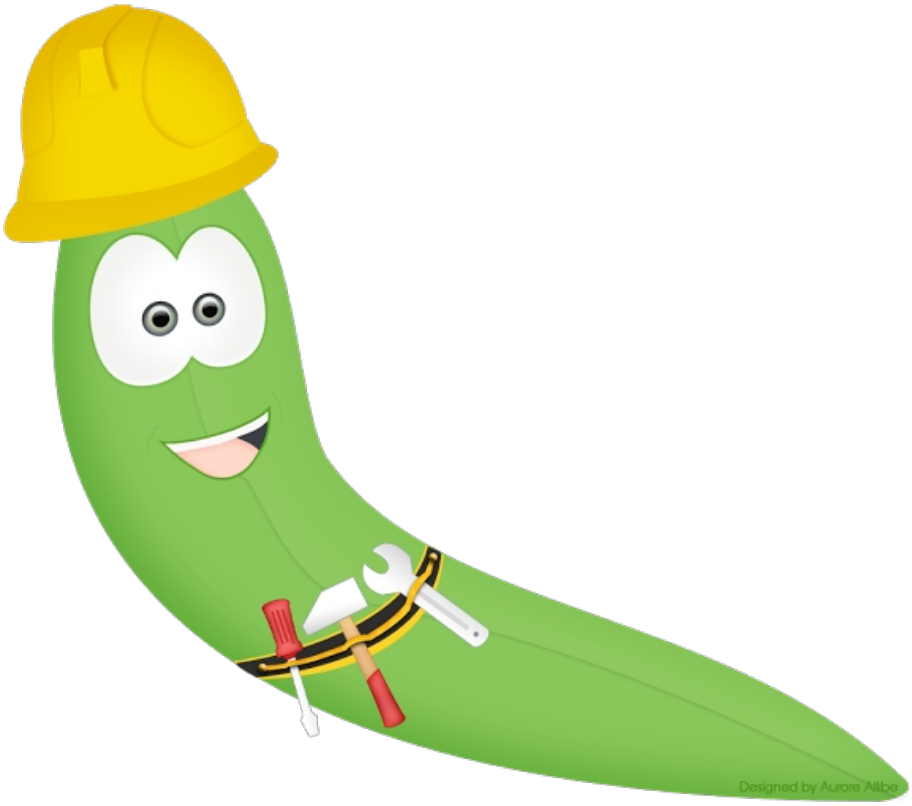


Communications instantanées pour le portail Silverpeas

Rapport de stage,

Master IC2A DCISS RS 2015/2016,

Rémi Pace

Table des matières

Introduction

L'objet de ce stage est le renouvellement de la messagerie instantanée du portail collaboratif Silverpeas.

L'ensemble des travaux présentés ci-dessous sont disponibles en ligne:

- Messagerie instantanée sur un espace de démonstration : https://im.silverpeas.net
- Messagerie instantanée intégrée au portail Silverpeas : https://im.silverpeas.net/silverpeas
- Code source : https://github.com/remipassmoilesel/djoe

Nous allons tout d'abord détailler le contexte du projet, en présentant l'entreprise Silverpeas et son portail collaboratif, puis présenter et justifier les choix technologiques qui ont été fait, pour enfin décrire la chronologie de travail effectuée.

Logiciel libre

L'ensemble des travaux réalisés et détaillés dans ce rapport ont été réalisés à l'aide de projets libres et à sources ouvertes. Systèmes d'exploitations, langages de script et de programmation, logiciels serveurs, infographies, bureautique, navigateurs, environnements de développement intégrés, tous les composants du projet et les outils utilisés sont librement téléchargeables et redistribuables.

Contexte du stage

Présentation de l'entreprise Silverpeas

Silverpeas est une entreprise qui emploie 12 personnes basée à Europole. Elle conçoit un portail collaboratif sous licence libre (GPLv3) et propose des services autour : installation et mise en œuvre, formation à l'utilisation et à l'installation, développement, adaptations graphiques, support. L'entreprise à été fondée en 2000 et a été une des premières a proposer un portail collaboratif en France.

Silverpeas à fait le choix d'utiliser et de promouvoir des technologies libres. L'entreprise propose son portail en téléchargement libre sur Github et sur le site communautaire www.silverpeas.org. Un installeur ainsi qu'une image Docker permettent à des utilisateurs non experts d'essayer la plateforme, ce qui assure une bonne visibilité à l'entreprise et à son produit.

Présentation du portail Silverpeas

Le but du portail collaboratif Silverpeas est d'améliorer les performances des organisations (associations, entreprises, services publics , ...) en optimisant la collaboration et la gestion des connaissances. Pour ce faire, elle fournit clé en main une trentaine d'applications environ:

- indexation et rassemblement d'informations sur les acteurs d'une organisation,
- gestion électronique de documents : édition en ligne non simultanée, versionnage, partage en ligne, indexation ...
- gestion de projet : création d'espaces de travail dédiés, édition de diagrammes de Gantt, ...
- organisation en interne : almanach, quiz, workflow, calendrier de partage de salles et de véhicules, ...
- concertation : enquêtes, sondages, forums, commentaires ...
- communications externes : création de sites web, …

L'entreprise Silverpeas à fait le choix assumé d'utiliser et de proposer du logiciel libre. Cette stratégie permet aux utilisateurs potentiels de pouvoir découvrir la plateforme et son fonctionnement en effectuant une installation de découverte à l'aide d'un installeur et d'une image Docker.

L'entreprise propose d'héberger sa plateforme ou de l'installer sur les serveurs des clients. En effet, bien qu'une installation de découverte soit accessible à un débutant, une installation de qualité production requiert des connaissances techniques avancées et une bonne connaissance du portail Silverpeas.

Public du portail

La plateforme Silverpeas s'adresse avant tout aux organisations de taille moyenne à grande. Le client ayant le plus d'utilisateurs emploie 30 000 personnes. Les domaines d'utilisation sont variés : industrie, santé, recherche technologique ou agricole, conseils généraux, transports publics, ...

