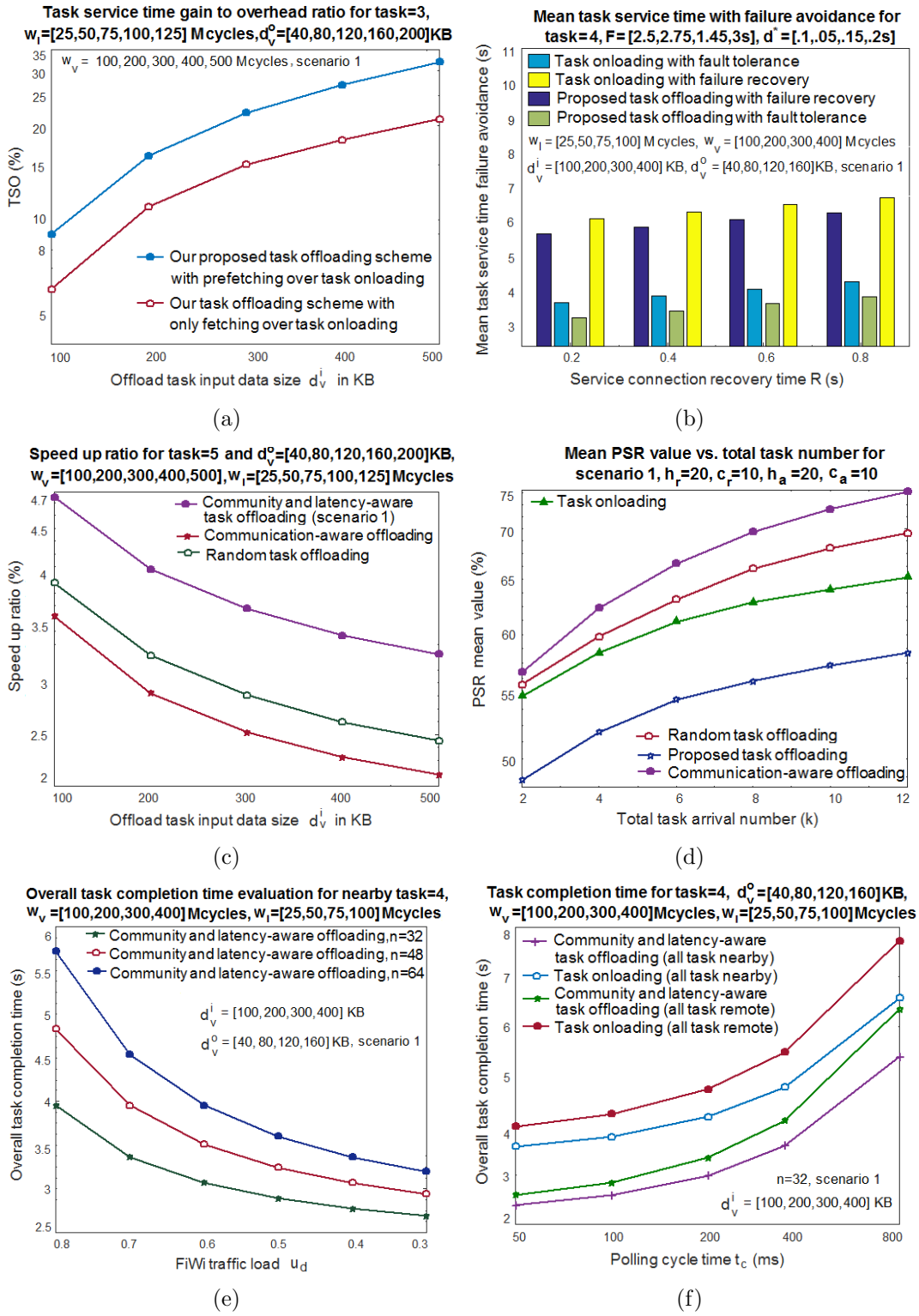


Figure 5.8: Task service time gain to overhead ratio (TSO), speed up ratio, mean task service time with failure avoidance, PSR, and overall task completion time evaluation of our proposed community and latency-aware task offloading scheme.

task execution re-starts from the last checkpoint after recovery from the connection failure. Fig. 5.8(c) reveals that for an increasing offload task input data size (d_v^i), the maximum speed up ratio (s_u) is obtained with our proposed community and latency-aware task offloading scheme. For instance, for d_v^i set to 100 MB, the speed up ratio of our proposed scheme, communication-aware, and random task offloading schemes equals 4.7, 3.6, 3.9, respectively.

Fig. 5.8(d) examines the mean task PSR ($\hat{\mu}_s$) performance of different HART task execution schemes. Importantly, note that a small mean PSR value indicates the suitability of a HART task execution scheme. The figure shows that the mean PSR value increases for an increasing task number (k) in all considered schemes. We observe that our proposed task offloading scheme achieves a lower PSR value ($\hat{\mu}_{off}$) than its counterparts. This is because both minimum task workload processing time and task service time result in a smaller mean PSR value. Next, Fig. 5.8(e) depicts the overall task completion time ($\overline{t}_{tct,n}^k$) variation in our proposed community- and latency-aware task offloading scheme under varying FiWi traffic loads (u_d) and ONU-MPP numbers (n). We observe that small values of n and u_d translate into a shorter task completion delay in our proposed community- and latency-aware task offloading. Finally, Fig. 5.8(f) highlights the overall task completion time performance of our proposed task offloading and onloading schemes versus polling cycle time (t_c). The figure shows that the overall task completion time delay is higher in all compared schemes for large t_c. Importantly, we observe that both remote ($\overline{t}_{tct,r}^k$) and nearby ($\overline{t}_{tct,n}^k$) task completion times are minimal in our proposed community- and latency-aware task offloading scheme. For instance, for $t_c = 400$ ms and $k = 4$, the gain of the nearby and remote task completion time achieved in our proposed task offloading scheme is approximately 23.7% and 24.1% higher than in the task onloading scheme, respectively.

5.7 Conclusions

We proposed a community- and latency-aware multi-task scheduling scheme for collaborative HART task execution across FiWi enhanced Tactile Internet infrastructures. To accomplish multiple HART task execution in a resource- and time-efficient manner, our proposed task scheduling scheme selects both optimal multi-task scheduling order and suitable task processing nodes for different HART tasks. To reap the benefits from task prefetching to execute multiple HART tasks, we presented a novel prefetching-aware bandwidth allocation scheme that copes with both conventional broadband and task offloading data traffic at the same time. We developed a comprehensive analytical model to investigate the performance of our proposed community- and latency-aware task offloading scheme in terms of mean task service time, delay and power saving ratio, task prefetching time efficiency, task service time gain to overhead ratio, among others. Our presented results provide insights into selection of suitable

robot/agent resources for our proposed community and latency-aware task offloading scheme by taking different arrived HART task number, task requirements, and host and community robot/agent resource availabilities, into account. Our obtained results show that for a typical system of 32 ONU-MPPs and a polling cycle time of 100 ms, our proposed task offloading scheme achieves up to 31.3% and 32.7% task completion time gain over the task onloading scheme for nearby and remote task execution, respectively. The results demonstrate that for a typical task offload input data size of 500 MB, our proposed task offloading scheme with task prefetching capability offers a 11% higher task service time gain to overhead ratio than a conventional fetching based scheme. Furthermore, our findings suggest that for failure avoidance, our proposed fault tolerance mechanism is more effective in the considered task offloading scheme than alternative failure recovery mechanisms. Thus, our proposed community- and latency-aware task offloading scheme leveraging on both fault tolerance and task prefetching capability is a promising solution for low-latency HART collaboration in the emerging Tactile Internet.

Chapter 6

User Preference Aware Task Coordination and Proactive Bandwidth Allocation in a FiWi Network Infrastructures

6.1 Preamble

This chapter contains material extracted from the following paper:
[J6] M. Chowdhury and M. Maier, "User Preference Aware Task Coordination and Proactive Bandwidth Allocation in a FiWi Based Human-Agent-Robot Teamwork Ecosystem," *IEEE Transactions on Network and Service Management,* Oct. 2018 (in revision)[115].

6.2 Introduction

With the rise of smart mobile devices, demands for mobile applications in our everyday life have experienced significant growth during the last decade. Due to their resource limitations, however, many mobile devices may not be able to provide high quality-of-experience (QoE) to human users for computation-intensive task execution. To alleviate the burden of resource-constrained mobile devices, the concept of mobile-edge computing (MEC) has recently emerged, which allows mobile devices to offload computation tasks to nearby edge cloud servers for processing. MEC offers several cloud services, e.g., caching, computation processing, to mobile devices via decentralized cloudlets at the edge of the network, e.g., base station [22]. Decentralized cloudlets generally experience a lower task offloading communication latency than traditional remote clouds. Hence, task offloading to edge cloudlets is beneficial for handling many interactive cyber-physical system (CPS) applications that harness human-machine interaction, including virtual and augmented reality.

The importance of decentralized cloudlets is also witnessed in emerging *Tactile Internet* applications, where ultra-low latency communication services allow humans to remotely steer/control virtual/physical objects (e.g., robots) in real time in order to perform non-local tasks [5]. Note that the ultimate long-term goal of the Tactile Internet is to enable new goods and services that require human expertise in the coordination of human-robot symbiosis for the sake of complementing humans rather than substituting for them. The resultant collaborative human-machine activities are the object of *human-agent-robot teamwork (HART)* research, where the efficient allocation of task requests of humans to suitable machines (e.g., robots) and agents (e.g., cloudlets) is essential [55]. In a HART ecosystem, a task can be either (*i*) a physical task (e.g., lifting an object), (*ii*) a digital task (e.g., object detection), or (*iii*) a hybrid task that includes both physical and digital subtasks (e.g., sensing object at a given task location and detecting the sensed object). Note that unlike performing a physical task, the execution of a digital task does not necessarily require the presence of a robot/agent at the given task location. Further, note that a digital task may comprise either caching (e.g., audio/video/data content download), computation (e.g., object detection), or both (e.g., object detection from a captured image and caching content of the detected object).

While collaborative HART holds great promise for mobile users (MUs) requesting task execution, an unsuitable task assignment to robots/agents may lead to a higher task execution delay and inefficient resource utilization. To avoid these shortcomings, design of a suitable robot selection strategy for the allocation of both local and non-local tasks, taking a variety of different task characteristics (e.g., execution deadline) and robot properties (e.g., computation speed) into account is mandatory. Nevertheless, limited resources of robots may become a crucial bottleneck for the proper execution of different computation-intensive HART tasks. To render the execution of HART tasks more efficient, a collaborative computing strategy, where both robots and cloud agents jointly process a given MU's HART task may be suitable. Note that only robots residing near the given task location are eligible for executing physical tasks, whereas digital tasks may be offloaded to any suitable agent, either nearby or remote, for processing. Hence, to the best of authors knowledge, performance evaluation of multiple HART tasks by taking into account both dedicated and non-dedicated robot/cloud resources and MUs different preferences (delay vs. monetary cost saving preferences) was missing in the existing literature.

• **Delay Preference:** When requesting the execution of a delay-sensitive HART task (e.g., face recognition for real-time food/pizza delivery), a given MU may prefer a lower task execution delay over the incurred monetary cost. Hence, to meet the low-delay requirement, the utilization of both dedicated and non-dedicated robot/cloud resources as well as the preemptive bandwidth assignment for a given delay tolerance may become mandatory. Note that the

difference between dedicated and non-dedicated robot/cloud resources is that a robot/cloud server may be dedicated to a single task, i.e., only a single user can use the robot/cloud server or shared (i.e., non-dedicated) and thus used by multiple users. However, MUs have to pay an additional monetary cost for utilizing both dedicated and non-dedicated robots/clouds and preempting bandwidth resources.

• **Monetary Cost Saving Preference:** Beside delay-sensitive tasks, HART tasks (e.g., participatory sensing, image recognition) may not require time-critical execution. For such delay-tolerant tasks, monetary cost saving may be the primary concern for MUs. Thus, MUs may utilize only dedicated robots/clouds and non-preemptive bandwidth resources for the execution of their delay-tolerant tasks.

At present, no existing study deals with the problem of delay-sensitive and delay-tolerant caching and computing HART task execution considering preemptive/non-preemptive bandwidth allocation. To avoid additional delay and monetary costs while mitigating MUs' different task requests, in this work we propose a user preference aware task coordination strategy. Our proposed strategy not only enables the proper selection of dedicated/non-dedicated robots/agents for different HART tasks but also allows for efficient task offloading by exploiting a given MU's cost saving preferences. Specifically, we develop an analytical framework to examine the performance trade-off between delay cost saving (DCS) and monetary cost saving (MCS) schemes for the execution of different HART tasks, taking dedicated/non-dedicated cloud agents with/without caching capabilities into account and comparing the following three different DCS and MCS multi-task offloading schemes: (*i*) maximum throughput and minimum delay (MTMD), (*ii*) maximum throughput (MT), and (*iii*) minimum delay (MD).

The remainder of the chapter is structured as follows. In Section 6.3, we review prior art and outline open research challenges. In Section 6.4, we describe our proposed preference aware HART task coordination policy along with the considered communications network infrastructure. Section 6.5 presents the analytical model to evaluate the performance of our proposed scheme. Section 6.6 presents our obtained results and findings. Finally, Section 6.7 concludes the chapter.

6.3 Prior Art and Open Challenges

A significant body of research studies exist in the literature that focus on the problem of whether or not to utilize infrastructure-based cloud [29], cloudlet [28], [107] or infrastructure-less mobile ad-hoc cloud [116] resources for the execution of an MU's computation task. Some computation offloading studies investigate the problem of whether to offload the full or only a part of the MU's task to a suitable cloud agent for processing [30], [102]. The majority of these studies aim at selecting a suitable cloud server for task offloading with the objective of

minimizing either the task response time [28], [30] or the mobile device's energy consumption [74], [17]. For instance, in [112] and [113], a suitable cloud server is selected for computation task execution to minimize the task workload processing delay and offloading communication delay, respectively.

Investigations of selecting a suitable task offloading node while achieving both lower task workload processing and communication delays for executing different caching and computing HART tasks are not available. Furthermore, most existing studies focused on evaluating the task offloading performance by assuming available bandwidth resources for task offloading (upload and download) activities. Note, however, that immediate task offloading to cloud servers may not always be possible due to the lack and uncertainty of available bandwidth resources and network connectivity. Thus, for designing a proper task offloading node selection scheme, the waiting delay for an upcoming transmission opportunity, task offloading communication delay, workload processing delay as well as involved monetary cost need to be taken into account as well. Toward this end, beside proper cloud agent selection, previously proposed task offloading schemes [31], [32], [33] aimed at resolving the problem of selecting the suitable wireless interface (4G LTE-Advanced or WiFi) for either task data uploading or downloading, thought not both at the same time. Further, the authors in [117] proposed an energy-efficient delayed network selection scheme that optimizes the trade-off between energy consumption and transmission delay during the task data uploading process. In [118], an online task offloading policy was proposed that maximizes the amount of data offloaded through the WiFi network interface. The work in [119] aimed at achieving maximum throughput for caching content download through the suitable wireless network interface. Investigations of achieving both maximum throughput and minimum task execution delay for multiple HART tasks over integrated fiber-wireless (FiWi) network infrastructures, which consist of a fiber backhaul and wireless front-end, is an open research challenge. Moreover, the question of how to design an adaptive bandwidth allocation policy for the execution of an MU's delay-sensitive and delay-tolerant HART tasks without affecting their conventional broadband access services remains another open research challenge.

6.4 FiWi Enhanced Tactile Internet Infrastructure for Preference Aware HART Task Coordination

6.4.1 Network Architecture

Fig. 6.1 depicts our considered FiWi enhanced Tactile Internet infrastructure for HART task coordination leveraging both dedicated and non-dedicated cloud agent/robot resources. The optical fiber backhaul consists of time and wavelength division multiplexing (T/WDM) IEEE

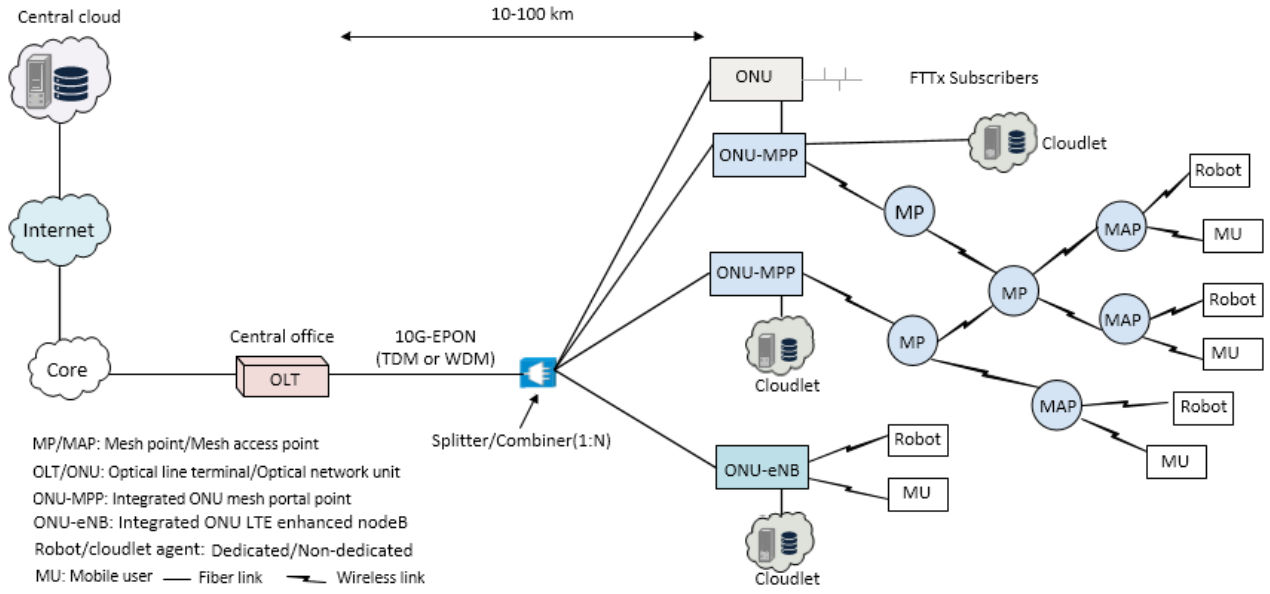


Figure 6.1: FiWi enhanced Tactile Internet infrastructure for preference aware HART task coordination.

