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

NOUVELLE APPROCHE DE SIGNALISATION POUR LES COMMUNICATIONS EN TEMPS-RÉEL

Par

Samah Ouhmama

Mémoire présenté pour l'obtention du grade de
Maître es Sciences, M.Sc.
en télécommunications

Jury d'évaluation

Examinateur externe	Prof. Roch Glitho Concordia University
Examinateur interne	Prof. Amar Mitiche INRS-EMT
Directeur de recherche	Prof. Jean-Charles Grégoire INRS-EMT

Remerciements

Au terme de ce travail, je tiens à exprimer ma profonde gratitude à toutes les personnes qui ont contribuées, de près ou de loin, à l'accomplissement de ce travail.

Je tiens à coeur à adresser mes sincères remerciements à l'égard de mon directeur de recherche, M. Jean-Charles Grégoire. Ses directives et conseils précieux, sa patience, sa disponibilité, son appui et son suivi ont rendu possible la réalisation de ce projet de maîtrise. Ses questions pertinentes ont toujours été un déclencheur de plus de curiosité et d'enrichissement.

Mes remerciements s'adressent aussi à M. Amar Mitiche et à M. Roch Glitho pour avoir accepté de faire partie du jury d'évaluation.

Un spécial hommage est rendu à tous les professeurs de l'INRS-EMT et au personnel du centre. Qu'ils trouvent ici l'expression de ma sincère reconnaissance.

Résumé

Les communications en temps réel suscitent toujours davantage l'intérêt des chercheurs. Cette technologie permet à des clients de communiquer et d'échanger interactivement des informations. Dans une perspective Web, l'environnement WebRTC permet des communications en temps réel via les navigateurs. Pour pouvoir entrer en communication, les clients doivent échanger quelques informations, par un processus de signalisation. Il permet de créer le lien de communication ainsi que de coordonner les différents composants nécessaires pour compléter une communication. Le protocole de signalisation le plus utilisé est SIP/SDP. Le format de description des paramètres de la session SDP présente plusieurs faiblesses et inconvénients. Dans ce mémoire, nous proposons une approche de signalisation générique. Nous décomposons, de manière modulaire et incrémentale, l'information contenue dans le format SDP et l'adaptons à des situations qui ne sont pas nécessairement « de bout en bout ». L'information sera concise et mieux structurée tout en respectant les besoins d'une grande variété d'applications.

Mots-clés Signalisation, WebRTC, SIP/SDP, communications pair à pair, client-serveur.

Étudiant	Directeur de recherche

Table des matières

Liste des figures

Liste des tableaux

Glossaire

3GPP	3rd Generation Partnership Project
B2BUA	Back to Back User Agent
Client	Machine qui envoie des demandes à la centrale.
Connexion	Liaison entre deux ou plusieurs clients. Une session doit être ouverte pour pouvoir établir une connexion.
HTML5	HyperText Markup Language 5
HTTP	HyperText Transfer Protocol
ICE	Interactive Connectivity Establishment
IETF	Internet Engineering Task Force
IMS	IP Multimedia Subsystem
IP	Internet Protocol
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
JSON	JavaScript Object Notation
LAN	Local Area Network
NAT	Network Address Translation
NSIS	Next Steps in Signaling
Opération	Requête parvenant du client et destinée à être exécutée au niveau du serveur.
OTT	Over-The-Top
P2P	Pair à pair
Présence	État de l'usager qui reflète la possibilité de le joindre ou non.
REST	Representational State Transfer
RTC	Real-Time Communication
SDP	Session Description Protocol
Session	Période de temps pendant laquelle le client est en interaction avec un autre programme qualifié de serveur. Durant la session, le client peut réaliser une ou plusieurs opérations.
SIP	Session Initiation Protocol

Terminal	Machine à partir de laquelle le client est connecté à la centrale.
VoIP	Voix sur IP
VVoIP	Voix et vidéo sur IP
W3C	World Wide Web Consortium
WebRTC	Web Real-Time Communication
XML	Extensible Markup Language

Chapitre 1

Introduction

1.1 Contexte de travail

Les communications en temps réel (Real Time Communication, RTC) sont un mode de communication où les usagers peuvent échanger interactivement des informations, de manière instantanée ou avec un délai négligeable. Au fil du temps, ce type de communication a connu une grande évolution et nous témoignons de nos jours d'une large diversité de communications via Internet. Les usagers peuvent clavarder en temps réel en utilisant la messagerie instantanée. Les services voix et vidéo sur IP (Voice and Video over IP, VVoIP) quand à eux nécessitent d'installer des applications (le cas de Skype par exemple) ou de passer par des plugins (exemple: Flash, Silverlight). Une nouvelle technologie, appelée WebRTC (Web Real-Time Communication) voit donc le jour permettant aux usagers de communiquer directement via navigateurs, sans installer de composants supplémentaires. En effet, le WebRTC fait objet de plusieurs travaux de recherche ces derniers temps (1), (2) et (3). Les spécifications de cet environnement ont été rédigées par le World Wide Web Consortium (W3C). WebRTC prend en charge les applications d'appels téléphoniques, le transfert de fichiers ainsi que les applications de chat audio/vidéo.

Dans un environnement WebRTC, les communications interactives sont intégrées dans une plateforme qui offre un ou plusieurs services de base, autres que la communication. Nous disons que la communication est « embarquée » (4) ou « imbriquée ». WebRTC fournit un moyen d'intégrer les modes de communication (audio, vidéo et données) au navigateur ainsi qu'un mécanisme pour

pouvoir les activer. Les communications interactives présentent un enrichissement des services déjà offerts par la plateforme existante et bénéficient de son support pour plusieurs fonctionnalités telles que l'identification du client et la vérification de sa présence.

Avant tout échange de média entre usagers, une coordination est nécessaire, qui opère par un processus de signalisation. La signalisation concerne la création du lien de communication ainsi que l'échange des messages de contrôle. Pour le cas d'un appel voix sur IP (Voice over IP, VoIP) (5) par exemple, le protocole de signalisation le plus utilisé est le protocole d'initiation de la session (Session Initiation Protocol, SIP)(6). Le protocole de description de la session (Session Description Protocol, SDP)(7) est utilisé en conjonction avec SIP pour décrire et présenter les différents types de média supportés et les propriétés d'un point de communication. Pour les environnements WebRTC, le processus de signalisation n'a pas été spécifié par W3C. Les développeurs peuvent choisir le protocole de signalisation qu'ils estiment efficace pour leur implémentation ou utiliser un mécanisme propriétaire.

Des travaux dont le but est d'améliorer la signalisation pour les communications en temps réel existent. Citons l'exemple du groupe de travail de l'IETF, Next Steps in Signaling (NSIS) (8), qui a été formé en 2001. L'objectif principal de ce projet était de définir un nouveau protocole de signalisation pour les communications Internet. Les efforts du groupe NSIS ont produit un framework générique (9) mais qui n'a jamais abouti dans la pratique. Dans les faits, SIP s'est imposé comme solution unique.

1.2 Problématique

SDP est utilisé pour décrire les paramètres d'initiation d'une session multimédia. Alors qu'il est désigné comme un format descriptif, son utilisation dans un environnement interactif présente plusieurs inconvénients et reste trop attachée au modèle offre-réponse (10). Nous développerons ce modèle dans le chapitre qui suit. Une analyse (11) approfondie du format SDP révèle plusieurs limites et problématiques:

— il ne possède pas de structure;

— il est difficile à comprendre;

— ses attributs peuvent prendre plusieurs interprétations;

— il est redondant;
— il est difficile à modifier.

1.3 Objectif et contribution du travail

L'objectif principal est de passer d'une signalisation complexe reposant sur SDP à une signalisation générique, en permettant et en favorisant les communications pair à pair (peer-to-peer, P2P). Nous proposerons un autre format, où l'information sera concise et mieux structurée, tout en respectant les besoins d'une grande diversité d'applications.

Pour ce faire, nous exploitons de la diversité des informations contenues dans un document SDP, à savoir: l'adresse IP, la version IP, les caractéristiques du terminal, les médias et les codecs pris en charge par le navigateur, le choix du média, le choix du codec, le port où le média va être reçu, la bande passante maximale ainsi que d'autres informations de connectivité et de sécurité. Nous définissons donc trois grandes classes d'informations:

— *Informations statiques*: ce sont des informations qui ne changent pas le long d'une session.
— *Paramètres de communication*: elles incluent des informations relatives à une communication entre deux clients.
— *Modification de communication*: informations sur les changements pendant la communication.

  Les modifications pendant la communication peuvent être des variations actives, où le client est à l'origine des changements, ou des variations automatiques, initiées par le réseau.

  — variations actives:
    — ajouter/enlever la vidéo;
    — tourner le terminal (changement d'orientation);
    — activer la touche de discrétion;
    — ajouter des canaux audio.
  — variations automatiques:
    — adaptation de la qualité du codec;
    — adaptation aux conditions du réseau.

Notre sujet se place dans la catégorie des travaux visant la performance du mécanisme de signalisation des communications multimédia. Nous avons établi le contexte général de notre travail et avons survolé les limites du format SDP. Nous procéderons à une analyse et une critique plus détaillées de ce protocole au cours du chapitre qui suit.

1.4 Structure du document

Le présent rapport est divisé en six chapitres. Dans le chapitre qui suit, nous allons expliquer quelques notions de base nécessaires à la compréhension de la problématique. Nous allons établir un ensemble de spécifications du modèle de signalisation que nous proposons. Le troisième chapitre portera sur une étude de cas où nous décrivons une communication entre deux clients, selon un scénario choisi. Dans le quatrième chapitre, nous illustrons le concept décrit en présentant les flux entre les entités ainsi que quelques messages de signalisation. L'étude du scénario nous permettra d'élaborer une généralisation sous forme de discussion qu'on traitera au cours du cinquième chapitre. Enfin, nous simulons le scénario décrit afin de prouver la performance de la signalisation proposée, ce qui fera objet du dernier chapitre.

Chapitre 2

Notions de base et spécification des besoins

Dans ce chapitre, nous posons le cadre de notre travail, nous spécifions les besoins et proposons un modèle global de communication entre clients. Nous verrons quelques notions, dans leur cadre général, et spécifiquement pour des communications directes en temps réel WebRTC.

2.1 Signalisation

En téléphonie, la signalisation implique la gestion des ressources nécessaires pour l'établissement d'un canal de communication ainsi que le contrôle d'appel. Pour que deux clients puissent communiquer, ils doivent se mettre d'accord sur un canal commun. Si le client A souhaite communiquer avec le client B, alors ce dernier doit fournir une « adresse » à laquelle le client A peut le joindre (cette adresse correspond au numéro de téléphone pour le cas de la téléphonie traditionnelle).

Par analogie, avant de pouvoir communiquer directement, de pair à pair, les clients WebRTC doivent obligatoirement échanger quelques informations qui aideront à établir la communication. Trois principales classes d'information sont échangées, à savoir:

— Messages de contrôle de session: initialiser et fermer un lien de communication et rapporter les erreurs;

— Configuration réseau: quelle est l'adresse IP et sur quel port le client est-il en écoute?

— Compatibilité des médias: quels codecs sont utilisés?

Plusieurs protocoles de signalisation existent et le plus utilisé dans l'Internet est le protocole SIP. Ce dernier utilise SDP pour décrire les détails de l'appel, sous un modèle d'offre-réponse. Nous présenterons dans ce qui suit le format SDP et verrons ses limites et inconvénients.

2.2 SDP

Spécifié par l'IETF, SDP (7) est un protocole de description, conçu à l'origine pour être utilisé par des applications de diffusion. Son usage s'est généralisé avec le protocole de signalisation SIP. SDP permet de décrire les propriétés des sessions multimédia, les différents types de média supportés et les propriétés d'un point de communication: son adresse IP, le port d'écoute, les codecs à utiliser ainsi que les différents candidats-relais pour établir la communication.

Les environnements WebRTC utilisent une architecture JSEP (Javascript Session Establishment Protocol)(12) basée sur le modèle offre-réponse afin d'établir des sessions multimédia. Nous détaillerons ce modèle dans ce qui suit.

2.2.1 Modèle offre-réponse

Dans ce modèle, le client initiateur génère une description de la session multimédia qu'il désire partager avec le deuxième client. Cette description est appelée une offre. Le client visé par cette offre peut l'accepter ou la refuser. Si l'offre est acceptée, le client récepteur génère une description de la session (selon l'offre reçue). Nous présentons en listings 2.1 et 2.2 (tirés de (13)) deux exemples de messages SDP où le premier est une offre et le deuxième en est la réponse. Dans cet exemple, Alice et Bob présentent l'ensemble des codecs audio et vidéo supportés, à savoir pour l'audio les codecs, PCMU, PCMA et iLBC. En réponse à cette offre, Bob accepte un seul codec audio (PCMU).

Listing 2.1 – Exemple d'une offre SDP

```
v=0
o=alice 2890844526 2890844526 IN IP4 host.atlanta.example.com
s=
c=IN IP4 host.atlanta.example.com
t=0 0
m=audio 49170 RTP/AVP 0 8 97
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:97 iLBC/8000
m=video 51372 RTP/AVP 31 32
a=rtpmap:31 H261/90000
a=rtpmap:32 MPV/90000
```

Listing 2.2 – Exemple d'une réponse SDP

```
v=0
o=bob 2808844564 2808844564 IN IP4 host.biloxi.example.com
s=
c=IN IP4 host.biloxi.example.com
t=0 0
m=audio 49174 RTP/AVP 0
a=rtpmap:0 PCMU/8000
m=video 49170 RTP/AVP 32
a=rtpmap:32 MPV/90000
```

Notons que le modèle offre-réponse s'étend à la négociation d'autres paramètres tels que la sécurité et la qualité de service (quality of service, QoS). En effet, les postes peuvent négocier le mécanisme de QoS à utiliser pour un média donné. Nous présentons ci-dessous un exemple d'échange offre-réponse pour la négociation de QoS. Cet exemple (tiré de (14)) utilise les attributs « qos-mech-send » et « qos-mech-recv ».

L'initiateur génère une offre décrivant les mécanismes de QoS qu'il pourra utiliser. Souhaitant utiliser le protocole RSVP, il le cite donc en premier:

```
m=audio 50000 RTP/AVP 0
a=qos-mech-send: rsvp nsis
a=qos-mech-recv: rsvp nsis
```

En réponse à cette offre, le client avise l'initiateur qu'il peut supporter uniquement le mécanisme NSIS:

```
m=audio 55000 RTP/AVP 0
a=qos-mech-send: nsis
a=qos-mech-recv: nsis
```

2.2.2 Analyse et critique du format SDP

Une analyse de la structure de ce protocole permet d'identifier les principales informations véhiculées:

- des lignes globales comportant principalement:
  - des informations sur le protocole: version du protocole;
  - des informations sur la session: nom de la session, son objet et la durée prévue. Aussi, le responsable de la session et la bande passante nécessaire pour la session.
- des informations sur le média: type du média utilisé (audio, vidéo) ainsi que l'offre de codecs pour chaque média;
- des informations de connectivité: les paramètres du protocole ICE pour les environnements WebRTC par exemple;
- d'autres informations sur le protocole de transport utilisé et les paramètres de sécurité.

SDP est composé de plusieurs types de lignes d'informations présentant plusieurs inconvénients:

- le format, même si présenté sous forme de texte, n'est pas très facile à comprendre;
- le format n'a pas de structure, les données sont une séquence de lignes dénotées par un attribut particulier;
- certaines informations sont redondantes et même annoncées par d'autres protocoles (la bande passante à titre d'exemple);
- l'ensemble du message SDP doit être renvoyé avec tout changement dans la session, ce qui rendra certainement le traitement du message plus long et la détection de changements plus compliquée;
- la signification de chaque ligne d'attribut peut changer selon les lignes qui les précèdent.

De plus, SDP n'a pas été conçu pour un modèle offre-réponse, mais simplement pour caractériser des sources. Tout ceci pousse à se demander si l'adoption de ce format ne diminue pas la performance du mécanisme de signalisation. En effet, l'utilisation de SDP avec les environnements WebRTC peut être abandonnée. Ceci a fait l'objet de discussions au sein de l'IETF (15).

2.3 Modèles de communication

Notons que nous dirons qu'une communication est symétrique si les deux parties A et B (émetteur et récepteur) peuvent indifféremment initier la communication. Chacune des parties a l'information nécessaire pour communiquer avec l'autre.

Nous identifions trois modèles d'appel possibles, à savoir les communications trapézoïdales, les communications triangulaires et celles asymétriques avec un serveur. Nous ne nous intéressons pas à l'étude du modèle pair-à-pair. Dans les faits, ce modèle n'existe pas dans l'Internet car trop de dispositifs bloquent les communications. Un relai est toujours présent. Bien qu'il soit plus réaliste dans un contexte d'entreprise, il n'en reste pas moins que des fonctionnalités élémentaires telles que l'inscription et l'authentification doivent être assurées par une entité intermédiaire.

2.3.1 Communication trapézoïdale

Ce type de communication est symétrique et est utilisé par SIP/SDP. La figure 2.1 illustre ce modèle. Un client doit au préalable être inscrit au niveau du serveur, communément appelé mandataire (proxy). Les deux clients peuvent être dans des domaines différents, ce qui explique l'existence de deux mandataires. Aussi, l'identification requiert des relais de haut niveau.

Le client appelant initie la demande d'appel qui sera relayée vers la destination. La communication suit le modèle offre-réponse dans le sens où la négociation des paramètres de l'appel se fait de client à client. L'appel est routé à sa destination à l'aide des serveurs 1 et 2 qui servent de relais durant la communication.

2.3.2 Communication triangulaire

Les communications sont symétriques et peuvent être relayées ou facilitées. La figure 2.2 illustre une communication triangulaire. Les deux clients A et B doivent être inscrits au niveau du serveur pour pouvoir initier ou accepter une communication.

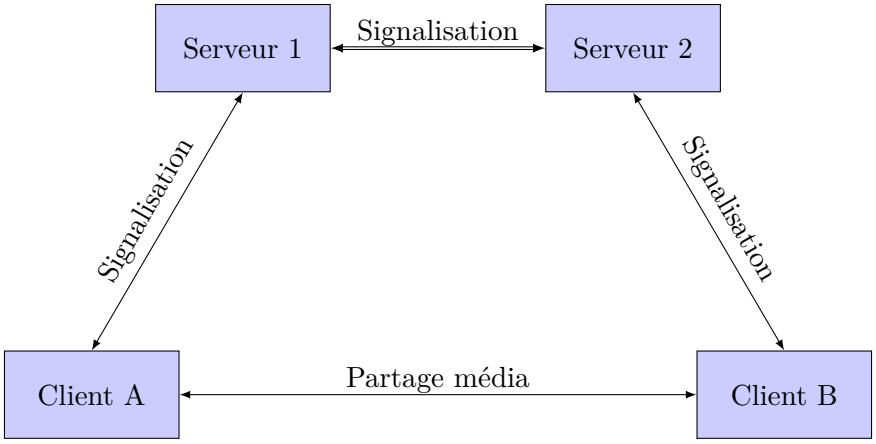


Figure 2.1 – Modèle en trapèze

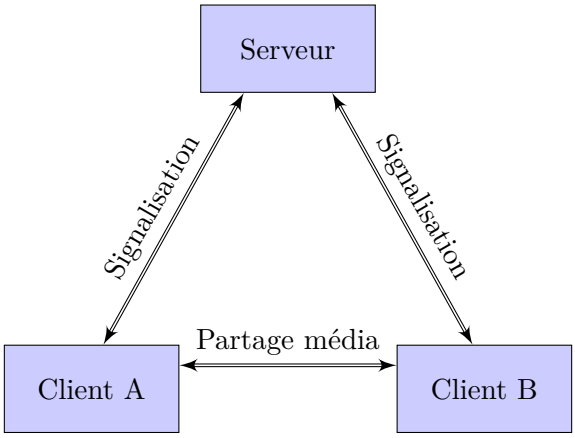


Figure 2.2 – Modèle avec serveur intermédiaire

Cas des communications relayées

C'est une simplification des communications trapézoïdales où les deux clients appartiennent au même domaine. Le serveur sert dans ce cas de relais de communication. Les paramètres d'appel sont négociés et établis de bout en bout. Tout comme le modèle précédent, ce type de communication suit un modèle offre-réponse.

Cas des communications facilitées

Ce type de communication est basé sur le modèle client-serveur. Le serveur intermédiaire unique prend en charge la négociation des paramètres de l'appel. Plusieurs simplifications peuvent être faites, ce qui va être exploré plus en détail dans notre exemple. Une étape d'initialisation est né-

cessaire où le client se connecte au serveur et envoie des informations statiques: informations sur le terminal, informations sur les médias supportés, informations de connectivité ainsi que la pile de protocoles disponible pour IPv4 et IPv6.

Pour établir une communication, le client A (initiateur) exprime son souhait de partager une session multimédia avec le client B en envoyant une demande de connexion au serveur. Ce dernier, ayant déjà les informations des deux clients, décide des paramètres de la communication. La négociation est totalement prise en charge par le serveur qui crée et contrôle les communications. Le routage d'appel disparait donc avec ce modèle, le serveur n'est pas un relai de communication mais est au coeur de la négociation des messages de contrôle d'appel.

Cette approche est utilisée par les services « over-the-top » (OTT), déployés par dessus une infrastructure réseau banalisée. Nous détaillerons ce modèle, selon un cas choisi, dans le chapitre qui suit.

2.3.3 Communication asymétrique avec serveur

Dans ce modèle, présenté en figure 2.3, l'appel est destiné à un serveur qui prend en charge la création, l'établissement et le contrôle de la communication. Nous dirons que la communication est asymétrique dans le sens où l'appelé n'a pas l'information nécessaire pour joindre l'appelant (client). Cette approche utilise un modèle client-serveur. Le client a le rôle d'initier la communication.

Ce modèle de communication est utilisé, à titre d'exemple, par les services de commerce électronique et les centres d'appel.