Étant donné que le projet est distribué sous licence libre et accessible à tous, et qu'aucunes statistiques ne sont collectées, il est difficile de définir un public exact tant au niveau de l'expérience utilisateur attendue que des matériels et logiciels utilisés. C'est pourquoi l'ergonomie est prise en charge par un ergonome spécialisé et prend en compte les retours des utilisateurs.

Conception modulaire

Le portail est conçu de manière modulaire, avec un module central chargé de la gestion des ressources et de la liaison entre composants, puis plusieurs composants périphériques constituant les applications et services métier de Silverpeas.

Le portail est avant tout destiné à des poste de bureautique et n'a pas d'interface adaptative pour périphériques mobiles. Cependant, une application est disponible.

Silverpeas utilise les technologies Java Édition Entreprise et HTML5. Le portail fonctionne sur un serveur d'application WildFly (la version open source de Jboss EAP).

Intégration système

Une attention particulière à été portée sur le respect des normes et standards. De ce fait, le portail s'intègre sur de nombreuses plate-formes et peut profiter des infrastructures logicielles éventuellement présentes : annuaires LDAP, bases de données métier, ...

Objectifs du stages

L'objectif final du stage est la production d'une messagerie instantanée un pour un intégrée au portail. Cependant, les objectifs fixés évoluent au fur et à mesure de l'avancée du projet.

Objectifs primaires

L'objectif principal du stage est le remplacement de la messagerie instantanée du portail Silverpeas. Un utilisateur du portail doit avoir le moyen de communiquer avec un autre utilisateur en instantané de manière fluide et fiable.

Une première liste de caractéristiques indispensables a été dressée au commencement du stage :

- L'utilisation de technologies et de protocoles standardisés et ouverts
- L'utilisation de technologies compatibles avec la licence GPLv3 du portail Silverpeas
- Une compatibilité maximale avec les navigateurs web courants
- L'utilisation de moyens de communication sécurisés
- La possibilité d'échanger du texte entre deux personnes
- L'utilisation de notifications claires (arrivée de message, ...)
- Une bonne fluidité d'utilisation

Public visé

L'entreprise Silverpeas ne recueille pas de statistiques sur ses utilisateurs. De plus, étant donné que la plateforme est distribuée sous licence GPLv3 il est difficile de déterminer quels sont les utilisateurs cibles, et quels matériels ou logiciels ils utilisent.

Par conséquent, la messagerie devra être compatible avec un maximum de navigateurs différents et à destination d'un public varié.

Existant

L'existant sera peu détaillé car très rudimentaire. La messagerie actuelle du portail Silverpeas permet en théorie l'envoi de messages d'une personne vers une autre, et la notification d'utilisateurs.

Cette messagerie instantanée à été développée ex-nihilo, sans utilisation de framework ou de protocoles existants. Elle à été développée avec les technologie d'alors c'est à dire JSP, HTML et Javascript.

En pratique la messagerie est devenue inopérante, étant donné que la messagerie utilise des fenêtres « pop-up » et que ces dernières sont systématiquement bloquées par les navigateurs modernes. De plus l'intégration de cette messagerie est peu poussée et son accès non évident.

Veille concurrentielle

Une veille concurrentielle à été menée en parallèle de la veille technologique, auprès de professionnels de la collaboration en entreprise mais aussi auprès d'autres systèmes proposant des communications en temps réel.

En tout 12 systèmes on été étudiés en détail, et de nombreux autres ont été survolés : Google +, Hubl'in, Jalios, Citadel, Big Blue Button … Le résultat de l'étude a fait l'objet d'un document publié sur le portail Silverpeas.

Cette veille a montré tout d'abord un manque d'intérêt de la part des concurrents directs de Silverpeas pour les communications en temps réel. Ceci peut s'expliquer déjà par la nature des plate-formes proposées (orientées sur la gestion des connaissances dans le long terme) mais aussi par la complexité de mise en œuvre de communications en temps réel fiables.

De plus, cette veille technologique à montré clairement que l'utilisation de plugins tels que Flash ou Java est encore généralisée bien que fortement dépréciée. Malgré l'émergence de nouvelles technologies cohérentes et largement compatibles ces pratiques persistes.

Un protocole minimal à été décidé pour encadrer l'étude des systèmes cibles :

1. Description succincte de la messagerie instantanée et de son essai (MI)
2. Description des fonctionnalités de la MI
3. Points forts et points faibles de la MI
4. Observation des technologies employées, basée sur une observation sommaire des codes sources accessibles
5. Captures d'écran de l'interface

Objectifs additionnels

Étant donné les possibilités des technologies actuelles, et le nombre de projets libres de qualité disponibles sur le web, il a tout de suite été envisagé d'intégrer un projet existant puis de l'étendre. Les fonctionnalités suivantes ont été acceptés comme nécessaires et accessible sur la durée du stage:

- API de manipulation de la messagerie, pour une bonne intégration au portail existant
- messagerie textuelle instantanée de groupe
- édition de texte collaborative
- appels voix et vidéo, et visioconférence

L'opportunité d'Etherpad

Lors de la phase de veille technologique, il est apparut qu'une fonction d'édition de texte collaborative pourrait être ajoutée à la messagerie à moindres coûts. En effet, Etherpad est éditeur de texte collaboratif distribué sous licence libre permettant de modifier un document texte à plusieurs en temps réel.

Le projet étant bien structuré et facile d'intégration, des essais en parallèles de la messagerie ont été décidés pour observation et éventuelle intégration. L'objectif final est de proposer un espace « brouillon » ou plusieurs personnes peuvent rédiger un texte ensemble.

Gestion du projet

Répartition des charges

Le projet est réalisé en autonomie sous la supervision de l'architecte logiciel et du directeur produit.

Documentation

Tous les travaux effectués doivent laisser des traces visibles et évaluables par toute l'équipe. Dès le premier temps de veille, les résultats de recherche ont été rassemblés dans des documents publiés en ligne sur le portail de l'entreprise Silverpeas.

Depuis le début de la deuxième phase d'essais et de développement tous les travaux produits sont publiés et mis à disposition en test en ligne sur un serveur dédié.

Rencontre hebdomadaire

Une rencontre hebdomadaire à été convenue pour discuter de l'avancée du projet et des priorités du moment. La rencontre est facilitée par la publication des travaux et de leur avancée en ligne.