802.3av 10 Gb/s Ethernet Passive Optical Network (10G-EPON) with a fiber backhaul range of 10-100 km between the central optical line terminal (OLT) and remote optical network units (ONUs). The central OLT connects to three different subsets of ONUs through a 1:N optical splitter/combiner. The first subset of ONUs serves fixed wired FTTx subscribers, e.g., fiber-to-the-home (FTTH). To provide MUs with WLAN and cellular services, the second and third subsets of ONUs are attached to an IEEE 802.11s mesh portal point (ONU-MPP) and a cellular base station (ONU-eNB), respectively. At the wireless front-end, the ONU-MPP connects with the wireless mesh network (WMN) through intermediate mesh points (MPs) and mesh access points (MAPs). MPs serve as relay nodes, which forward packets between MPPs and MAPs. MAPs provide wireless access services to associated MUs as well as WiFi enabled robots. To allow inter-ONU communication with broadband and cloud offloading services, so-called interconnected fiber (IF) links may be used between pairs of neighboring ONUs. For remote cloud services, central cloud servers are attached to the OLT via dedicated fiber links. Further, to provide cloud computing and caching services to MUs/robots at the network edge, multiple dedicated/non-dedicated cloudlets are connected to the ONU-MPPs/ONU-eNBs via dedicated fiber links [52].

6.4.2 Preference Aware HART Task Coordination

In this section, we present our proposed user preference aware HART task coordination scheme, whose pseudo-code is given in Algorithm 8. In our proposed task coordination scheme, initially

Algorithm 8 Preference-Aware HART Task Coordination

Notation: A set of full task request (n_t), physical ($n_s \in n_t$)/digital sub-task ($n_c \in n_t$) number, total dedicated robot (β), non-dedicated robot (Θ), dedicated local cloudlet agent ($\bar{\alpha}$), non-dedicated local and non-local cloudlet agent (σ), dedicated remote cloud (τ) agent, preference policy P=delay cost saving (DCS)/monetary cost saving (MCS)

1: **for** each arrived full task request $\forall \delta_i \in n_t$ **do**
2: collect each dedicated and non-dedicated robots ($r \in f = \beta + \Theta$) and agents ($\theta \in \hat{v} = \bar{\alpha} + \sigma + \tau$) busy time, distance, and CPU cycles information
3: check user preference (P) for each task request (δ_i)
4: **if** user preference $P == MCS$ **then**
5: **for** each physical sub-task $s_i \in n_s$ **do**
6: compute physical sub-task (s_i) processing delay ($t^r_{\gamma,s}$) for each dedicated robot ($r \in \beta$)
7: assign s_i to dedicated robot ($r \in \beta$) with lower $t^r_{\gamma,s}$
8: **end for**
9: **for** each digital sub-task $c_i \in n_c$ **do**
10: compute digital sub-task (c_i) processing delay ($t^{\theta}_{\gamma,c}$) for each dedicated agent ($\theta \in z = \tau + \bar{\alpha}$), assign c_i to dedicated agent with lower $t^{\theta}_{\gamma,c}$
11: the agent completes the computation sub-part of digital sub-task and checks caching content availability of digital sub-task
12: **if** caching content is available **then**
13: transfer the caching content of c_i to MU
14: **else**
15: collect caching content from other agent with min distance and transfer to MU
16: **end if**
17: assign delay-tolerant task offload time slot for c_i based on proactive bandwidth assignment policy
18: **end for**
19: **end if**
20: **if** user preference $P == DCS$ **then**
21: **for** each physical sub-task $s_i \in n_s$ **do**
22: compute dedicated and non-dedicated robots ($r \in f$) physical task processing delay ($t^r_{\Omega,s}$), assign s_i to the robot ($r \in f$) with lower $t^r_{\Omega,s}$ value
23: **end for**
24: **for** each digital sub-task $c_i \in n_c$ **do**
25: compute dedicated and non-dedicated agents ($\theta \in \hat{v}$) digital sub-task processing delay ($t^{\theta}_{\Omega,c}$), assign c_i to the agent ($\theta \in \hat{v}$) with lower $t^{\theta}_{\Omega,c}$ value
26: the agent completes the computation sub-part of digital sub-task and checks caching content availability of digital sub-task
27: **if** required caching content is available **then**
28: transfers the caching content of c_i to MU
29: **else**
30: collect caching content from another agent with min distance and transfer to MU
31: **end if**
32: assign delay-sensitive task offload time slot for c_i based on proactive bandwidth assignment policy
33: **end for**
34: **end if**
35: **end for**

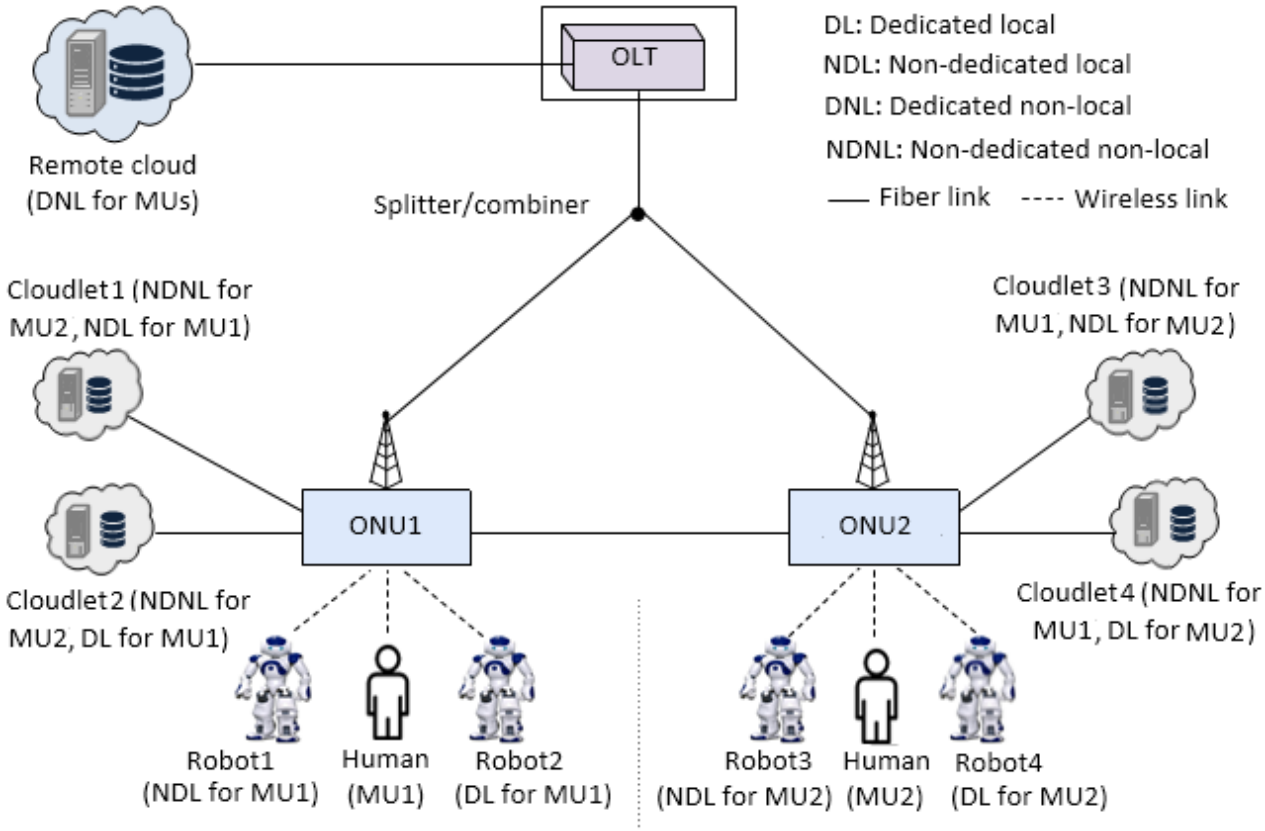


Figure 6.2: Local and non-local dedicated/non-dedicated robots and agents.

an MU sends a full HART task execution request (physical and digital sub-tasks) to the task coordinator co-located at each ONU. Next, the task coordinator selects a suitable robot and cloud agent for the execution of each physical and digital sub-task, respectively. Note that to satisfy given user preferences our proposed HART task coordination applies the following two policies: (i) minimizing task execution delay and (ii) maximizing monetary cost saving for different HART tasks, as explained in greater detail in the following.

A. Delay Cost Saving Policy (DCS): In this policy, after receiving a given MU's HART task request message, the task coordinator sends the task request message to all dedicated and non-dedicated actors (i.e., local and non-local robots/agents). Then, all actors send their task response message to the task coordinator, which comprises information abouth their busy time, location, and task processing speed (CPU cycles). After collecting all actors' response message, the task coordinator calculates their predicted task processing delay. Subsequently, the task coordinator selects a suitable robot offering minimum processing time $t^{r}_{\Omega,s}$ for the physical sub-task and a suitable agent offering minimum processing time $t^{\theta}_{\Omega,c}$ for executing the digital sub-task, respectively. Note that unlike non-local actors, local actors are located within the ONU's coverage area, where the physical sub-task needs to be performed, as illustrated in Fig. 6.2. For

the selection of suitable actors, our proposed task coordination scheme considers the following cases. If local dedicated actors are available, the task coordinator selects suitable actors only from the set of local dedicated actors. Otherwise, the task coordinator selects suitable actors from both dedicated and non-dedicated (local and non-local) actors. The selected robot executes the physical sub-task, generates the physical sub-task output (digital sub-task input), and offloads the digital sub-task input to the selected agent for further processing. The selected agent executes the computation sub-part (e.g., face detection) of the digital sub-task and checks whether caching the content (e.g., information about detected face) of the computation sub-part result is possible or not. If caching is possible, the selected agent transfers the cached content back to the MU during the task result download sub-slot. Otherwise, the initially selected agent first fetches the cached content from another agent (cloudlet or remote cloud) and then transfers it back to the corresponding MU.

Due to the use of different cloud agents for executing digital sub-tasks, our proposed DCS policy can be divided into two categories: (*i*) DCS with local/non-local cloudlet caching and (*ii*) DCS with remote cloud caching. In addition, given different priorities for executing multiple HART tasks, we consider the following three variant schemes of our DCS policy: (*i*) maximum throughput and minimum task execution delay (MTMD) based scheme, (*ii*) maximum throughput (MT) based scheme, and (*iii*) minimum task workload processing delay (MD) based scheme. Note that in the case of a multiple-task request arrival, maximum throughput can be ensured by using the network interface offering the higher data rate for task request transmission and task up/downloading, whereas minimum task workload processing delay can be achieved by selecting the most powerful robot/cloud agent for each task.

B. Monetary Cost Saving Policy (MCS): In this scheme, after receiving a given MU's task execution request, the task coordinator sends the task request message only to dedicated actors. The dedicated actors in turn send their task response message to the task coordinator, including information about their busy time, location, and task processing speed (CPU cycles). Next, the task coordinator computes the required physical ($t^r_{\gamma,s}$) and digital sub-task ($t^\theta_{\gamma,c}$) processing time for each dedicated robot and agent, respectively. Subsequently, the task coordinator selects a suitable dedicated robot that provides a lower $t^r_{\gamma,s}$ for the physical sub-task and a suitable agent that provides a lower $t^\theta_{\gamma,c}$ for executing the digital sub-task, respectively. The selected robot then executes the physical sub-task and offloads the digital sub-task to the selected cloud agent for further processing. Next, the selected agent processes the computation sub-part of the digital sub-task and checks whether caching the content of the computation sub-part result can be done or not. If caching the content is possible, the initially selected agent transfers the cached content to the MU during the task result download sub-slot. If the caching content isn't possible, the initially selected agent collects the cached content

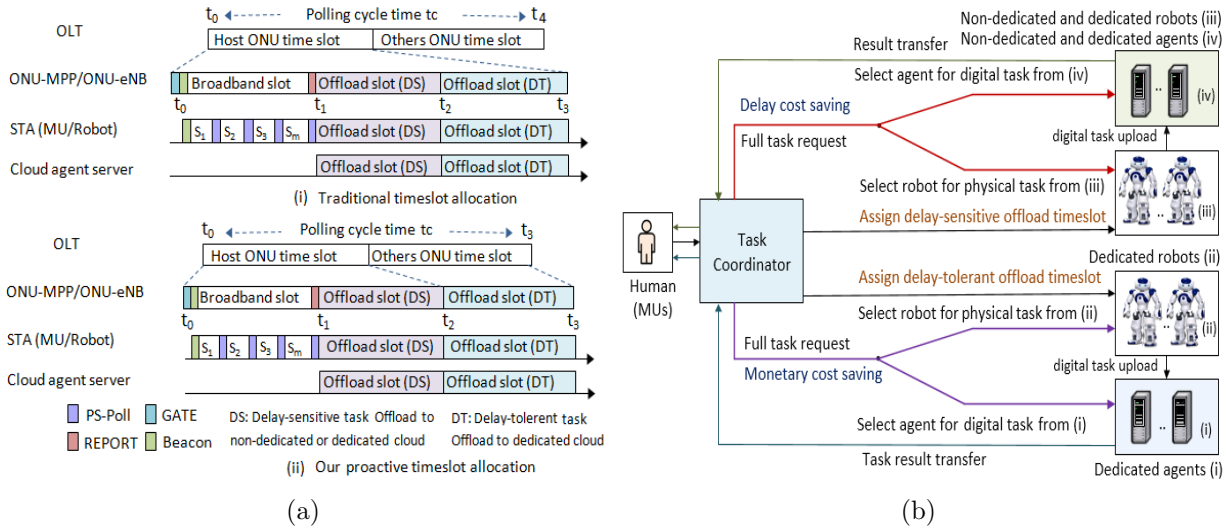


Figure 6.3: (a) Proactive bandwidth allocation scheme and (b) task coordination scheme.

from another dedicated agent and transfers it to the intended MU. Note that depending on the selection of different dedicated agents, there exist two variations of our MCS policy: (*i*) MCS with cloudlet caching and (*ii*) MCS with remote cloud caching. Hence, given different priorities for task execution there exist the following three schemes of our proposed MCS policy: (*i*) maximum throughput and minimum delay (MTMD), (*ii*) maximum throughput (MT), and (*iii*) minimum task workload processing delay (MD) based schemes.

6.4.3 Proactive Bandwidth Allocation Scheme

Fig. 6.3(a) depicts our two-layer TDMA based proactive bandwidth allocation scheme in greater details for execution of different HART tasks. It differs from the traditional TDMA based model in several ways. First, in our proposed scheme, we divide task offloading users into two groups (see Fig. 6.3(b)), namely, delay-sensitive users (applying the DCS policy) and delay-tolerant users (applying the MCS policy). Second, in our proposed scheme, DCS policy users offload their delay-sensitive digital sub-tasks to suitable dedicated or non-dedicated agents during the associated ONU's offload time-slot. Conversely, MCS policy users offload their delay-tolerant digital sub-tasks to suitable dedicated agents only during another ONU's time-slot by using the dedicated point-to-point IF fiber connections. Thus, by performing delay-tolerant task offloading during another ONU's time-slot, our proposed scheme is able to save both bandwidth and monetary cost for MCS policy users.

In our proposed two-layer TDMA scheme, the first TDMA layer is used for the optical fiber backhaul, whereby the OLT allocates upstream (US) timeslots to all ONUs via IEEE 802.3ah multipoint control protocol (MPCP) messages (`REPORT` and `GATE`). The second TDMA layer

is used to operate the wireless part, whereby ONUs assign both broadband and task offload sub-slots to their associated MUs/robots via IEEE 802.11 messages (`Beacon` and `PS-Poll`). Note that each ONU sends a `REPORT` message to the OLT in order to notify the OLT about its bandwidth requirement in the next polling cycle, whereas a `GATE` message is sent downstream by the OLT to inform all ONUs about their assigned time-slot. After receiving the `GATE` message from the OLT, each ONU extracts its broadband and task offload time-slot schedule. Subsequently, each ONU assigns a broadband and task offload sub-slot to its associated MUs/robots based on their instantaneous traffic demand via a `PS-Poll` message. Next, the ONU broadcasts a `Beacon` message to its associated MUs/robots to inform them about their broadband and task offload sub-slot schedule.

6.5 Performance Analysis

In this section, we develop an analytical model to evaluate the performance of our proposed user preference aware task coordination scheme in terms of various key performance metrics.

6.5.1 Delay Analysis

First, we analyze the aggregate task execution delay $t^{\delta}_{\kappa,\mu}$ for both DCS and MCS policies, which comprises the following three delay components: (i) task request message dissemination delay t^{δ}_{rq}, (ii) actor selection delay $t^{\delta}_{\kappa,ser}$, and (iii) full HART (physical and digital) task processing delay $t^{\hat{\delta}}_{\kappa,\mu}$.