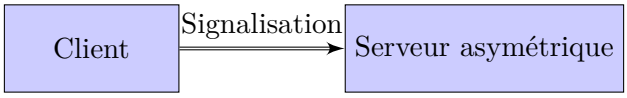


Figure 2.3 – Modèle asymétrique avec serveur

Prenons l'exemple d'un client qui se connecte au site Web d'un fournisseur de service et veut parler à un représentant. Le client, à l'aide de l'interface, peut appuyer sur le bouton désigné à cet effet. Le client spécifie avec sa demande de connexion quelques paramètres tels que la langue, le département et le type de la demande. La demande de connexion est transmise au serveur qui

l'ajoute à une file d'attente. Quand un représentant, répondant aux préferences client, est disponible, la demande est relayée à l'agent avec les informations du client.

Aucune étape d'inscription du client n'est nécessaire pour ce modèle de communication. Toutefois, quelques services, comme les services bancaires, peuvent exiger que le client se soit identifié au préalable. Cette étape offrira une meilleure expérience client mais ne changera pas la communication en tant que telle.

2.4 Conclusion

Dans ce chapitre, nous avons présenté le mécanisme de signalisation et plus précisément le protocole SDP. Nous avons analysé ce format de description de la session multimédia et avons décelé plusieurs inconvénients. En effet, conçu à l'origine pour une utilisation unique à savoir la description des caractéristiques d'un point de communication, SDP n'est certainement pas adéquat pour le modèle offre-réponse. Lors de la négociation, plusieurs informations doivent être retransmises. Toutefois, SDP ne fournit aucune indication précise sur le changement qui a été fait.

Après avoir analysé le protocole SDP et vu que son utilisation au cours de la signalisation présente plusieurs inconvénients, nous avons identifié quelques modèles de communication possibles. Ces modèles nous permettront de voir par la suite une manière par laquelle SDP pourrait être simplifié. Le chapitre qui va suivre portera sur une étude de cas où nous verrons en détail le modèle de communication avec serveur intermédiaire et plus précisément le cas d'une communication facilitée.

Chapitre 3

Description et simulation d'un cas d'étude

Dans ce chapitre, nous étudions l'information requise pour l'établissement d'une communication et sa mise en forme, indépendamment de SDP. Pour ce faire, nous prenons un cas d'étude qui correspond aux modèles OTT, qui ont l'intérêt d'offrir plusieurs variantes et donc de pouvoir exploiter la souplesse que l'on a. Nous décrivons et simulons une communication entre deux clients désirant transmettre des données, de la voix ou de la vidéo. À travers ce scénario, nous verrons en profondeur, par l'exemple, les différents éléments de la communication. Nous proposerons en fin de ce chapitre une mise en œuvre du scénario décrit.

3.1 Scénario choisi

Dans ce scénario, nous adopterons le modèle de communication avec serveur intermédiaire, le cas d'une communication facilitée 2.3.2. Nous avons choisi de décrire et de présenter quelques éléments de simulation d'un cas d'étude. Ceci nous permettra d'avoir une idée pour simplifier SDP pour diverses situations ainsi que de bien clarifier le concept. Les scénarios choisis exploitent un serveur intermédiaire, unique - au moins conceptuellement - correspondant au modèle utilisé par la plupart des plateformes supportant les communications OTT.

Notre système comporte deux entités principales: Client et Centrale.

Centrale: la centrale est une entité intermédiaire qui facilite les appels, sauvegarde l'information et exécute des opérations telles que la négociation des paramètres d'appel. La centrale assure un mécanisme permettant de transformer une signalisation complexe en de simples opérations.

Client: peut désigner l'utilisateur physique ou la machine qui envoie des demandes à la centrale. Les communications seront établies entre deux clients.

Nous verrons dans ce qui suit les étapes typiques d'une communication ainsi que les différents messages échangés entre les entités.

3.2 Étapes de communication

Nous proposons les grandes étapes d'une communication, chacune avec les opérations associées. Pour toutes ces opérations, il y a une précondition commune que la centrale soit démarrée et en écoute. Nous présenterons les prérequis (informations requises), les informations connues de la centrale à la fin de chacune des opérations ainsi que le contenu des messages de signalisation. Les messages transportés combinent des éléments de SIP (opération) et des éléments de SDP.

Notons que chaque message échangé aura un identificateur. En effet, ceci permettra de détecter s'il y a une perte de messages; de plus, la centrale a besoin d'envoyer une réponse (succès ou échec de la requête): l'identificateur permettra donc de faire référence au message requête (provenant du client). Nous pouvons proposer que chaques messages requête-réponse aient le même identificateur (ou des identificateurs successifs).

Avant qu'un client A puisse entrer en communication avec un autre client B, le client A doit vérifier, à l'aide de la centrale, si le client B est présent. Les deux clients doivent au préalable s'enregistrer au niveau de la centrale pour qu'elle puisse les reconnaitre par la suite. Cette opération d'enregistrement (qu'on appelera inscription) n'est pas une opération faisant partie de la communication mais en est un prérequis. Nous définissons l'opération d'inscription comme suit:

Inscription: action d'inscrire un utilisateur, accédant à la centrale pour la première fois. Le but est d'avoir des droits d'utilisation en tant que client et de pouvoir identifier le client lors d'une prochaine connexion.

Nous identifions donc trois étapes principales: inscription, session et connexion.

3.2.1 Opérations relatives à l'inscription

L'étape d'inscription comprend deux opérations principales:

— **Nouvelle inscription**

Opération initiée par le client qui doit s'inscrire auprès de la centrale. Cette dernière reçoit les demandes d'inscription des clients, vérifie l'existence du compte et fournit au client le résultat de sa requête. L'information sur les utilisateurs est persistante.

— Prérequis: le client devrait au préalable connaitre le nom d'hôte de la centrale.

— Messages de signalisation:

Client à la centrale: le message comprend des informations générales sur l'usager: mot de passe, nom, prénom et adresse courriel. Le nom d'usager permettra d'identifier de façon unique les clients enregistrés.

Centrale au client: la centrale répond avec un code de succès ou d'échec de l'inscription.

— À la fin de cette opération: en cas de succès, la centrale a l'information sur l'identité du client.

— **Suppression d'une inscription**

Opération qui peut être initiée par le client. Le but est de supprimer une inscription qui a été conclue avec succès (client a reçu la confirmation de l'inscription).

3.2.2 Opérations relatives à la session

Les messages transmis durant les étapes relatives aux sessions et aux communications sont propres à chaque session et sont donc temporaires. Trois opérations peuvent être associées à l'étape de la session:

— **Ouverture de session**

Cette opération est initiée par le client et comprend l'authentification (voir Annexe A). Le but est de vérifier l'identité du client et d'autoriser des communications.

- Prérequis: le client doit être déjà inscrit.
- Messages de signalisation:

  Client à la centrale: les messages peuvent varier selon le mécanisme utilisé. En effet, plusieurs mécanismes d'authentification existent. Citons par exemple:

  - Authentification simple: l'authentification repose sur un seul élément; exemple: le message comprendra le mot de passe de l'usager.
  - Authentification forte: mécanismes qui requièrent au moins deux facteurs d'authentification comme le mécanisme de défi-réponse.

  Centrale au client: la centrale envoie un code de succès ou d'échec d'ouverture de la session. En cas de succès, un identificateur de la session est généré par la centrale afin d'en garantir l'unicité. La centrale peut attribuer à la session une durée de vie.
- Informations connues de la centrale à la fin de l'authentification: un canal de communication permettant l'échange des messages, ainsi que le nombre de sessions ouvertes par le client.

— **Inscription du terminal**

Cette opération est initiée par le client et vient juste après l'authentification. Le client inscrit son terminal en envoyant à la centrale l'information sur les médias supportés ainsi que la liste des codecs de chacun des médias. En réponse, la centrale informe le client des informations de connectivité, paramètres du protocole ICE pour les clients WebRTC.

Pourquoi recueillir les candidats ICE?

Un client, n'ayant aucune information sur la présence ou l'absence des NAT (Network Address Translation) ni sur leur emplacement, risque de fournir son addresse IP privée comme point de contact, ce qui empêchera l'établissement d'une connexion directe avec un autre client (il serait impossible à l'autre bout de le joindre et d'établir un flux média). Les protocoles ICE/TURN/STUN sont utilisés à cet effet pour les sessions multimédia en temps réel basées sur le protocole de transport UDP. Collectivement, ils permettent de fonctionner avec n'importe quel type de NAT/Firewall et permettent au client de contourner les limitations du réseau.

- Prérequis: client authentifié.
- Messages de signalisation:

Client à la centrale: l'identificateur de la session, médias supportés, liste des codecs pour chaque média.

Centrale au client: la centrale répond au client avec un message contenant l'information sur l'identificateur de la session, succès/échec de réception des informations. En cas de succès, la centrale informe le client des adresses de relais de transport.

— Informations connues à la fin de cette opération: la centrale a l'information sur les médias et codecs supportés par le client.

— **Rupture de session**

Ou fin de session, opération qui peut être initiée par le client ou par la centrale:

— déconnexion du client de la centrale;

— la centrale peut mettre fin à la session si un temps alloué à la session expire.

La centrale maintient à jour la liste des présences des clients.

— Prérequis: une session est ouverte par le client.

— Messages de signalisation:

Client à la centrale: le message comprend l'action de suppression et la référence à la session concernée (identificateur de la session).

Centrale au client: la réponse de la centrale comprend l'état de l'opération (succès ou échec) et l'identificateur de la session.

— Informations connues de la centrale: nombre de sessions ouvertes par le client. La centrale a aussi l'information sur la présence du client.

3.2.3 Opérations relatives à la connexion

Nous identifions quatre opérations principales.

— **Initiation d'une connexion**

Un client peut envoyer à la centrale une demande de connexion avec un autre client dont la présence a été vérifiée (voir Annexe B). Plusieurs connexions peuvent être établies au cours de la même session.

Les connexions sont elles séquentielles ou parallèles?

Notons ce que nous entendons par connexions parallèles: deux ou plusieurs connexions pouvant être établies simultanément. Les connexions séquentielles désignent une exécution successive de connexions. Autrement dit, le client ne peut initier une connexion que si aucune autre connexion n'est en cours.
Les connexions sont ici parallèles (le client peut mettre en attente un appel qui est en cours et initier une nouvelle connexion A ou A/V).

Enjeux d'établir plusieurs connexions

Nous supposons que la centrale est capable de gérer autant de demandes de connexions possibles et que la bande passante du client peut satisfaire les demandes de connexions (le cas échéant, les flux média vont être interrompus en gardant les connexions actives). Le problème qui pourrait se poser est du côté client: disponibilité des ressources du client (CPU, réseau, écran). Le client doit contribuer à ce niveau.
Afin d'initier une connexion:

— Prérequis: le client B doit être présent.

— Messages de signalisation:

  Plusieurs messages sont échangés. La figure 3.1 présente la machine d'état englobant les états et les messages pour cette opération.

  Client à la centrale: le message de demande de connexion indique le client à appeler (client B). Il comporte aussi le média à partager (choisi par le client A), l'adresse IP du client A, le port de réception du média ainsi que l'identificateur de la session.

  Centrale au client: la centrale échange avec les clients plusieurs messages provisoires avant d'établir la connexion. Tous ces messages réfèrent à la session en mentionnant l'identificateur de la session.

  — La centrale achemine la demande de connexion au client B. Le message d'invitation contient l'information sur le client appelant ainsi que le média à partager.
  — Le client B peut accepter la demande de connexion. Il envoie un message à la centrale contenant son adresse IP et le port de réception du média.
  — Après avoir reçu la réponse du client B, la centrale envoie une notification au client A l'informant si la demande de connexion est acceptée ou refusée.
  — La centrale, ayant déjà les informations des codecs supportés par les deux clients, établit un codec commun pour le média choisi. Elle envoie par la suite à chacun des

deux clients la description de l'autre client (adresse IP et port de réception du média) ainsi que le codec à utiliser. La centrale attribue enfin un identificateur de connexion. Cet identificateur sera unique et commun entre les deux clients pour cette connexion puisqu'il s'agit de la même connexion.

Dans cette approche, le client communique à coup sûr. Une alternative serait de proposer la connexion des deux côtés, et d'annuler si l'un des clients refuse. Le nombre d'opérations ne devrait néanmoins pas être sensiblement différent.

Intérêt de l'identificateur de connexion: nous verrons par la suite que les clients pourront faire une variation en cours d'appel (ajout d'un flux média, changement de codec). Le client devrait donc aviser la centrale du changement. Puisqu'un client peut avoir plusieurs connexions au cours de la même session, un identificateur de connexion s'avère obligatoire. Aussi, le client doit informer la centrale en cas de rupture de connexion.

— Informations connues de la centrale à la fin de cette opération: la centrale connait le nombre de connexions actives pour chacun des clients A et B.

— **Rejet d'une connexion**

Le rejet d'une connexion peut être fait de la part de la centrale ou du client:

— Côté client:
  — Choix du client: le client refuse l'appel.
  — Le client ne répond pas: la centrale attribue un temps de vie à la demande de connexion durant lequel le client doit donner suite à la demande.
  — Rejet de l'appel dans le cas où l'appareil client n'est plus capable d'accepter de nouvelles connexions.
— Côté centrale:
  — La centrale reçoit la demande de connexion du client initiateur, mais avant de l'acheminer au deuxième client, ce dernier n'est plus présent.
  — Le client peut envoyer à la centrale une information de statut (occupé, ne pas déranger). Sur la base de cette information, la centrale peut décider de refuser ou d'accepter la demande de connexion.

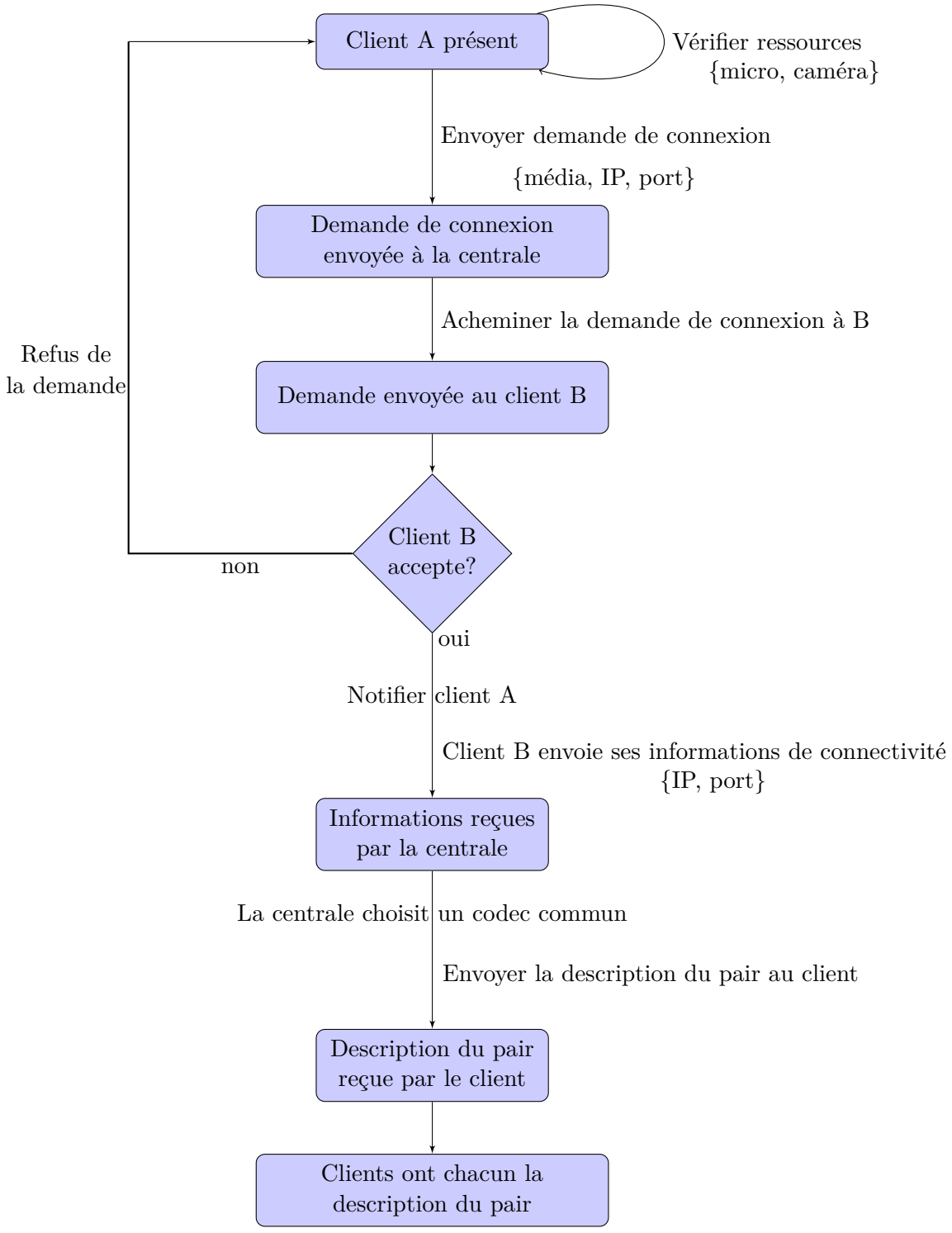


Figure 3.1 – Machine d'état pour l'initiation de connexion

— **Modification de paramètres**

Cette opération correspond à une variation lors d'une communication: ajout, suppression ou modification d'un paramètre au cours de la communication (changement de codec par

exemple, adaptation aux conditions du réseau). L'opération peut être initiée par le client ou par la centrale.

- Prérequis: la connexion est en cours.
- Messages de signalisation:

  Les messages peuvent être échangés de bout en bout ou par l'intermédiaire de la centrale. Pour notre cas d'étude, nous choisissons une synchronisation de bout en bout.

  Un client ne peut pas exécuter automatiquement une variation sans passer par ces deux étapes:

  - Annoncer: le client voulant faire une modification doit envoyer une requête de modification et en attendre la réponse. Le message contient l'information sur la variation (le nouvel état du paramètre à varier), ainsi que les identificateurs de la session et de la connexion.
  - Accepter: le client recevant une requête de modification répond avec un message d'acceptation (ou refus) de la modification.

  L'avantage de se coordonner de bout en bout est de permettre aux clients de modifier une connexion courante en cas de redémarrage de la centrale.

  Les clients avisent par la suite la centrale du changement par rapport à la connexion.
- Informations connues de la centrale: nouvel état du paramètre modifié.

- **Rupture de connexion**

  Cette opération est établie de bout en bout. Les clients peuvent rompre l'appel et aviser la centrale par la suite; elle n'intervient pas au cours de la rupture. La centrale a le rôle de mettre à jour la liste des connexions actives après chaque rupture de connexion.
  - Prérequis: la connexion est en cours.
  - Messages de signalisation:

    Client à la centrale: identificateurs de sessions et de connexion, action de supprimer la connexion.

    Centrale au client: pas de message de retour (le précédent est un message informatif de la part du client).
  - Informations connues de la centrale:
    - nombre de connexions actives pour le client;
    - nouvel état du client.

3.3 Problèmes divers

3.3.1 Cas du crash de la centrale

En cas de crash de la centrale, les connexions en cours peuvent être préservées si un système de récupération a été implanté. Sinon, l'information sur les connexions en cours sera perdue. Ces questions sont hors de portée de notre étude.

3.3.2 Administration de la centrale

La centrale peut avoir un utilisateur avec des droits d'administration qui pourra exécuter les opérations suivantes:

— lister les sessions courantes;
— lister les connexions en cours;
— voir l'état actuel d'un client (en appel, occupé), sa disponibilité, ses statistiques;
— arrêter la centrale.

3.3.3 Cas d'erreur

Pour toute étape de communication, la centrale peut envoyer un message d'erreur, suite à une erreur de traitement ou à la non compréhension d'une information qui lui a été transmise (le message reçu n'est pas au bon format par exemple). Le message d'erreur indique le type d'erreur rencontrée. Le client en est informé car il pourrait être censé réagir, selon l'erreur qui a été rencontrée.

3.4 Implantation du scénario

3.4.1 Utilisation de Websocket

Avec le Web traditionnel, un client envoyant une requête au serveur ne peut faire une autre action qu'après avoir reçu une réponse à sa première requête. Ceci cause une longue attente au niveau du client. Ce problème a été résolu avec l'introduction de l'objet XMLHttpRequest (16) qui assure

un fonctionnement asynchrone. Un mécanisme permettant des connexionx bidirectionnelles entre le client et le serveur a été récemment intégré dans la plupart des navigateurs Web. En effet, Websocket, protocole normalisé par l'IETF (17), permet de créer et d'ouvrir des canaux de communications bidirectionnelles entre les navigateurs et les services Web. Les messages sont transmis de façon simultanée en utilisant des connexions TCP. Les services Web peuvent envoyer directement aux navigateurs des messages à n'importe quel moment sans que le client ait initié une requête. Ainsi, les clients peuvent être notifiés par d'éventuels changements d'état au niveau du service Web.

3.4.2 Préparation de l'environnement

SocketIO

Nous utilisons, pour la simulation, SocketIO (18), une bibliothèque Javascript qui permet des communications en temps réel en assurant des connexions de longue durée. Si Websocket est déjà supporté par le client (ce qui est le cas pour la plupart des navigateurs Web récents), alors SocketIO choisit automatiquement ce protocole. Sinon, SocketIO a recours à d'autres techniques et elle use la solution la plus efficace. En utilisant SocketIO, on est sûr que tout navigateur client serait capable de se connecter au serveur.