Feuille de route

Dès la fin de la phase de veille une feuille de route à été mise en place sous la forme d'un document Etherpad (édition collaborative en ligne). Dans ce document sont consignés chaque jour le travail en cours et le travail effectué.

Recueil des idées

Un document collaboratif à également été mis en ligne sur la plateforme de démonstration de messagerie pour recueillir les impressions et les éventuels problèmes rencontrés lors des essais.

Calendrier du projet

Un calendrier a été élaboré. Les étapes du projet ont été ordonnées de manière à positionner les phases les plus risquées au plus tôt.

- 3 à 4 semaines de veille technologique
- choix de projets de messagerie aboutis et mise en ligne pour essais
- évaluation technique des projets

- première intégration dans le portail Silverpeas
- mise en place du projet choisi avec infrastructure de production
- extension du projet choisi

Gestion des sources

Le code source du projet est géré avec l'outil Git, mis en ligne sur le Github et sauvegardé en ligne sur deux serveurs distincts.

Contraintes de communications en temps réel pour le web

Les communications instantanées pour le web sur un réseau public posent plusieurs problèmes. Avant de voir les principaux protocoles étudiés il est nécessaire de prendre connaissance des contraintes inhérentes à la mise en place d'une messagerie instantanée.

Ces problèmes sont abordés ci-dessous dans le contexte du projet de stage, c'est à dire dans le cadre de l'utilisation d'un navigateur web et de transports utilisés pour le web.

Signalement

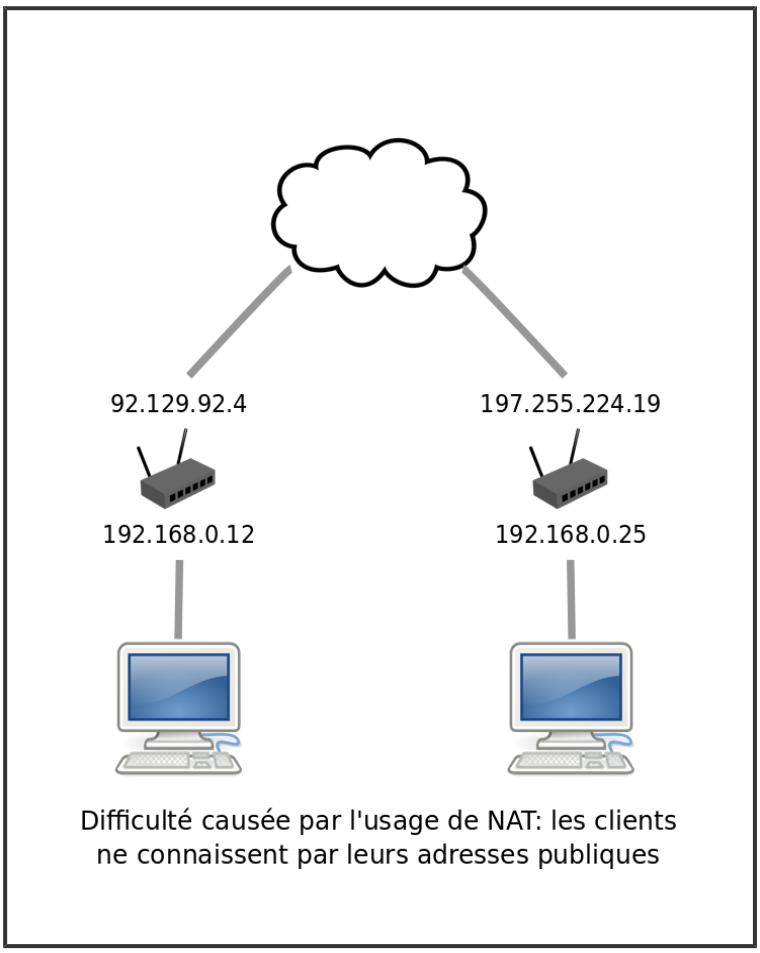
Sur un réseau où plusieurs clients souhaitent communiquer entre eux, le signalement permet aux clients de se découvrir et d'échanger des informations sur leurs capacités multimédia.

Les opérations de signalement peuvent être intégrées dans certains protocoles et certains logiciels serveurs.

Diversités des topologies réseau

Ce point est particulièrement contraignant puisque omniprésent en environnement d'entreprise et tout à fait imprévisible :

- Premièrement l'utilisation de NAT conduit un client à ne pas connaître son adresse IP publique, ce qui peut empêcher la communication entre clients. En effet si un client transmet une adresse IP privée, elle sera inutilisable sur un autre réseau.
- Deuxièmement l'utilisation de pare-feux modifie ou empêche le transit d'informations par certains ports ou certains modes de transport.



Après somme de toutes les informations analysées, un seuil de conditions restrictives à été fixé en dessous duquel la messagerie ne sera pas opérante. A partir de ce seuil sont garantit :

- Le trafic HTTP sur le port 80 des deux clients
- Le trafic HTTPS sur le port 443 des deux clients

Dans la majorité des situations seront disponibles également :

- Le trafic UDP sur le port 443 des deux clients
- Le trafic en TCP et UDP en paire à paire

Dans le cas de technologies abordées ci-dessous, des logiciels serveurs spécifiques sont disponibles pour répondre à ces problématiques.

Diversité des systèmes clients

La diversité des systèmes d'exploitation et des navigateurs web utilisés par les clients d'une application de messagerie cause de grandes disparités dans les capacités des clients et leurs compatibilités.

Les problèmes de compatibilité concernent surtout le transit de médias, le transit de texte étant mieux géré. Jusqu'à très récemment pour résoudre ce problème, deux stratégies était employées :

- Un développement spécifique pour chaque plateforme
- L'utilisation de plugins permettant d'harmoniser les capacités des différents clients et d'ajouter un niveau d'abstraction au problème

La deuxième solution n'est plus vraiment d'actualité bien qu'encore employée, puisque les plugins utilisés sont désormais activement dépréciés par les grand navigateurs (Java et Flash).