(i) Task request dissemination delay: The task request message dissemination delay t^{δ}_{rq} denotes the time interval between task request message generation by the MU and task request message reception by the task scheduler at the host ONU. Thus, t^{δ}_{rq} includes two delay components, namely, the waiting delay for transmission opportunity (t^{b}_{w}) and communication delay (t^{x}_{tra}) for transferring the task request from the MU to the task scheduler. The calculation of t^{b}_{w} and t^{x}_{tra} will be described in greater detail in Section 6.5.3.

(ii) Actor selection delay: After receiving the task request message from the MU, the task scheduler at the host ONU selects suitable actors (robot and agent) for each full task execution. The process of selecting suitable actors involves the exchange of task request ($\hat{t}_{rq}$), actors' response (t_{ar}), and task assignment confirmation (t_{ac}) messages between the task scheduler and selected actors. If the total number of active users $\hat{k}$ (for DCS policy $\hat{k}$=f+$\hat{v}$ and for MCS policy $\hat{k}$=z+β), latency (task processing time) comparison time of two actors t_{cp}, and total number of selected actors y are known, the actor selection delay in DCS ($t^{\delta}_{\Omega,ser}$)

and MCS policy ($t^{\delta}_{\gamma,ser}$) are given by

$$t^{\delta}_{\kappa,ser} = \begin{cases} (f+\hat{v}) \cdot (\hat{t}_{rq} + t_{ar} + t_{cp}) + y \cdot t_{ac}, & \text{if } \kappa = \Omega \\ (\beta+z) \cdot (\hat{t}_{rq} + t_{ar} + t_{cp}) + y \cdot t_{ac}, & \text{if } \kappa = \gamma. \end{cases} \tag{6.1}$$

(*iii*) **Task processing delay:** Next, we calculate the full task processing delay ($t^{\widehat{\delta}}_{\kappa,\mu}$) that comprises both the robot's physical sub-task ($t^{r}_{\kappa,s}$) and the agent's digital sub-task processing delay ($t^{\theta}_{\kappa,c}$), whereby $t^{\widehat{\delta}}_{\kappa,\mu}$ is given by $t^{\widehat{\delta}}_{\kappa,\mu}$=$t^{r}_{\kappa,s}$+$t^{\theta}_{\kappa,c}$=$t^{b}_{r}$+$t^{\Psi}_{r}$+$t^{\phi}_{r}$+$t^{\theta}_{\kappa,c}$. The robot's physical sub-task processing time is equal to $t^{r}_{\kappa,s}$=t^{b}_{r}+t^{Ψ}_{r}+t^{ϕ}_{r}=t^{b}_{r}+$\frac{d_{\Psi}}{q_r}$+$\frac{w_{\phi}}{\nu_r}$, where t^{b}_{r}, t^{Ψ}_{r}, t^{ϕ}_{r}, d_{Ψ}, w_{ϕ}, q_r, ν_r denote the robot's busy time, task location traverse time (t^{Ψ}_{r}=$\frac{d_{\Psi}}{q_r}$), physical sub-task workload processing time (t^{ϕ}_{r}=$\frac{w_{\phi}}{\nu_r}$), Euclidean distance between task and robot locations, physical sub-task workload, and the robot's moving and computation processing speed, respectively. For the DCS policy, a suitable robot is selected by checking all dedicated and non-dedicated robots' physical sub-task processing times as follows: $t^{r}_{\Omega,s}$=min$\{t^{1}_{\Omega,s}, ..t^{f}_{\Omega,s}\}$, $r = 1, 2, \ldots, f$. Conversely, for the robot selection using the MCS policy, only dedicated robots are examined: $t^{r}_{\gamma,s}$=min$\{t^{1}_{\gamma,s}, ..t^{\beta}_{\gamma,s}\}$, $r = 1, 2, \ldots, \beta$.

For the digital sub-task execution using the DCS and MCS policies, the robot that executes the physical sub-task initially uploads the digital sub-task input to the cloud agent. Next, the selected cloud agent processes the computation and caching sub-part of the digital sub-task. Thus, by taking the task offloading communication, computation processing, and caching delays into account, the digital sub-task processing delay is given by $t^{\theta}_{\kappa,c}$=t^{b}_{θ}+t^{ψ}_{θ}=t^{b}_{θ}+$t^{w}_{\kappa,\theta}$+t^{u}_{θ}+t^{π}_{θ}+t^{ϕ}_{θ}+t^{d}_{θ}, where t^{b}_{θ}, $t^{w}_{\kappa,\theta}$, t^{u}_{θ}, t^{π}_{θ}, t^{ϕ}_{θ}, and t^{d}_{θ} represent the agent's busy time, transmission opportunity delay, digital sub-task uploading, digital sub-task computation processing (t^{π}_{θ}=$\frac{w_{\pi}}{\nu_{\theta}}$), cache lookup, and task result download delays, respectively. In the case of our DCS policy, a suitable cloud agent θ is selected by comparing all dedicated and non-dedicated agents' digital sub-task processing time as follows: $t^{\theta}_{\Omega,c}$=min$\{t^{1}_{\Omega,c}, ..t^{\hat{v}}_{\Omega,c}\}$, where $\theta = 1, 2, \ldots, \hat{v}$. Whereas for our MCS policy, a suitable agent is selected only from the dedicated agents: $t^{\theta}_{\gamma,c}$=min$\{t^{1}_{\gamma,c}, ..t^{z}_{\gamma,c}\}$, where $\theta = 1, 2, \ldots, z$. Finally, by taking the task request transmission, actor selection as well as physical and digital sub-task processing delays into account, the full task execution delay in the case of our DCS ($t^{\delta}_{\Omega,\mu}$) and MCS policies ($t^{\delta}_{\gamma,\mu}$) is obtained as follows:

$$t^{\delta}_{\kappa,\mu} = \begin{cases} t^{\delta}_{rq} + t^{\delta}_{\Omega,ser} + t^{r}_{\Omega,s} + t^{\theta}_{\Omega,c}, & \text{if } \kappa = \Omega \\ t^{\delta}_{rq} + t^{\delta}_{\gamma,ser} + t^{r}_{\gamma,s} + t^{\theta}_{\gamma,c}, & \text{if } \kappa = \gamma. \end{cases} \tag{6.2}$$

6.5.2 Caching Content Access Latency

In this section, we analyze the caching content access latency by using the so-called average memory access time (AMAT) formula, in which the cache hit $\hat{h}$ and miss ratio $\hat{m}$ play a

significant role. To see this, note that if the caching content is available at the initially selected agent ($\hat{m}$=0), the caching content access latency is equal to the agent's cache look-up delay t^l_θ, i.e., the time required to match a request to the related response. Otherwise, if the caching content is unavailable ($\hat{m}$=1) at the initially selected agent (θ), an additional time is required to fetch the caching content from another agent (θ^*). Thus, considering both cases, the total caching content access latency is equal to t^ϕ_θ=t^l_θ+$\hat{m}$·($t^w_{\theta\to\theta^*}$+$t^{\hat{d}}_{\theta\to\theta^*}$+$t^l_{\theta^*}$), where $t^w_{\theta\to\theta^*}$ and $t^l_\theta/t^l_{\theta^*}$ denote the access delay (maximum t_c) and cache lookup delay at the initially/newly selected agent, respectively. For instance, for a face recognition cache, we have t^l_θ=$t^l_{\theta^*}$=$f(s_v)$, where $f(s_v)$ is a monotonically increasing function of the cache size s_v [120]. Further, $t^{\hat{d}}_{\theta\to\theta^*}$ denotes the communication delay for fetching the caching content from another agent, which is given by

$$t^{\hat{d}}_{\theta\to\theta^*} = \begin{cases} \frac{g_w+s_v}{\Phi_{ct}} + \frac{g_w+s_v}{\Phi_{ct}} + t_p & \text{if } \theta^* = l_{ct}, \\ h^c_n \cdot \frac{g_w+s_v}{\phi_{fl}} + \frac{2g_w}{\Phi_{ct}} + \frac{2s_v}{\Phi_{ct}} + t_{\hat{p}} & \text{if } \theta^* = n_{ct}, \\ h^c_r \cdot \frac{g_w+s_v}{\Psi_{fl}} + \frac{g_w+s_v}{\Phi_{ct}} + \frac{g_w+s_v}{\Phi_{cl}} + t^*_p & \text{if } \theta^* = n_{cl}, \end{cases} \tag{6.3}$$

where g_w, s_v, h^c_n/h^c_r, $t_p/t_{\hat{p}}/t_{p^*}$, and $\Phi_{ct}/\phi_{fl}/\Psi_{fl}/\Phi_{cl}$ represent the computation sub-part output data size, caching content data size, hop distance between host ONU and nearby ONU (non-local cloudlet location)/host ONU and the OLT (remote cloud location), propagation delay for local cloudlet l_{ct}/non-local cloudlet n_{ct}/remote cloud n_{cl} caching, transmission capacity of the link between the ONU and cloudlet, inter ONU, the ONU and OLT, OLT and remote cloud, respectively.

To compute the cache miss ratio, we assume that agents' caching content files (e.g., audio, text, video of the recognized face) available for download are stored in the form of a library $\vartheta = 1, 2, \ldots, V$. More specifically, each file has an average size of s_v bits and different popularity. The probability of cached content $v = 1, 2 \ldots, V$ being requested for download follows a Zipf distribution [121], given by $P_\vartheta(v)$=$\frac{\hat{\sigma}}{v^\epsilon}$, where $\hat{\sigma}$=$\big(\sum_{i=1}^{V} \frac{1}{i^\epsilon}\big)^{-1}$ and ϵ describes the steepness of the distribution $P_\vartheta(v)$. Provided that n_f=$\frac{c_\theta}{s_v}$ is the total number of caching content files at the agent, n_a is the agent's number, and c_θ is the agent's caching content capacity, the cache hit ratio equals $\hat{h} = 1 - \hat{m} = 1 - \frac{\sum_{v=n_a \cdot n_f}^{V} \frac{1}{v^\epsilon}}{\sum_{v=1}^{V} \frac{1}{v^\epsilon}}$.

6.5.3 Mean Task Offload Delay

A task offload packet may suffer from offloading delay during its transmission to an agent, as shown in Figs. 6.4 and 6.5 for the DCS and MCS policies, respectively. The mean task offload delay generally comprises three delay components. The first delay component (t_{u1}) is the time interval between the arrival of a task offload packet (A) at an MU and the bandwidth reservation request (`PS-Poll`) transmission (R). If the task request packet is generated by the MU after the current cycle `PS-Poll` message, the MU waits for the maximum cycle period

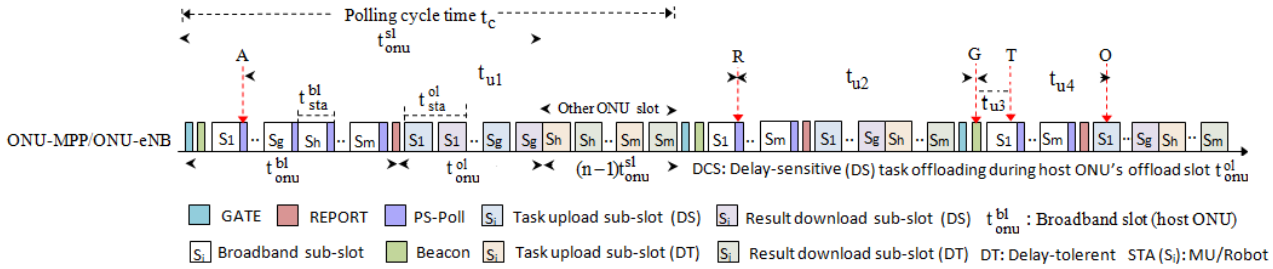


Figure 6.4: Mean offload packet delay components in DCS policy.

(t_{u1}=t_c) to transmit a bandwidth reservation (`PS-Poll`) message. On average, t_{u1} is equal to $\frac{t_c}{2}$. The second delay component refers to the time interval between the bandwidth reservation (R) and grant (G) messages. In the following, let STA be either an MU or a robot. For STA_1, t_{u2} is then equal to $(t_c - t^{bl}_{sta})$. For the m-th STA, t_{u2} equals $(t_c - m \cdot t^{bl}_{sta})$. Thus, on average we have $t_{u2} = \frac{mt_c}{m} - \frac{(1+2+...+m-1+m)t^{bl}_{sta}}{m} = t_c - \frac{(m+1)t^{bl}_{sta}}{2}$. The third delay component differs in the DCS and MCS policies. In our DCS policy (see Fig. 6.4), the third delay component (t_{u4}) denotes the time interval between a STA's grant (G) message and delay-sensitive task offload sub-slot start time (O). For STA_1, t_{u4} is equal to $m \cdot t^{bl}_{sta} + t^{msg}_{pon}$. For STA_2, t_{u4} equals $m \cdot t^{bl}_{sta} + t^{msg}_{pon} + t^{ol}_{sta}$. If the total delay-sensitive task offload sub-slot (t^{ol}_{sta}) is h, we obtain on average $t_{u4} = m \cdot t^{bl}_{sta} + t^{msg}_{pon} + \frac{(0+1+2+...+h-1)\cdot t^{ol}_{sta}}{h} = m \cdot t^{bl}_{sta} + t^{msg}_{pon} + \frac{(h-1)t^{ol}_{sta}}{2}$. By contrast, in our MCS policy (see Fig. 6.5), the third delay component (t_{u5}) is equal to the time interval between the grant (G) message and delay-tolerant task offload sub-slot start time (O). For the first STA, we have $t_{u5} = t^{sl}_{onu} - t^{msg}_{pon} - t^{msg}_{wl}$. If the total delay-tolerant task offload sub-slot (t^{ol}_{sta}) is k and delay-sensitive offload sub-slot is h, we obtain on average t_{u5}=$t^{sl}_{onu} - t^{msg}_{pon} - t^{msg}_{wl} + \frac{(0+1+2+...+k-1)\cdot t^{ol}_{sta}}{k}$. By summing up all three delay components, the mean task offload delay in DCS ($t^{w}_{\Omega,\theta}$) and MCS ($t^{w}_{\gamma,\theta}$) policies is given by

$$t^{w}_{\kappa,\theta} = \begin{cases} t_{u1} + t_{u2} + t_{u4}, & \text{if } \kappa = \Omega \\ t_{u1} + t_{u2} + t_{u5}, & \text{if } \kappa = \gamma. \end{cases} \tag{6.4}$$

Next, let us calculate the STA's task request dissemination delay (t^{δ}_{rq}), which includes the bandwidth opportunity delay (t^{b}_{w}) and upstream (US) task request traverse time (t^{x}_{tra}). If the task request packet is generated by the STA after the current cycle broadband sub-slot, then t^{b}_{w}=t_{u1}+t_{u2}+t_{u3} includes three delay components, where t_{u1} is the time interval between the task request packet arrival at the STA (A) and bandwidth request transmission (R), t_{u2} is the time interval between the bandwidth request transfer (R) and grant (G) message, and t_{u3} is the time interval between the grant (G) message and the STA's broadband (T) sub-slot start time. Note that for STA_1, t_{u3} is equal to zero. Whereas for STA_m, t_{u3} equals $(m-1)t^{bl}_{sta}$. On average, we have $t_{u3} = \frac{(0+1+2+...+m-1)t^{bl}_{sta}}{m} = \frac{(m-1)t^{bl}_{sta}}{2}$. By summing up all three

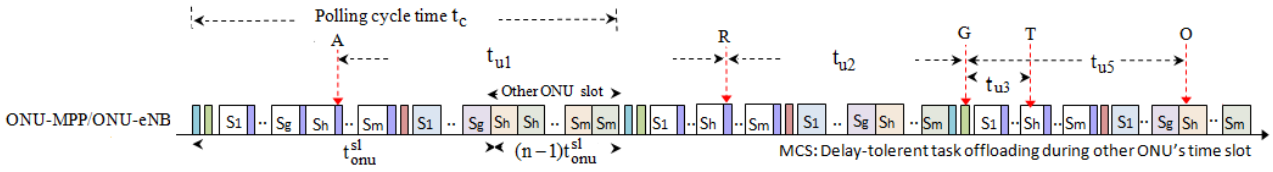


Figure 6.5: Mean offload packet delay components in MCS policy.

delay components, t^b_w is obtained as $t^b_w = t_{u1} + t_{u2} + t_{u3}$. Finally, we calculate t^x_{tra}, which denotes the required time to transfer the STA's task request message from the MU to the task scheduler at the host ONU. Hence, we have: $t^x_{tra}{=}t^{bl}_{sta}{=}t^u_{mu\to h_o}{+}t^p_{mu\to h_o}$, where $t^u_{mu\to h_o}$ and $t^p_{mu\to h_o}$ denote the US transmission and the total propagation delay that incurs during the task request transfer process, respectively.