Langages et outils utilisés

La démarche utilisée pour implémenter le serveur de signalisation et le client est présentée en Annexe D. Nous y présentons aussi quelques captures de la réalisation. Le langage utilisé pour la simulation de la centrale est Python 3.4. Du côté client nous avons utilisé le langage de balisage HTML5 (19) afin de créer et structurer des pages Web. HTML5 a l'avantage d'offrir un contenu riche sans besoin de plugins. La version actuelle offre plusieurs fonctionnalités (animation, musique, etc) et est conçue pour différentes plateformes (PC, téléphones intelligents et même pour Smart TV). Nous avons utilisé Javascript/CSS afin de donner un style aux pages HTML et les transformer en des interfaces utilisateur interactives. De plus, un système de gestion de base de données a été utilisé afin de pouvoir stocker et gérér quelques informations clients qui sont persistantes.

Pour la présentation des messages de signalisation nous avons eu recours au format de données textuelles JSON (JavaScript Object Notation). Nous illustrons quelques exemples des messages de signalisation au cours du prochain chapitre.

3.5 Conclusion

Dans ce chapitre, nous avons regardé de plus près les informations nécessaires à l'établissement d'une communication. Nous avons décrit un scénario de communication ainsi que des éléments de sa simulation. L'objectif est de simplifier le format SDP pour divers problèmes, ce qui fera l'objet du reste du document. Le chapitre qui suit portera sur une illustration du concept qu'on a décrit avec des messages qui peuvent être traités de manière générale. Nous regarderons la performance de notre approche par rapport à l'utilisation de SIP/SDP.

Chapitre 4

Illustration du concept

Dans le chapitre précédent, nous avons présenté le contexte et les principes directeurs. Le présent chapitre portera sur une illustration de l'approche déjà décrite. Nous allons extraire les messages eux-mêmes et verrons les échanges entre les entités.

Le processus d'offre-réponse propose que le client initiateur envoie la description de la session de communication qu'il veut partager. Cette description (offre), incluant le média et les codecs à utiliser, les relais de transport ainsi que d'autres informations, est relayée au client récepteur. Ce dernier génère une description de la session, appelée une réponse, qui va être relayée à l'initiateur de l'offre. Le concept que nous proposons dans ce chapitre ne suit pas cette logique (l'offre-réponse). Nous supposons qu'au lieu que les descriptions soient générées par les clients, une entité intermédiaire (déjà présente pour d'autres finalités) supporte le processus de la négociation des paramètres de la session multimédia. Cette entité doit connaitre au préalable quelques éléments statiques tels que les codecs supportés par les clients. Elle décide des paramètres à utiliser pour la session et en informe les clients. Autrement dit, avec notre approche, l'offre est pré-déterminée et ne doit pas être produite en temps-réel.

4.1 Échanges entre les entités

Dans cette section, nous présentons quelques diagrammes de séquences décrivant les échanges entre le client et la centrale. Nous donnons la structure des messages de signalisation pour les différentes étapes de la communication.

4.1.1 Quelques notions et spécifications

Notons que les informations échangées entre entités peuvent être communiquées selon deux approches:

— **Approche 1**: existence d'un protocole responsable de la signalisation, comme c'est le cas pour SIP. L'action est définie par les messages du protocole.
— **Approche 2**: le protocole est neutre et est simplement responsable du transport (fiable) des messages.

Pour l'approche 1, les messages seront similaires à l'approche 2 mais en l'absence de quelques paramètres tels que l'action. Nous optons pour l'approche 2 afin d'illustrer le cas d'étude choisi. Quelle que soit l'étape de la communication, le message a une structure systématique, ce qui facilitera son traitement et permettra sa compréhension. Le listing 4.1 présente la structure globale d'un message, où:

« **H** »: représente l'entête du message, il permet d'identifier le message, spécifie son le type ainsi que l'action desirée s'il y a lieu. D'autres informations peuvent être incluses selon le contexte.

« **D** »: représente le corps du message qui va inclure des paramètres explicites et implicites. Sa structure diffère selon l'étape de la communication. Ce champs n'est pas obligatoire et donc sera absent si aucune information n'y sera incluse.

Tous les messages de signalisation que nous présenterons pour ce cas d'étude seront illustrés sous format JSON. Tout autre format de représentation de structures de données peut être utilisé, tel que XML. Nous présentons dans le tableau 4.1 l'ensemble des abréviations utilisées des différents paramètres définissant un message.

Listing 4.1 – Structure globale d'un message

```
{
  H:{
  },
  D:{
  }
}
```

Nous aurons besoin d'identifier la transaction car un client pourra interagir dans différentes transactions en même temps et les réponses qu'il reçoit doivent spécifier de quelle transaction il s'agit. Nous noterons cet identificateur par « **TID** ». La requête initiale, l'ensemble des réponses/messages provisoires ainsi que la réponse finale auront le même TID.

Un message doit spécifier le type qu'on notera par « **T** », ceci permettra à l'entité (client ou centrale) d'avoir l'information sur l'objet du message. Les valeurs possibles de ce paramètre sont définies dans le tableau 4.2. Le type peut être:

- une requête **« R »**: ce type de message requiert une réponse. Nous incluons les transmissions et retransmissions d'une requête déjà existante, puisque le contenu du message retransmis ne va pas différer.
- une réponse à une requête **« Rs »**, incluant le cas de succès ou d'échec de la requête (le message reçu a été traité et compris à la réception, exemple d'échec de la requête: erreur client).

  Une réponse peut être finale ou intermédiaire:
  - finale: indiquant la fin de la transaction;
  - intermédiaire: ce message est provisoire et est utilisé pour indiquer l'état de la requête.
- un message de notification où l'émetteur informe d'un état et n'attend pas de réponse, on le notera par **« Nt »**;
- une erreur **« Er »**: le message reçu ne peut pas être traité.

Si un message est de type requête « R », une action est requise. Nous notons l'action par « **A** ». Le tableau 4.2 spécifie l'ensemble des actions possibles.

Tableau 4.1 – Principaux paramètres composant un message

Paramètre	Usage
H	*header*, représente l'entête du message.
D	*data*, représente le corps du message.
TID	*transaction identifier*, sert à identifier la transaction.
T	*type*, définit le type du message JSON.
A	*action*, l'action à exécuter.
St	*status*, comporte un code/message donnant l'état de la requête.
RID	*retransmission identifier*, distingue les nouvelles requêtes des requêtes retransmises de la même transaction. Il est nécessaire si le message est de type requête.
SID	*session identifier*, identifie de manière unique une session entre le client et la centrale.
M	*media*, selon le contexte du message, ce paramètre peut informer des médias et codecs supportés par le terminal client, ou dans le cas d'une invitation, indiquer le média à partager.
Cd	*client device*, identifie le terminal utilisé par le client et informe de sess caractéristiques.
Ss	*screen size*, taille d'écran du terminal client.
CID	*connection identifier*, identifie de manière unique une connexion entre deux clients.

Pour les messages de type **Rs**, un statut doit être spécifié. Nous le notons par « **St** ». Il comporte un code de succès/erreur avec le message adéquat. À l'aide du code, l'entité peut savoir s'il s'agit d'une réponse finale ou provisoire.

Tableau 4.2 – Valeurs possibles de certains paramètres d'un message

Paramètre	Valeurs possibles	Utilisation
T	**R**	*Request.*
	Rs	*Response.*
	Nt	*Notification.*
	Er	*Error.*
A	**register**	inscription d'un nouveau client.
	login	ouverture de session d'un client enregistré.
	add	ajouter/inscrire un terminal.
	invite	invitation à une nouvelle connexion.
	log out	fermeture de session.
	disconnect	demande de fin de connexion.
	activate	demande d'activation de flux.
	update	demande de changement au cours de la communication.
	pause	demande d'arrêt du flux média.
	resume	demande de reprise du flux média

Cas où aucune réponse n'est reçue suite à une requête

L'initiateur d'une requête suppose que sa requête n'a pas été reçue s'il ne reçoit pas de réponse après un certain délai. Après ce délai, l'initiateur peut retransmettre sa requête. Le message retransmis va incrémenter un indice de retransmission qu'on notera par « **RID** ». Nous présentons en Annexe C quelques diagrammes de séquencement expliquant la nécessité de ce paramètre. RID est initialisé par 0 avec la requête initiale.

4.1.2 Préparation à l'appel

Une étape avant service existe et peut être intégrée afin de pouvoir identifier le client. Nous présentons en Annexe E les échanges entre le client et la centrale pour les opérations d'inscription et d'ouverture de la session. D'autres formes et alternatives existent selon les besoins de l'implémentation, par exemple si la communication est embarquée dans un autre service de type commerce électronique.

Quand une session est ouverte avec succès, un identificateur de session « **SID** » est généré par la centrale (E.5 présente un exemple de message de succès d'ouverture de la session). SID aide à créer un contexte pour les communications à venir, et agit donc comme « jeton d'accès ». La centrale l'envoie dans le corps du message de succès d'ouverture de la session. Tous les messages de signalisation échangés entre les entités vont contenir dorénavant cet identificateur (qui sera inclus dans l'entête des messages).

Après avoir reçu le jeton de la session, le client récupère automatiquement les informations concernant le terminal et les médias supportés et les transmet par la suite à la centrale. Ces informations restent statiques le long de la session, d'où l'intérêt de les identifier à ce niveau. Nous nommons cette opération l'inscription du terminal.

Inscription du terminal

À cette étape, le client définit les possibilités du terminal et des médias. Une manière de le faire, quand on exploite une plateforme WebRTC, est de générer une offre SDP et d'en retirer l'information désirée. Dans l'API WebRTC, l'opération RTCPeerConnection (20) permet de créer une offre en générant un objet SDP. Cet objet contient la configuration de la session incluant la description des médias et des options des codecs qui sont pris en charge. De plus, l'objet donne l'information sur tous les candidats qui ont été recueillis par l'agent ICE. Ces candidats représentent des addresses de relais de transport pour chacun des médias. Pour notre cas d'étude, c'est la centrale qui est responsable d'envoyer les adresses des candidats ICE au client. La figure 4.1 illustre cette opération.

Nous présentons en listing 4.2 un exemple de message d'inscription du terminal, envoyé du client à la centrale. Ce message inclut des données informatives. Le corps du message comporte deux principaux blocs d'informations:

— **« M »**: décrit les médias et codecs supportés ainsi que d'autres paramètres et options.
La présentation des données recueillies (médias et codecs supportés) est conforme aux notations de l'IETF (21) et de IANA (22).
Exemple pour la présentation de l'information sur le codec:
<encoding name>/<clock rate> [/<encoding parameters>].

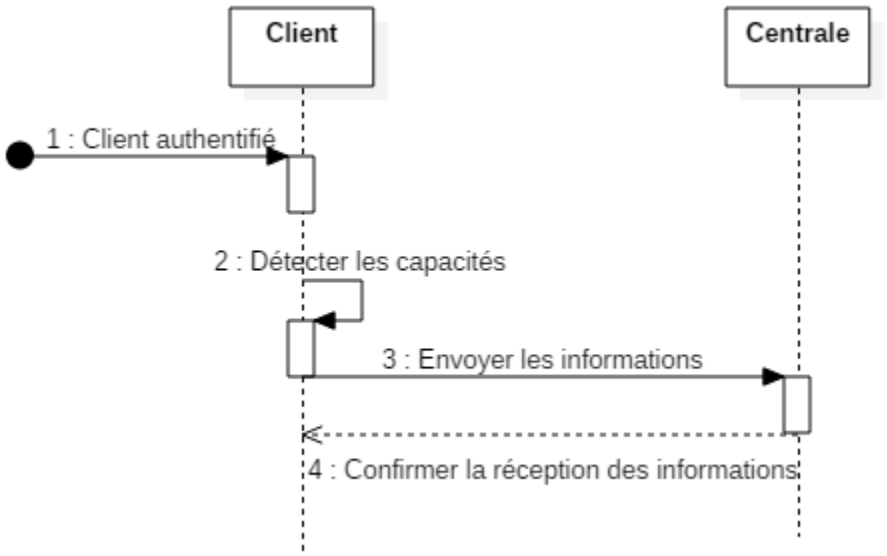


Figure 4.1 – Diagramme de séquences pour l'inscription du terminal

— **Cd**: un client peut être connecté depuis plusieurs terminaux lors de la même session, d'où le besoin d'identifier le terminal. Ce bloc donne les caractéristiques du terminal utilisé comme l'orientation et la taille d'écran. Nous notons par:

  — **« mode »**: orientation du terminal informant sur les types de rotation possibles: « portrait », « landscape » ou « seascape ».

  — **« Ss »**: le client indique à la centrale la résolution d'image maximale qu'il supporte. L'information sur la taille d'écran est négociée (au niveau de la centrale) afin de minimiser les distortions d'image lors de la communication vidéo et assurer ainsi une meilleure qualité chez le client. Nous proposons de garder les notations de l'IETF (23).

Listing 4.2 – Inscription du terminal

```
{
  H:{
      ID:{
          SID:12455565,
          TID:5,
          RID:0
      },
      T:R,
      A:"add"
  },
  D:{
      M:{
          audio:{
              codecs:["Opus/48000/2", "PCMU/8000" , "PCMA/8000", "CN/32
                  000"]
          },
          video:{
              codecs:["VP8/90000", "VP9/90000" , "red/90000", "ulpfec/9
                  0000"]
          }
      },
      Cd:{
          ID:5,
          properties:{
              mode:{"portrait", "landscape"},
              Ss:""
          },
          client:["Chrome", "52.0.2743.116"]
      }
  }
}
```

Enfin, le terminal informe sur le logiciel utilisé qu'on indique par « client ». Il correspond au type du navigateur Web utilisé (Mozilla, Chrome ou autre) ainsi que sa version.

La centrale confirme l'inscription du terminal client, correspondant au message 4 de la figure 4.1. Ce message, présenté dans le listing 4.3, est de type réponse. Le statut indique l'état du traitement de la requête d'inscription du terminal. La centrale inclut dans sa réponse les informations de connectivité à savoir les adresses des relais de transport ainsi que d'autres paramètres du protocole ICE.

Listing 4.3 – Réponse à l'inscription du terminal

```
{
  H:{
       ID:{
            SID:12455565,
            TID:5
       },
       T:Rs,
       St:[200, "request processed successfully"]
  },
  D:{
       connectivity:{
            candidates:"",
            parameters:""
       }
  }
}
```

Rupture de session

Le listing 4.4 présente un message de demande de rupture de session, envoyé du client à la centrale. Le client décide de mettre fin à la session en appuyant sur le bouton désigné.

Listing 4.4 – Rupture de la session

```
{
  H:{
       ID:{
            SID:12455565,
            TID:6,
            RID:0
       },
       T:R,
       A:"logout"
  }
}
```

Suite à la demande de rupture de session, la centrale répond avec un message de confirmation et met à jour la liste des présences.

4.1.3 Initiation d'une connexion

L'ensemble des messages que nous présentons pour cette opération correspondent à la machine d'état déjà décrite dans 3.1.

Supposons un client A dans son état initial « présent ». Pour demander une connexion audio ou audio/vidéo avec un client B, le client A devrait au préalable vérifier la présence du client B. Le client A vérifie les ressources internes nécessaires pour cette connexion (disponibilité du micro et de la caméra). Il envoie par la suite la demande de connexion à la centrale en spécifiant le média qu'il désire partager, son adresse IP et le port où il désire recevoir le média. Nous présentons en listing 4.5 un exemple de message d'invitation à une communication audio.

Dans le corps du message, le client appelant envoie trois principaux blocs d'information, à savoir: le client à appeler, le média et les informations de connectivité. Le client à appeler est indiqué par « **To** », la valeur que ce paramètre contient peut correspondre au nom d'usager unique du client. Dans les informations de connectivité, le client informe sur le chemin de transport qu'il souhaiterait utiliser pour recevoir le média, annonçant ainsi la version IP qu'il utilise, son adresse

IP et le port de réception du média. Le message n'a pas besoin de présenter les informations de transport sous forme de « Clé:Valeur ». La centrale devrait être capable de déceler ce que chacune de ces informations désigne. Le message indique aussi le média choisi par le client, son mode de transmission est « sendrecv » indiquant que le client souhaite envoyer et recevoir la voix. Nous spécifions ce paramètre car au cours d'une communication, le client peut mettre l'appel en attente ou passer au mode discrétion (mute) et donc ce paramètre doit être signalé à l'autre bout. Il ne devrait pas avoir de confusion entre le mode utilisé ici et celui représentant la taille d'écran lors de l'inscription du terminal, puisque chacun est utilisé dans un contexte différent.

Listing 4.5 – Invitation au client B

```
{
  H:{
      ID:{
          SID:12455565,
          TID:7,
          RID:0
      },
      T:R,
      A:"invite"
  },
  D:{
      To:"client B",
      M:{
          type:"audio",
          mode:"sendrecv"
      },
      Transport:["IPv4", "207.162.11.201", "8000"]
  }
}
```

La centrale achemine l'invitation reçue à la destination (client B). Le corps du message informe sur l'origine de l'invitation et le média à partager. Nous illustrons en listing 4.6 un exemple de message d'acheminement de l'invitation de la centrale au client B. Le message est de type requête.

Les identificateurs de session, SID, et de transaction, TID, sont propres aux communications entre la centrale et le client B.

Listing 4.6 – Acheminement de l'invitation

```
{
  H:{
       ID:{
            SID:798898,
            TID:11,
            RID:0
       },
       T:R,
       A:"invite"
  },
  D:{
       from:"client A",
       M:{
            type:"audio",
            mode:"sendrecv"
       }
  }
}
```

Après réception de l'invitation, le client B peut accepter ou refuser la demande de connexion. Si la demande est acceptée, le client B envoie ses informations de connectivité, à savoir l'adresse IP et le port de réception. Le listing 4.7 présente un exemple du message d'accord envoyé à la centrale.

La centrale récupère les informations reçues de la part du client B. Elle communique aux deux clients A et B la description de l'autre client ainsi que le codec commun pour la communication désirée (des informations directives). La centrale génère aussi un jeton de connexion. Nous présentons un exemple de message de description envoyé de la centrale au client appelant (A) en listing 4.8. Le corps de ce message contient la description du client B (informations de connectivité), le codec

choisi et l'identificateur de la connexion « CID ». Ce dernier identifie une connexion particulière (entre deux clients) au cours de la session et est partagé par les clients de la même connexion.

Listing 4.7 – Acceptation de l'invitation

```
{
  H:{
       ID:{
            SID:798898,
            TID:11
       },
       T:Rs,
       St:[200, "OK"]
  },
  D:{
       Transport:["IPv4", "92.153.171.111", "8080"] }
}
```

Listing 4.8 – Description du client

```
{
  H:{
       ID:{
            SID:12455565,
            TID:7
       },
       T:Rs,
       St:[200, "OK"]
  },
  D:{
       CID:"conn1",
       Transport:["IPv4", "92.153.171.111", "8080"],
       audio: "Opus/48000/2"}
}
```

Avant de commencer l'échange du média, il faut s'assurer que le flux a été accepté. Le client initiateur (ici le client A) doit envoyer un message au client B l'avisant du succès de l'établissement des paramètres du média. Ce message est de type notification « Nt », ne nécessitant pas de réponse. Il peut être envoyé de bout en bout ou par l'intermédiaire de la centrale. L'exemple 4.9 est un message envoyé du client A à la centrale.

Suite à la réception de cette notification, la centrale met l'état des deux clients « en appel ».

Listing 4.9 – Notification: flux accepté

```
{
  H:{
      ID:{
          SID:12455565,
          TID:8,
          CID:"conn1"
      },
      T:Nt,
      A:"activate"
  }
}
```

4.1.4 Durant la communication

Modification de paramètres

Le client peut faire une variation au cours d'une communication. Le message de modification référencie au paramètre à modifier et spécifie la variation à faire. La synchronisation peut se faire de bout en bout.

Exemple de modification:

Nous présentons un exemple de modification initié par le client. L'opération se fait en trois étapes comme expliqué précédemment: annoncer, accepter et exécuter.

Listing 4.10 – Annoncer une modification

```
{
  H:{
       ID:{
            SID:12455565,
            TID:9,
            CID:"conn1",
            RID:0
       },
       T:R,
       A:"update"
  },
  D:{
       audio:{
            mode:"inactive"
       }
  }
}
```

Annoncer: le message 4.10 présente une demande de mise en attente de l'appel . Le message est de type requête dont l'action est « update ». Le client spécifie le paramètre à varier dans le corps du message avec la nouvelle valeur désirée. Ici, le client veut changer le mode à inactif. La présentation se fait sous forme « Clé:Valeur » et la clé est unique pour une connexion donnée, donc il n'y aurait pas de confusion.

Si le client demande de faire plusieurs variations en même temps, il va falloir spécifier toutes les variations dans le corps du message, chacune avec sa clé.

Accepter: le client B accepte la demande de variation provenant du client initiateur. Le message est présenté en listing 4.11.

Listing 4.11 – Accepter une modification

```
{
  H:{
       ID:{
            SID:798898,
            TID:12,
            CID:"conn1"
       },
       T:Rs,
       St:[200, "OK"]
  }
}
```

Le message peut aussi comprendre la variation approuvée (dans le cas où plusieurs variations ont été demandées par le client).

Exécuter ou confirmer la variation:

Après avoir reçu la confirmation du client B, le client A envoie une notification indiquant qu'il reconnait la réponse de B et confirme qu'il exécute la mise en attente de l'appel. Ici le message est de type notification dont l'action est « pause » indiquant que le partage média va s'arrêter.

Listing 4.12 – Exécuter la modification

```
{
  H:{
       ID:{
            SID:12455565,
            TID:9,
            CID:"conn1"
       },
       T:Nt,
       A:"pause"
} }
```

Pour la reprise de l'appel, les deux clients doivent également échanger trois messages, dont la structure est similaire aux précédents.

Rupture de connexion

Les deux clients peuvent arrêter la connexion de bout en bout et aviser ensuite la centrale de leur nouvel état. La figure 4.2 présente le diagramme de séquencement pour cette opération.