Protocoles étudiés

Veille technologique

Durant la phase de veille plusieurs protocoles et plusieurs projets ont été étudiés. Les technologies ont été répartis en quatre catégories : les protocoles, les bibliothèques (de plus bas niveau), les frameworks (de plus haut niveau), et les projets complets.

Un protocole minimaliste à été mis en place pour étudier ces technologies :

- La technologie est elle bien documentée ?
- La technologie est elle « vivante », c'est à dire soutenue et en évolution ?
- Qu'elle est sa compatibilité avec les navigateurs courants ?
- A quels types d'échanges est-elle adaptée ?
- Quelles infrastructures sont nécessaires à son fonctionnement ?

Le résultat complet de ces recherches à fait l'objet d'un document publié sur le portail Silverpeas. Ne seront présentés ci-dessous que les principaux protocoles étudiés, ainsi que deux projets de client libres. N'ont pas été étudiés tous les protocoles ou projets propriétaires, ainsi que les protocoles non destinés au web.

SIP/SIMPLE

Session Initiation Protocol est un protocole standardisé de manipulation de sessions multimédias. Destiné originellement à la téléphonie IP, SIP sert à la localisation et à l'authentification des utilisateurs. SIP peut être utilisé comme méthode de signalement uniquement ou peut être étendu pour transporter des données texte.

Dans le cas de communications multimédia, SIP est utilisé uniquement comme méthode de signalement, le transport des donnés média n'étant pas assuré par le protocole. SIP est un protocole client / serveur mais est utilisable également en paire à paire.

Les messages SIP sont d'un format proche du HTTP avec des codes de réponse similaires. Grâce à l'extension SIMPLE (SIP for Instant Messaging and Presence Leveraging Extensions) le protocole SIP peut être étendu pour répondre aux besoins basiques des messageries instantanées textuelles.

Infrastructure nécessaire

Serveur SIP

Application au projet

SIP peut être un bon choix pour le développement d'une messagerie instantanée simple mais n'est pas la meilleure option disponible dans le cadre de ce projet.

En effet le protocole SIP/SIMPLE semble bien moins structuré que d'autres protocoles et n'implémente que peu de fonctionnalités. L'utilisation de SIP pour la messagerie instantanée implique l'implémentation d'aspects complexes de la messagerie instantanée comme par exemple la gestion des

droits d'utilisateurs.

WebRTC

WebRTC est une interface de programmation applicative permettant de manipuler des flux de médias entre navigateurs web en paire à paire. WebRTC se présente sous la forme d'un ensemble d'outils accessibles en Javascript. L'objectif de cette technologie est de simplifier l'utilisation de flux média dans les navigateurs, sans utilisation de plugins externes.

WebRTC permet d'échanger des flux multimédias variés (texte, flux binaires arbitraires et flux vidéos) mais est particulièrement optimisé pour la manipulation de vidéos. WebRTC ne spécifie pas les méthodes de transport ou de signalement dans un but de non redondance : la priorité est laissée aux technologies existantes. Cependant, dans la pratique le transport UDP est privilégié bien que d'autres modes soient disponibles.

Objectif de simplification mais infrastructure lourde

Bien que WebRTC soit censé rendre autonome les navigateurs web pour le transit de médias en paire à paire, plusieurs difficultés font que l'infrastructure serveur nécessaire à l'utilisation de cette API est lourde.

En effet, étant donné la diversité des topologies réseau, et notamment l'utilisation courante de NAT et de pare-feux, plusieurs logiciels serveurs sont nécessaires.

On peut distinguer deux embûches majeures dans les communications WebRTC :

- **Un client est derrière un NAT et ne connaît pas son adresse IP publique**

  Dans ce cas lorsque que le client A est négocie une connexion avec le client B, le client A envoi son adresse privée qui est inutilisable pour le client B.

  Pour pallier ce problème il est possible de mettre à disposition des clients un serveur **STUN** (Simple Traversal of UDP through NATs). Ainsi, en effectuant une requête vers un de ces serveurs un client pourra découvrir son adresse publique.

  Un serveur STUN est peu consommateur de ressources et son utilisation permet à 86 % des connexions initiées de réussir (source : www.html5rocks.com)

- **Un client est derrière un pare-feu et ses communications sont limitées**

  Dans cette catégorie de cas variés un client peu ne pas avoir accès à une certaine plage de ports, ou le trafic peut être limité au transport TCP.

  Pour pallier ce problème, il est possible de mettre à disposition des utilisateurs un serveur **TURN** (Traversal Using Relays around NAT)**.** Le serveur TURN, en plus de proposer les services d'un serveur STUN, permet de relayer des médias en transit entre deux destinataires. Un serveur TURN peut adapter les modes de transports en fonction de son environnement.

  Le travail d'un serveur TURN peut être lourd si beaucoup de clients l'utilisent simultanément mais l'utilisation de ce serveur permet le transit de médias dans beaucoup de cas difficiles et avec de bonnes performances.