6.5.4 Communication Delay

In this subsection, we compute task offloading communication delay $t^o_\theta = t^u_\theta + t^d_\theta$, which comprises both the digital sub-task upload delay (t^u_θ) from robot to cloud agent and the task result download delay (t^d_θ) from agent θ to MU. Hence, t^o_θ is obtained as follows:

$$t^o_\theta = \begin{cases} \frac{g_i+s_v}{\psi_w} + \frac{g_i+s_v}{\Phi_{ct}} + t_{p_1} & \text{if } \theta = l_{ct}, \\ \frac{g_i+s_v}{\psi_w} + h^c_n \cdot \frac{g_i+s_v}{\phi_{fl}} + \frac{g_i+s_v}{\Phi_{ct}} + t_{p_2} & \text{if } \theta = n_{ct}, \\ \frac{g_i+s_v}{\psi_w} + h^c_r \cdot \frac{g_i+s_v}{\Psi_{fl}} + \frac{g_i+s_v}{\Phi_{cl}} + t_{p_3} & \text{if } \theta = n_{cl}, \end{cases} \tag{6.5}$$

where g_i, s_v, h^c_n, h^c_r, $\psi_w/\Phi_{ct}/\phi_{fl}/\Psi_{fl}/\Phi_{cl}$, t_{p_1}, t_{p_2}, and t_{p_3} denote the digital sub-task input and output data size, hop distance between host ONU h_o (physical task location) and nearby ONU d_o (non-local cloudlet location), hop distance between host ONU and OLT (remote cloud location), transmission capacity of the link between STA and ONU, ONU and cloudlet, inter-ONU, ONU and OLT, OLT and remote cloud, total propagation delay for task offloading to local cloudlet l_{ct} ($t_{p_1}{=}2t^p_{r\to h_o}{+}2t^p_{h_o\to l_{ct}}$), non-local cloudlet n_{ct} ($t_{p_2}{=}2t^p_{r\to h_o}{+}2t^p_{h_o\to d_o}{+}2t^p_{d_o\to n_{ct}}$), and remote cloud n_{cl} ($t_{p_3}{=}2t^p_{r\to h_o}{+}2t^p_{h_o\to olt}{+}2t^p_{olt\to r_{cl}}$), respectively.

6.5.5 Monetary Cost

In this subsection, we analyze the expected monetary cost for both DCS and MCS policy based HART task execution. Note that the monetary cost calculation the execution of multiple full HART tasks consists of two parts: (i) monetary cost of using actors (cloud agents and robots) for physical and digital sub-task processing ($p^j_{\kappa,w}$) and (ii) monetary cost of using bandwidth resources for task request transmission and task offloading ($p^\epsilon_{\kappa,b}$). The monetary cost of the

physical and digital sub-task workload processing using DCS and MCS policies is given by

$$p^{j}_{\kappa,w} = \begin{cases} \big(\hat{n}_s \cdot \tilde{\omega}_1 \cdot \vec{p}_d \cdot t^{r}_{\Omega,s} + \bar{n}_s \cdot \hat{\omega}_2 \cdot \vec{p}_n \cdot t^{r}_{\Omega,s} + \\ \hat{n}_c \cdot \upsilon_1 \cdot \tilde{p}_d \cdot t^{\theta}_{\Omega,c} + \bar{n}_c \cdot \upsilon_2 \cdot \tilde{p}_n \cdot t^{\theta}_{\Omega,c}\big), & \text{if } \kappa = \Omega \\ \big(\hat{n}_s \cdot \alpha_1 \cdot \vec{p}_d \cdot t^{r}_{\gamma,s} + \hat{n}_c \cdot \upsilon_1 \cdot \tilde{p}_d \cdot t^{\theta}_{\gamma,c}\big), & \text{if } \kappa = \gamma, \end{cases} \tag{6.6}$$

where $\hat{n}_s/\bar{n}_s$ and $\hat{n}_c/\bar{n}_c$ denote the number of physical and digital sub-tasks that use dedicated/non-dedicated robots and agents ($\hat{n}_s$,$\bar{n}_s \in n_s$ and $\hat{n}_c$,$\bar{n}_c \in n_c$), respectively. Further, $\tilde{\omega}_1/\hat{\omega}_2$ and υ_1/υ_2 represent the monetary cost per unit time (second) for using dedicated/non-dedicated robots and agents, respectively, while $t^{r}_{\Omega,s}/t^{r}_{\gamma,s}$ and $t^{\theta}_{\Omega,c}/t^{\theta}_{\gamma,c}$ denote the selected robot's physical and agent's digital sub-task processing time in our DCS/MCS policy, respectively. Moreover, let $\vec{p}_d/\vec{p}_n$ and $\tilde{p}_d/\tilde{p}_n$ be the probability that a dedicated/non-dedicated robot and agent is selected, respectively. Note that the probability that a dedicated/non-dedicated actor (robot/agent) selected is equal to 1, if the corresponding robot's/agent's physical/digital task processing time is minimum among all actors.

Similarly, the total monetary cost of using bandwidth resources in our DCS and MCS policies is obtained as follows:

$$p^{\epsilon}_{\kappa,b} = \begin{cases} \big(\Delta_1 \cdot p_w \cdot t^{x}_{trs} \cdot \vec{n}_t + \Delta_2 \cdot p_c \cdot t^{x}_{trs} \cdot \tilde{n}_t \\ + \vec{n}_c \cdot \Lambda_1 \cdot p_w \cdot t^{o}_{\theta} + \tilde{n}_c \cdot p_c \cdot \Lambda_2 \cdot t^{o}_{\theta}\big), & \text{if } \kappa = \Omega \\ \big(\Delta_1 \cdot p_w \cdot t^{x}_{tra} \cdot \vec{n}_t + \Delta_2 \cdot p_c \cdot t^{x}_{tra} \cdot \tilde{n}_t \\ + \vec{n}_c \cdot p_w \cdot \vec{\lambda}_1 \cdot t^{o}_{\theta} + \tilde{n}_c \cdot p_c \cdot \tilde{\lambda}_2 \cdot t^{o}_{\theta}\big), & \text{if } \kappa = \gamma, \end{cases} \tag{6.7}$$

where $\vec{n}_t/\tilde{n}_t$ and $\vec{n}_c/\tilde{n}_c$ denote the number of tasks that use the WiFi/cellular network interface for task request transmission and task offloading, respectively. Furthermore, t^{x}_{tra} and t^{o}_{θ} represent the task request transmission and task offloading communication (t^{o}_{θ}=t^{u}_{θ}+t^{d}_{θ}) delay, respectively. Moreover, Δ_1 and Δ_2 represent the monetary charge per unit time for task request transmission using the WiFi and cellular interface, respectively. Let Λ_1/Λ_2 and $\vec{\lambda}_1/\tilde{\lambda}_2$ be the monetary cost per unit time for delay-sensitive and delay-tolerant task offloading using the WiFi/cellular interface, respectively. The probability that the WiFi and cellular network interface is used for task request transmission and task offloading is given by p_w and p_c, respectively. We note that an MU uses the WiFi network interface ($p_w = 1$) only if its utilization ($t^{w}_{\alpha} \geq \vec{t}^{\delta}_{\kappa,\mu} > \tilde{t}^{\delta}_{\kappa,\mu}$) provides the minimum task execution delay and the MU's associated MAP residence time is sufficient ($t^{w}_{\alpha} = \frac{1}{\lambda_w}$) to perform task execution. Otherwise, the MU uses the cellular network interface ($t^{\delta}_{\kappa,\mu} = \tilde{t}^{\delta}_{\kappa,\mu}$) provided it offers the minimum task execution delay ($t^{c}_{\alpha} \geq \tilde{t}^{\delta}_{\kappa,\mu} > \vec{t}^{\delta}_{\kappa,\mu}$). Note that $\vec{t}^{\delta}_{\kappa,\mu}$ and $\tilde{t}^{\delta}_{\kappa,\mu}$ denote the task execution delay, if the WiFi ($t^{\delta}_{\kappa,\mu} = \vec{t}^{\delta}_{\kappa,\mu}$) and cellular interface ($t^{\delta}_{\kappa,\mu} = \tilde{t}^{\delta}_{\kappa,\mu}$) is used as a wireless medium for task request transmission and offloading, respectively.

6.5.6 Time and Monetary Cost Saving Ratios

In the following, we calculate the time saving ratio (TSR) p^{Ω}_{sd} and monetary cost saving ratio (MSR) p^{γ}_{sm} for both DCS and MCS policy based HART task execution. We define p^{γ}_{sm} as the ratio of monetary cost gain using our MCS policy for task execution $(p^{j}_{\Omega,w} - p^{j}_{\gamma,w} + p^{\epsilon}_{\Omega,b} - p^{\epsilon}_{\gamma,b})$ and the total monetary cost for the case of DCS policy based execution $(p^{j}_{\Omega,w} + p^{\epsilon}_{\Omega,b})$. With n_t denoting the total task number, p^{γ}_{sm} is given by

$$p^{\gamma}_{sm} = \frac{\sum_{i=1}^{n_t} \left(p^{j}_{\Omega,w} - p^{j}_{\gamma,w} + p^{\epsilon}_{\Omega,b} - p^{\epsilon}_{\gamma,b}\right)}{\sum_{i=1}^{n_t} \left(p^{j}_{\Omega,w} + p^{\epsilon}_{\Omega,b}\right)} \times 100\%. \tag{6.8}$$

Hence, p^{Ω}_{sd} is defined as the ratio of time cost gain for DCS policy based task execution ($t^{\delta}_{\gamma,\mu}$-$t^{\delta}_{\Omega,\mu}$) and task execution time cost for our MCS policy ($t^{\delta}_{\gamma,\mu}$=t^{δ}_{rq}+$t^{\delta}_{\gamma,ser}$+$t^{r}_{\gamma,s}$+$t^{\theta}_{\gamma,c}$). Thus, p^{Ω}_{sd} is given by

$$t^{\Omega}_{sd} = \frac{\sum_{i=1}^{n_t} \left(t^{\delta}_{\gamma,\mu} - t^{\delta}_{\Omega,\mu}\right)}{\sum_{i=1}^{n_t} \left(t^{\delta}_{rq} + t^{\delta}_{\gamma,ser} + t^{r}_{\gamma,s} + t^{\theta}_{\gamma,c}\right)} \times 100\%. \tag{6.9}$$

6.5.7 Energy Cost

Next, we compute the energy consumption cost of both DCS ($e^{\tau}_{\Omega,\mu}$) and MCS ($e^{\tau}_{\gamma,\mu}$) policy based task execution given by

$$e^{\tau}_{\kappa,\mu} = \begin{cases} \sum_{i=1}^{n_t} \left(e^{\delta}_{rq} + e^{\delta}_{\Omega,ser} + e^{r}_{\Omega,s} + e^{\theta}_{\Omega,c}\right), & \text{if } \kappa = \Omega \\ \sum_{i=1}^{n_t} \left(e^{\delta}_{rq} + e^{\delta}_{\gamma,ser} + e^{r}_{\gamma,s} + e^{\theta}_{\gamma,c}\right), & \text{if } \kappa = \gamma, \end{cases} \tag{6.10}$$

where e^{δ}_{rq}, $e^{\delta}_{\kappa,ser}$, $e^{r}_{\kappa,s}$, and $e^{\theta}_{\kappa,c}$ are the STA's (i.e., MU or robot) energy consumption during task request dissemination, actor selection, and physical/digital sub-task execution, respectively. Further, note that e^{δ}_{rq} is given by $e^{\delta}_{rq} = e_b + e_{tra} = \hat{e}_i \cdot t^{b}_{w} + e_{\psi} \cdot t^{x}_{tra}$, where e_b and e_{tra} denote the energy consumption of task request buffering and transmission delay, respectively. We calculate the STA's energy consumption during actor selection using our DCS ($e^{\delta}_{\Omega,ser} = e_{rq} + e_{ar} + e_{ac} = f \cdot e_{\Phi} \cdot \hat{t}_{rq} + f \cdot e_{\psi} \cdot t_{ar} + e_{\Phi} \cdot t_{ac}$) and MCS policy ($e^{\delta}_{\gamma,ser} = \hat{e}_{rq} + \hat{e}_{ar} + \hat{e}_{ac} = \beta \cdot e_{\Phi} \cdot \hat{t}_{rq} + \beta \cdot e_{\psi} \cdot t_{ar} + e_{\Phi} \cdot t_{ac}$), where $e_{rq}/\hat{e}_{rq}$, $e_{ar}/\hat{e}_{ar}$, $e_{ac}/\hat{e}_{ac}$, and f/β denote the energy consumption during task request reception, response transmission, assignment message reception, and total number of robots, respectively. Hence, the STA's energy consumption during physical sub-task execution ($e^{r}_{\kappa,s}$) is equal to $e^{r}_{\kappa,s}$=$e_{\chi} \cdot t^{r}_{\kappa,s}$ =e^{b}_{r}+e^{Ψ}_{r}+e^{ϕ}_{r}=$\hat{e}_i \cdot t^{b}_{r}$+$e_{\varphi} \cdot$ t^{Ψ}_{r}+$e_{\sigma} \cdot t^{\phi}_{r}$, whereby $t^{r}_{\kappa,s}$ represents the physical sub-task execution time. For our DCS and MCS policy, $t^{r}_{\kappa,s}$ is equal to $t^{r}_{\Omega,s}$ and $t^{r}_{\gamma,s}$, respectively. Moreover, let e_{χ}, e^{b}_{r}, e^{Ψ}_{r}, and e^{ϕ}_{r} be the average energy consumption (per second) during physical sub-task execution, the robot's busy state ($e^{b}_{r} = \hat{e}_i \cdot t^{b}_{r}$), task location traverse ($e^{\Psi}_{r} = e_{\varphi} \cdot t^{\Psi}_{r}$), and physical sub-task workload processing delay ($e^{\phi}_{r} = e_{\sigma} \cdot t^{\phi}_{r}$), respectively. The STA's energy consumption during digital sub-task execution is obtained as

$e^{\theta}_{\kappa,c}$ in DCS ($e^{\theta}_{\kappa,c} = e^{\theta}_{\Omega,c}$) and MCS policy ($e^{\theta}_{\kappa,c} = e^{\theta}_{\gamma,c}$). Further, $e^{\theta}_{\kappa,c}$ is given by $e^{\theta}_{\kappa,c} = \hat{e}_h \cdot t^{\theta}_{\kappa,c} = e^{b}_{\theta} + e^{w}_{\theta} + e^{o}_{\theta} + e^{\pi}_{\theta} + e^{\phi}_{\theta} = \hat{e}_i \cdot (t^{b}_{\theta} + t^{w}_{\kappa,\theta}) + e_{\psi} \cdot t^{u}_{\theta} + e_{\Phi} \cdot t^{d}_{\theta} + \hat{e}_i \cdot (t^{\pi}_{\theta} + t^{\phi}_{\theta})$, where $\hat{e}_h$ and $t^{\theta}_{\kappa,c}$ are the average energy consumption and time required for digital sub-task execution, respectively, and e^{b}_{θ}, e^{w}_{θ}, e^{o}_{θ}, e^{π}_{θ}, and e^{ϕ}_{θ} denote the energy consumption in the agent's busy state (e^{b}_{θ}=$\hat{e}_i \cdot t^{b}_{\theta}$), offloading waiting ($e^{w}_{\theta} = \hat{e}_i \cdot t^{w}_{\kappa,\theta}$), communication delay ($e^{o}_{\theta} = e_{\psi} \cdot t^{u}_{\theta} + e_{\Phi} \cdot t^{d}_{\theta}$), computation processing ($e^{\pi}_{\theta} = \hat{e}_i \cdot t^{\pi}_{\theta}$), and cache look-up delay ($e^{\phi}_{\theta} = \hat{e}_i \cdot t^{\phi}_{\theta}$), respectively.

6.5.8 Communication to Computation Ratio

In this sub-section, we analyze another important performance metric: the communication to computation ratio (C2R). C2R is defined as the ratio of task execution communication delay (t^{ϑ}_{κ}) and task workload processing time (t^{ϵ}_{κ}) and may be applied in our DCS ($\kappa = \Omega$) and MCS ($\kappa = \gamma$) policies, whereby $C2R_{\kappa}$ is given by

$$C2R_{\kappa} = \frac{t^{\vartheta}_{\kappa}}{t^{\epsilon}_{\kappa}} = \frac{\sum_{i=1}^{n_t} t^{\delta}_{rq} + t^{\delta}_{\kappa,ser} + t^{w}_{\kappa,\theta} + t^{o}_{\theta}}{\sum_{i=1}^{n_t} t^{\phi}_{r} + t^{\pi}_{\theta} + t^{\phi}_{\theta}}. \tag{6.11}$$

6.5.9 Task Offload Gain to Overhead Ratio

Finally, we introduce another major performance metric, namely, the so-called task offloading time gain to overhead ratio (TGO). For the computation of TGO, we consider both task offloading and non-offloading versions of our DCS and MCS policies. More specifically, in the non-offloading version of our DCS/MCS policy, the selected robot executes the full task. Conversely, in the task offloading version of our DCS/MCS policy, the selected robot and agent process the physical and digital sub-task, respectively.