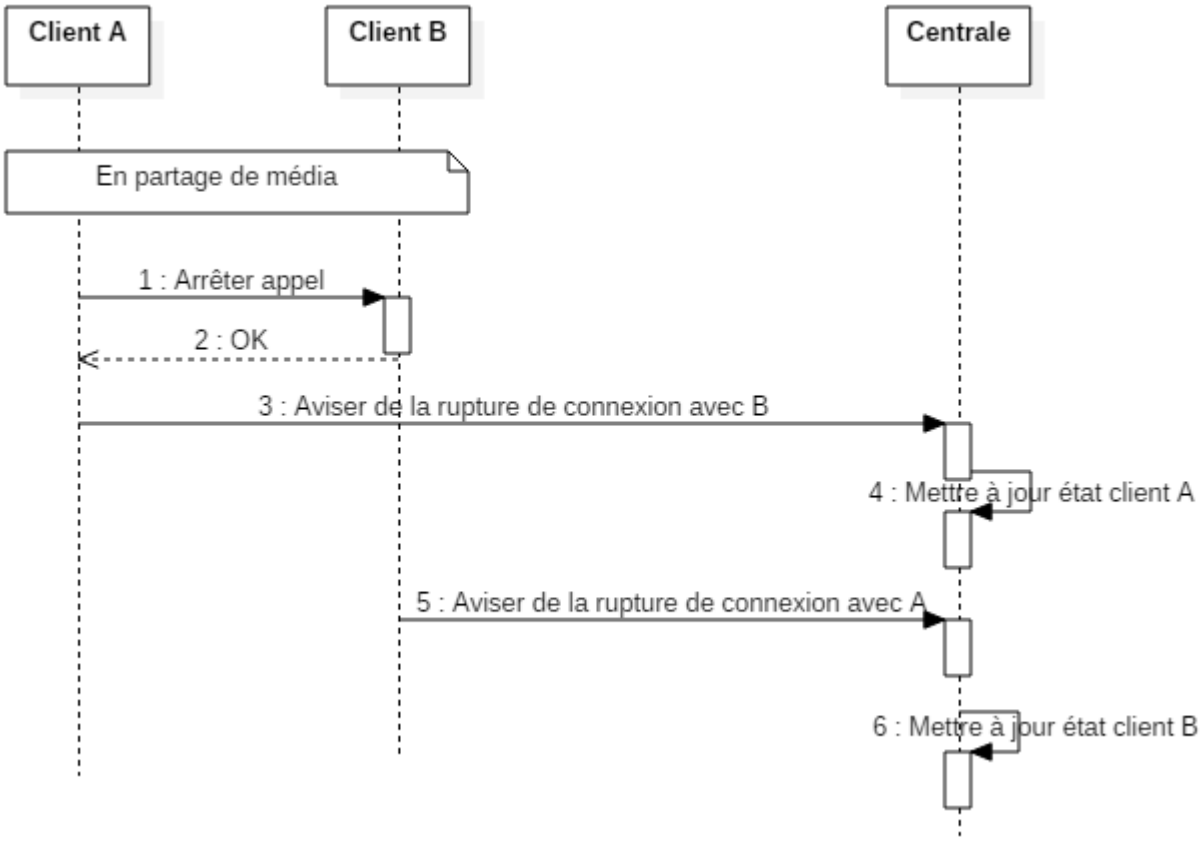


Figure 4.2 – Diagramme de séquences pour la rupture de connexion

Le message 4.13 est une demande du client B de mettre fin à l'appel. Le client A accepte la demande de rupture de connexion, sa réponse est présentée en listing 4.14.

Listing 4.13 – Demande d'arrêt de la connexion

```
{
  H:{
      ID:{
          SID:798898,
          TID:13,
          CID:"conn1",
          RID:0
      },
      T:R,
      A:"disconnect"
  }
}
```

Listing 4.14 – Réponse à la demande de rupture de connexion

```
{
  H:{
      ID:{
          SID:12455565,
          TID:10,
          CID:"conn1"
      },
      T:Rs,
      St:[200, "OK"]
  }
}
```

Enfin, les deux clients avisent la centrale de la fin de connexion. Le listing 4.15 est un avis de rupture de connexion envoyé du client à la centrale. Il est de type notification.

Listing 4.15 – Avis de rupture de connexion

```
{
  H:{
       ID:{
            SID: 798898,
            TID: 14,
            CID:"conn1"
       },
       T:Nt,
       A:"disconnect"
  }
}
```

4.2 Conclusion

Dans ce chapitre, nous avons étudié, par l'exemple, une nouvelle possibilité de signalisation, moins complexe et mieux structurée. Une forme de restructuration de l'information communément trouvée dans SDP, mais aussi d'autres éléments véhiculés par SIP. Nous avons simplifié SDP par d'autres types de messages mais qui peuvent être traités de manière générique. Dans le chapitre suivant, nous verrons quelques cas d'étude que nous étudierons brièvement. La visée principale est de prouver que ce concept reste agile face à divers scénarios de communications embarquées.

Chapitre 5

Discussion / Généralisation

Dans le chapitre précédent, nous avons travaillé en détail sur un scénario afin de bien illustrer le concept décrit dans le chapitre 3. Nous avons vu les échanges entre les entités et avons détaillé les messages de signalisation. Le présent chapitre portera sur d'autres cas d'étude, que nous traitons de manière plus succincte. Le but est d'illustrer la souplesse, la flexibilité et les gains du concept proposé.

5.1 Étude d'autres scénarios

Nombreuses sont les plateformes qui supportent l'intégration des services de communication interactive, comme c'est le cas pour les géants des réseaux sociaux Facebook et Twitter ou encore Gmail. Ces plateformes fournissent déjà des opérations de base telles que l'inscription, l'authentification et la présence. Contrairement aux modèles de communication traditionnels, les communications interactives deviennent une valeur ajoutée à la plateforme. Dans les scénarios que nous choisissons d'étudier, les communications interactives sont embarquées au lieu d'être le coeur du service. Ces cas d'étude suivent un modèle client-serveur plutôt que pair à pair, ce qui ouvre la porte à plusieurs simplifications. En effet, les communications sont créées et contrôlées par le serveur et dans certains cas, comme le commerce électronique et WebRTC, établies par la plateforme elle-même. La négociation des paramètres d'appel ne se fait pas de bout en bout comme c'est le cas pour le modèle offre-réponse. Le routage d'appel disparait donc, contrairement aux modèles trapézoïdaux.

Nous choisissons d'étudier les scénarios suivants:

1. partage de média avant établissement de la communication;
2. appel initié par un tiers;
3. communication asymétrique.

Pour chaque cas, nous verrons:

— les étapes nécessaires pour établir une communication;
— le nombre de messages qui doivent être échangés pour accomplir une étape;
— ce qui change dans les échanges de messages et le contenu d'un message par rapport au cas d'étude déjà traité.

Les illustrations présentées pour chaque cas d'étude sont des possiblités, d'autres alternatives peuvent exister. Nous verrons dans le chapitre suivant une comparaison approfondie des messages nécessaires à l'accomplissement d'une étape, selon notre concept et le modèle trapézoïdal.

5.1.1 Étape d'initialisation

Une étape « avant service » est nécessaire et est commune pour tous les scénarios que nous traitons. Cette étape inclut une inscription de l'équipement où le client informe le serveur de signalisation des modes de communication possibles. Autrement dit, les médias et les codecs supportés ainsi que d'autres caractéristiques du terminal. Découvrir quels codecs sont supportés par le terminal est un sujet problématique. Nous supposons que le client est capable de questionner la plateforme et de découvrir les codecs supportés. Dans le cas de WebRTC, ceci se fait par la récupération d'un corps SDP. Toujours dans un contexte WebRTC, le processus d'inscription de l'équipement est similaire à celui précédemment étudié dans le chapitre 3 (3.2.2). À la fin de cette phase, le serveur informe le client de ses informations de connectivité correspondant aux candidats ICE pour les clients WebRTC.

5.1.2 Cas 1: partage de média avant établissement de la communication

Des informations peuvent être transmises de façon restreinte avant que l'appel soit accepté. Un exemple récent de ce type de communication est l'option « Knock-Knock » (26) offerte par l'application Google Duo. Cette option permet à l'appelé de voir qui l'appelle; une forme d'aperçu vidéo en temps réel. Le protocole SIP/SDP a un genre de facilitateur sous la forme de « early media » (27), à savoir un mécanisme qui permet d'implanter et d'autoriser l'échange d'information avant l'établissement complet d'une connexion. D'autres systèmes peuvent utiliser ce modèle de communication, par exemple les services exigeant un degré élevé de sécurité. Le client, voulant communiquer avec un service, initie la communication. Une identification faciale du client est faite avant d'acheminer l'appel au destinataire. Le système utilise le média reçu, dans ce cas la vidéo, afin de pouvoir identifier le client de manière plus sécure.

Le média provisoire est partagé de l'initiateur au récepteur; le sens inverse peut être techniquement possible mais présente des soucis de confidentialité. De plus que les paramètres transmis au cours de l'étape d'initialisation (5.1.1), les clients sous ce modèle doivent indiquer s'ils supportent ou non la réception d'un média provisoire pour d'éventuelles communications subséquentes.

Établissement d'une communication

Nous présentons dans la figure 5.1 un flux de communication entre deux clients pour une communication réussie avec un partage de média provisoire. « G » représente une centrale Google ou autre.

— **Messages avant média provisoire**: (Messages 1 – 5)

1. Le client A, désirant communiquer avec le client B, envoie une requête d'invitation à G. Le contenu du message d'invitation est similaire à celui présenté dans le chapitre 3, contenant l'information sur le client à appeler, l'adresse IP du client appelant, le média à partager et le port de réception du média.
2. G envoie une invitation à une session de média provisoire au client B. Le message donne l'information sur le client appelant et le type du média que A désire partager.
3. Le client B confirme être prêt à recevoir le média provisoire; il fournit avec sa réponse les informations nécessaires à la réception du média (adresse IP et port de réception

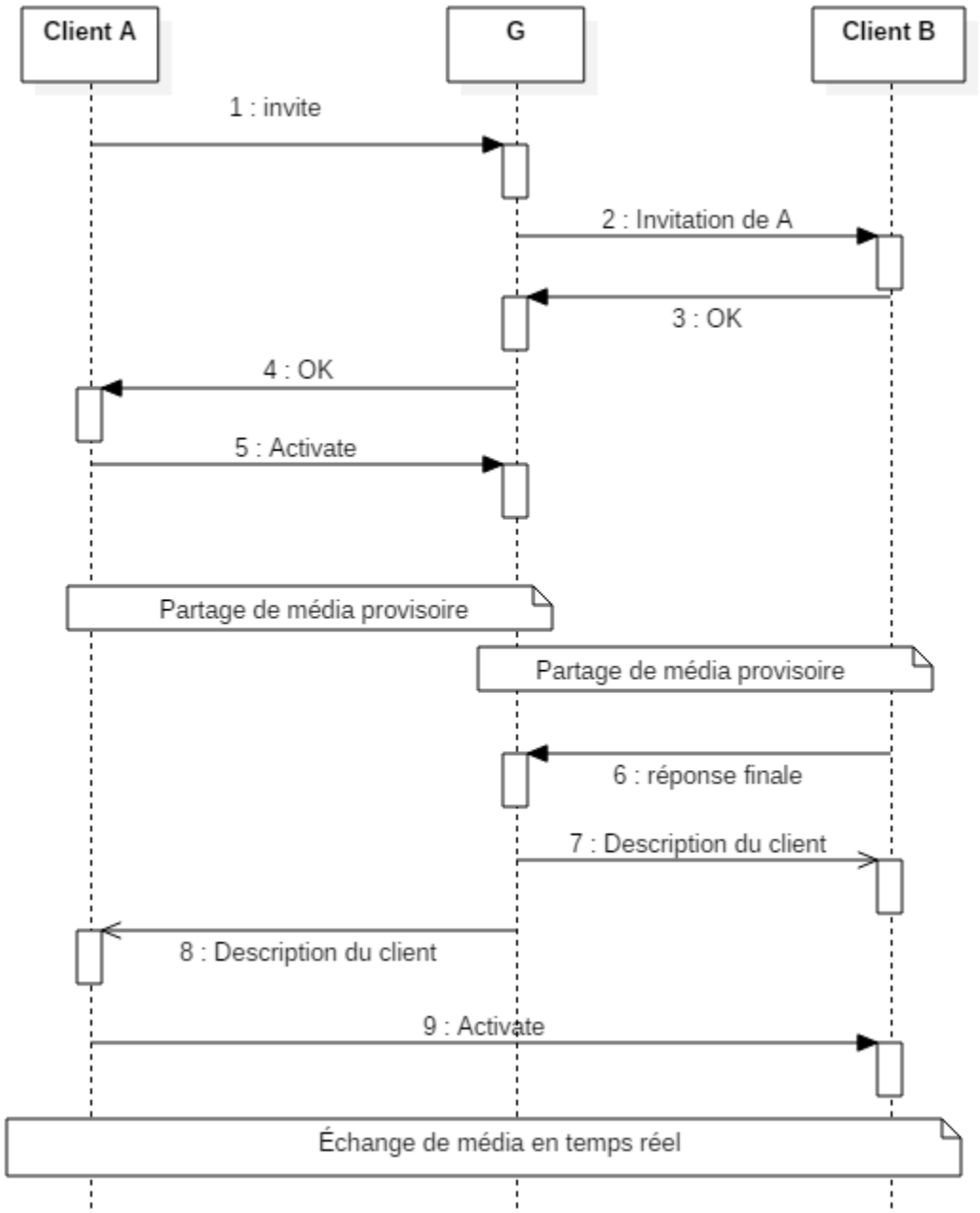


Figure 5.1 – Partage média avant établissement de l'appel

du média provisoire). Cette réponse ne représente pas une réponse finale à l'invitation initiale.

4. Le serveur G répond à l'invitation du client A (cette réponse confirme au client A que le client B est présent et que G a l'information nécessaire pour transmettre au client B le média provisoire).

5. Le client A envoie une notification pour indiquer que le partage de média provisoire va commencer.

— **Partage du média provisoire**:

Avant que G ne reçoive une réponse finale à l'invitation envoyée à B, le média est partagé, dans une seule direction, du client initiateur A vers le client B. Le partage peut être fait de P2P ou par l'intermédiaire de G. Nous présentons ce qui change pour ces deux possibilités:

— de pair à pair: dans ce cas, le message d'invitation acheminé à B (message 2) doit contenir l'information sur le codec à utiliser pour le média provisoire. G a besoin d'informer le client A du chemin de transport à utiliser pour envoyer le média au client B; cette information est incluse avec le message 4. L'activation du flux, message 5, est envoyée directement de A à B.

— avec l'intermédiaire de G: la centrale G peut aussi jouer le rôle d'un serveur média, comme indiqué dans la figure 5.1. L'avantage de cette solution est que B n'a pas besoin d'avoir l'information sur le(s) codec(s) à utiliser puisque G prendra en charge le transcodage du média.

— **Réception d'une réponse finale et établissement de la communication**:

6. La succession des messages (6 – 9) dans la figure 5.1 est similaire aux échanges déjà présentés dans le chapitre 3, si ce n'est que G n'a pas besoin d'aviser le client initiateur A que B a répondu à l'invitation. En effet, la réception du message de description présente pour A une réponse positive et finale à l'invitation initiale.

   À la réception d'une réponse finale de la part de B, G crée la connexion, décide du codec commun à utiliser pour la communication et envoie la description de chacun des clients à l'autre extrémité. Comme décrit dans le chapitre 3, la description comprend l'adresse IP de l'autre client ainsi que le(s) codec(s) à utiliser pour cette communication. Le port de réception du média est le même que celui communiqué précédemment pour la réception du média provisoire. Si le client souhaite recevoir le média sur un autre port, alors une resynchronisation doit avoir lieu.

   Aucune négociation des paramètres de l'appel n'est faite de bout en bout, tout est totalement pris en charge par G. Le serveur de signalisation G doit être capable de déterminer la possibilité de réaliser ce service, sur la base des informations sur les terminaux et les préférences des usagers.

5.1.3 Cas 2: communications asymétriques

Ce modèle n'exige aucune forme de présence. En effet, un client n'a pas besoin d'être identifié avant de pouvoir effectuer une communication. Le client joue le rôle d'initiateur de l'appel. La communication est asymétrique dans le sens où l'appelé n'a pas l'information pour initier une communication avec le client. Les communications asymétriques sont souvent utilisées par diverses solutions de communication comme les centres d'appel, le commerce électronique et le support à la clientèle.

Prenons l'exemple d'un client qui navigue sur le site Web d'un fournisseur de service et veut parler à un représentant. Avant que la communication passe à l'appelé (agent), une communication vocale peut être établie avec un outil de synthèse vocale ou des messages enregistrés. Ce modèle de communication nécessite l'existence d'un contrôleur central capable de réacheminer l'appel sans implication de l'appelant, contrôler la communication ainsi que de demeurer présent comme relai de la communication. Il joue le rôle d'intermédiaire permanent dans la communication. Ceci implique donc la présence d'un « B2BUA » (relais voix dos à dos) qui prend en charge la création de l'appel et fournit un contrôle centralisé des communications. Il participe activement à l'établissement de l'appel. Tous les messages de signalisation passent et sont traités par le B2BUA.

Établissement d'une communication

La figure 5.2 présente le séquencement des messages échangés entre les entités pour une communication asymétrique réussie.

Dans un modèle « click-to-call », le client va sur la page du service et demande de parler à un représentant en cliquant sur le bouton installé à cet effet. Ceci envoie une requête HTTP au B2BUA (message 1 de la figure 5.2). Cette requête est une invitation à une nouvelle connexion, contenant les chemins de transport média désirés par le client. En effet, nous supposons que la demande d'inscription du terminal se fait en début de session et donc les relais ICE sont déjà connus par le client. La demande de connexion (message 1 de la figure 5.2) spécifie aussi le type du média que le client veut partager pour la communication avec l'agent. Le type des communications offertes est déterminé et défini par l'entreprise. De plus, le corps du message peut inclure les préférences client (langue, département, type de la demande).

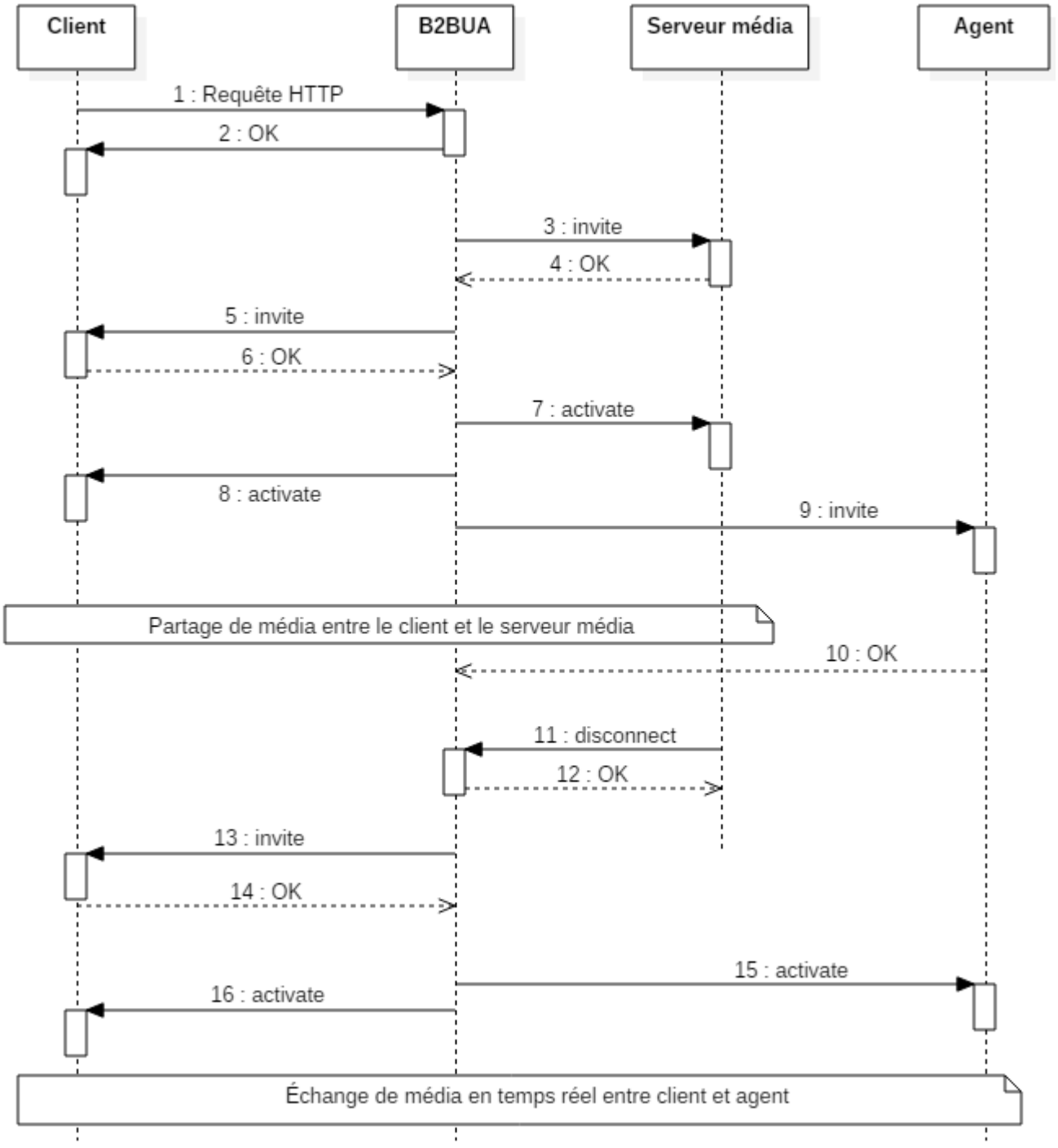


Figure 5.2 – Communication asymétrique

— **Établissement d'une communication avec le serveur média**: (Messages 3 – 8)
Avant que l'appel passe à un agent, une communication vocale est établie en premier lieu avec un outil de messages enregistrés. En effet, le B2BUA met le client en communication

avec un serveur média. Le média, par défaut audio, est partagé dans une seule direction, du serveur média au client. Le B2BUA prend en charge la création et l'établissement de cette communication. Ceci a l'avantage d'offrir au client une meilleure expérience puisque l'attente sera réduite. Le fournisseur de service pourra en profiter pour passer au client des messages de divertissement ou d'information.