Procédure ICE

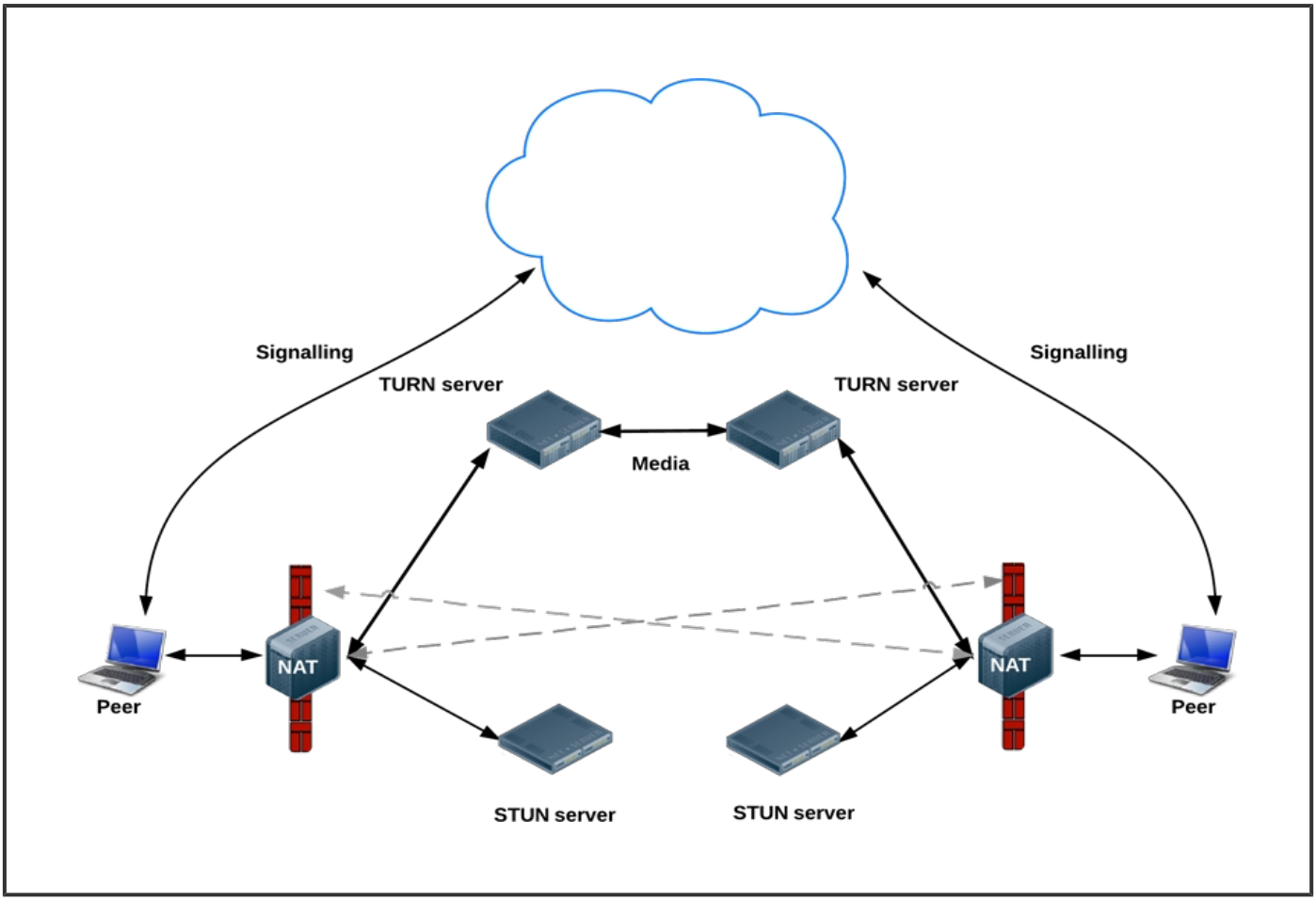
ICE (Interactive Connectivity Establisment) est une procédure de détermination automatique du meilleur moyen pour deux clients de se joindre. Plusieurs solutions sont étudiées successivement pour

déterminer la meilleure, de la moins contraignante pour les serveurs distants à la plus contraignante.

Globalement la méthode ICE agit comme suit :

- Les clients essaient de se connecter en paire à paire avec les adresses à leur disposition
- En cas d'échec chaque client effectue une requête STUN pour obtenir son adresse publique et retente une connexion
- En cas d'échec le serveur TURN relaie le flux entre les clients sur une plage définie de ports (par défaut 49152-65535)
- En cas d'échec le serveur relaie le flux entre les clients sur le port 80 ou 443

Schématisation d'une installation STUN / TURN



Source : http://www.html5rocks.com/en/tutorials/webrtc/infrastructure/

Infrastructure nécessaire

Serveur de signalement : pour la découverte de paires

Serveur STUN: pour permettre aux clients de découvrir leur adresse IP publique

Serveur TURN : pour permettre aux clients de communiquer même lorsqu'il sont situés dans des environnements contraignants.

Serveur HTTP utilisant TLS (HTTPS): indispensable pour que les navigateurs autorisent l'accès aux périphériques de capture multimédia.

Serveur MCU (Multi Control Unit): optionnel, permet la transformation des flux vidéos à la volée. Les transformations possibles sont nombreuses mais la plupart sont destinées à adapter la vidéo aux périphériques cibles (ex : mobiles) ou à l'ajout d'informations (ex : réalité augmentée)

Application au projet

WebRTC est le meilleur choix disponible pour le transport de données multimédia mais n'est pas le choix optimal pour une messagerie texte.

En effet l'utilisation de WebRTC pour la messagerie instantanée implique l'implémentation d'aspects complexes de la messagerie instantanée comme par exemple la gestion des droits d'utilisateurs.

De plus WebRTC étant une norme récente, elle est susceptible d'évoluer rapidement dans le temps et il peut être risqué d'appuyer tout un projet de messagerie sur cette norme dans son état actuel.

XMPP

XMPP est un ensemble de protocoles standardisés dédiés aux communications en temps réel. Dérivé du XML, le XMPP est structuré et hautement extensible. Destiné premièrement à la messagerie textuelle instantanée, il a peu à peu été étendu pour supporter :

- la messagerie instantanée de groupe
- les droits d'utilisateurs : propriété d'une conversation, bannissement, administration, …
- la gestion d'une liste de contacts
- un système de présences et de statuts complexes (composition, en activité, ...)
- le transfert de données multimédia : fichiers, sons, vidéo…
- les communications de service
- ….

Le XMPP peut être utilisé pour gérer de petites plate-formes de messagerie instantanée simples mais ce standard est destiné à structurer des communications à grande échelle sur des réseaux publiques décentralisés.

Fonctionnement structuré

Le XMPP spécifie des aspects complexes des communications en temps réel comme la gestion des droits d'utilisateur, la conversation texte de groupe, le signalement de paires multimédia, le système de publish-suscribe, …

XMPP prend en compte trois types d'acteurs :

- les serveurs : qui permettent la mise en relation de clients et de composants
- les utilisateurs : personnes physiques connectées à l'aide d'un client
- les composants : des utilisateurs non humains

La prise en compte des composants non-humains est une force du protocole. Cela permet d'effectuer du traitement de données et des tâches d'administration indépendamment des implémentations

choisies. Par exemple, implémenter l'ajout d'un utilisateur sur un serveur en respectant le protocole XMPP permettra de changer de logiciel serveur à la différence de l'utilisation d'une API spécifique.