TGO is defined as the ratio of task execution time gain ($\hat{t}^{\delta}_{\kappa,n} - t^{\delta}_{\kappa,\mu}$) and offloading communication overhead in DCS/MCS task offloading policy. Thus, TGO is obtained as fellows:

$$TGO_{\kappa} = \frac{\hat{t}^{\delta}_{\kappa,\mu} - t^{\delta}_{\kappa,\mu}}{t^{\delta}_{\kappa,od} - \hat{t}^{\delta}_{\kappa,od}}, \tag{6.12}$$

where $t^{\delta}_{\kappa,\mu}$ and $t^{\delta}_{\kappa,od}$ represent the full task execution delay and communication overhead ($t^{\delta}_{\kappa,od} = t^{\delta}_{rq} + t^{\delta}_{\kappa,ser} + t^{w}_{\kappa,\theta} + t^{o}_{\theta}$) in the DCS ($\kappa$=$\Omega$) and MCS ($\kappa$=$\gamma$) task offloading policies. Consequently, $\hat{t}^{\delta}_{\kappa,\mu}$ and $\hat{t}^{\delta}_{\kappa,od}$ denote the full task execution delay and communication overhead in the non-offloading version of our DCS and MCS policies. Further, note that $\hat{t}^{\delta}_{\kappa,od}$ is given by $\hat{t}^{\delta}_{\kappa,od} = t^{\delta}_{rq} + \hat{t}^{\delta}_{\kappa,ser} + t^{w}_{\kappa,r} + t^{d}_{r}$, where $\hat{t}^{\delta}_{\kappa,ser}$ denotes the suitable robot's selection time in the non-offloading version of DCS ($\hat{t}^{\delta}_{\kappa,ser} = f \cdot \hat{t}_{rq} + f \cdot t_{ar} + f \cdot t_{cp} + y \cdot t_{ac}$) and MCS ($\hat{t}^{\delta}_{\kappa,ser} = \beta \cdot \hat{t}_{rq} + \beta \cdot t_{ar} + \beta \cdot t_{cp} + y \cdot t_{ac}$). Let t^{d}_{r} denote the task result transfer delay in the non-offloading version of our DCS and MCS policies, which is given by $t^{d}_{r} = \frac{s_v}{\psi_w} + \frac{s_v}{\psi_w} + t_{p4}$. Moreover, $\hat{t}^{\delta}_{\kappa,\mu}$ is

Table 6.1: Parameters and default values for evaluation of user preference-aware task and resource assignment scheme

Notation	**Definition**	**Default values/units**
n_t, w_π, w_ϕ, s_v	Task number, digital and physical sub-task workload, digital sub-task output data size	1-20, M cycles, KB (vary)
$e_\psi, e_\Phi, \hat{e}_i, e_\varphi, e_\sigma$	STA's average energy consumption (per second) for digital task upload, task result download, waiting delay, physical task location traversing, task workload processing	0.1W, 0.05W, 0.001W, .7W, .5W
$m, q_r, d_\Psi, \epsilon,$ $\nu_r/\nu_\theta, c_\theta$	STAs under each ONU, robot moving speed and distance from task location, steepness of Zipf distribution, robot/agent task processing speed, agent storage capacity	1-20, 1-10m/s, 1-100m, .5-1, 500/3200MHz, 3-10GB
$\beta/\Theta, z/\hat{v}$	Number of dedicated/non-dedicated robot, dedicated/total cloud agent	1-20, 1-6 (vary)
$x_w, y_f, \bar{\alpha}/\sigma/\tau$	Transmission capacity of Wireless and fiber link ($x_w = \psi_w$, $y_f = \Phi_{ct}/\phi_{fl}/\Psi_{fl}/\Phi_{cl}$), dedicated local cloudlet/non-dedicated cloudlet/dedicated remote cloud server number	6900(WLAN)/ 300(cellular) Mbps, 10 Gb/s(fiber),1-3
$t_c, t^c_\alpha/t^w_\alpha$	Polling cycle time, STA's cellular/WiFi residence time	100-800 ms, 0-15s (random)
$t^{msg}_{wl}, t^{msg}_{pon}, t^{\delta}_{\kappa,ser}$ g_i, s_w, t_d	Wireless (e.g., `PS-Poll`), MPCP message length (`GATE,REPORT`), actors selection delay, digital sub-task input, and computation sub-part output data size, task deadline	0.512μs, 0.231μs, μs, 50-1000 KB, 5-100 KB, 5-15s
$h^c_n/h^c_r, V/n_a,$ $t_{p_1}/t_{p_2}/t_{p_3}$	Hop distance between host ONU and nearby ONU/OLT, number of caching content/cloudlet, propagation delay for local cloudlet/non-local cloudlet/cloud offloading	2/4, 1000/4, ms (vary)
$\tilde{\omega}_1/\hat{\omega}_2, \upsilon_1/\upsilon_2,$ $\Delta_1/\Delta_2, \Lambda_1/\Lambda_2,$ $\vec{\lambda}_1/\tilde{\lambda}_2$	Monetary cost (per second) for dedicated/non-dedicated robot and agent, task request transfer, delay-sensitive, delay-tolerant task offloading using WiFi/cellular interface	.002/.008 \$ and .002/.008 \$, .01\$, .01\$, .001\$
$\hat{t}_{rq}/t_{ar}/t_{cp}/t_{ac}$ $t_p/t_{\hat{p}}/t^*_p,$	Task request/response/comparison/ack message duration, propagation delay caching (local →local cloudlet/local→non-local cloudlet/local cloudlet → remote cloud),	0.17/0.12/.20/0.12 μs, ms (vary)

given by $\hat{t}^{\delta}_{\kappa,\mu} = t^{\delta}_{rq} + \hat{t}^{\delta}_{\kappa,ser} + t^{r}_{\kappa,s} + t^{r}_{\kappa,c}$, whereby $t^{r}_{\kappa,s}$ and $t^{r}_{\kappa,c}$ represent the execution time of the physical sub-task ($t^{r}_{\kappa,s} = t^{b}_{r} + t^{\Psi}_{r} + t^{\phi}_{r}$) and digital sub-task ($t^{r}_{\kappa,c} = t^{\pi}_{r} + t^{\phi}_{r} + t^{w}_{\kappa,r} + t^{d}_{r}$) in the non-offloading version of our DCS and MCS policies, respectively, whereas t^{b}_{r}, t^{Ψ}_{r}, t^{ϕ}_{r}, t^{π}_{r}, t^{ϕ}_{r}, $t^{w}_{\kappa,r}$, and t^{d}_{r} denote the robot's busy time, task location traverse time, physical sub-task workload processing, and digital sub-task workload processing ($t^{\pi}_{r} = \frac{w_{\pi}}{\nu_{r}}$), cache look-up ($t^{\phi}_{r} = t^{l}_{\theta}$), task result buffering ($t^{w}_{\kappa,r} = t^{w}_{\kappa,\theta}$), and task result transfer delay, respectively.

6.6 Results

In this section, we present numerical results to investigate the performance of our proposed DCS and MCS based task execution schemes. Table 6.1 summarizes the parameters and their default values in accordance with [30], [17], [119], [120], and [121].

Assumptions and simulation setup: We assume that each full HART task request consists both physical (capturing an image at a location) and digital (detection of object from captured image and caching content access of the detected object) sub-task. Due to movement facilities physical sub-task can be executed only by robots, whereas location independent digital sub-task can be executed by robots/cloud agents based on their computing and caching abilities. Note that, output of the physical sub-task (captured image) is the input data size of digital sub-task that can be offloaded to cloud agent/nearby robots for processing. For delay-sensitive DCS policy users, the main requirements of the HART task execution are the selection of suitable actors from both dedicated and non-dedicated actors and preemptive bandwidth resource assignment. Whereas, for the delay-tolerant MCS policy users, the main requirements of the HART task execution are the utilization of only dedicated actors for suitable actors selection and non-preemptive bandwidth resource assignment. Note that, in this work ONU's are inter-connected by interconnected fiber links. Robots and MU's device is connected with the ONU-MPP/ONU-eNB's through the wireless link. Whereas, ONU's use dedicated point-to-point fiber links to transfer/receive their offloaded task input/output data to/from cloudlet server. The computation processing speed of the CPU of each robot and cloud agent server is set to 500 and 3200 MHz, respectively. The fiber backhaul transmission capacity is set to 10 Gb/s in both uplink and downlink, whereas the fiber backhaul length between the OLT and ONUs is 20 Km. At the wireless front-end, maximum data rates of 300 Mb/s (cellular link) and 6900 Mb/s (WLAN physical line rate) is considered. The fiber backhaul range between the ONU to cloudlet server and OLT to remote cloud server is 1 Km and 10 Km, respectively. Hence, the MAP radius, ONU-MPP/ONU-eNB coverage area, and density of MAPs within each ONU-MPP coverage area is set to 100 m, 9 km^2, and 3, respectively. The full HART task arrival number, physical and digital sub-task workload, offloaded (digital) task input and output data size values are chosen randomly in the range of

1-20, 50-1000 M cycles, 50-1000 KB, 10-200 KB, respectively. A STA's average energy consumption (per second) for digital sub-task upload, result download, waiting (idle) time, task workload processing is set to 0.1W, 0.05W, 0.001W, and 0.5W, respectively, similar to [17]. The polling cycle time and task deadline values are varied within the range of 100-800 ms and 5-30s, respectively. Moreover, we assume that the monetary cost of using non-dedicated actors (owned by network operators) is higher than that of dedicated actors (owned by MUs). We also assume that the usage of a delay-sensitive offload time sub-slot implies higher monetary costs than the usage of a delay-tolerant offload time subslot. Thus, to highlight the impact of the examined cloud/robot selection and usage of bandwidth resources on the performance, the average monetary cost value (per second) for dedicated/non-dedicated actor usage as well as delay-sensitive and delay-tolerant task offloading via the WiFi/cellular interface are set to 0.002\$/0.008\$, 0.01\$/0.001\$, respectively. To visualize the effect of the actors' impact on the performance, the number of dedicated/non-dedicated robots and cloud agents is varied within the range of 1-20 and 1-6, respectively. The duration of the MPCP (t_{msg}^{pon}) and WLAN messages (t_{msg}^{wl}) is set to 0.512µs and 0.231µs, respectively, similar to [105]. The remaining parameters and their default values are described below in Figs. 6.6 and 6.7. The system settings used in the evaluation (Scenario 1) is given by: $\beta = 20$, $\Theta = 12$, $z = 2$, $\hat{v} = 6$, $w_\pi = w_\phi = 100, 50, 200, 150, 400, 300, 250, 350, 450, 500, 600, 550, 650, 800, 750, 700, 950, 1000,$ $850, 900$ Mcycles, $s_v = 20, 10, 40, 30, 80, 60, 50, 70, 90, 100, 120, 110, 130, 160, 150, 140, 190, 200,$ $170, 180$KB.

Performance analysis: Figs. 6.6(a) and 6.6(b) depict the average aggregate task execution time performance for our proposed DCS and MCS based task execution policies, respectively. Both figures indicate that for an increasing task arrival number, the average task execution delay increases in all compared versions of DCS and MCS policies, i.e., MTMD, MT, and MD. Specifically, we observe that for both DCS and MCS based task execution, the MTMD based scheme achieves a significantly lower time delay than its counterparts. Further, we observe that for both DCS and MCS based task execution, the MD based scheme results in the second lowest average task execution time. By contrast, the MT based scheme shows an inferior average task execution time performance. This is due to the fact that in the MTMD based scheme both robot and cloud agent are selected by taking not only their task workload processing delay but also the incurred waiting and communication delay into account. Furthermore, the network interface (wired/wireless) providing the highest data rate is selected for multiple-task offloading activities. In the MTMD based scheme, robot/cloud agent and bandwidth resources are assigned to multiple tasks based on their lower task execution deadline requirement. Hence, in the MT based scheme, the selection of the highest available data rate is ensured for each task offloading activity. Note that the MT based scheme

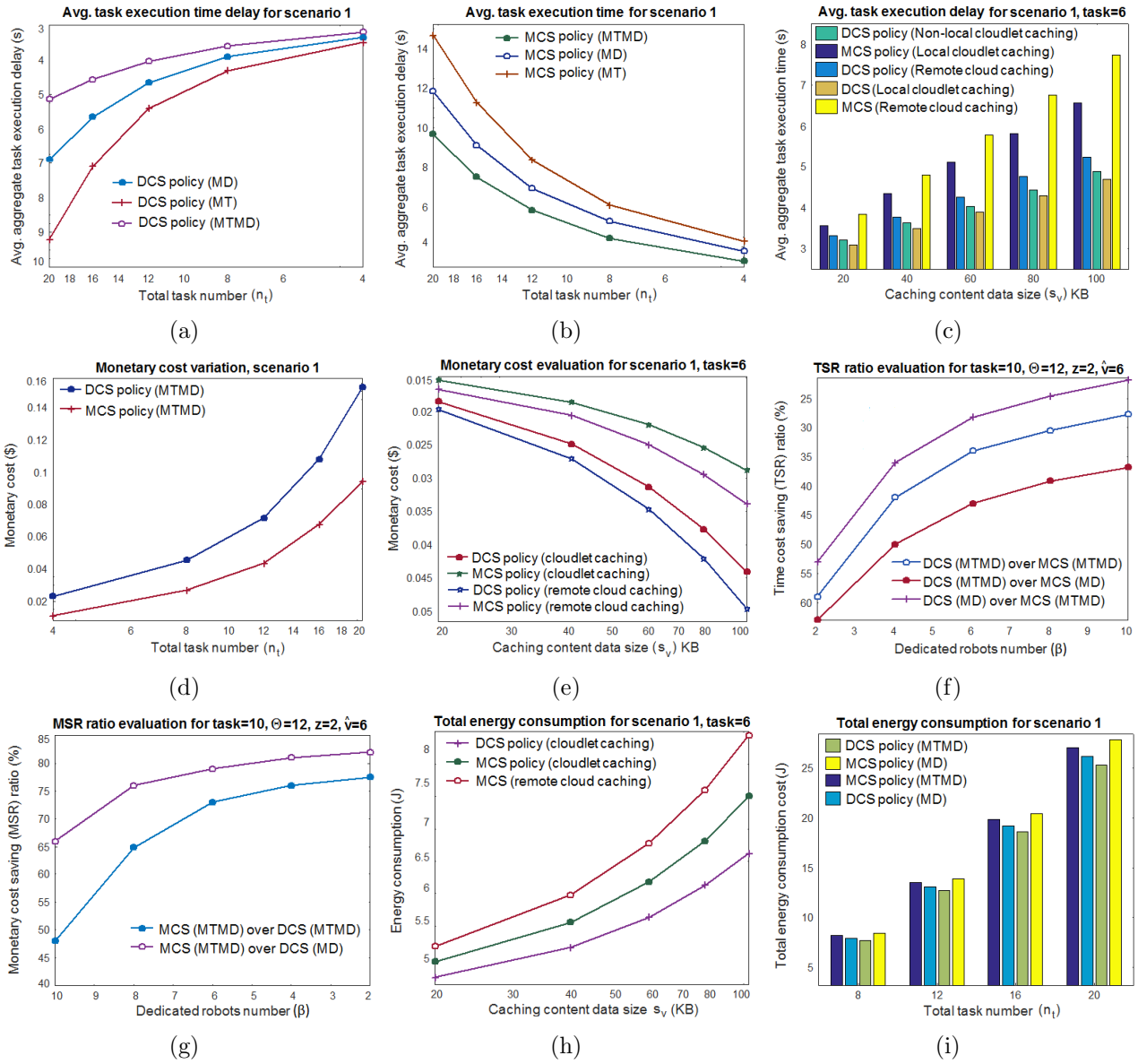


Figure 6.6: Average task execution time, monetary cost, task execution time cost saving ratio, monetary cost saving ratio, and total energy consumption cost performance for scenario 1

assigns suitable actors (robot/cloud agent) for each task based in random order. Hence, in the MD based scheme, the task scheduler selects suitable actors with a higher task processing speed for each arriving task. As a result, the MD based scheme minimizes the task workload processing delay rather than the full task execution time, which consists of both the task workload processing and offloading communication delays. From Figs. 6.6(a) and 6.6(b) we also observe that for varying task numbers, the DCS task execution policies outperform their MCS based counterparts in terms of task execution time. This is because DCS policy users give preemptive access to actors and bandwidth resources for executing their delay-sensitive tasks.

Fig. 6.6(c) examines the suitability of different cloud agent selections for the execution of digital tasks using our proposed DCS and MCS policies. The figure shows that for an increasing caching content data size (s_v), the average task execution time increases rapidly for all compared DCS and MCS policies. The figure also indicates that if the priority is achieving a lower task execution delay, the DCS (local cloudlet caching) scheme is more suitable than all its counterparts. We notice that the MCS (remote cloud caching) scheme experiences the maximum task execution delay due to its higher task offloading communication overhead. For instance, for $n_t = 6$ and $s_v = 60$ KB, the DCS (local cloudlet caching) scheme yields approximately a 3.46%, 8.45%, 23.67%, and 32.52% higher task execution delay gain than the DCS (non-local cloudlet caching), DCS (remote cloud caching), MCS (local cloudlet caching), and MCS (remote cloud caching) schemes, respectively. Fig. 6.6(d) depicts the monetary cost versus total task number (n_t) performance for both DCS (MTMD) and MCS (MTMD) policies. The figure reveals that for both DCS (MTMD) and MCS (MTMD) policies, the monetary cost remains low for low values of n_t, but rapidly increases for larger n_t. Moreover, for varying task numbers, the MCS (MTMD) policy outperforms the DCS (MTMD) policy in terms of minimum monetary cost. This is because unlike the DCS (MTMD) policy, the MCS (MTMD) policy relies on dedicated robots/agents and non-preemptive bandwidth resources for their requested task execution. Fig. 6.6(e) shows the monetary cost versus caching content data size (s_v) performance for different DCS and MCS policies. We observe that for both higher and lower s_v, the MCS (cloudlet caching) and DCS (remote cloud caching) schemes offer the lowest and highest monetary cost, respectively.

Figs. 6.6(f) and 6.6(g) clearly show that a shortage of dedicated robots (β) has a detrimental impact on the task execution time and monetary cost saving performance of our proposed DCS and MCS policies. Both figures show that for different β, the time and monetary cost saving ratio is maximum in the DCS (MTMD) and MCS (MTMD) policies, respectively. Note that a lower dedicated robots availability results in a higher task execution delay in the MCS (MTMD) than DCS (MTMD) policy. Hence, the use of both dedicated and non-dedicated

actors cause additional monetary cost in the DCS policy, as opposed to the MCS (MTMD) policy. For instance, $n_t = 10$ and $\beta = 6$, the time cost saving ratio in the DCS (MTMD) policy compared with the MCS (MTMD) and MCS (MD) policies is approximately 34% and 45%, respectively. Hence, for $n_t = 10$ and $\beta = 6$, the monetary cost saving ratio in the MCS (MTMD) policy compared with the DCS (MTMD) and DCS (MD) policies is 74% and 80%, respectively.