3. Le B2BUA envoie respectivement au client et au serveur média des invitations à la communication vocale (messages 3 et 5) incluant l'information sur le codec à utiliser pour cette communication vocale. L'invitation envoyée au serveur média contient aussi le chemin de transport à utiliser pour envoyer le média au client. Le média sera envoyé dans une seule direction, donc il n'y aurait pas besoin d'informer le client des informations de transport du serveur média.
4. Après avoir reçu l'acceptation des invitations (messages 4 et 6), le B2BUA notifie le client et le serveur média que la communication vocale va commencer (messages 7 et 8).

Au cours de la communication vocale entre le client et le serveur média, le B2BUA envoie une invitation à l'agent (message 9) l'informant du média que le client veut partager pour cette communication (média choisi avec la requête initiale (message 1)). Cette invitation informe aussi l'agent du chemin de transport à utiliser pour joindre le client ainsi que le(s) codec(s) à utiliser pour cette communication. L'agent répond (message 10) en informant le B2BUA de l'adresse IP et du port où il souhaite recevoir le média. En effet, nous supposons que les terminaux des représentants sont déjà enregistrés au niveau du B2BUA et donc chaque représentant connait les relais de transport qui peuvent être utilisés pour recevoir un média.

À ce niveau, le B2BUA attend d'avoir une confirmation du serveur média que la session de communication vocale est achevée (messages 11 et 12) avant d'établir une communication entre le client et l'agent.

— **La communication passe à l'agent**:

Après avoir reçu la réponse du l'agent (message 10), le B2BUA invite le client à la communication (message 13). Le média pour cette communication est le même que celui choisi précédemment par le client. Cette invitation informe le client du codec à utiliser ainsi que des informations de transport de l'agent. Après réception du message d'acceptation de la communication (message 14), le B2BUA envoie une notification respectivement au client et à l'agent pour que le partage du média puisse commencer.

5.1.4 Cas 3: l'appel est initié par un tiers

Ce scénario peut correspondre aux options de rappel, par exemple: un client appelle un service qui n'est pas disponible ou dont le temps d'attente est élevé. Le service propose donc au client de le rappeler après un certain délai. Ce modèle de communication est également utilisé par les plateformes de conférences où une entité intermédiaire, qu'on appelera également une centrale, est capable de créer une communication qui se déroulera entre d'autres parties.

Établissement d'une communication

Nous présentons dans la figure 5.3 le séquencement des messages échangés pour une communication établie par une tierce partie. Le flux de communication est similaire au cas d'étude décrit dans le chapitre 3, si ce n'est que les invitations sont initiées par la centrale elle-même.

Nous supposons que la centrale a l'information pour vérifier la présence des clients A et B. Pour les services de vidéo conférence par exemple, un lien unique est fourni au client. Une fois que le client accède au lien, la centrale se charge de le joindre à la conférence. L'initialisation (inscription du terminal) peut être faite à ce niveau.

Les messages d'invitation (messages 1 et 3) informent les clients du(es) média(s) qui peuvent être partagés au cours de cette communication. Le choix final du média reste à la discrétion du client. Les réponses des clients (messages 2 et 4) contiennent, de plus que le choix du média à partager, l'information sur le chemin de transport que le client souhaite utiliser pour recevoir le média.

Les messages qui suivent sont similaires au cas d'étude 3. Après avoir envoyé à chacun des clients la description de l'autre bout, la centrale les notifie que le partage du média va commencer (messages 7 et 8). L'activation du média est envoyée de la centrale au client.

Les scénarios traités au cours de ce chapitre ont été simplement choisis de manière à prouver la validité du concept quelque soit le modèle de communication. En effet, nous avons essayé de voir d'autres cas qui suivent différents modèles de communication autres que celui décrit lors du chapitre 3. Nous avons prouvé qu'une bonne partie de la logique de notre approche reste la même pour chacun de ces scénarios.

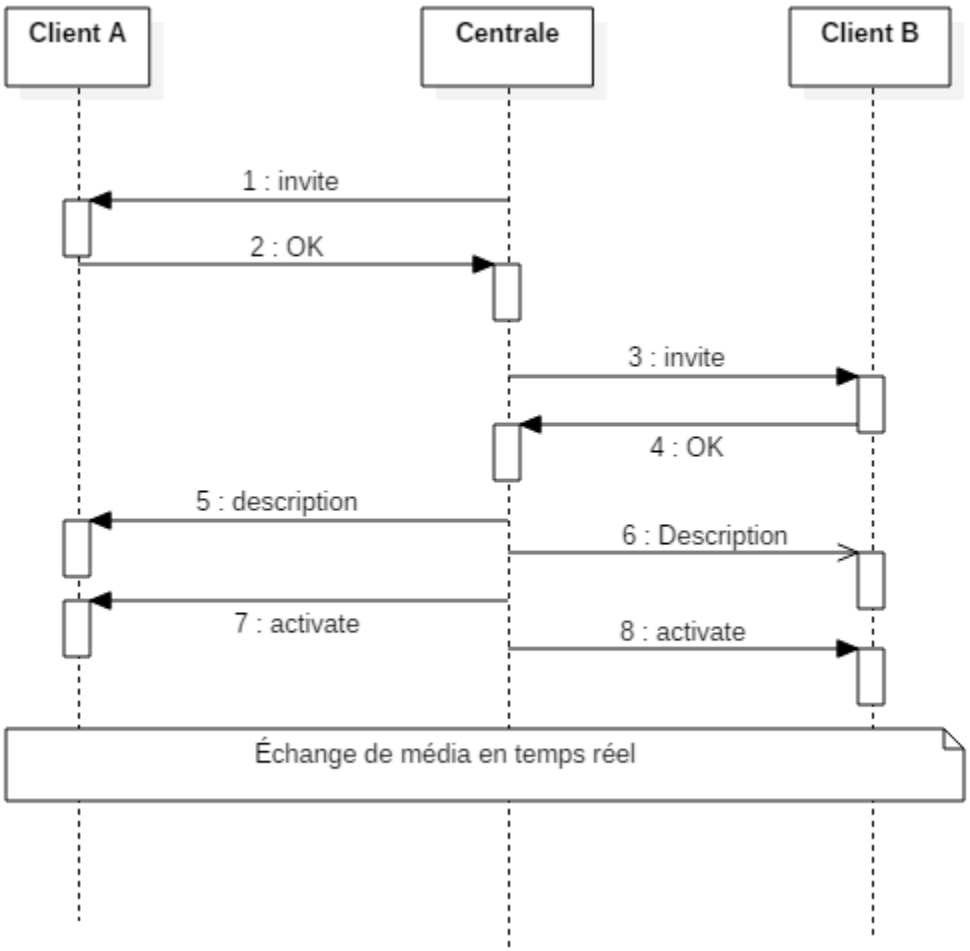


Figure 5.3 – Communication initiée par une tierce partie

5.2 Discussion

Pour les différents cas d'étude traités, nous remarquons qu'une bonne partie de messages de signalisation est similaire à ce qu'on a déjà décrit et montré dans le chapitre 3. Par contre, la logique de contrôle est différente. D'autres simplications peuvent être considérées, à titre d'exemple, la centrale peut envoyer les descriptions avec les messages d'invitations. Elle décide du codec à utiliser pour chaque média offert et en informe les clients. Les réponses des clients donneront simplement l'information sur le chemin de transport préféré. L'établissement de la communication se fera avec des messages de signalisation ayant plus de contenu mais avec une latence plus faible. Si l'un des clients refuse l'appel alors le processus sera annulé.

Le processus de signalisation est souvent associé avec une machine d'état qui capte les différents états de l'entité ainsi que les opérations qui lui sont offertes pour chacun des états. Notre concept s'affranchit du modèle d'appel qui est intégré à SIP, ce qui est utile pour des modèles alternatifs. Nous n'abordons pas le modèle d'appel dans notre étude; néanmoins, l'application devra le tenir en compte.

5.3 Compatibilité des codecs

Le partage du média en temps réel est de pair à pair. Nous avons vu avec les modèles précédents que les clients ne font pas de négociation des paramètres de la session. Au contraire, une entité intermédiaire (serveur de signalisation) prend en charge la négociation. En effet, ce serveur de signalisation, ayant au préalable l'information sur les modes de communication possibles, décide du(es) codec(s) à utiliser pour la communication. Si pour une communication donnée, le serveur de signalisation ne peut pas établir un codec commun entre les clients, alors la communication doit échouer. Pour pallier à ce problème, nous proposons l'existence d'un relais qui, outre les fonctions ICE, permet également de faire du transcodage du média. D'un côté pratique, tous les clients WebRTC (navigateurs) supportent l'utilisation du codec ITU-T G.711. Toutefois, nous ne pouvons pas forcer l'utilisation de ce codec en raison de son support par la plupart des terminaux. Le choix du codec doit être dynamique et varie selon le contexte d'utilisation. À titre d'exemple, G.711 est consommateur de la bande passante mais donne une meilleure qualité, alors que le codec G.729 consomme moins de bande passante mais avec un léger déclin de la qualité.

5.4 Conclusion

Nous avons vu à travers quelques scénarios, traités brièvement, la flexibilité du concept proposé. Dans les scénarios choisis, la signalisation est gérée par une entité qui ne fait pas partie de la communication en tant que telle mais qui reste fondamentale pour le support des communications. Le chapitre suivant portera sur une comparaison des deux approches de signalisation. Nous prouvons les gains de l'utilisation de notre concept par rapport à SIP/SDP.

Chapitre 6

Gains et faiblesses par rapport à SIP/SDP

Nous comparons notre approche par rapport à l'utilisation du protocole SIP/SDP. Nous verrons les gains de notre concept et prouvons que nous transmettons moins d'informations dans les messages de signalisation. Nous comparons le nombre de messages échangés durant une étape de communication et discutons les faiblesses de notre approche.

6.1 Comparaison du concept avec SIP/SDP

Afin de pouvoir comparer la méthode de signalisation proposée avec l'utilisation de SIP/SDP, nous étudierons trois principaux critères, à savoir:

— les étapes nécessaires pour l'établissement d'une communication;
— le nombre de messages échangés au cours d'une étape;
— la taille du message de signalisation.

6.1.1 Étapes d'établissement d'une communication

Pour que deux clients puissent communiquer, ils doivent s'inscrire et créer un contexte pour la communication (ouverture de session). Notre concept propose une étape d'inscription du terminal

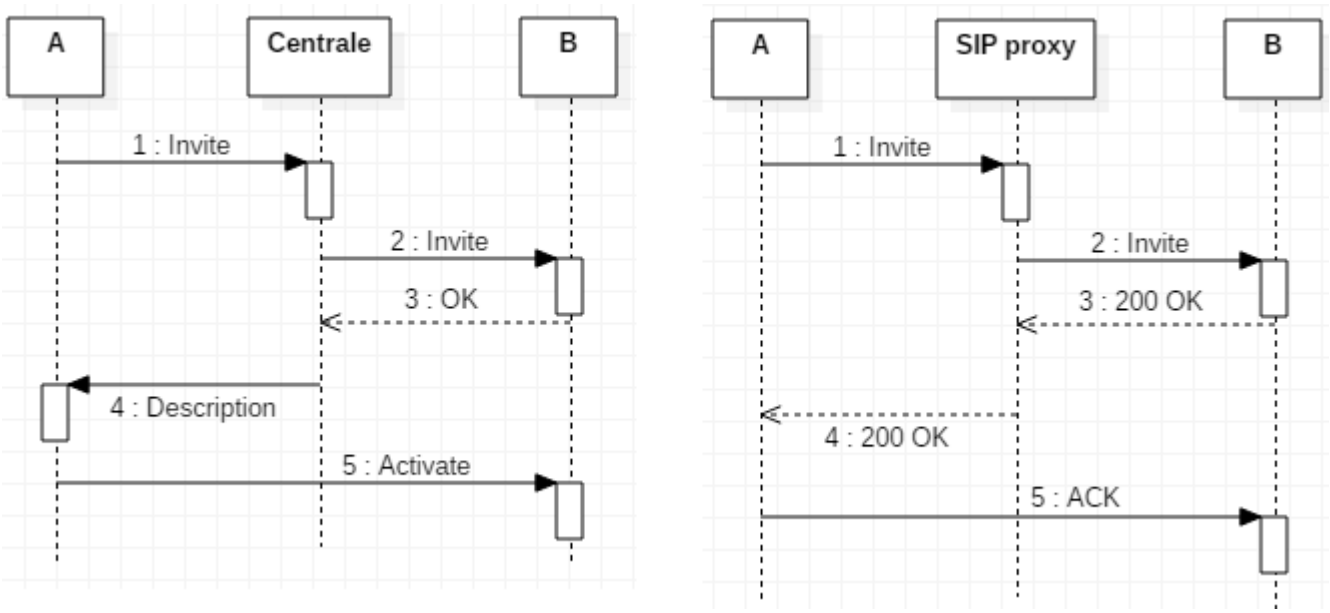


Figure 6.1 – Flux minimal d'une communication pour les deux approches

où le client informe la centrale de ses informations statiques (médias, codecs et caractéristiques du terminal). Cette opération est comparable au message « REGISTER », de SIP, qui est combiné au « login ». Ces étapes sont similaires pour les deux protocoles (notre approche et SIP/SDP).

6.1.2 Nombre de messages par étape

Nous comparons, selon les deux approches, les messages qui doivent être échangés entre l'initiateur et le récepteur afin d'accomplir une opération. Nous choisissons d'étudier les deux opérations: établissement d'une connexion et modification en cours d'appel. Pour la modification en cours de la communication, nous verrons l'exemple de la mise en attente de l'appel.

— **Établissement d'une connexion**: la figure 6.1 présente une comparaison du flux d'établissement d'une communication selon les deux approches. L'invitation ou l'offre initiale est relayée, pour le protocole SIP, de bout en bout. Le proxy ne participe pas à la négociation des paramètres de la communication. À la réception d'un message « invite », le client peut générer une réponse provisoire « 180 Ringing », message non obligatoire. Aussi, le client peut envoyer un message SIP « 100 Trying » avant le message « 180 Ringing », ce message étant facultatif. Nous comparons ici juste les messages minimaux nécessaires à la réalisation de l'opération.

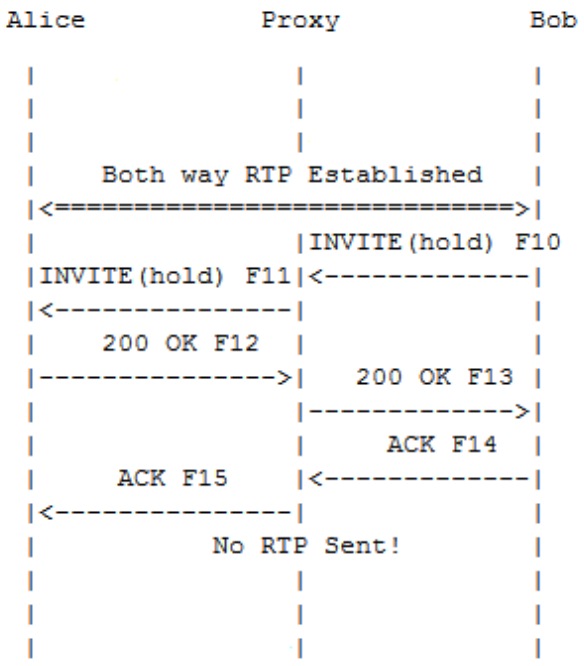


Figure 6.2 – Flux de communication pour la mise en attente de l'appel (SIP)

Le mécanisme d'offre-réponse exige donc, au minimum, quatre messages pour l'établissement de l'appel, de plus que le message d'activation du flux. Notre approche propose d'échanger également quatre messages pour établir la communication. La principale différence est que l'offre ne provient pas de l'initiateur de l'appel mais plutôt de la centrale qui décide des paramètres à utiliser pour cette communication.

— **Modification en cours de la communication**: pour la mise en attente d'un appel, les clients SIP échangent trois messages (INVITE, 200 OK et ACK) qui sont relayés par le proxy SIP. La figure 6.2 (tirée de (24)) présente un exemple de messages échangés lors de cette opération.

Le flux de communication pour la mise en attente de l'appel, dans le cas de notre approche, a été détaillé dans le chapitre précédent. Nous supposons que la modification est initiée par le client avec une synchronisation de bout en bout. Les clients échangent trois messages (annonce, acceptation et confirmation du changement). La confirmation du changement est en effet un acquittement indiquant que le partage de média va s'arrêter. Il est équivalent au message ACK pour SIP/SDP. Une notification est par la suite envoyée à la centrale afin

d'aviser du changement effectué. À noter que cette notification n'ajoute pas de temps à l'établissement d'appel.

Nous résumons dans le tableau 6.1 les messages de signalisation pour les deux opérations étudiées.

Tableau 6.1 – Comparaison entre SIP/SDP et notre approche

	SIP/SDP	Notre approche
Établir une connexion	5 messages	5 messages
	INVITE (2)	Invite (2)
	200 OK (2)	OK
		description du client
	ACK	Activate (flux accepté)
Mise en attente	3 messages	4 messages
	INVITE	Annoncer
	200 OK	Accepter
	ACK	Exécuter
		Notifier la centrale

Améliorations sur les messages supplémentaires

Des messages supplémentaires peuvent être échangés au cours de l'établissement de l'appel. Le but est d'informer sur un état provisoire du traitement d'une requête. À titre d'exemple, la centrale dans 6.1 peut envoyer un message provisoire au client indiquant que l'invitation a été reçue. Ce message permettra d'éviter les multiples transmissions de l'invitation. Nous illustrons en listing 6.1 un exemple de ce message avec notre approche. De même, le client B peut accuser la réception de l'invitation avant d'envoyer une réponse finale. Ces réponses provisoires comportent, tout comme le message « OK », un statut sous forme de code/message. Les codes et les messages utilisés peuvent être les mêmes que ceux utilisés par le protocole HTTP.

Pour SIP, des messages, dont le principe est similaire, sont utilisés en cours de l'établissement de la communication. Nous présentons en listing 6.2 (tiré de (25)) un exemple de message

« 100 Trying », envoyé du proxy au client suite à un message « Invite ». Le message ne contient pas un message SDP.

Listing 6.1 – Accusé de réception de l'invitation

```
{
  H:{
      ID:{
          SID:798898,
          TID:11
      },
      T:Rs,
      St:[100, "Continue"]
  }
}
```

Listing 6.2 – Exemple de message provisoire « 100 Trying »

```
SIP/2.0 100 Trying
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9;
    received=192.0.2.101
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 1 INVITE
Content-Length: 0
```

6.1.3 Taille du message de signalisation

Afin de pouvoir voir si notre approche permet une diminution de la quantité d'informations transmises, nous étudions deux exemples d'opérations: initiation d'une connexion et modification en cours d'une communication.

Il n'est pas évident de comparer la taille (en bytes) de deux messages chacun avec un contexte différent, nous comparons plutôt le nombre d'informations/paramètres transmis avec chacun de ces messages de signalisation.

Initiation d'une connexion

Nous présentons en listing 6.3 (tiré de (25)) un exemple de message d'invitation avec le protocole SIP/SDP.

En plus de l'entête qui est lourd, le corps de ce message donne plusieurs informations à savoir:

— l'identificateur de la session et sa version;

— le type du réseau, le type de l'adresse IP ainsi que l'adresse du client qui a crée le document SDP;

— l'adresse IP où le média va être envoyé/reçu;

— le type du média et le port où le client souhaite le recevoir;

— le protocole de transport ainsi que la liste des codecs supportés.

La plupart de ces informations vont être retransmises au cours de la négociation, alors que certaines d'entre elles ne sont même pas pertinentes pour l'offre-réponse.