Chaque utilisateur est identifié par une chaîne de caractères appelée Jabber ID (JID) sous la forme :

```
identifiant_utilisateur@service-xmpp/ressource

/* Où 'ressource' représente une chaîne de caractères unique par client connecté, un
   utilisateur pouvant utiliser plusieurs clients simultanément. */
```

Étant donné que les réseaux XMPP sont considérés comme publics, deux utilisateurs doivent être abonnés entre eux pour communiquer.

Réseau décentralisé

XMPP à été conçu pour fonctionner de manière décentralisée. Les serveurs respectant les normes peuvent communiquer entre eux sur un port dédié pour faire transiter des informations.

Ainsi un client situé sur un domaine A pourra contacter un client dans un domaine C, l'information pouvant transiter par un domaine B ou plusieurs autres domaines.

Les standards XMPP sont utilisés par de très nombreuses organisations pour structurer les échanges textuels (Google, Facebook), et ils sont activement maintenus ce qui permet une grande compatibilité d'infrastructure.

Utiliser XMPP et respecter ses normes peut permettre de communiquer à partir d'un domaine privé avec des personnes utilisant un autre réseau public comme un de ceux spécifiés sur cette liste : https://list.jabber.at

Les bases du protocole

L'ensemble des standards XMPP reposent sur trois types de messages. Les messages sont appelés des « stances ».

<presence>

La stance de présence est utilisée pour envoyer régulièrement des informations sur le statut d'un client. Les statuts peuvent être textuels, à destination d'utilisateurs humain, ou codifiés à destination de machines.

Lorsqu'un client souhaite communiquer sa présence il l'envoie au serveur XMPP qui se charge de la répartir vers tous ses abonnés.

Le message de présence permet aussi de s'abonner aux informations de présence d'un autre utilisateur, ce qui en fait un contact.

Exemple de présence :

```
<presence>
      <show>away</show>
      <status>at the ball</status>
</presence>
```

Le format de la présence est strict et les normes recommandent de ne pas tenter d'extension sur ce type de message.

<message>

La stance message est le transport privilégié d'informations à destination d'utilisateurs. Il peut être de type :

- 'chat' : communication entre deux utilisateurs
- 'groupchat' : communication entre un groupe d'utilisateurs
- ou de types plus spécifiques pour communications de service

Le message peut comporter un ou plusieurs élément body qui sont les seuls devant être affichés, sous certaines conditions. De cette manière, la stance message est facilement extensible pour pouvoir supporter par exemple le transport de métadonnées (favoris, mots clefs, ...).

La stance de message peut également transporter des média. En effet dans le cas de l'utilisation de certains modes de transport, un flux binaire peut être converti en base 64 et intégré dans une stance de message.

Exemple de message :

```
<message from='bingley@netherfield.lit/drawing_room'
       to='darcy@pemberley.lit'
       type='chat'>
       <body>Come, Darcy, I must have you dance.</body>
       <thread>4fd61b376fbc4950b9433f031a5595ab</thread>
</message>

<message from='bennets@chat.meryton.lit/mrs.bennet'
       to='mr.bennet@longbourn.lit/study'
       type='groupchat'>
       <body>
              We have had a most delightful evening, a most excellent ball.
       </body>
</message>
```

<iq>

IQ pour « Information / Query » est une stance destinée aux requêtes de service XMPP. Toute stance IQ doit obligatoirement recevoir une réponse, même sous forme d'erreur.

Un exemple parlant de l'utilisation de stances IQ est la découverte de services (XEP 0030). Un client XMPP connecté à un réseau peut découvrir les services disponibles offerts par le serveur pour adapter son comportement.

Exemple de requête IQ :

```
< ! – Requête de découverte de services -->
<iq type='get'
    from='romeo@montague.net/orchard'
    to='plays.shakespeare.lit'
    id='info1'>
  <query xmlns='http://jabber.org/protocol/disco#info'/>
</iq>

< !– Réponse -->
<iq type='result'
    from='plays.shakespeare.lit'
    to='romeo@montague.net/orchard'
    id='info1'>
  <query xmlns='http://jabber.org/protocol/disco#info'>
    <identity
        category='conference'
```

```
        type='text'
        name='Play-Specific Chatrooms'/>
    <identity
        category='directory'
        type='chatroom'
        name='Play-Specific Chatrooms'/>

    < !-- Services offerts -->

    <feature var='http://jabber.org/protocol/disco#info'/>
    <feature var='http://jabber.org/protocol/disco#items'/>
    <feature var='http://jabber.org/protocol/muc'/>
    <feature var='jabber:iq:register'/>
    <feature var='jabber:iq:search'/>
    <feature var='jabber:iq:time'/>
    <feature var='jabber:iq:version'/>

  </query>
</iq>
```

X-Forms

Pour compléter ces stances et pour conserver une architecture globale largement compatible, les X-Forms sont utilisés dans les négociations.

Un X-Forms est un formulaire XML permettant de spécifier quelles informations et quels formats d'informations sont attendus en réponse à une communication. L'utilisation de X-forms permet une bonne structure de communication indépendante des différentes implémentations possibles de composants.

Dans l'exemple de la recherche d'utilisateurs (XEP 0055), un client peut demander quels sont les utilisateurs inscrit à un serveur. Si ce client ne souhaite obtenir que les résultats avec une adresse mail contenant le mot 'pierre' alors deux solutions sont envisageables :

- Implémenter 'en dur' sur le serveur une procédure de communication ou le client doit spécifier les critères particuliers à appliquer aux adresses mail. Dans ce cas les développeurs du client pourront être contraint de changer éventuellement d'implémentation en cas de changement de logiciel serveur ou d'évolution de la norme
- Utiliser X-Forms pour spécifier quels champs de recherche peuvent être utilisés et de quelle manière selon cette chronologie :
    1. Le client effectue une requête de préparation
    2. Le serveur retourne un formulaire X-Forms détaillant les informations utilisables
    3. Le client adapte son comportement en fonctions des possibilités offertes, puis retourne le formulaire rempli
    4. Le serveur répond à la requête