Fig. 6.6(h) examines the impact of varying caching content data sizes (s_v) on the energy consumption cost for different DCS and MCS policies. We observe that the energy consumption cost is lower for small s_v and higher for large s_v in the considered DCS and MCS policies. We also note that the DCS (cloudlet caching) scheme achieves the minimum energy consumption cost of all compared approaches. Fig. 6.6(i) depicts the energy consumption cost versus task number (n_t) for both DCS and MCS based policies. The figure shows that the energy consumption cost rises rapidly for an increasing n_t in all compared policies. Note that for different n_t, the DCS (MTMD) policy offers a lower energy consumption cost than the alternative policies. Also note that due to the higher task execution delay, the MCS based policies suffer from a higher energy consumption cost than the alternative policies. For instance, with $n_t = 12$, the energy consumption gain of the DCS (MTMD) policy over the MCS (MTMD), DCS (MD), and MCS (MD) policy equals 6.71%, 3.4%, and 9.42%, respectively.

Next, in Fig. 6.7(a), we investigate the mean task offload delay performance of our proposed DCS and MCS policies. The figure shows that for varying task numbers, the mean task offload delay is the lowest in our DCS (MTMD) and highest in our MCS (MTMD) policies. This is because MCS policy users can offload their tasks to a dedicated cloud agent only after the completion of DCS policy users' task offloading. Fig. 6.7(b) quantifies the impact of a varying task number (n_t) on the TGO performance of both DCS (MTMD) and MCS (MTMD) based task offloading policies and compares it with their non-offloading (only robot based execution) counterparts. We notice that for an increasing task number, the TGO ratio increases rapidly in all compared schemes. We observe that for small and large values of n_t, the maximum TGO ratio is obtained in the DCS (MTMD) policy due to its lower digital task processing overhead. For instance, for $n_t = 4$, the TGO ratio in the DCS (MTMD) and MCS (MTMD) policies is approximately 62% and 46%, respectively. Fig. 6.7(c) depicts the average caching content access delay for our DCS and MCS policies. We observe from the figure that for an increasing caching content data size (s_v), the caching content access delay grows rapidly in all compared schemes. In addition, we note that the caching content access delay becomes the lowest in the DCS policy, if the host local cloudlet fetches cached content from another local cloudlet. The figure also shows that the caching content access delay becomes the highest in the MCS policy, if the host local cloudlet fetches cached content from the remote cloud server.

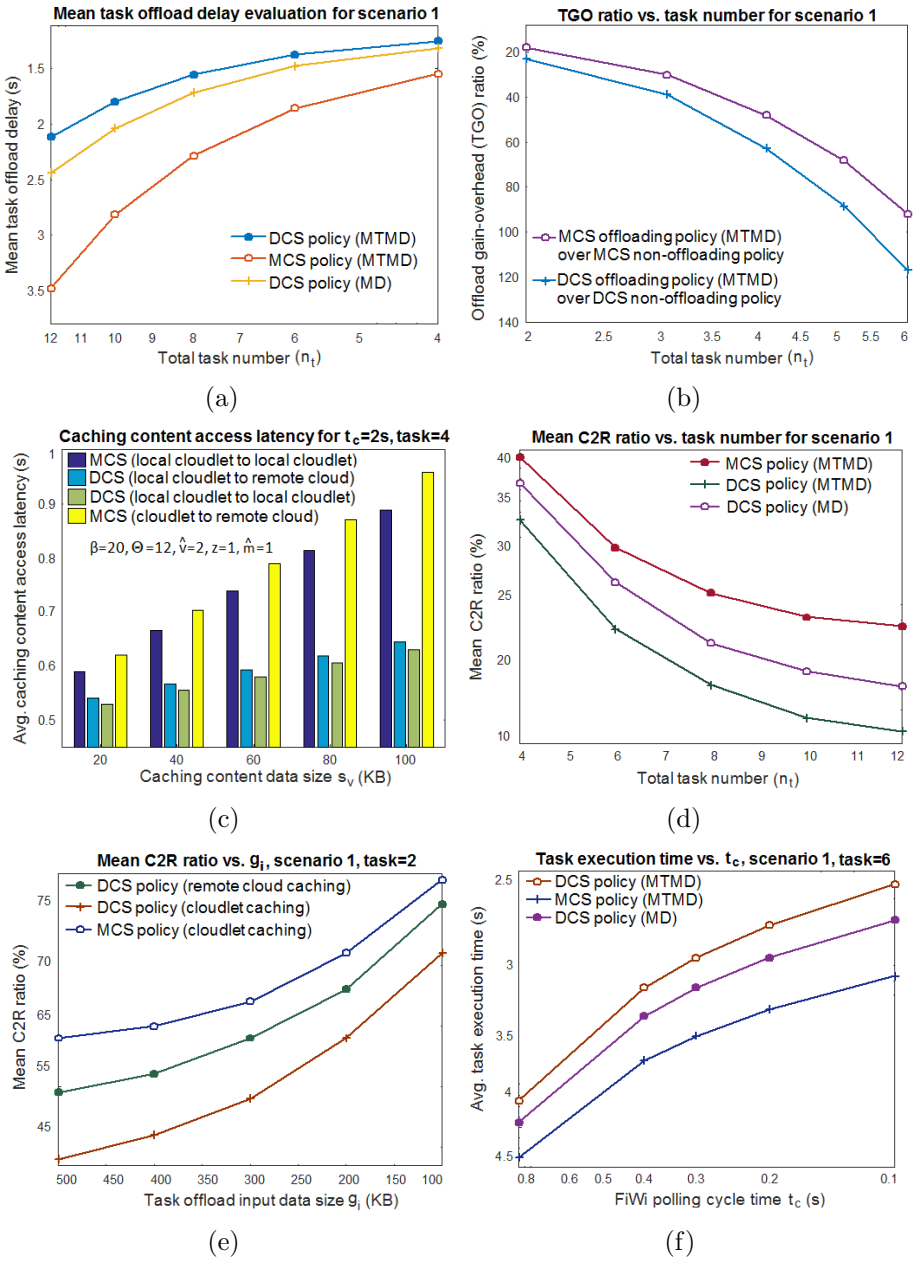


Figure 6.7: Mean Task offload delay, TGO ratio, C2R ratio, caching content access latency, and average task execution time performance.

For instance, for $s_v = 80$ MB, $\hat{m} = 1$, and $n_t = 4$, the DCS (local cloudlet to local cloudlet) scheme achieves an approximately 25% and 31% higher caching content access delay gain than the MCS (local cloudlet to local cloudlet) and MCS (local cloudlet to remote cloud) schemes, respectively.

Fig. 6.7(d) depicts the impact of varying task numbers on the mean C2R ratio for DCS and MCS based policies. Note that a task execution scheme with a lower C2R ratio incurs a lower communication overhead, which is more beneficial for executing delay-sensitive HART tasks. The figure shows that the mean C2R ratio decreases for an increasing task number. This is is due to the fact that the task workload processing time is inversely proportional to the C2R ratio in all compared policies. Note that the DCS (MTMD) policy offers a smaller mean C2R ratio than the other alternative policies. Fig. 6.7(e) examines the mean C2R ratio versus task offload input data size (g_i) performance for different DCS and MCS policies. The figure shows that for varying g_i, the mean C2R ratio becomes minimum in the DCS (cloudlet caching) policy, as opposed to both DCS (remote cloud caching) and MCS (cloudlet caching) policies. This is because the DCS (cloudlet caching) policy experiences a smaller digital task offloading delay than both DCS (remote cloud caching) and MCS (cloudlet caching) policies. For instand, for $g_i = 400$ KB and $n_t = 2$, the DCS (cloudlet caching) scheme achieves a 7% and 12% lower mean C2R ratio than the DCS (remote cloud caching) and MCS (cloudlet caching) schemes, respectively. Finally, Fig. 6.7(f) illustrates the impact of the polling cycle time (t_c) on the average task execution delay of our proposed DCS and MCS policies. We notice that the average task execution delay increases for increasing t_c. Furthermore, we observe that for small and large values of t_c, the DCS (MTMD) policy achieves a higher average task execution time gain than the alternative policies. For instance, for $t_c = 0.3$ s, the average task execution time gain of the DCS (MTMD) policy over the MCS (MTMD) policy equals 15.42%, as opposed to only 6.03% over the DCS (MD) policy. This result indicates that for the execution of delay-sensitive tasks, the DCS (MTMD) policy is the superior solution.

6.7 Conclusions

In this chapter, we investigated the performance of user preference aware HART task execution over FiWi enhanced network infrastructures for the emerging Tactile Internet. To minimize the task execution delay of DCS policy users, our proposed task coordination scheme selects suitable actors by using both dedicated and non-dedicated actors. Conversely, to maximize the monetary cost saving of MCS policy users, our proposed scheme selects appropriate actors only from the set of dedicated actors. Furthermore, we presented a proactive bandwidth allocation scheme that assigns preemptive and non-preemptive bandwidth resources to DCS and MCS policy users, respectively. We also developed an analytical framework to evaluate

the performance of our DCS and MCS policy based task execution in terms of monetary cost and task execution time saving ratio, energy consumption, mean task offload delay, TGO and C2R ratios, and caching content access delay. Our obtained results show that for a typical number of 10 tasks and 8 available dedicated robots, the DCS (MTMD) policy exhibits a higher task execution time saving ratio of 30.5% and a lower monetary cost saving ratio of 63.6% than the MCS (MTMD) policy. Unlike alternative approaches, our findings indicate that the MTMD policy is useful for both DCS and MCS policy users due to its minimum task execution time and monetary cost. Our proposed user preference aware task coordination policy thus represents a promising solution to reduce both task execution delay and monetary cost for emerging Tactile Internet applications.

Chapter 7

Conclusions and Future Research

This chapter summarizes the contributions of this thesis and outlines some future research directions in the field of HART-centric task coordination over FiWi enhanced network infrastructures.

7.1 Conclusions

Unlike the IoT without any human involvement in its underlying machine-to-machine communications, the Tactile Internet involves the HART-centric collaboration and thus allows for a human-centric design approach towards creating and consuming novel immersive experiences via the Internet. This thesis tried to shed some light on the augmentation (i.e., extension of capabilities) of the human through the HART-centric collaborative task execution framework. To reap the benefits from human-machine convergence, this thesis presented a suitable task coordination framework for efficiently orchestrating the real-time collaboration among human mobile users, centralized and decentralized computational agents (cloud/cloudlets), and collaborative robots (cobots) across converged FiWi enhanced network infrastructures. In light of the emerging Tactile Internet moving towards decentralization based on edge computing, intelligent base stations, collaborative cloud computing (robots and cloudlets), the inherent distributed processing and storage capabilities of FiWi enhanced networks were exploited for the execution of local and non-local HART-centric tasks. The doctoral thesis focused on HART task coordination over FiWi enhanced networks focusing on three major issues, namely, power and latency-aware task assignment, failure avoidance, and prefetching-aware bandwidth resource assignment.

For the cost-effective HART task execution over FiWi enhanced networks, delay and power saving issues must be handled in a comprehensive fashion by taking into account task properties, dynamically changing bandwidth availabilities, and collaborative node resources (robot and cloud agents). To do so, this thesis proposed a novel HART task coordination scheme

for FiWi enhanced networks that assigns not only tasks to suitable actors but also bandwidth resources for task offloading/result transfer activities. The proposed HART task coordination scheme achieves more than 53% delay cost saving while saving monetary costs of more than 48% (in typical scenarios).

Different problems were explored and suitable schemes were proposed for the latency-aware HART task execution over FiWi enhanced networks. More specifically, to render the human-to-robot task allocation process more efficient, in Chapter 2 we proposed a local and non-local task allocation scheme for MUs' requested task execution according to several key design parameters such as the availability, skill set, distance to task location, and remaining energy of robots. Furthermore, to reduce failures during task execution, we presented a neighboring robot assisted failure reporting mechanism. Our results show that the estimated minimum execution time-based robot selection outperforms traditional minimum distance and priority based selection schemes in terms of end-to-end delay and average residual energy. Moreover, we observed that the non-local task allocation delay is higher than the local task allocation delay.

In Chapter 3, we presented a collaborative computing strategy that combines suitable host robot selection for sensing sub-task execution and collaborative node selection for computation sub-task offloading. We exploited conventional cloud servers, decentralized cloudlets, and neighboring robots as collaborative nodes for computation offloading in support of a host robot's requested computation sub-task execution. The results of both collaborative and non-collaborative task execution schemes demonstrate that for a typical scenario the collaborative task execution scheme improves the task response time delay by up to 8.75% and the energy consumption by up to 14.98% compared to the non-collaborative task execution scheme.

For resource-efficient task execution, Chapter 4 proposed a context-aware task migration scheme for efficiently orchestrating the real-time collaboration among human mobile users, central and decentralized computational agents (cloud/cloudlets), and collaborative robots (cobots) across converged FiWi communications infrastructures. We investigated the problem of whether and, if so, when and where a HART-centric task should be best migrated to. For resource-efficient task execution, the migration decision is made according to given task processing capabilities of cloud/cloudlet agents and cobots, task execution deadline, energy consumption of involved cobots and mobile devices, and task migration latency. Our results show that for a typical cognitive sub-task input data size of 600 MB, the cobot-to-agent (cloudlet near task location) cognitive task migration scheme achieves more than 20% task response time improvement and 23% energy savings over the traditional non-migration scheme. The results also show that intra-agent cognitive sub-task migration achieves a higher task response time gain than inter-agent migration.

In Chapter 5, we investigated a community- and latency-aware multiple HART task scheduling scheme by using real-time information about arriving task requests for both isolated and clustered robots/agents. More specifically, we investigated the optimal multi-task scheduling order and resource assignment strategy for task on- and offloading based HART task execution with task prefetching and fault tolerance capabilities. To reap the benefits from task prefetching for the execution of multiple HART tasks, we presented a novel prefetching-aware bandwidth allocation scheme that copes with conventional broadband and task offloading data traffic at the same time. Our presented results show that for a typical system of 32 ONU-MPPs and a polling cycle time of 100 ms, our proposed prefetching-aware task offloading scheme achieves up to 31.3% and 32.7% task completion time gain over the task onloading scheme for nearby and remote HART task execution, respectively.

Lastly, in Chapter 6, to achieve minimum task execution delay and monetary cost, we developed a user preference-aware HART task coordination framework that selects appropriate dedicated/non-dedicated robot/cloud agents for executing different caching and computing delay-sensitive and delay-tolerant HART tasks. Further, to cope with limited bandwidth resources, we proposed a proactive bandwidth allocation policy for both delay-sensitive and delay-tolerant HART task execution. We observed that for a task number of 10 and 8 available dedicated robots, our proposed DCS (MTMD) policy exhibits an up to 30.5% higher time saving ratio and a 63.6% lower monetary cost saving ratio over the alternative MCS (MTMD) policy.

7.2 Future Research

Due to their coverage and capacity advantages, FiWi enhanced networks have great potential to ensure QoS for several emerging local and non-local HART-centric applications, thus creating new opportunities for several industries including manufacturing, industrial automation, education, transport, entertainment, and health-care. Importantly, the integration of human users, robots, remote cloud and decentralized cloudlet resources over FiWi enhanced networks creates a powerful paradigm for not only the emerging Tactile Internet but also for future network and communications research. Consequently, the contributions made in this thesis can be extended for additional improvements. Some interesting future research directions that may build on our proposed schemes are described below in greater detail.

(i) **Self-aware HART task coordination:** With the advent of safe collaborative robots and agents, their seamless integration into human teams as teammates starts to gain steam as part of the vision of the emerging Tactile Internet, which lies at the nexus of computerization, automation, and robotization. While necessary, low task execution time and ultra-reliable human-robot-agent connectivity are not sufficient to unleash the full potential of the resultant

human-agent-robot teamwork (HART) applications. The automation of various physical and digital HART tasks with self-aware requirements is doable by state-of-the-art agents and robots. Regardless of whether a technological advance is labor-saving or capital-saving, skill-biased or not, and regardless of the speed with which robots or other machines approach or exceed human skill sets, the key to the effect of the new technologies on human wellbeing is who owns the technologies [122]. If other persons owned our replacement technologies, we would become jobless. Instead, if users owned them, humans would have their current earnings and their time freed from labor to seek other productive activity. To unleash the full potential of HART applications, one future research direction involves the development of self-aware HART task coordination schemes for physical and digital task execution based on the shared use of user- and network-owned robots/agents. HART members are assumed to be self-aware about their respective goals, application needs, capabilities, and constraints. Further, through communication, they can establish a collective context-awareness with the objective of minimizing the completion time of tasks by robots and agents, which may be either user-owned or network-owned. Beside the minimization of the task completion time, another major objective of this work may be the minimization of energy consumption and operational expenditures (OPEX) of physical/digital task execution by mobile robots and agents. Specifically, the question of when, how, and under which circumstances user-ownership of mobile robots and cloud agents becomes beneficial in terms of OPEX per executed task represents an interesting research problem.

(*ii*) **Online-to-Offline/Offline-to-Online (O2O) communication**: The increasing demands for online resources (e.g., cloud, cloudlet resources) create a huge challenge for MUs' task execution due to their preferred energy consumption and data usage cost plan. As a solution, online-to-offline/offline-to-online (O2O) communication allows HART members to utilize both online (e.g., cloud resources) and offline resources (e.g., own or nearby mobile devices) for task execution. O2O communications aims to maximize the use of offline and online resources by allowing collaboration and communication among each HART member in order to achieve win-win situations [123]. By leveraging O2O service migration with resource awareness, HART members may share their bandwidth, computation, and storage resources to stimulate beneficial cooperation, which can cut down users energy consumption and data usage cost. Hence, to cope with an insufficient energy and data usage budget of MUs, research in the area of O2O based joint task migration along with appropriate bandwidth sharing schemes is another promising direction.