Listing 6.3 – Exemple de message d'invitation avec SIP/SDP

```
INVITE sip:bob@biloxi.example.com SIP/2.0
Via: SIP/2.0/TCP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
Max-Forwards: 70
Route: <sip:ss1.atlanta.example.com;lr>
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76s1
To: Bob <sip:bob@biloxi.example.com>
Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 2 INVITE
Contact: <sip:alice@client.atlanta.example.com;transport=tcp>
Proxy-Authorization: Digest username="alice",
 realm="atlanta.example.com",
 nonce="wf84f1ceczx41ae6cbe5aea9c8e88d359", opaque="",
 uri="sip:bob@biloxi.example.com",
 response="42ce3cef44b2250c6a6071bc8"
Content-Type: application/sdp
Content-Length: 151

v=0
o=alice 2890544526 2890844526 IN IP4 client.atlanta.example.com
s=-
c=IN IP4 192.0.2.101
t=0 0
m=audio 49172 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

Pour pouvoir comparer entre l'utilisation de SIP/SDP et notre approche, nous présentons en listing 6.4 un exemple de message d'invitation avec notre concept. L'entête de ce message permet d'identifier le client ainsi que le type du message et la transaction à laquelle il appartient. Le corps du message donne l'information sur six principaux paramètres: le client à appeler, informations sur le type du média et les informations de transport (la version du protocole IP utilisé, l'adresse IP et le port).

Listing 6.4 – Exemple d'invitation avec notre approche

```
{
  h:{
      ID:{
          SID:12455565,
          TID:7,
          RID:0
      },
      type:"R",
      a:"invite"
  },
  d:{
      to:"client B",
      m: {
          type:"audio",
          mode:"sendrecv"
      },
      Transport:["IPv4", "207.162.11.201", "8000"]
  }
}
```

Modification en cours d'appel

Avec SIP/SDP, pour qu'un client puisse faire une modification en cours de la communication, il doit renvoyer toute la description déjà transmise. Dans l'exemple 6.5, Alice demande d'établir deux

flux audio avec Bob (offre initiale). Après avoir accepté cette offre 6.6, Bob décide de mettre le flux audio en attente; il envoie donc une demande de modification sous forme d'une deuxième offre 6.7. Alice répond alors avec un message SDP similaire à son offre initiale 6.8. Cet exemple est tiré de (13).

Listing 6.5 – Offre SDP initiale

```
v=0
o=alice 2890844526 2890844526 IN IP4 host.atlanta.example.com
s=
c=IN IP4 host.atlanta.example.com
t=0 0
m=audio 49170 RTP/AVP 0 97
a=rtpmap:0 PCMU/8000
a=rtpmap:97 iLBC/8000
m=audio 49172 RTP/AVP 98
a=rtpmap:98 telephone-event/8000
a=sendonly
```

Listing 6.6 – Réponse SDP à l'offre initiale

```
v=0
o=bob 2808844564 2808844564 IN IP4 host.biloxi.example.com
s=
c=IN IP4 host.biloxi.example.com
t=0 0
m=audio 49172 RTP/AVP 97
a=rtpmap:97 iLBC/8000
m=audio 49174 RTP/AVP 98
a=rtpmap:98 telephone-event/8000
a=recvonly
```

Listing 6.7 – Demande de mise en attente

```
v=0
o=bob 2808844564 2808844565 IN IP4 host.biloxi.example.com
s=
c=IN IP4 host.biloxi.example.com
t=0 0
m=audio 49172 RTP/AVP 97
a=rtpmap:97 iLBC/8000
a=sendonly
m=audio 49174 RTP/AVP 98
a=rtpmap:98 telephone-event/8000
a=recvonly
```

Listing 6.8 – Réponse à la demande de mise en attente

```
v=0
o=alice 2890844526 2890844527 IN IP4 host.atlanta.example.com
s=
c=IN IP4 host.atlanta.example.com
t=0 0
m=audio 49170 RTP/AVP 0 97
a=rtpmap:0 PCMU/8000
a=rtpmap:97 iLBC/8000
m=audio 49172 RTP/AVP 98
a=rtpmap:98 telephone-event/8000
a=sendonly
```

Au cours de cette demande de modification (mise en attente de l'appel); la plupart des informations renvoyées avec le message de modification sont redondantes: informations de connectivité, médias, codecs et leurs paramètres.

Listing 6.9 – Demande de mise en attente

```
{
  h:{
      ID:{
          SID:12455565,
          TID:9,
          CID:"conn1",
          RID:0 },
      type:"R",
      a:"update"
  },
  d:{
      audio:{
          mode:"inactive" }
  }
}
```

Notre méthode propose de faire le changement plus rapidement. Les variations durant la communication peuvent être faites de bout en bout. Le message de modification spécifie juste la variation à

faire et il y a moins d'informations à traiter. Nous présentons en listing 6.9 un exemple de demande de mise en attente de l'appel.

6.2 Gains par rapport à SIP/SDP

Nombre de messages échangés

Le protocole SIP/SDP se base sur un modèle pair à pair où chaque message est relayé de bout en bout. Le modèle P2P suppose que les terminaux sont toujours présents ou joignables ce qui est rarement le cas dans un environnement Web. Les scénarios étudiés précédemment se basent sur une structure client-serveur qui est plus facile à implémenter surtout avec l'existence d'une plateforme qui supporte déjà des opérations de base.

En utilisant notre approche, les messages ne sont pas relayés entre les extrémités mais sont plutôt contrôlés et traités par un serveur intermédiaire. Ceci a l'avantage de centraliser la négociation et de pallier aux inconvénients du routage des messages selon le mécanisme d'offre-réponse. Le nombre de messages échangés pour établir l'appel est le même pour les deux approches. La différence principale réside en la restructuration de l'information. Au besoin, les deux approches peuvent intégrer d'autres messages facultatifs. Pour le cas d'une variation au cours de la communication, notre concept utilise un message de plus que SIP/SDP. Rappelons que ce message n'a aucun impact sur le temps d'établissement.

Informations transmises dans le message de signalisation

Nous avons restructuré les données de signalisation en informations statiques et informations dynamiques. Le client inscrit son terminal en fournissant un ensemble d'informations statiques par rapport à la session. Cette réorganisation des informations permet de réduire la taille du message puisqu'il y aura moins d'informations échangées au cours des communications. La comparaison établie dans 6.1 montre que, avec notre concept, moins d'informations sont échangées dans les messages de signalisation. En effet, notre approche se base sur l'abstraction et le recours à des variations minimales au lieu de renvoyer le document SDP en entier. Alors que pour SIP/SDP, les messages échangés contiennent plusieurs informations redondantes.

La taille des messages de signalisation est considérablement réduite avec notre concept. SIP/SDP se base sur un mécanisme d'offre-réponse, où la totalité du document SDP est échangée même si l'autre bout n'aura besoin que d'une information précise. Nous présentons en Annexe F un exemple (tiré de (28)) d'offre-réponse pour une session audio WebRTC. La totalité de ce message SDP doit être renvoyée avec tout changement dans la session. Par exemple, même après établissement du chemin de la communication, les relais ICE (lignes 17 et 18 de F.1) vont être retransmis avec chaque message, ce qui rendra le message encore plus lourd. Avec notre modèle et pour un changement au cours de la session par exemple, juste les informations permettant d'identifier le contexte du message (identification de la transaction, de l'opération) sont incluses dans le message avec la variation à faire. Nous avons illustré un exemple de message de modification dans 6.9.

Complexité des messages

Les informations transmises dans les messages de signalisation sont des données de forme textuelle, faciles à comprendre et à traiter. Nous proposons que les informations soient structurées de manière hiérarchique et arborescente. La présentation des informations échangées au cours de la signalisation est plus claire et structurée avec notre approche qu'avec SIP/SDP. Le format choisi doit être ouvert et indépendant des langages de programmations. Cette structure arborescente permettra de créer/traiter un changement plus facilement. En effet, le client peut annoncer le changement ou référer au paramètre dont il accepte le changement juste en spécifiant son chemin d'arborescence. Un exemple d'annonce de changement a été présenté dans 6.1.3.

— **Création d'un message / génération d'un message**
la création du texte n'est pas si différente de celle de SDP. Les informations sont indiquées clairement.

— **Traitement/analyse d'un message**
Pour le protocole SIP, la détection du changement au niveau de la réception est complexe. En effet, le fait d'introduire plusieurs lignes d'attributs, dont la signification varie selon le contexte des lignes qui précèdent, rend la compréhension du message plus difficile.
Notre approche propose l'utilisation de messages avec une structure hiérarchique. Le contenu du message devient donc plus facile à comprendre côté humain et à traiter par la machine. Le traitement devrait se faire sans ambiguïté. Nous pouvons introduire au niveau de l'entité

réceptrice un analyseur de texte comme celui intégré avec les navigateurs Web. L'analyse est plus directe et se fait en temps constant plutôt qu'en temps linéaire.

Erreur d'interprétation

Une entité qui ne comprend pas une information peut simplement l'ignorer et continuer l'analyse du reste des informations.

6.3 Conclusion

Dans ce chapitre, nous avons établi une comparaison entre les deux approches. Nous avons montré que le concept que nous proposons permet une facilité de traitement des messages de signalisation puisque les messages sont plus structurés et de taille considérablement réduite. Notre concept permet une diminution de la quantité d'informations transmises et du risque d'erreur d'interprétation. Le protocole SIP offre au cours du processus de négociation, une gestion des problématiques reliées à la connectivité. L'approche que nous proposons ne résout pas ce problème mais le pallie avec l'utilisation du protocole ICE/TURN/STUN. La faiblesse principale de notre approche est de s'écarter d'un modèle standardisé, au risque de compliquer l'interopératibilité. Il faut recréer certains mécanismes de robustesse que SIP offre.

Chapitre 7

Conclusion et perspectives

Le protocole SIP/SDP est devenu un standard de signalisation pour les communications interactives. SIP permet non seulement la gestion des ressources nécessaires à l'établissemet d'une session de communication, mais aussi le contrôle et la terminaison de l'appel. Ce protocole a dû s'adapter à diverses évolutions afin de répondre aux exigences des différents systèmes de communication. En effet, depuis sa conception, le protocole a connu plusieurs ajouts et/ou modifications, au point qu'il est devenu peu maniable.

Notre travail consistait à proposer une nouvelle perspective de signalisation pour les communications en temps réel. Nous nous sommes basés sur le standard SIP, protocole complexe reposant sur le format de description SDP. Nous avons restructuré l'information contenue dans SDP et l'avons adapté à des modèles qui ne sont pas nécessairement P2P. En effet, les communications interactives embarquées tendent à contourner le modèle trapézoïdale, basé sur l'offre-réponse. Nous nous sommes donc focalisés sur l'étude de scénarios suivants un modèle client-serveur.

Dans un premier temps, nous avons introduit quelques notions préliminaires nécessaires à la compréhension de ce document. Ainsi, nous avons défini, dans le deuxième chapitre, ce qu'est la signalisation. Nous avons introduit le protocole de description des sessions multimédia SDP et avons présenté le mécanisme de négociation utilisé, communément appelé l'offre-réponse. Une analyse du protocole SDP nous a permise de déceler plusieurs faiblesses. Nous avons regardé brièvement les modèles de communication les plus communs et avons vu ceux qui sont adoptés par l'environnement WebRTC. Ensuite, nous avons décrit, lors du troisième chapitre, une communication en temps réel

selon un cas d'étude choisi. Un serveur de signalisation a été simulé afin de permettre d'extraire l'ensemble des informations requises à l'établissement d'un appel. Après avoir décrit ce scénario, nous avons illustré le concept dans le quatrième chapitre. Nous avons présenté les échanges entre les entités et avons développé quelques messages de signalisation. Le cinquième chapitre a porté sur une discussion de la flexibilité de l'approche proposée, à travers une étude brève de quelques cas d'étude. Le but étant de prouver la souplesse du concept déjà décrit. Dans le dernier chapitre, nous avons comparé SIP/SDP et l'approche que nous proposons.

Notre étude nous a permis d'examiner de plus près le protocole de signalisation SIP. Ce protocole utilise un format de description, à la base déclaratif et qui, par le biais de l'offre-réponse, est utilisé dans un contexte interactif. Nous proposons un format mieux adapté et plus structuré. Nous nous sommes concentrés sur l'étude des modèles de communication reposant sur une infrastructure de base. Nous avons donc supposé un ensemble de simplifications qui peuvent être offertes par le modèle client-serveur. Le nouveau concept est plus approprié aux services OTT. Notre format propose des fonctionnalités qui ne sont pas supportées par SDP. En effet, SDP ne fournit pas des informations étendues sur les caractéristiques des terminaux clients. À titre d'exemple, l'option sur l'orientation du terminal n'a été ajoutée que récemment (29). D'autres aspects peuvent être explorés pour étendre ce travail. Une implémentation pratique de différents scénarios, selon notre concept et SIP, est un horizon possible à considérer. Ceci permettra une comparaison plus fiable entre les deux approches. Aussi, il s'avère intéressant d'étudier la faisabilité de combiner les deux approches. En effet, pour le cas des centres d'appel par exemple, où une infrastructure de téléphonie traditionnelle est déjà présente et implémentée, nous pouvons discuter des éléments requis pour coordonner les deux mécanismes de signalisation. En utilisant SIP d'une extrémité et notre approche de l'autre.

Références

[1] https://webrtc.org/ [Dernier accès le 11 Novembre 2016]

[2] J. Jang-Jaccard , S. Nepal, B.Celler, and B. Yan. “WebRTC-based video conferencing service for telehealth”, Computing, 98(1-2), 169-193, 2016.

[3] A. B. Johnston, D. C. Burnett. “WebRTC: APIs and RTCWEB protocols of the HTML5 real-time web”, Digital Codex LLC, 2012.

[4] J. C. Grégoire. “On embedded real-time media communications”, Proceedings of the 1st Workshop on All-Web Ral-Time Systems, ACM, April 2015.

[5] B. Goode. “Voice over internet protocol”, Proc. of the IEEE, Vol. 90, No.9, September 2002.

[6] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. “SIP: Session Initiation Protocol”, RFC 3261, IETF, June 2002.

[7] M. Handley, V. Jacobson. “SDP: Session Description Protocol.” RFC 2327, IETF, April 1998.

[8] R. Hancock, G. Karagiannis, J. Loughney and S. Van den Bosch, “Next Steps in Signaling (NSIS): Framework”, IETF, RFC 4080, June 2005.

[9] M. Femminella, R. Francescangeli, G. Reali and H. Schulzrinne, “Gossip-based signaling dissemination extension for next steps in signaling”, 2012 IEEE Network Operations and Management Symposium. IEEE, 2012. p. 1022-1028.

[10] J. Rosenberg, H. Schulzrinne. “An Offer/Answer Model with the Session Description Protocol (SDP).” RFC 3264, IETF, June 2002.

[11] https://webrtchacks.com/sdp-anatomy/ [Dernier accès le 11 Novembre 2016]

[12] J. Uberti, C. Jennings, and E. Rescorla, "Javascript Session Establishment Protocol", draft-ietf-rtcweb-jsep-12 (work in progress), October 2015.

[13] A. Johnston, R. Sparks. “Session Description Protocol (SDP) Offer/Answer Examples.” RFC 4317, IETF, December 2005.

[14] J. Polk, S. Dhesikan, and G. Camarillo. “Quality of Service (QoS) Mechanism Selection in the Session Description Protocol (SDP).” RFC 5432, IETF, March 2009.

[15] https://webrtcstandards.info/end-of-sdp-in-webrtc/ [Dernier accès le 11 Novembre 2016]

[16] Van Kesteren, A., Jackson, D. (2007). The xmlhttprequest object. World Wide Web Consortium, Working Draft WD-XMLHttpRequest-20070618.

[17] I. Fette, A. Melnikov. “The websocket protocol.” RFC 6455, IETF, December 2011.

[18] https://flask-socketio.readthedocs.org/en/latest/ [Dernier accès le 11 Novembre 2016].

[19] http://www.w3.org/TR/html5/ [Dernier accès le 17 Novembre 2016].

[20] https://www.w3.org/TR/webrtc [Dernier accès le 17 Novembre 2016]

[21] M. Handley, V. Jacobson, and C. Perkins. “SDP: Session Description Protocol”, RFC 4566, IETF, July 2006.

[22] Internet Assigned Numbers Authority (IANA) “Session Description Protocol (SDP) Parameters”, www.iana.org.

[23] I. Johansson, K. Jung. “Negotiation of Generic Image Attributes in the Session Description Protocol (SDP)”, RFC 6236, IETF, May 2011.

[24] A. Johnston, R. Sparks, C. Cunningham, S. Donovan, K. Summers. “Session Initiation Protocol Service Examples.” RFC 5359, IETF, October 2008.

[25] A. Johnston, S. Donovan, R. Sparks, C. Cunningham, K. Summers. “Session Initiation Protocol (SIP) Basic Call Flow Examples.” RFC 3665, IETF, December 2003.

[26] https://googleblog.blogspot.ca/2016/08/meet-google-duo-simple-1-to-1-video.html [Dernier accès le 17 Novembre 2016]

[27] J. Rosenberg. “SIP Early Media”, IETF Standard-Working-Draft, Internet Engineering Task Force, Jul. 13, 2001, pp. 1-23.

[28] S. Nandakumar, C. Jennings. “SDP for the WebRTC”, draft-nandakumar-rtcweb-sdp-02 (work in progress), July 2016.

[29] J. C. Grégoire. “SDP-based Signalling and OTT Services: History and Perspective on an Evolving Trend”, 19th International ICIN Conference - Innovations in Clouds, Internet and Networks - March 1-3, 2016, Paris.

[30] J. Franks, P. Hallam-Baker, J. Hostetler, S. Lawrence, P. Leach, A. Luotonen, and L. Stewart. “HTTP Authentication: Basic and Digest Access Authentication”, RFC 2617, IETF, June 1999.

[31] http://flask.pocoo.org/ [Dernier accès le 11 Novembre 2016]

[32] S. Pfeiffer, C. Pearce. “The definitive guide to HTML5 video.” Apress, 2010.

Annexe A

Authentification par défi

A.1 Description de l'algorithme

Le mode d'authentification par défi (challenge-response authentication) est un mécanisme largement adopté dans l'industrie. Ses spécifications ont été définies dans (30). Nous illustrons le principe du mécanisme dans la figure A.1.

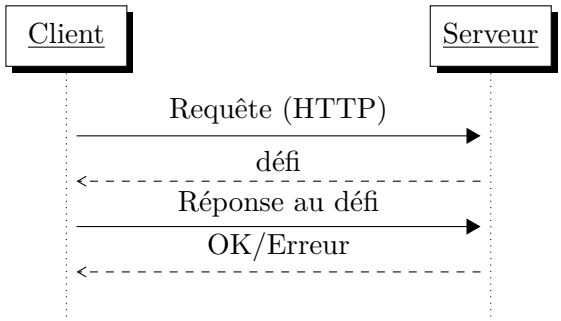


Figure A.1 – Authentification par défi

1. Pour pouvoir accéder à une ressource protégée, le client envoie une requête au serveur.
2. Le serveur reconnaît que la demande provenant du client requiert une authentification. Il répond à la requête du client par un défi. Ce dernier est généralement une chaîne de caractères aléatoires générée par le serveur.
3. Le client envoie une réponse au défi. Cette réponse combine le défi reçu avec un NIP qui est connu par les deux entités. Le but de cette combinaison est de ne pas divulguer l'information sur le réseau.
4. Le serveur traite la réponse reçue et envoie l'état au client (permission accordée ou erreur).

A.2 Implémentation du mécanisme (avec Python)

Définition des fonctions utilisées pour l'authentification

```
import hashlib
import string, random

def getchallenge():
    #genererer un 'String' aleatoire:
    challenge = map(lambda i: chr(random.randint(0, 255)), range(16))
    return "".join(challenge)

def getresponse(password, challenge):
    m = hashlib.md5()
    m.update(password.encode('utf-8'))
    m.update(challenge.encode('utf-8'))
    return m.digest()
```

— **Étape 1:** le client se connecte, il envoie une requête au serveur;

— **Étape 2:** le serveur génère un défi (challenge);

```
    challenge = getchallenge()
    print("Serveur:", repr(challenge))
```

— **Étape 3:** le client combine le défi reçu avec son mot de passe et fait le calcul de la réponse à transmettre;

```
    client_response = getresponse("client_password", challenge)
    print("Client:", repr(client_response))
```

— **Étape 4:** le serveur fait de même pour créer une réponse, elle compare par la suite sa réponse avec celle reçue du client. Elle donne enfin le résultat de cette comparaison.