Exemple de formulaire X-Forms :

```
<iq type='set'
    to='search.im.silverpeas.net'
    xmlns='jabber:client'
    id='de6a30a5-d3f3-4f28-bddd-33725ab86666:sendIQ'>
       <query xmlns='jabber:iq:search'>
             <x xmlns='jabber:x:data'
                type='submit'>
```

```
            <field type='hidden'
                      var='FORM_TYPE'>
                   <value>
                   jabber:iq:search
                   </value>
            </field>
            <field var='search'
                      type='text-single'>
                   <value>
                   *
                   </value>
            </field>
            <field var='Username'
                      type='boolean'>
                   <value>
                   1
                   </value>
            </field>
            <field var='Name'
                      type='boolean'>
                   <value>
                   1
                   </value>
            </field>
            </x>
      </query>
</iq>
```

Extensions du protocole: les XEP

Les fonctionnalités du protocole XMPP sont basiques, mais sont étendues par de nombreuses XEP : XMPP Extension Protocol. Une XEP est une spécification d'extension du protocole XMPP.

XMPP est un standard ouvert. Tout le monde peut proposer sa propre extension du protocole pour publication à la XMPP Standard Foundation à condition qu'elle respecte la XEP des XEP, la XEP 0001 Protocoles d'extension. En fonction de l'avancée d'une XEP, de sa compatibilité avec les autres XEP et les travaux à venir son état évolue de «expérimentale» à « active » ou à «obsolète»

Chaque extension respecte une structure définie avec :

- Une introduction
- Un type : information, standard, procédure, ...
- La spécification des objectifs
- Une description détaillée

Transports possibles

Pour acheminer des stances XMPP plusieurs types de transports sont disponibles :

- en « natif », c'est à dire en TCP à travers le port 5222. Méthode originelle non adaptée au web car non supportée par les navigateurs
- à travers WebSocket : le choix moderne de plus en plus apprécié. Cette solution est efficace mais encore indisponible pour certains navigateurs, et parfois bloquée par des pare-feux
- à travers HTTP avec BOSH (Bidirectional-streams Over Synchronous HTTP). Cette solution bien que plus ancienne et consommant plus de ressources est compatible avec tous les navigateurs web, et passe tous les pare-feux

De plus, le XMPP permet le transport de données multimédia. Le XMPP étant un format textuel il peut poser problème pour le transport de données binaires. Dans ce cas, deux types de transports sont possible :

- **« out-of-band » :** le plus efficace. Le transport est négocié entre deux client en XMPP mais le flux média est transmis par un moyen alternatif, parfois en paire à paire, parfois par un serveur relai : SOCKS5, WebRTC, …
- **« in-band » :** le flux de données est converti en base 64 puis segmenté et intégré dans le corps de stances « message ». Cette méthode est gourmande en ressource mais présente l'intérêt de pouvoir passer tous les pare-feux si le serveur est correctement configuré.

Remarque : La solution « in-band » n'est plus la meilleure solution de secours et a d'autres alternatives comme l'utilisation d'un serveur TURN.

Infrastructure nécessaire

Un serveur XMPP : pour l'échange de données

Application au projet

XMPP est le meilleur choix dans le cadre de ce projet pour la messagerie textuelle mais aussi en tant que composant de la messagerie multimédia.

Dans le cadre de la messagerie textuelles XMPP offre une excellente structure normée et extensible. De plus, étant donné que ces standards sont ouverts de nombreux logiciels serveurs de qualité professionnelle sont disponibles et implémentent des fonctionnalités complexes de la messagerie instantanée (messagerie de groupe, gestion des droits d'utilisateurs, ...)

De plus, dans le cadre de la messagerie multimédia, XMPP fournit une méthode de signalement efficace et permettra la découverte de paire et la négociation de flux multimédia entre les clients.

Choix et complémentarité des technologies

Dans le cadre du projet de remplacement de la messagerie instantanée, ce choix de technologie à été fait :

- XMPP à travers HTTP sera utilisé pour structurer la messagerie texte et servira également de méthode de signalement multimédia à travers son extension Jingle
- WebRTC sera utilisé pour les connexions multimédia

Externalisation des services multimédias

A la fin de la période de veille une question s'est posée naturellement : Quel sera le meilleur choix en terme d'infrastructure ? Proposer des services multimédia ou les externaliser ?

L'infrastructure nécessaire aux services multimédias est réputée complexe à mettre en œuvre et très gourmande en ressources. Pour simplifier l'utilisation de ces services plusieurs sociétés proposent des API en SaaS permettant de se libérer de ces contraintes à des prix compétitifs.

Cependant une externalisation totale des services de messagerie instantanée est inenvisageable car incompatible avec la philosophie du portail Silverpeas qui est de fournir une solution clef en main complète. De plus certains clients ayant des politiques strictes de sécurité il pourrait être délicat d'imposer l'utilisation de serveurs externes à l'entreprise.

Dans le but d'optimiser la qualité du service de messagerie multimédia, sans passer à coté de l'occasion de proposer éventuellement un service valorisable, une démarche à été convenue :

1. Mettre en place une messagerie texte structurée qui sera disponible dès l'offre la plus basique sur le portail Silverpeas
2. Puis dans un premier temps tenter de mettre en place l'infrastructure nécessaire aux communications multimédias en temps réel. Des tests seront ensuite effectués pour observer la quantité de ressources consommées.
3. Et dans un deuxième temps si la charge se révèle conséquente, découpler la gestion multimédia du client pour permettre l'alternance entre l'utilisation de services internes et externes. Ainsi l'offre de messagerie se divisera en deux :
    - Une messagerie texte et multimédia basique comprise dans l'offre de portail
    - L'utilisation de services externes en options pour une utilisation multimédia intensive

Les étapes du projet

Recherche de projets aboutis

Plusieurs projets ont été étudiés et deux seulement on été retenus. Les projets ont été sélectionnés sur la base de ces critères :

- Les projets utilisent les technologies sélectionnées
- Les projets n'utilisent aucun plugins et sont largement compatibles avec les grands navigateurs
- Les projets sont distribués sous licence libre compatible avec la licence GPLv3

Les deux projets retenus sont les suivants :

- **Kaiwa :** un client XMPP / WebRTC écrit en HTML5. Ce client est un travail dérivé du client O'Talk de l'entreprise américaine &yet. https://github.com/digicoop/kaiwa
- **JSXC :** JSXC est un projet indépendant Allemand. https://github.com/jsxc/jsxc

Mise en essai des deux projets

Dans le but de favoriser l'évaluation des projets par la pratique un serveur virtuel Ubuntu 14.04 à été dédié à la messagerie instantanée.