(*iii*) **Human-machine interaction based on mixed reality applications**: Virtual reality (VR) creates a computer generated 3D environment, which can be experienced by human users. VR enables users to observe not only the defined objects but also the real environments

and its objects based on their choices. Remote robotic surgery simulation, flight simulation, and science fiction movies are some application areas of VR [124]. In augmented reality (AR), the real environment is augmented by computer-generated virtual data (e.g., visual, auditory, haptic). AR represents the immersive aspects of the real environment [125], [126]. AR changes people's perception about the real-world environment. Conversely, VR technology changes the real environment with a 3D simulated environment. Mixed reality (MR) integrates the immersive capabilities of VR with AR so that data can be transferred back into the physical world. More specifically, MR allows not only the combination of real and virtual worlds but also allows the coexistence of real and virtual objects and their real-time interaction. Thus, by allowing for the collaboration and interaction of humans with real and virtual worlds (e.g., machines), MR creates a new medium for both consumer and enterprise domains [127]. For the cost-effective deployment of mixed reality applications, the development of adaptive real-time synchronization, communication, and computation techniques opens up a multitude of future research opportunities.

(*iv*) **Optimization techniques for human-machine coactivity**: Another future scope of our work is to develop and implement suitable optimization techniques (e.g., particle swarm optimization) to minimize both network and cloud resource usage cost for human-machine coactivity based task execution, while satisfying the task execution requirements, i.e., the deadline.

(*v*) **Realization of very low-latency and high reliability requirements for haptic communications based applications**: To achieve very low end-to-end latency of 1 ms for real-time haptic communications based applications (e.g., remote robot steering and control, transfer of touch senses), different challenges and requirements need to be addressed properly. For example, at the physical layer one way packet transmission needs to satisfy the transmission duration of 100 µs (packet lengths). To meet this requirement, each packet duration should not over 33-µs packet duration. This is because for one-way packet transmission latency calculation some additional latencies need to be included: protocol processing, encoding at the transmitter, and decoding at the receiver. Note that, in current LTE cellular systems, the duration of one orthogonal frequency division multiplexing (OFDM) symbol alone is close to 70-µs long. Thus, research in the area of cellular physical layer for very low-latency haptic communications needs to be revisited. To achieve 1 ms round-trip latency, another major requirement is that the control server needs to be placed within 150 Km distance from the tactile point of interaction. Further, the satisfaction of carrier grade reliability (failure rate of 10^{-7}) for haptic/tactile senses transmission/reception over cellular networks is another key challenge. To achieve optimal results for real-time haptic communication research in the area of development of advanced tactile/haptic devices, perception-based encoder/decoder,

advanced visual-haptic multiplexing scheme, collaborative multi-user haptic operation management, and priority based radio resource management scheme is mandatory.

(*vi*) **Advanced artificial-intelligence (AI) based prediction techniques for haptic applications**: Development of advanced caching, computing, and user-oriented traffic management system at the network edge (cellular/WiFi base station) would improve the de-congestion problem of the core network [35]. Most importantly, development and implementation of artificial-intelligent engines at the network edge can lower the end-to-end latency by predicting the haptic/tactile experience, i.e. acceleration of movement on one end and the force feedback on the other. Note that, currently simple linear regression algorithms are used to predict the movement and reaction for fairly repetitive skill set driven actions between tactile/haptic devices that requires 10-100 milliseconds. Hence, more advanced artificial Intelligence based prediction technique needs to be developed to reduce the end-to-end latency in different real-time scenarios where the predicted action/reaction values are deviating from the real-time values, so that some coefficients of the predicting models are need to be updated and transmitted to the other end for corrections and damage recovery.

Bibliography

[1] M. Condoluci, T. Mahmoodi, E. Steinbach, and M. Dohler, "Soft Resource Reservation for Low-Delayed Teleoperation Over Mobile Networks," *IEEE Access*, vol. 5, pp. 10445 – 10455, May 2017.

[2] G. Fettweis and S. Alamouti, "5G: Personal Mobile Internet Beyond What Cellular Did to Telephony," *IEEE Communications Magazine*, vol. 52, no. 2, pp. 140–145, Feb. 2014.

[3] ITU-T Technology Watch Report, "The Tactile Internet," pp. 1–24, Aug. 2014.

[4] G. Fettweis, "The Tactile Internet: Applications and Challenges," *IEEE Vehicular Technology Magazine*, vol. 9, no. 1, pp. 64–70, Mar. 2014.

[5] M. Maier, M. Chowdhury, B. P. Rimal, and D. P. Van, "The Tactile Internet: Vision, Recent Progress, and Open Challenges," *IEEE Communications Magazine*, vol. 54, no. 5, pp. 138–145, May 2016.

[6] M. Maier, A. Ebrahimzadeh, and M. Chowdhury, "The Tactile Internet: Automation or Augmentation of the Human?," *IEEE Access*, vol. 6, no. 1, pp. 41607–41618, Dec. 2018.

[7] M. Satyanarayanan, R. Schuster, M. Ebling, G. Fettweis, H. Flinck, K. Joshi, and K. Sabnani, "An Open Ecosystem for Mobile-Cloud Convergence," *IEEE Communications Magazine*, vol. 53, no. 3, pp. 63–70, Mar. 2015.

[8] Mobile-Edge Computing (MEC) Industry Initiative, Introductory Technical White Paper, "Mobile-Edge Computing," *The European Telecommunications Standards Institute Portal*, Sep. 2014.

[9] H. Beyranvand, M. Lévesque, M. Maier, J. A. Salehi, C. Verikoukis, and D. Tipper, "Toward 5G: FiWi Enhanced LTE-A HetNets With Reliable Low-Latency Fiber Backhaul Sharing and WiFi Offloading," *IEEE/ACM Transactions on Networking*, vol. 25, no. 2, pp. 690–707, Apr. 2017.

[10] A. Aijaz, "Toward Human-in-the-Loop Mobile Networks: A Radio Resource Allocation Perspective on Haptic Communications," *IEEE Transactions on Wireless Communications*, IEEE Early Access, June 2018.

[11] M. Johnson, J. Bradshaw, P. Feltovich, C. Jonker, B. Riemsdijk, and M. Sierhuis, "Autonomy and Interdependence in Human-Agent-Robot Teams," *IEEE Intelligent Systems*, vol. 27, no. 2, pp. 43–51, Apr. 2012.

[12] S. Giordani, M. Lujak, and F. Martinelli, "A Distributed Algorithm for the Multi-Robot Task Allocation Problem," *Trends in Applied Intelligent Systems, LNCS, Springer*, vol. 6096, pp. 721–730, June 2010.

[13] G. Li, Y. Tamura, and H. Asama, "Dynamical Task Allocation and Reallocation Based on Body Expansion Behavior for Multi-robot Coordination System," *in Proc., IEEE ICMA*, pp. 537–542, Beijing, China, Aug. 2011.

[14] U. Gurel, N. Adar, and O. Parlaktuna, "Priority-Based Task Allocation in Auction-Based Applications," *in Proc., IEEE INISTA*, pp. 1–5, Albena, Bulgaria, June 2013.

[15] R. Goyal, T. Sharma, and R. Tiwari, "Priority Based Multi Robot Task Assignment," *Advances in Swarm Intelligence, LNCS, Springer*, vol. 7331, pp. 554–563, June 2012.

[16] P. Patil, A. Hakiri, and A. Gokhale, "Cyber Foraging and Offloading Framework for Internet of Things," *in Proc., IEEE Computer Software and Applications Conference (COMPSAC)*, pp. 359–368, Atlanta, USA, June, 2016.

[17] W. Zhang, Y. Wen, and D. Oliver Wu, "Energy-efficient Scheduling Policy for Collaborative Execution in Mobile Cloud Computing," *in Proc., IEEE International Conference on Computer Communications (INFOCOM)*, pp. 190–194, Turin, Italy, Apr. 2013.

[18] T. Penner *et al.*, "Transient Clouds: Assignment and Collaborative Execution of Tasks on Mobile Devices," *in Proc., IEEE GLOBECOM*, pp. 2801–2806, Austin, TX, USA, Dec. 2014.

[19] L. Gkatzikis and I. Koutsopoulos, "Mobiles on Cloud Nine: Efficient Task Migration Policies for Cloud Computing Systems," *in Proc., IEEE International Conference on Cloud Networking (CloudNet)*, pp. 204–210, Luxembourg, Oct. 2014.

[20] M. Mishra, A. Das, P. Kulkarni, and A. Sahoo, "Dynamic Resource Management Using Virtual Machine Migrations," *IEEE Communications Magazine*, vol. 50, no. 9, pp. 34–40, Sep. 2012.

[21] S. Shaw and A. Singh, "Use of Proactive and Reactive Hotspot Detection Technique to Reduce the Number of Virtual Machine Migration and Energy Consumption in Cloud Data Center," *Computers and Electrical Engineering Journal*, vol. 47, pp. 241–254, Oct. 2015.

[22] T. Rodrigues, K. Suto, H. Nishiyama, and N. Kato, "Hybrid Method for Minimizing Service Delay in Edge Cloud Computing through VM Migration and Transmission Power Control," *IEEE Transactions on Computers*, vol. 66, no. 5, pp. 810–819, May 2017.

[23] M. Alicherry and T. V. Lakshman, "Network Aware Resource Allocation in Distributed Clouds," *in Proc., IEEE INFOCOM*, pp. 963–971, March 2012.

[24] I. Yaqoob, E. Ahmed, A. Gani, S. Mokhtar, and M. Imran, "Heterogeneity-Aware Task Allocation in Mobile Ad Hoc Cloud," *IEEE Access*, vol. 5, pp. 1779–1795, Feb. 2017.

[25] Z. Lu, J. Zhao, Y. Wu, and G. Cao, "Task Allocation for Mobile Cloud Computing in Heterogeneous Wireless Networks," *in Proc., IEEE ICCCN*, pp. 1–9, Las Vegas, Aug. 2015.

[26] T. Shi, M. Yang, X. Li, Q. Lei, and Y. Jinag, "An Energy-Efficient Scheduling Scheme for Time-Constrained Tasks in Local Mobile Clouds," *Pervasive and Mobile Computing*, vol. 27, pp. 90–105, Apr. 2016.

[27] X. Zhu, C. Chen, L. Yang, and Y. Xiang, "Angel: Agent-Based Scheduling for Real-Time Tasks in Virtualized Clouds," *IEEE Transactions on Computers*, vol. 64, no. 12, pp. 3389–3403, Dec. 2015.

[28] L. Gkatzikis and I. Koutsopoulos, "Migrate or Not? Exploiting Dynamic Task Migration in Mobile Cloud Computing Systems," *IEEE Wireless Communications Magazine*, vol. 20, no. 3, pp. 24–32, June 2013.

[29] L. Zhou, Z. Yang, J. Rodrigues, and M. Guizani, "Exploring Blind Online Scheduling for Mobile Cloud Multimedia Services," *IEEE Wireless Communications*, vol. 20, no. 3, pp. 54–61, June 2013.

[30] H. Zhang, Q. Zhang, and X. Du, "Toward Vehicle-Assisted Cloud Computing for Smartphones," *IEEE Transactions on Vehicular Technology*, vol. 64, no. 12, pp. 5610–5618, Dec. 2015.

[31] H. Wu and K. Wolter, "Stochastic Analysis of Delayed Mobile Offloading in Heterogeneous Networks," *IEEE Transactions on Mobile Computing*, vol. 17, no. 2, pp. 461–474, Feb. 2018.

[32] Y. Im *et al.*, "AMUSE: Empowering Users for Cost-Aware Offloading with Throughput-Delay Tradeoffs," *IEEE Transactions on Mobile Computing*, vol. 15, no. 5, pp. 1062–1076, May 2016.

[33] F. Mehmeti and T. Spyropoulos, "Performance Analysis of "On-the-Spot" Mobile Data Offloading," *in Proc. IEEE Global Communications Conference*, pp. 1577–1583, Atlanta, USA, 2013.

[34] K. Moskvitch, "Tactile Internet: 5G and Cloud on Steroids," *IET Engineering and Technology Magazine*, vol. 10, no. 4, pp. 48–53, May 2015.

[35] M. Simsek, A. Aijaz, M. Dohler, J. Sachs, and G. Fettweis, "5G-Enabled Tactile Internet," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 3, pp. 460–473, Mar. 2016.

[36] M. Dohler, T. Mahmoodi, M. A. Lema, M. Condoluci, F. Sardis, K. Antonakoglou, and A. Aghvami, "Internet of Skills: Where Robotics Meets AI, 5G, and the Tactile Internet," *in Proc., European Conference on Networks and Communications (EuCNC)*, pp. 1–5, Oulu, Finland, June 2017.

[37] A. Aijaz, M. Dohler, H. Aghvami, V. Friderikos, and M. Frodigh, "Realizing the Tactile Internet: Haptic Communications over Next Generation 5G Cellular Networks," *IEEE Wireless Communications Magazine*, vol. 24, no. 2, pp. 82–89, Apr. 2017.

[38] D. Szabó, A. Gulyás, F. Fitzek, and D. Lucani, "Towards the Tactile Internet: Decreasing Communication Latency with Network Coding and Software Defined Networking," *in Proc., IEEE European Wireless Conference*, pp. 428–433, Budapest, Hungary, May 2015.

[39] E. Wong, M. Dias, and L. Ruan, "Predictive Resource Allocation for Tactile Internet Capable Passive Optical LANs," *IEEE/OSA Journal of Lightwave Technology*, vol. 35, no. 13, pp. 1–13, July 2017.

[40] A. Aijaz, "Towards 5G-enabled Tactile Internet: Radio Resource Allocation for Haptic Communications," *in Proc., IEEE Wireless Communications and Networking Conference*, pp. 1–6, Doha, Qatar, Apr. 2016.

[41] C. She, C. Yang, and T. Quek, "Uplink Transmission Design With Massive Machine Type Devices in Tactile Internet," *in Proc., IEEE Global Communications Conference (GLOBECOM) Workshops*, pp. 1–6, Washington, USA, Dec. 2016.

[42] C. She and C. Yang, "Energy Efficient Design for Tactile Internet," *in Proc., IEEE/CIC International Conference on Communications in China (ICCC)*, pp. 1–6, Chengdu, China, Jul. 2016.

[43] C. She, C. Yang, and T. Quek, "Cross-Layer Transmission Design for Tactile Internet," *in Proc., IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, Washington, USA, Dec. 2016.

[44] Y. Feng, C. Jayasundara, A. Nirmalathas, and E. Wong, "Hybrid Coordination Function Controlled Channel Access for Latency-Sensitive Tactile Applications," *in Proc., IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, Marina Bay Sands, Singapore, Dec. 2017.

[45] Y. Feng, C. Jayasundara, A. Nirmalathas, and E. Wong, "IEEE 802.11 HCCA for Tactile Applications," *in Proc., IEEE International Telecommunication Networks and Applications Conference (ITNAC)*, pp. 1–3, Melbourne, Australia, Nov. 2017.

[46] M. Maier, "FiWi Access Networks: Future Research Challenges and Moonshot Perspectives (Invited Paper)," *in Proc., IEEE International Conference on Communications (ICC), Workshop on Fiber-Wireless Integrated Technologies, Systems and Networks*, pp. 371–375, Sydney, Australia, June 2014.

[47] A. Ebrahimzadeh, M. Chowdhury, and M. Maier, "The Tactile Internet over 5G FiWi Architectures," *Optical and Wireless Convergence for 5G Networks and Beyond, Wiley-IEEE Press*, pp. 1–31, Abdelgader M Abdalla (Editor), Jun. 2018, in print.

[48] F. Aurzada, M. Lévesque, M. Maier, and M. Reisslein, "FiWi Access Networks Based on Next-Generation PON and Gigabit-Class WLAN Technologies: A Capacity and Delay Analysis," *IEEE/ACM Transactions on Networking*, vol. 22, no. 4, pp. 1176–1189, Aug. 2014.

[49] M. Maier and B. P. Rimal, "The Audacity of Fiber-Wireless (FiWi) Networks: Revisited for Clouds and Cloudlets," *China Communications, Feature Topic on Optical Interconnection Networks for Cloud Data Centers*, vol. 12, no. 8, pp. 33–45, Aug. 2015.

[50] China Mobile Research Institute, "C-RAN: The Road Towards Green RAN," *White Paper*, Oct. 2011.

[51] Nokia Networks, "Intelligent Base Stations," *White Paper*, Feb. 2013.

[52] B. P. Rimal, D. P. Van, and M. Maier, "Mobile Edge Computing Empowered Fiber-Wireless Access Networks in the 5G Era," *IEEE Communications Magazine*, vol. 55, no. 2, pp. 192–200, Feb. 2017.

[53] D. C. Engelbart, "Augmenting Human Intellect: A Conceptual Framework," *Stanford Research Institute Summary Report AFOSR-3233*, Oct., 1962.

[54] J. C. R. Licklider, "Man-Computer Symbiosis," *IRE Transactions on Human Factors in Electronics*, vol. HFE-1, Mar., 1960.

[55] J. Bradshaw, V. Dignum, C. Jonker, and M. Sierhuis, "Human-Agent-Robot Teamwork," *IEEE Intelligent Systems*, vol. 27, no. 2, pp. 8–13, Apr. 2012.

[56] J. Bradshaw, P. Feltovich, M. Johnson, and L. Bunch, "Coordination in Human-Agent-Robot Teamwork," *in Proc., International Symposium on Collaborative Technologies and Systems (CTS)*, pp. 467–476, Ervine, USA, May 2008.