```
    central_response = getresponse("client_password", challenge)
    if central_response == client_response:
        print ("Serveur:", "Login OK")
        // Notifier le client
```

Annexe B

Vérification de la présence

L'opération de vérification de la présence, même si non liée directement à la connexion, pourrait être gérée par la centrale. Au cours de l'initiation de la connexion, la centrale a besoin de vérifier la présence du client avant d'acheminer une demande de connexion . Pour vérifier les présences, la centrale doit au préalable garder une table des clients authentifiés. Nous entendons par vérifier la présence d'un client, voir son statut, s'il est connecté à la centrale ou non. Nous présentons dans la figure B.1 le séquencement de messages pour cette opération. Nous supposons que le client A connait au préalable le nom d'usager du client B.

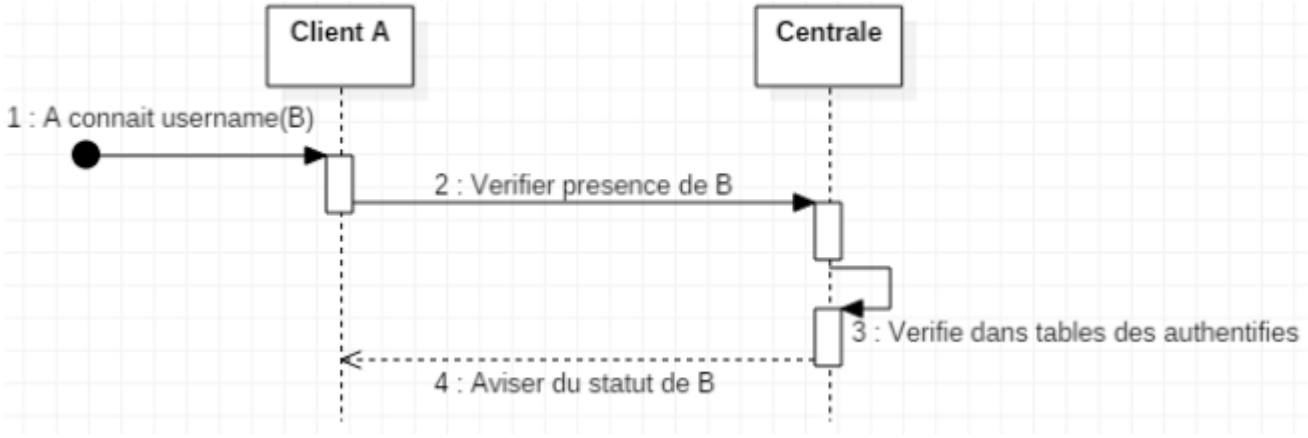


Figure B.1 – Diagramme de séquences pour la vérification des présences

Annexe C

Indice de retransmission

C.1 Intérêt de l'indice de retransmission

C.1.1 Côté récepteur de la requête

Dans la figure C.1, le client initie une requête et attend un certain délai prescrit. Si aucune réponse n'a été reçue, il retransmet sa requête en incrémentant l'indice de retransmission (RID). Au niveau de la réception, si RID est différent de zéro, la centrale vérifie si un message ayant le même contenu a été déjà traité. Si c'est le cas, la centrale n'a pas besoin de traiter encore une fois le message reçu; elle envoie la réponse déjà envoyée précédemment. Le fait d'avoir inclu cet indice réduit le temps (et le nombre) de traitement des requêtes au niveau de la centrale.

C.1.2 Côté initiateur de la requête

Dans la figure C.2, après avoir attendu un certain délai, le client initiateur de la requête suppose que sa requête initiale n'a pas été reçue. Le client retransmet donc sa requête en incrémentant RID. À la réception, la centrale traite les requêtes reçues et en envoie les réponses. Après avoir reçu la réponse au premier message, le client peut décider de ne pas considérer la deuxième réponse puisque le contenu des deux réponses est le même.

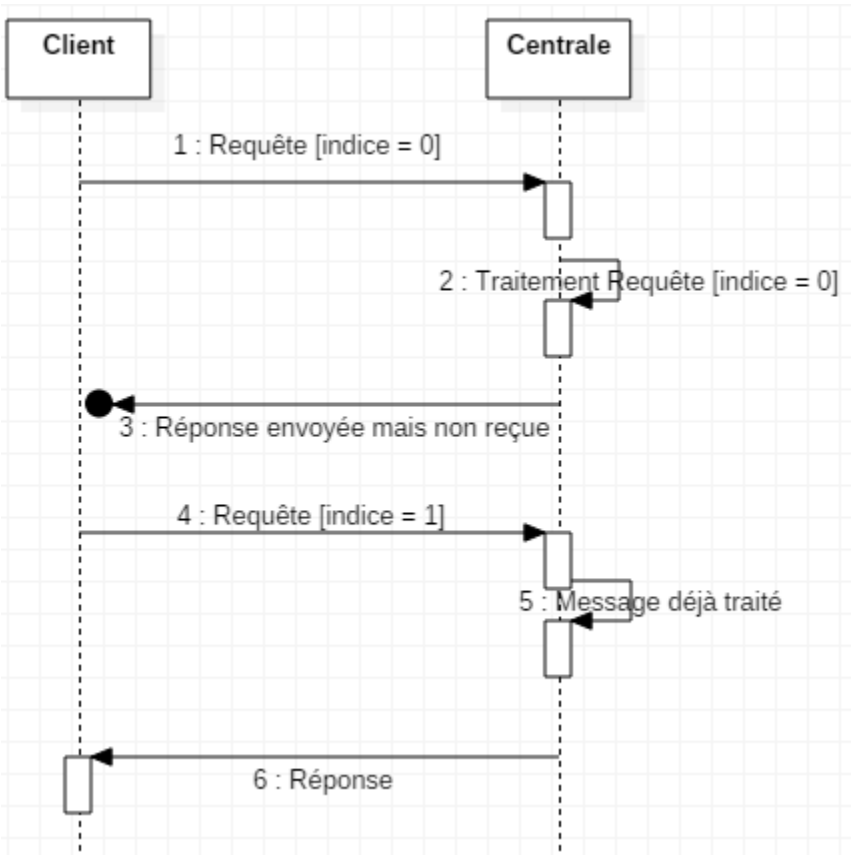


Figure C.1 – Intérêt du RID au niveau du récepteur

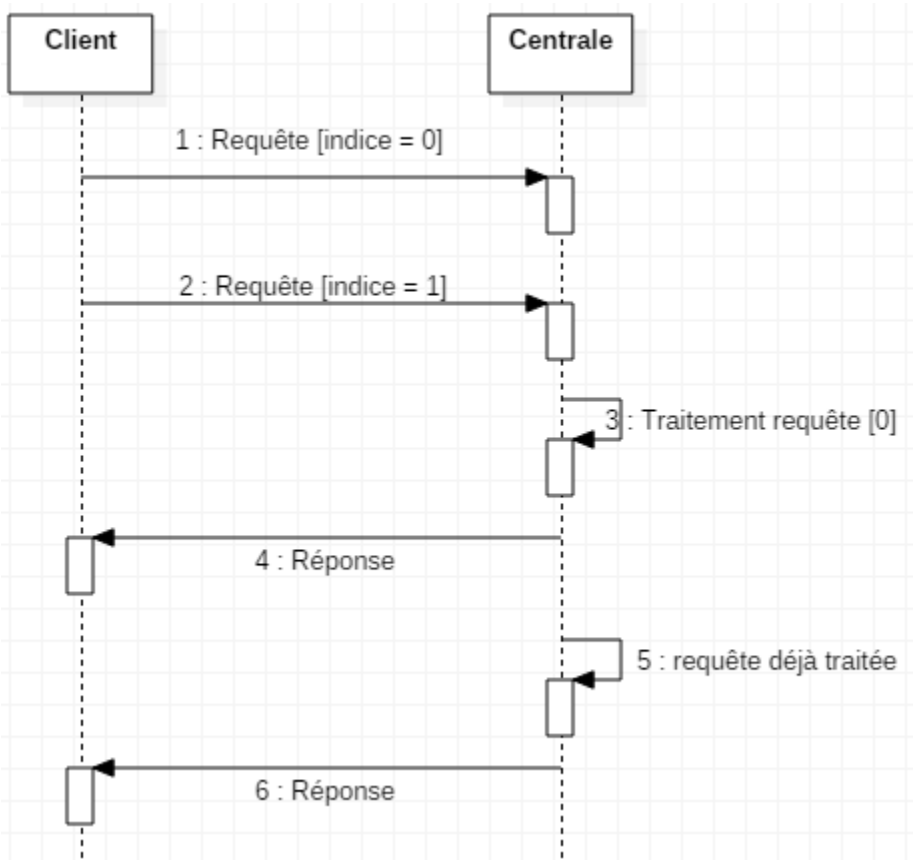


Figure C.2 – Intérêt du RID au niveau de l'émetteur

Annexe D

Simulation du scénario

Nous présentons quelques éléments de simulation d'une communication entre un client et une centrale.

D.1 Configuration de la centrale

1. Importer les bibliothèques qui seront utilisées, principalement Flask (31) et socketIO (18):

```
import time
import MySQLdb
from threading import Thread
try:
    import json
except ImportError:
    import simplejson as json

from flask import Flask, render_template, session,
                request, jsonify
from flask_socketio import SocketIO, emit, join_room,
        leave_room, close_room, rooms, disconnet
```

2. Créer une instance de la classe Flask:

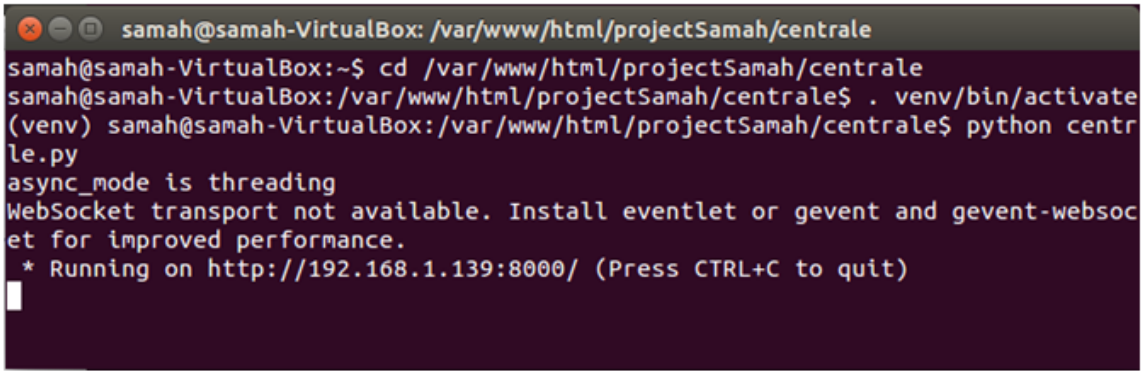
```
    app = Flask(__name__)
```

3. Spécifier le chemin de la page:

```
    @app.route('/')
    def index():
        global thread
        // du code //
        return render_template('index.html')
```

4. Centrale démarrée et en écoute, on spécifie l'adresse IP et le port d'écoute:

```
    if __name__ == '__main__':
        socketio.run(app, "192.168.1.139", port=8000)
```



D.2 Configuration du client

Le client accède à l'adresse 192.168.1.139:8000

Identification des moyens

Un code JavaScript est exécuté afin de détecter les codecs média disponibles. Nous présenons en listing 1 un exemple du code utilisé pour extraire la liste des codecs audio (code tiré de (32)).

Listing 1:

```
// pour les codecs audio
var types = new Array();
types[0] = "audio/ogg";
types[1] = "audio/ogg; codecs=vorbis";
types[2] = "audio/mpeg";
types[3] = "audio/mpeg; codecs=mp3";
types[4] = "audio/wav";
types[5] = "audio/wav; codecs=1";
types[6] = "audio/mp4";
types[7] = "audio/mp4; codecs=aac";
types[8] = "audio/x-m4b";
types[9] = "audio/x-m4b; codecs=aac";
types[10] = "audio/x-m4p";
types[11] = "audio/x-m4p; codecs=aac";
types[12] = "audio/aac";
types[13] = "audio/aac; codecs=aac";
types[14] = "audio/x-aac";
types[15] = "audio/x-aac; codecs=aac";
// creer un element audio
var audio = document.createElement('audio');
// tester les types et creer une liste de codecs supportes
var codecs_audio = new Array();
for (i=0; i<types.length; i++){
    var support = audio.canPlayType(types[i]);
    if (support == "probably"){
        codecs_audio.push(types[i]);
    }
    if (support == "maybe"){
        codecs_audio.push(types[i]);
    }
}
```

- génération d'un identifiant client. Ce dernier prépare les informations à envoyer à la centrale:

```
// choix d'un identifiant de la session
ID_session = Math.random().toString(36).substring(7);
var description = {'IDdeSession':ID_session,
    'audio':codecs_audio, 'video':codecs_video}
var JSONObject = JSON.stringify(description);
```

- création d'un objet 'EventEmitter' au niveau du client:

```
    // the socket.io documentation recommends sending an explicit
    //package upon connection, this is specially important when
    //using the global namespace
    var socket = io.connect('http://'+ document.domain + ':'
        + location.port + namespace);
```

- un auditeur (listener) à cet événement est joint au niveau de la centrale:

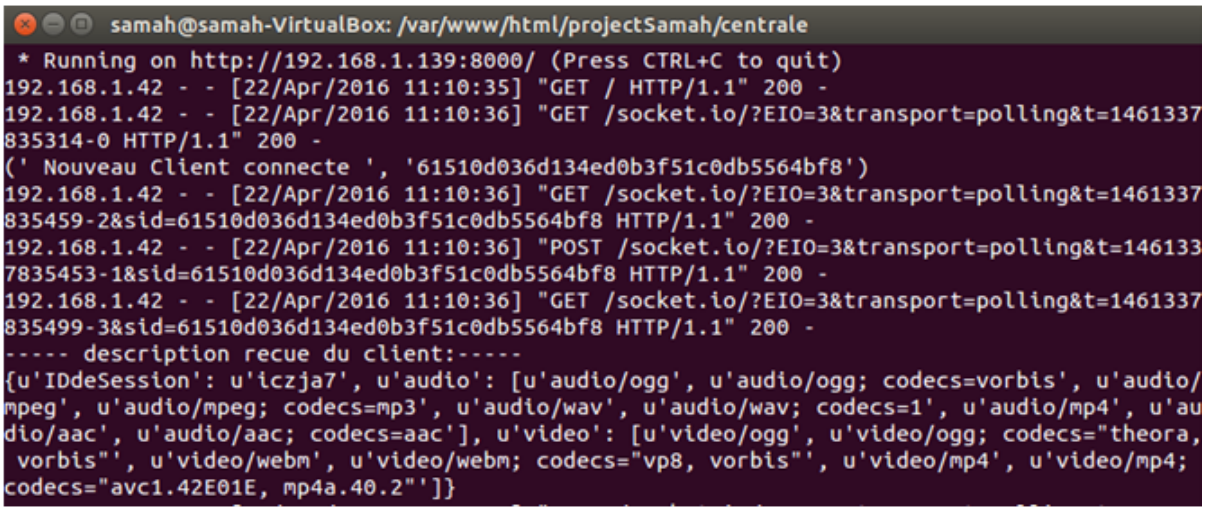
```
    @socketio.on('connect', namespace='/test')
    def test_connect():
        print("Nouveau client connecte", request.sid)
        # ici on recupere l'ID du socket
        currentSocketId = request.sid
        list_sockets.append(currentSocketId)
        emit('my response', {'data': 'Vous etes connecte',
            'count':0})
```

- la centrale répond avec un message de confirmation de l'ouverture de session, ce message sera affiché sur la page du client.

- le client répond par la suite avec sa description qu'il a regroupé en format JSON:

```
    // event handler for new connections
    socket.on('connect', function(){
        socket.emit('my event', {data: JSONObject});
    });
```

- la centrale reçoit l'objet JSON contenant l'information sur les médias disponibles, chacun avec les codecs offerts:



- la centrale a l'information sur la description du client, elle récupère l'ID de la session et prépare une liste des sessions actives:

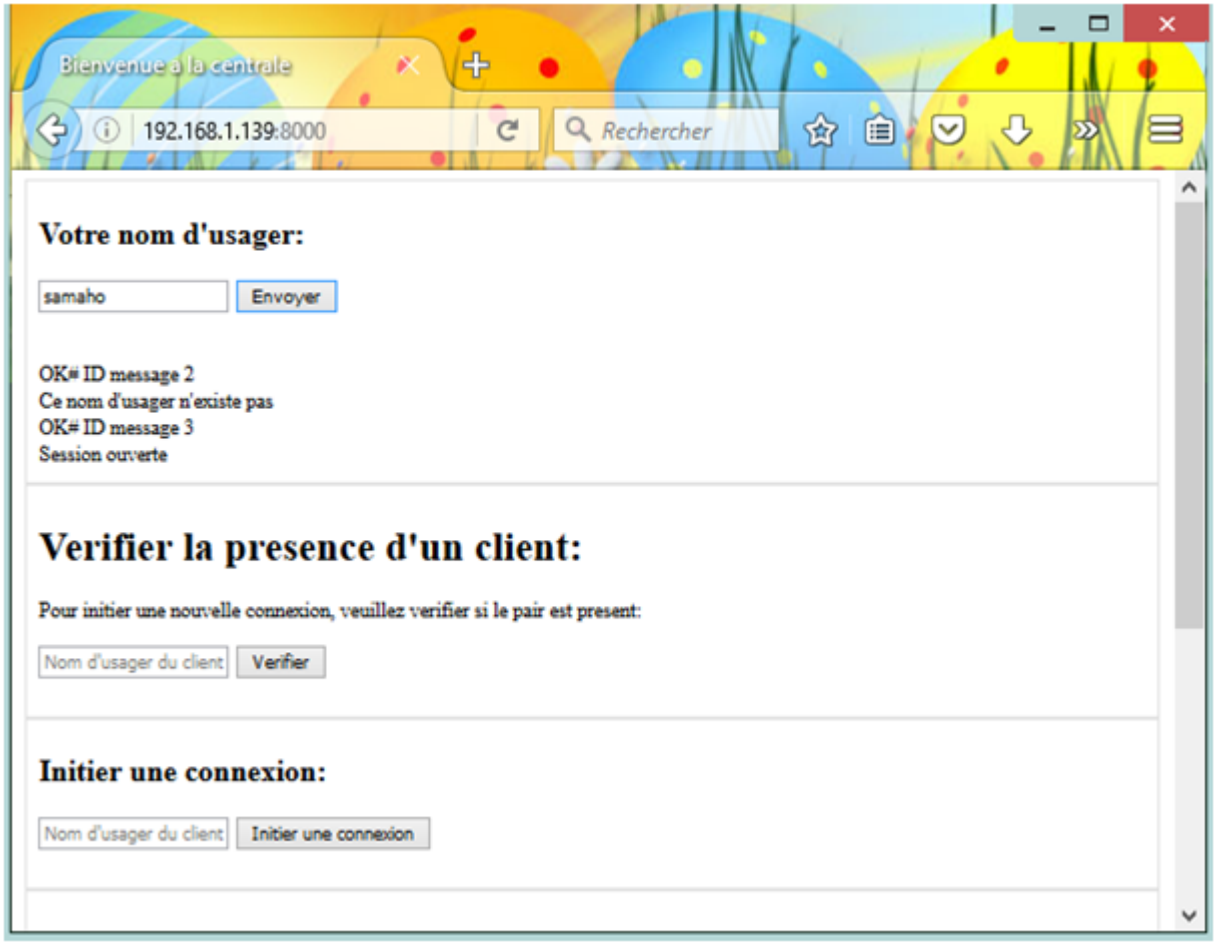
```
    @socketio.on('my␣event', namespace='/test')
    def test_message(message):
        session['receive_count'] = session.get('receive_count',0) + 1
        description_client = json.loads(message['data'])
        print('------␣description␣recue␣du␣client:␣------')
        print(description_client)
        ID_session = description_client['IDdeSession']
        sessions_actives.append(ID_session)
        emit('my␣response',
            {'data':message['data'], 'count': session['receive_count']})
```

Ouverture de session / authentification client

Processus de vérification de l'authentification du client:

```
    @socketio.on('my␣event2', namespace='/test')
    def get_username(message):
        session['receive_count'] = session.get('receive_count',0) + 1
        usernam = message['data']
        # connect to database
        db = MySQLdb.connect(host="localhost", user="samah",
             passwd="****", db="save_form")
        # create a cursor object. It will let you execute
        # all the queries you need
        cur = db.cursor()
        # use the SQL command
        cur.execute("SELECT␣username␣FROM␣abonne")
        data = cur.fetchall()
        for row in data:
            liste_username.append(row[0])
        db.close()
        if usernam in liste_username:
            global myusername
            print('element␣trouve␣dans␣la␣BD')
            clients_authentifies.append(usernam)
            myusername = usernam
            mydict[str(myusername)] = request.sid
            print(mydict)
            emit('my␣response␣user',{'data':'Session␣ouverte',
                'count':session['receive_count']})
        else:
            print("element␣n'existe␣pas")
            emit('my␣response␣user',{'data':'Ce␣nom␣d␣usager
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣n␣existe␣pas', 'count':session['receive_count']})
```

Le client est avisé par la suite que la session est ouverte, il peut maintenant vérifier si un client est présent et initier de nouvelles connexions:



Rupture de session

Le client peut se déconnecter de la centrale en cliquant sur le bouton 'Fin de session' qui fait appel au code JavaScript suivant:

```
    $('form#disconnect').submit(function(event){
      socket.emit('disconnect request', {data: $('#emit_data').val()});
      return false;
    });
```

Cet événement déclenche au niveau de la centrale la requête de déconnexion suivante:

```
@socketio.on('disconnect request', namespace='/test')
def disconnect_request(message):
    session['receive_count'] = session.get('receive_count',0)+1
    username_deconnect = message['data']
    emit('my response',
         {'data':'Disconnected!','count':session['receive_count']})
    print('Deconnecte')
    clients_authentifies.remove(username_deconnect)
    disconnect()
```

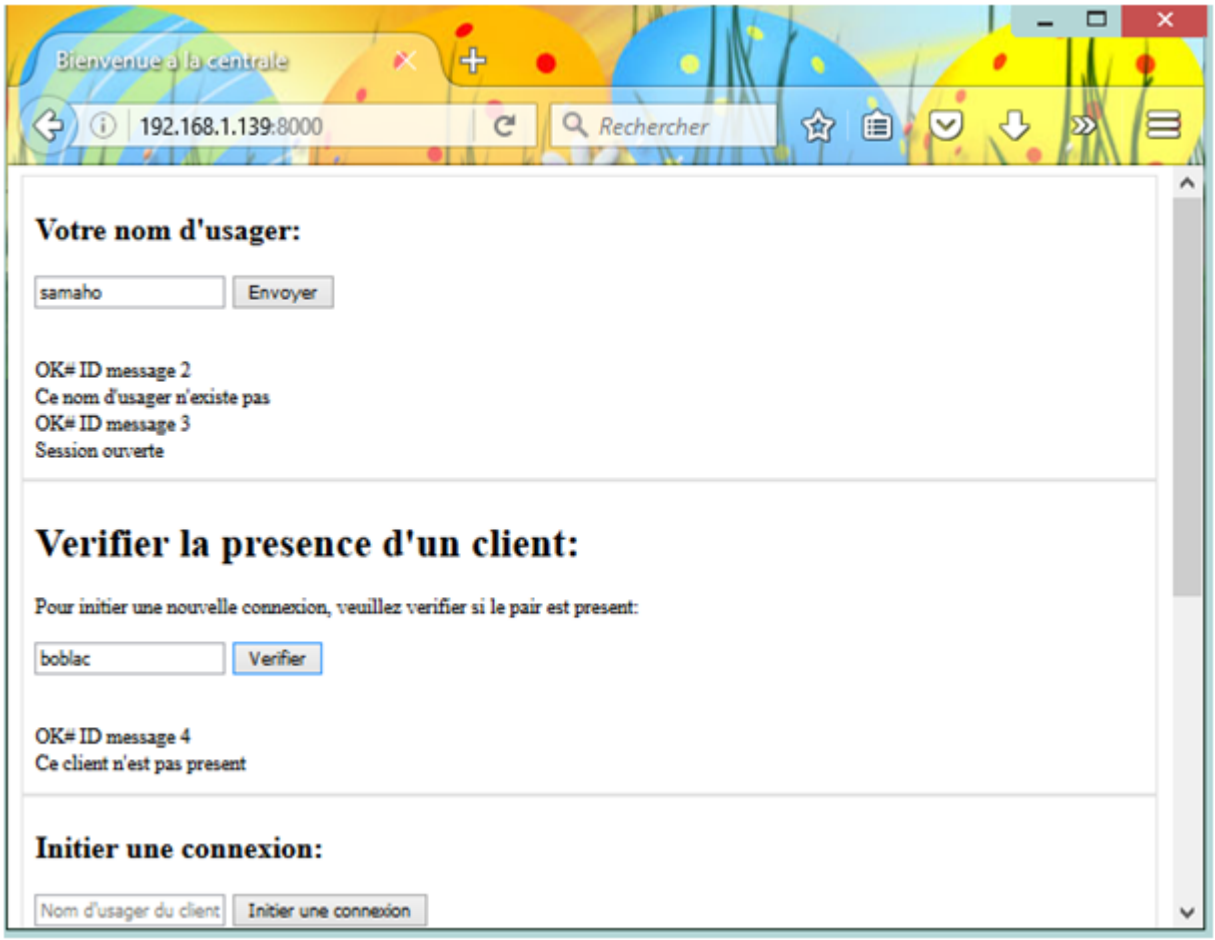
La centrale a maintenant l'information sur le client déconnecté:

```
('Client disconnected', '9a8c5419836545b99602dbb2bafa27d2')
192.168.1.42 - - [22/Apr/2016 14:37:46] "POST /socket.io/?EIO=3&transport=polling&t=146135
0266023-14&sid=9a8c5419836545b99602dbb2bafa27d2 HTTP/1.1" 200 -
192.168.1.42 - - [22/Apr/2016 14:37:46] "GET /socket.io/?EIO=3&transport=polling&t=1461350
260126-13&sid=9a8c5419836545b99602dbb2bafa27d2 HTTP/1.1" 200 -
192.168.1.42 - - [22/Apr/2016 14:37:46] "POST /socket.io/?EIO=3&transport=polling&t=146135
0266094-15&sid=9a8c5419836545b99602dbb2bafa27d2 HTTP/1.1" 400 -
```

Elle confirme au client sa déconnexion en lui émettant un message qui s'affiche sur sa page. Toutes les autres opérations vont être dorénavant inaccessibles.

Vérifier une présence

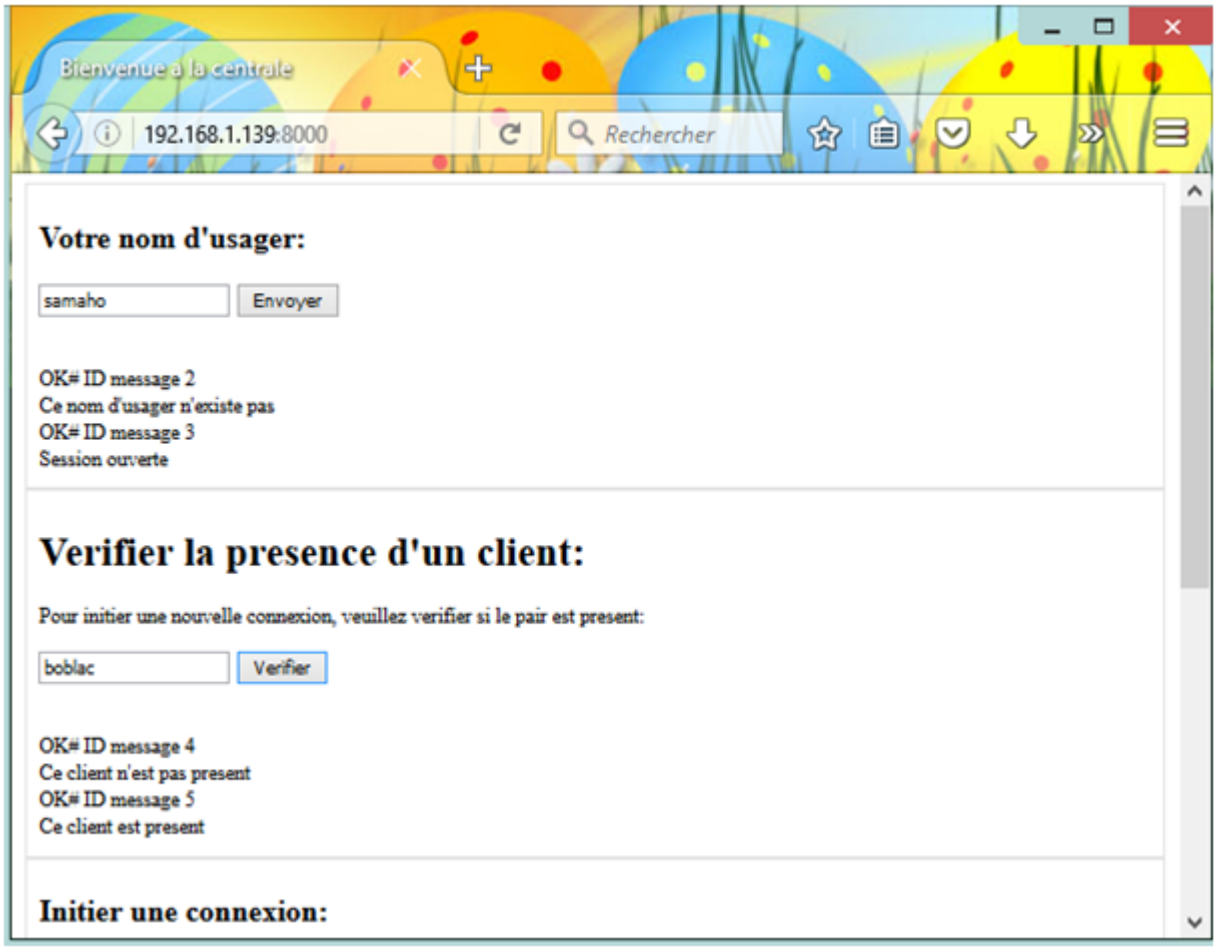
Un client demande la présence d'un autre client:



La centrale reçoit la demande de vérification de la présence d'un client:

```
    @socketio.on('my␣event␣presence', namespace='/test')
    def test_message_pre(message):
        session['receive_count'] = session.get('receive_count',0)+1
        print('Demande␣de␣presence␣recue:')
        demande = message['data']
        if demande in clients_authentifies:
            emit('my␣response␣presence',
            {'data':'Ce␣client␣est␣present',
               'count':session['receive_count']})
        else:
            emit('my␣response␣presence',
            {'data':'Ce␣client␣n'est pas present',
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣'count':session['receive_count']})
```

Elle vérifie et envoie la réponse; ici le client a demandé deux fois la présence du client 'boblac' qui n'était pas encore connecté durant la première demande de présence (message ID : 4).



Initier une connexion

Pour initier une connexion, le client doit entrer le nom d'usager du client avec le quel il veut initier une nouvelle connexion.
On vérifie d'abord si des codecs audio ont été détectés, si la liste des codecs est vide alors la demande d'initiation de la connexion va être refuser. Sinon, le client prépare un objet à envoyer à la centrale, contenant:

— l'adresse IP: c'est une simulation d'un cas où les clients ne sont pas derrière un NAT, on peut laisser la centrale découvrir l'adresse IP du client initiateur.
— le nom de la personne à appeler;
— le média choisi par défaut qui est vidéo (partage du micro et de la caméra);
— le port d'écoute.

```
        $('form#emitconnexion').submit(function(event) {
            // verifier si il a micro
            if (codecs_audio == []){
                document.getElementById("testcodec").innerHTML =
                  "Aucun codec disponible pour etablir un appel";
            }else{
            //prepare adresse IP, appelant, appele, port, media
                caller = $('#emit_data').val();
                calling = $('#ask_new_connexion').val();
                myIPadress = IPaddr;
                port = myport;
                media_to_share = media;
                // make a json format of those informations
                var json_initiate = {'Caller':caller,'Calling':calling,
                    'MyIPadress':myIPadress, 'Port':port,
                    'Media_to_share':media_to_share }
                var JSONObject_initiate = JSON.stringify(json_initiate);
                // send json format with event
                socket.emit('my event connexion',
                    {data: JSONObject_initiate} );
                return false;
            }
        });
```

La centrale récupère les informations envoyées du client:

```
    @socketio.on('my event connexion', namespace='/test')
    def test_message_cnnx(message):
    demande_connexion = json.loads(message['data'])
    print(demande_connexion)
    # la centrale verifie si la personne a appeler est toujours presente
    if demande_connexion['Calling'] in clients_authentifies:
        emit('my response_initiate_connexion',
        {'data':'client a appeler est disponible',
            'count': session['receive_count']})
        callingId = mydict.get(myusername)
        print('ID du socket a appeler',callingId)
        if callingId in list_sockets:
           listes_clients_appeles = []
           listes_clients_appeles.append(callingId)
           for item in listes_clients_appeles:
              emit('tester', {'data':"Vous avez: demande de connexion",
                    'count':session['receive_count']});
    else:
        print("impossible d'acheminer cette demande, client",
            demande_connexion['calling'], "n'est pas present" )
```

Annexe E

Avant la communication

Avant que deux clients puissent entrer en communication, deux étapes sont nécessaires. Le client doit s'inscrire au niveau de la centrale et ouvrir par la suite une session.

E.1 Inscription

La figure E.1 présente les échanges entre les différentes entités lors de l'inscription.

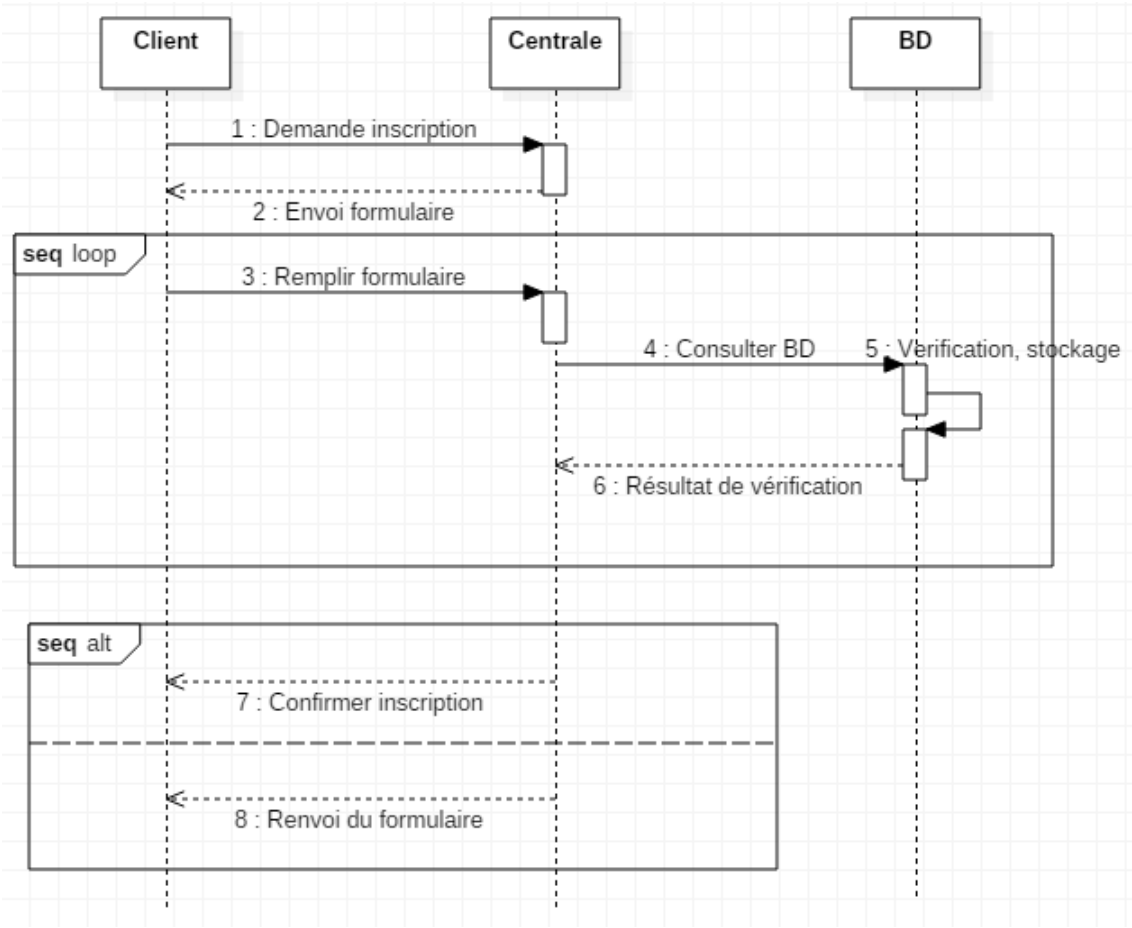


Figure E.1 – Diagramme de séquences pour l'inscription

Le client accède à la page web de la centrale où il a la possibilité de demander une inscription. L'inscription est obligatoire pour tous les usagers qui veulent ouvrir une session. Une fois que la demande est faite (bouton d'inscription cliqué), le client reçoit le formulaire d'inscription. Il remplit donc et envoie le formulaire à la centrale.
Les informations d'inscription sont: le nom et le prénom du client, son émail, le nom d'usager et mot de passe (ces informations sont requises). Dans notre cas et pour illustration, le formulaire propose de fournir d'autres informations telles que l'adresse du client et sa date de naissance.
Nous présentons en listing E.1 un exemple de message d'envoi d'informations d'inscription qui est équivalent au message 3 du diagramme E.1. Le message est de type requête et dont l'action est « register ». Le corps du message spécifie l'ensemble des informations remplies dans le formulaire d'inscription.
L'indice de retransmission « RID » indique 0 puisque la requête est initiale. Si aucune réponse n'est reçue, le client pourra retransmettre le même message en incrémentant cet indice.
À noter que le mot de passe peut être transmis de manière plus sécure.
Les informations contenues dans « divers » représentent des données facultatives mais qui peuvent être demandées et sauvegardées.

Listing E.1 – Message d'inscription

```
{
  h:{
      ID:{
          TID: 1,
          RID:0
      },
      type:"R",
      a:"register"
  },
  d:{
    usager:{
      username: "Boblac",
      prenom: "Bob",
      nom: "Lacroix",
      email: "bob@inrs.ca",
      password: "********",
      divers: {
        Birth_date: "14-5-1980",
        Address: {
        }
      }
    }
  }
}
```

Après réception du message d'inscription, la centrale exécute une boucle de vérification pour s'assurer que le client a entré tous les champs requis à l'inscription et que les informations entrées ne correspondent pas à un compte déjà existant. La centrale envoie une requête à la base de données (BD) pour s'assurer de l'unicité du nom d'usager. Si ce dernier figure déjà

dans la BD, alors un message d'erreur est affiché au client avec une possibilité de remplir de nouveau le formulaire. Sinon, les informations sont stockées dans la BD et on redirige le client à la page d'authentification. Les seules informations qui seront sauvegardées de manière persistante sont celles relatives à l'inscription de l'abonné. Les autres informations échangées entre le client et la centrale seront propres à la session. Le choix de la base de données peut varier selon le nombre d'abonnés prévu.

E.2 Authentification / Ouverture de session

Après inscription, le client pourra s'authentifier à tout moment. Le but principal est de vérifier l'identité du client. Nous utilisons une authentification minimale par « défi-réponse ». La figure E.2 présente le diagramme de séquencement de cette opération.

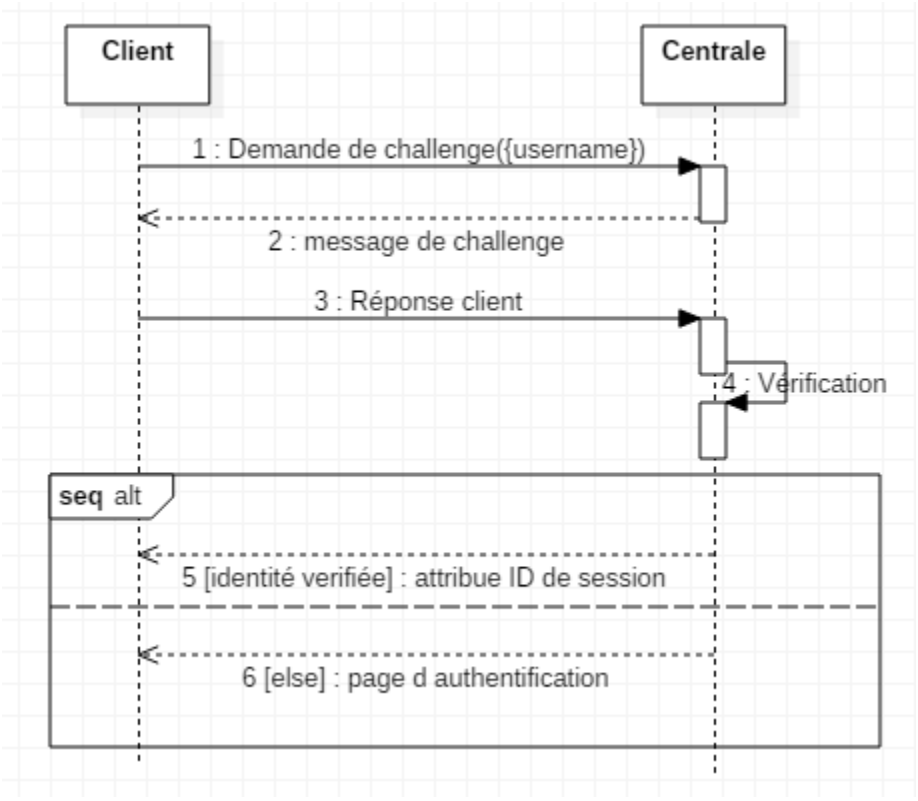


Figure E.2 – Diagramme de séquences pour l'ouverture de session

Les deux entités (client - centrale) utilisent durant ce mécanisme, de plus que le mot de passe, un message de défi (un NIP) pour renforcer la sécurité.
Le listing E.2 présente un exemple de message de demande d'ouverture de session, équivalent au message 1 du diagramme E.2.

Listing E.2 – Demande d'ouverture de session

```
{
  h:{
      ID:{
          TID: 3,
          RID:0
      },
      type:"R",
      a:"login"
  },
  d:{
      username:"Boblac"
  }
}
```

La centrale répond avec un message de défi, correspondant au message 2 de la figure E.2:

Listing E.3 – Envoi du défi

```
{
  h:{
      ID:{
          TID: 3
      },
      type:"rep",
      St:[407, "Authentication required"]
  },
  d:{
      challenge:"un_defi"
  }
}
```

Le client envoie sa réponse au défi:

Listing E.4 – Réponse au défi

```
{
  h:{
      ID:{
          TID: 3,
          RID:0
      },
      type:"R",
      a:"login"
  },
  d:{
      reponse:"reponse_client"
  }
}
```

La centrale donne suite à la demande du client (avec un message de succès ou d'échec d'ouverture de la session):

Listing E.5 – Cas de succès d'ouverture de la session

```
{
  h:{
      ID:{
          TID: 3
      },
      type:"rep",
      St:[200, "successful authentication"]
  },
  d:{
      SID:12455565
  }
}
```

Listing E.6 – Cas d'échec d'ouverture de la session

```
{
  h:{
      ID:{
          TID: 3
      },
      type:"rep",
      St:[401, "authentication required"]
  }
}
```

Annexe F

Exemple d'offre-réponse pour une communication audio WebRTC

Offre SDP:

Listing F.1 – Offre SDP

```
v=0
o=- 20518 0 IN IP4 0.0.0.0
s=-
t=0 0
a=group:BUNDLE audio
a=ice-options:trickle
m=audio 54609 UDP/TLS/RTP/SAVPF 109 0 8
c=IN IP4 24.23.204.141
a=rtcp:60065 IN IP4 24.23.204.141
a=mid:audio
a=msid:ma ta
a=sendrecv
a=rtpmap:109 opus/48000/2
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=maxptime:120
a=ice-ufrag:074c6550
a=ice-pwd:a28a397a4c3f31747d1ee3474af08a068
a=fingerprint:sha-256 19:E2:1C:3B:4B:9F:81:E6:B8:5C:F4:A5:A8:D8:73:04:BB:05
  :2F:70:9F:04:A9:0E:05:E9:26:33:E8:70:88:A2
a=setup:actpass
a=rtcp-mux
a=rtcp-rsize
a=rtcp-fb:109 nack
a=extmap:1 urn:ietf:params:rtp-hdrext:ssrc-audio-level
a=extmap:2 urn:ietf:params:rtp-hdrext:sdes:mid
a=ssrc:12345 cname:EocUG1f0fcg/yvY7
a=candidate:0 1 UDP 2122194687 192.168.1.4 61665 typ host
a=candidate:1 1 UDP 1685987071 24.23.204.141 54609
typ srflx raddr 192.168.1.4 rport 61665
a=candidate:0 2 UDP 2122194687 192.168.1.4 61667 typ host
a=candidate:1 2 UDP 1685987071 24.23.204.141 60065
typ srflx raddr 192.168.1.4 rport 61667
a=end-of-candidates
```

Réponse SDP:

Listing F.2 – Réponse SDP

```
v=0
o=- 16833 0 IN IP4 0.0.0.0
s=-
t=0 0
a=group:BUNDLE audio
a=ice-options:trickle
m=audio 49203 UDP/TLS/RTP/SAVPF 109 0 8
c=IN IP4 98.248.92.77
a=rtcp:49203 IN IP4 98.248.92.77
a=mid:audio
a=msid:ma ta
a=sendrecv
a=rtpmap:109 opus/48000/2
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=maxptime:120
a=ice-ufrag:05067423
a=ice-pwd:1747d1ee3474a28a397a4c3f3af08a068
a=fingerprint:sha-256 6B:8B:F0:65:5F:78:E2:51:3B:AC:6F:F3:3F:46:1B:35:DC:B8
  :5F:64:1A:24:C2:43:F0:A1:58:D0:A1:2C:19:08
a=setup:active
a=rtcp-mux
a=rtcp-rsize
a=rtcp-fb:109 nack
a=extmap:1 urn:ietf:params:rtp-hdrext:ssrc-audio-level
a=extmap:2 urn:ietf:params:rtp-hdrext:sdes:mid
a=ssrc:54321 cname:Q/NWs1ao1HmN4Xa5
a=candidate:0 1 UDP 2122194687 192.168.1.7 51556 typ host
a=candidate:1 1 UDP 1685987071 98.248.92.77 49203
  typ srflx raddr 192.168.1.7 rport 51556
a=end-of-candidates
```