[57] A. Khamis, A. Hussein, and A. Elmogy, "Multi-Robot Task Allocation: A Review of the State-of-the-Art," *Cooperative Robots and Sensor Networks, Springer*, vol. 604, pp. 31–51, May 2015.

[58] J. Liu, G. Shou, Y. Liu, Y. Hu, and Z. Guo, "Performance Evaluation of Integrated Multi-access Edge Computing and Fiber-Wireless Access Networks," *IEEE Access*, pp. 1–11, IEEE Early Access, Jun. 2018.

[59] T. Langford, Q. Gu, A. Rivera-Longoria, and M. Guirguis, "Collaborative Computing On-Demand: Harnessing Mobile Devices in Executing On-the-Fly Jobs," *in Proc., IEEE International Conference on Mobile Ad hoc and Sensor Systems (MASS)*, pp. 342–350, Hangzhou, China, Oct. 2013.

[60] H. Yu, M. Cheung, G. Iosifidis, L. Gao, L. Tassiulas, and J. Huang, "Mobile Data Offloading for Green Wireless Networks," *IEEE Wireless Communications*, vol. 24, no. 4, pp. 31–37, Aug. 2017.

[61] H. Ko, J. Lee, and S. Pack, "Performance Optimization of Delayed WiFi Offloading in Heterogeneous Networks," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 10, pp. 9436–9447, Oct. 2017.

[62] K. Lee, J. Lee, Y. Yi, I. Rhee, and S. Chong, "Mobile Data Offloading: How Much Can WiFi Deliver?," *IEEE/ACM Transactions on Networking*, vol. 21, no. 2, pp. 536–550, Apr. 2013.

[63] F. Mehmeti and T. Spyropoulos, "Performance Modeling, Analysis, and Optimization of Delayed Mobile Data Offloading for Mobile Users," *IEEE/ACM Transactions on Networking*, vol. 25, no. 1, pp. 550–564, Feb. 2017.

[64] F. Rebecchi, M. Amorin, V. Conan, A. Passarella, R. Bruno, and M. Conti, "Data Offloading Techniques in Cellular Networks: A Survey," *IEEE Communications Surveys and Tutorials*, vol. 17, no. 2, pp. 580–603, May. 2015.

[65] S.-I. Sou and Y.-T. Peng, "Performance Modeling for Multipath Mobile Data Offloading in Cellular/Wi-Fi Networks," *IEEE Transactions on Communications*, vol. 65, no. 9, pp. 3863–3875, Sep. 2017.

[66] 3rd Generation Partnership Project (3GPP), "Access Network Discovery and Selection Function (ANDSF), Management Object (MO)," *V 13.3.0, TS 24.312*, pp. 1–55, Jun. 2016.

[67] 3rd Generation Partnership Project (3GPP), "Technical Specification Group Services and System Aspects, Local IP Access and Selected IP Traffic Offload (LIPA-SIPTO)," *Rel. 10, V10.0.1, TR 23.829*, pp. 1–43, Oct. 2011.

[68] C. B. Sankaran, "Data Offloading Techniques in 3GPP Rel-10 Networks: A Tutorial," *IEEE Communications Magazine*, vol. 50, no. 6, pp. 46–53, June 2012.

[69] R. Wolski, C. Krintz, and D. Nurmi, "Using Bandwidth Data to Make Computation Offloading Decisions," *in Proc., IEEE International Symposium on Parallel and Distributed Processing*, pp. 1–8, Miami, USA, Apr. 2008.

[70] K. Kumar, Y. Lu, and B. Bhargava, "A Survey of Computation Offloading for Mobile Systems," *Mobile Networks and Applications Journal*, vol. 18, no. 1, pp. 129–140, Feb. 2013.

[71] E. Cuervo, D. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "MAUI: Making Smartphones Last Longer With Code Offload," *in Proc., ACM Conference on Mobile Systems, Applications, and Services (MobiSys)*, pp. 49–62, San Francisco, USA, Jun. 2010.

[72] B. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, "CloneCloud: Elastic Execution Between Mobile Device and Cloud," *in Proc., ACM European Conference on Computer Systems (EuroSys)*, pp. 301–314, Salzburg, Austria, Apr. 2011.

[73] S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang, "ThinkAir: Dynamic Resource Allocation and Parallel Execution in the Cloud for Mobile Code Offloading," *in Proc., IEEE International Conference on Computer Communications (INFOCOM)*, pp. 945–953, Orlando, USA, Mar. 2012.

[74] K. Kumar and Y. Lu, "Cloud Computing for Mobile Users: Can Offloading Computation Save Energy?," *IEEE Computer*, vol. 43, no. 4, pp. 51–56, Apr. 2010.

[75] I. Osunmakinde and V. Ramharuk, "Development of a Survivable Cloud Multi-Robot Framework for Heterogeneous Environments," *International Journal of Advanced Robotic Systems*, vol. 11, no. 10, pp. 1–22, Oct. 2014.

[76] C. Magurawalage, K. Yang, L. Hu, and J. Zhang, "Energy Efficient and Network-Aware Offloading Algorithm for Mobile Cloud Computing," *Computer Networks*, vol. 74, no. B, pp. 22–33, Dec. 2014.

[77] J. Liu, Y. Mao, J. Zhang, and K. Letaief, "Delay-Optimal Computation Task Scheduling for Mobile-Edge Computing Systems," *in Proc., IEEE International Symposium on Information Theory (ISIT)*, pp. 1451–1455, Barcelona, Spain, July 2016.

[78] W. Zhang, Y. Wen, J. Wu, and H. Li, "Toward a Unified Elastic Computing Platform for Smartphones With Cloud Support," *IEEE Network*, vol. 27, no. 5, pp. 34–40, Oct. 2013.

[79] M. Ra, A. Sheth, L. Mummert, P. Pillai, D. Wetherall, and R. Govindan, "Odessa: Enabling Interactive Perception Applications on Mobile Devices," *in Proc., ACM Conference on Mobile Systems, Applications, and Services (MobiSys)*, pp. 43–56, Washington, USA, June 2011.

[80] Nokia Siemens Networks, "Liquid Applications System Description," pp. 1–40, Jul. 2014.

[81] The European Telecommunications Standards Institute (ETSI) Industry Specification Group (ISG), "Mobile-Edge Computing - Introductory Technical White Paper," pp. 1–36, Sep. 2014.

[82] Cisco Systems Inc., *URL: https://developer.cisco.com/site/iox/*, Accessed on Oct. 20, 2016.

[83] L. Tang and S. He, "Multi-User Computation Offloading in Mobile Edge Computing: A Behavioral Perspective," *IEEE Network*, vol. 32, no. 1, pp. 48–53, Jan. 2018.

[84] S. Ko, K. Huang, S. Kim, and H. Chae, "Energy Efficient Mobile Computation Offloading via Online Prefetching," *in Proc., IEEE International Conference on Communications (ICC), Wireless Communications Symposium*, pp. 1–6, Paris, France, May 2017.

[85] X. Wang and M. Chen, "PreFeed: Cloud-Based Content Prefetching of Feed Subscriptions for Mobile Users," *IEEE Systems Journal*, vol. 8, no. 1, pp. 202–207, Mar. 2014.

[86] S. Ko, K. Huang, S. Kim, and H. Chae, "Live Prefetching for Mobile Computation Offloading," *IEEE Transactions on Wireless Communications*, vol. 16, no. 5, pp. 3057–3071, May 2017.

[87] M. Elbamby, M. Bennis, and W. Saad, "Proactive Edge Computing in Latency-Constrained Fog Networks," *in Proc., European Conference on Networks and Communications (EuCNC)*, pp. 1–6, Oulu, Finland, June 2017.

[88] B. Chowdhury, B. Biswal, and D. Mishra, "Development of Optimal Strategies for Task Assignment in Multirobot Systems," *in Proc., IEEE IACC*, pp. 1130–1135, Patiala, India, Mar. 2009.

[89] R. Mendonça, N. Nedjah, and L. Mourelle, "Efficient Distributed Algorithm of Dynamic Task Assignment for Swarm Robotics," *Computational Science and Its Applications, LNCS, Springer*, vol. 7971, pp. 500–510, June 2013.

[90] A. Jevtic, A. Gutierrez, D. Andina, and M. Jamshidi, "Distributed Bees Algorithm for Task Allocation in Swarm of Robots," *IEEE System Journal*, vol. 6, no. 2, pp. 296–304, June 2012.

[91] N. Kalra and A. Martinoli, "A Comparative Study of Market-Based and Threshold-Based Task Allocation," *Distributed Autonomous Robotic Systems, Book Chapter, Springer*, pp. 91–101, 2006.

[92] S. Trigui, A. Koubaa, O. Cheikhrouhou, H. Youssef, H. Bennaceur, M. Sriti, and Y. Javed, "A Distributed Market-Based Algorithm for Multi-Robot Assignment Problem," *Procedia Computer Science, Elsevier*, vol. 32, pp. 1108–1114, June 2014.

[93] D. P. Van, B. P. Rimal, S. Andreev, T. Tirronen, and M. Maier, "Machine-to-Machine Communications over FiWi Enhanced LTE Networks: A Power-Saving Framework and End-to-End Performance," *IEEE/OSA Journal of Lightwave Technology*, vol. 34, no. 4, pp. 1062–1071, Feb. 2016.

[94] B. P. Rimal, D. P. Van, and M. Maier, "Mobile-Edge Computing vs. Centralized Cloud Computing in Fiber-Wireless Access Networks," *in Proc., IEEE INFOCOM, Workshop on 5G and Beyond - Enabling Technologies and Applications*, pp. 1–6, San Francisco, CA, USA, Apr. 2016.

[95] F. Wang, G. Han, J. Jiang, and H. Qiu, "A Distributed Task Allocation Strategy for Collaborative Applications in Cluster-Based Wireless Sensor Networks," *International Journal of Distributed Sensor Networks*, vol. 2014, pp. 1–16, 2014.

[96] J. Yang, H. Zhang, Y. Ling, C. Pan, and W. Sun, "Task Allocation for Wireless Sensor Network Using Modified Binary Particle Swarm Optimization," *IEEE Sensors Journal*, vol. 14, no. 3, pp. 882–892, Mar. 2014.

[97] W. Zhang, R. Mallik, and K. B. Letaief, "Cooperative Spectrum Sensing Optimization in Cognitive Radio Networks," *in Proc., IEEE ICC*, pp. 3411–3415, May 2008.

[98] Y. Liang, Y. Zeng, E. Peh, and A. Hoang, "Sensing-Throughput Tradeoff for Cognitive Radio Networks," *IEEE Transactions on Wireless Communications*, vol. 7, no. 4, pp. 1326–1337, April 2008.

[99] A. Viguria *et al.*, "S+T: An Algorithm for Distributed Multirobot Task Allocation Based on Services for Improving Robot Cooperation," *in Proc., IEEE ICRA*, pp. 3163–3168, California, May 2008.

[100] B. Bellalta *et al.*, "On the Performance of Packet Aggregation in IEEE 802.11ac MU-MIMO WLANs," *IEEE Communications Letters*, vol. 16, no. 10, pp. 1588–1591, Oct. 2012.

[101] S. C. Jha, M. M. Rashid, and V. K. Bhargava, "Design of OMC-MAC: An Opportunistic Multi-channel MAC With QoS Provisioning for Distributed Cognitive Radio Networks," *IEEE Transactions on Wireless Communications*, vol. 10, no. 10, pp. 3414–3425, Oct. 2011.

[102] S. Deng, L. Huang, J. Taheri, and A. Zomaya, "Computation Offloading for Service Workflow in Mobile Cloud Computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 2, pp. 3317–3328, Dec. 2015.

[103] M. Hasan, E. Hossain, and D. Niyato, "Random Access for Machine-to-Machine Communication in LTE-Advanced Networks: Issues and Approaches," *IEEE Communications Magazine*, vol. 51, no. 6, pp. 86–93, June 2013.

[104] J. Rawadi, H. Artail, and H. Safa, "Providing Local Cloud Services to Mobile Devices With Inter-Cloudlet Communication," *in Proc., IEEE MELECON*, pp. 134–138, Lebanon, Apr. 2014.

[105] B. P. Rimal, D. P. Van, and M. Maier, "Mobile-Edge Computing Versus Centralized Cloud Computing Over a Converged FiWi Access Network," *IEEE Transactions on Network and Service Management*, vol. 14, no. 3, pp. 498–513, Sep. 2017.

[106] M. Chowdhury and M. Maier, "Community- and Latency-Aware Multi-Task Scheduling for HART Collaboration in FiWi Enhanced Networks," *IEEE Transactions on Cloud Computing*, November 2018, Submitted.

[107] Q. Xia *et al.*, "Online Algorithms for Location Aware Task Offloading in Two-Tiered Mobile Cloud Environments," *in Proc., IEEE/ACM 7th International Conference on Utility and Cloud Computing (UCC)*, pp. 109–116, London, Uk, Dec. 2014.

[108] L. Zhang, Z. Li, C. Wu, and M. Chen, "Online Algorithms for Uploading Deferrable Big Data to the Cloud," *in Proc., IEEE INFOCOM*, pp. 2022–30, Toronto, Apr. 2014.

[109] M. NoroozOliaee *et al.*, "Online Multi-Resource Scheduling for Minimum Task Completion Time in Cloud Servers," *in Proc., IEEE INFOCOM WKSHPS*, pp. 375–379, Toronto, April/May 2014.

[110] M. Dabbagh *et al.*, "Exploiting Task Elasticity and Price Heterogeneity for Maximizing Cloud Computing Profits," *IEEE Transactions on Emerging Topics in Computing*, vol. 6, no. 1, pp. 85–95, March 2018.

[111] X. Zhu, L. Yang, H. Chen, J. Wang, S. Yin, and X. Liu, "Real-Time Tasks Oriented Energy-Aware Scheduling in Virtualized Clouds," *IEEE Transactions on Cloud Computing*, vol. 2, no. 2, pp. 168–180, Apr.-Jun. 2014.

[112] L. Shi *et al.*, "Energy-Aware Scheduling of Embarrassingly Parallel Jobs and Resource Allocation in Cloud," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 6, pp. 1607–1620, June 2017.

[113] L. Yang, J. Cao, G. Liang, and X. Han, "Cost Aware Service Placement and Load Dispatching in Mobile Cloud Systems," *IEEE Transactions on Computers*, vol. 65, no. 5, pp. 1440–1452, May 2016.

[114] S. Ou *et al.*, "Performance Analysis of Fault-Tolerant Offloading Systems for Pervasive Services in Mobile Wireless Environments," *in Proc., IEEE ICC*, pp. 1856–1860, Beijing, China, May 2008.

[115] M. Chowdhury and M. Maier, "User Preference Aware Task Coordination and Proactive Bandwidth Allocation in a FiWi Based Human-Agent-Robot Teamwork Ecosystem," *IEEE Transactions on Network and Service Management*, Oct. 2018, in revision.

[116] W. Zhang, Y. Wen, and D. O. Wu, "Opportunistic Task Scheduling over Co-Located Clouds in Mobile Environment," *IEEE Transactions on Services Computing*, IEEE Early Access, 2017.

[117] P. Shu *et al.*, "eTime: Energy-efficient Transmission Between Cloud and Mobile Devices," *in Proc. IEEE INFOCOM*, pp. 195–199, Turin, Italy, 2013.

[118] H. Deng and I. Hou, "Online Scheduling for Delayed Mobile Offloading," *in Proc. IEEE INFOCOM*, pp. 1867–75, Hongkong, May 2015.

[119] S. Yang, C. Yeo, and B. Lee, "MaxCD: Efficient Multi-Flow Scheduling and Cooperative Downloading for Improved Highway Drive-Thru Internet Systems," *Computer Networks*, vol. 57, no. 8, pp. 1805–20, Jun. 2013.

[120] U. Drolia *et al.*, "Cachier: Edge-Caching for Recognition Applications," *in Proc., IEEE ICDCS*, pp. 276–286, Atlanta, Jun. 2017.

[121] Z. Luo *et al.*, "Energy-Efficient Caching for Mobile Edge Computing in 5G Networks," *Applied Sciences*, vol. 7, no. 6, pp. 1–13, May 2017.

[122] B. Kehoe, S. Patil, and K. Goldberg, "A Survey of Research on Cloud Robotics and Automation," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 2, pp. 398–409, Apr. 2015.

[123] J. Wan, Q. Jiang, and G. Pan, "Research on Performance Influence Factors of O2O Website With Interpretative Structure Model," *in Proc., International Conference on e-Business (WHICEB)*, pp. 31–41, China, May 2017.

[124] G. Chen and J. Chen, "Applying Virtual Reality to Remote Control of Mobile Robot," *in Proc., International Conference on Intelligent Technologies and Engineering Systems (ICITES)*, pp. 383–390, Kaohsiung, Taiwan, Apr. 2014.

[125] S. Chi, Y. Yuen, G. Aoyuneyong, and E. Johnson, "Augmented Reality: An Overview and Five Directions for AR in Education," *Journal of Educational Technology Development and Exchange (JETDE)*, vol. 4, no. 1, pp. 119–140, Oct. 2011.

[126] P. Parvinen, J. Hamari, and E. Poyry, "Introduction to the Minitrack on Mixed, Augmented and Virtual Reality," *in Proc., International Conference on System Sciences*, pp. 1395–1396, Hawaii, USA, Jan. 2018.

[127] M. Bekele, R. Pierdicca, E. Frontoni, E. Malinverni, and J. Gain, "A Survey of Augmented, Virtual, and Mixed Reality for Cultural Heritage," *Journal on Computing and Cultural Heritage (JOCCH)*, vol. 11, no. 2, pp. 1–36, Mar. 2018.