Création de deux images Docker pour essai

Afin de favoriser la cohabitation des deux applications sur un même serveur, et dans le but de se familiariser avec la structure des projets, une image Docker à été crée pour chacun des deux clients.

Docker est un système de conteneurisation d'application permettant d'empaqueter et de déployer des applications dans leur environnement. L'utilisation de Docker pour déployer deux applications aux besoins proches permet leur évolution dans des systèmes séparés, empêchant tout conflit.

Code source des projets :

https://github.com/remipassmoilesel/docker-kaiwa-openfire

https://github.com/remipassmoilesel/docker-jsxc-openfire-etherpad

Dockerfile d'essai du client Kaiwa. Un Dockerfile est une suite d'instructions permettant à l'application Docker de recréer l'environnement de l'application.

```
FROM ubuntu:14.04
MAINTAINER remipassmoilesel
LABEL name="Kaiwa/Openfire" description="Simple Dockerfile to test Kaiwa / Openfire"

# installer et configurer l'environnement
RUN apt-get update \
    && apt-get -y upgrade \
    && apt-get install -y openjdk-7-jre curl wget vim git xz-utils \
        sudo net-tools pwgen unzip openssh-server \
        logrotate supervisor language-pack-en software-properties-common \
        python-software-properties apt-transport-https ca-certificates

# installer Openfire
ADD opt.openfire/ /opt/openfire

# installer Node
RUN cd opt/ \
    && wget https://nodejs.org/dist/v6.2.0/node-v6.2.0-linux-x64.tar.xz \
    && tar -xf node-v6.2.0-linux-x64.tar.xz \
    && mv node-v6.2.0-linux-x64 node

# installer Kaiwa
ADD opt.kaiwa /opt/kaiwa/
RUN export PATH=$PATH:/opt/node/bin/ \
    && cd /opt/kaiwa/ \
    && /opt/node/bin/npm install

# fichier de lancement des serveurs
ADD opt.docker-entrypoint.sh /opt/docker-entrypoint.sh
RUN chmod +x /opt/docker-entrypoint.sh

# le script lancé au démarrage du conteneur
ENTRYPOINT /opt/docker-entrypoint.sh

# ouvrir les ports nécessaires
EXPOSE 80 7070 9090 9091 7443
```

Évaluation technique superficielle des projets

Kaiwa

Kaiwa est un client XMPP / WebRTC écrit en HTML5. Ce projet est très bien structuré et utilise des outils modernes: Node 6, NPM, Backbone, Jade, Stanza.io, Moonboots, … Le système est prévu pour fonctionner avec un serveur ExpressJS. Le code Javascript est clair, orienté objet, utilise les prototypes et les promesses, mais est très peu documenté.

Kaiwa est une excellente application mais elle est destiné à être utilisée en tant qu'application à part entière et demanderai un travail supplémentaire d'adaptation conséquent pour l'intégrer dans un système existant.

JSXC

JSXC est un client utilisant des outils plus anciens : Node 4, Grunt, StropheJS, Bower, JQuery … De plus son code n'est pas écrit en respectant le style orienté objet moderne de Javascript (le projet respectant une organisation en espaces de nom), il est fortement dépendant de JQuery et utilise la programmation événementielle de ce dernier.

Cependant, malgré ces défauts le projet est bien structuré et clair, et correctement documenté. Bien que moins moderne JSXC semble le meilleur candidat à l'intégration car, pour les mêmes services rendus, il nécessitera bien moins de travail d'adaptation étant donné qu'il a été conçu pour être intégré.

Fonctionnalités

Les deux projets proposent :

- de la messagerie texte un pour un
- de la messagerie texte de groupe
- la visioconférence un pour un
- le partage de fichiers

Choix d'un projet

Après mise en place des deux projets en ligne et premiers essais, la solution JSXC à été choisie comme première approche pour intégration.

Il a été convenu de faire une évaluation technique approfondie en effectuant des modification d'essais pour valider l'utilisation et l'extension de ce client.

En cas d'échec, la solution de repli sera le client Kaiwa.

Mise en place d'outils de diagnostic et de développement

Dans le but de mieux diagnostiquer les problèmes lors des installations et des développements à venir plusieurs outils ont été mis en place :

Une console XMPP : une application web permettant d'envoyer et de recevoir des stances XMPP. Cette console permet notamment de vérifier la syntaxe des requêtes envoyées et de visualiser les échanges. Travail dérivé du livre « Professional XMPP Programming with JavaScript and jQuery ».

Un observateur de trafic XMPP : une plugin JQuery développé pour l'occasion permettant d'afficher le trafic en cours d'une application XMPP. Utilisé pour repérer d'éventuels problèmes et observer le trafic.

BOSH-test : une application web simple permettant de diagnostiquer une connexion XMPP à travers HTTP. Travail dérivé du client JSXC.

ICE-test : une application web simple permettant de diagnostiquer une connexion STUN / TURN (S). Travail dérivé de https://webrtc.github.io

XMPP-Disco : une application Web permettant d'utiliser la XEP 0030 Service Discovery qui permet la découverte de capacité d'un serveur XMPP. Travail dérivé du livre « Professional XMPP Programming with JavaScript and jQuery ».

Une console d'événements JQuery : un plugin JQuery développé pour l'occasion permettant

l'observation des événements JQuery envoyés sur une page.

De plus, plusieurs scripts on été réalisés pour permettre une publication rapide des travaux en ligne notamment grâce à SSH et Rsync.

Première installation d'infrastructure

Dans le but d'explorer les possibilité de JSXC, une première installation en ligne complète à été effectuée. Le but de cette installation est de se rapprocher le plus possible des conditions réelles d'exploitation.

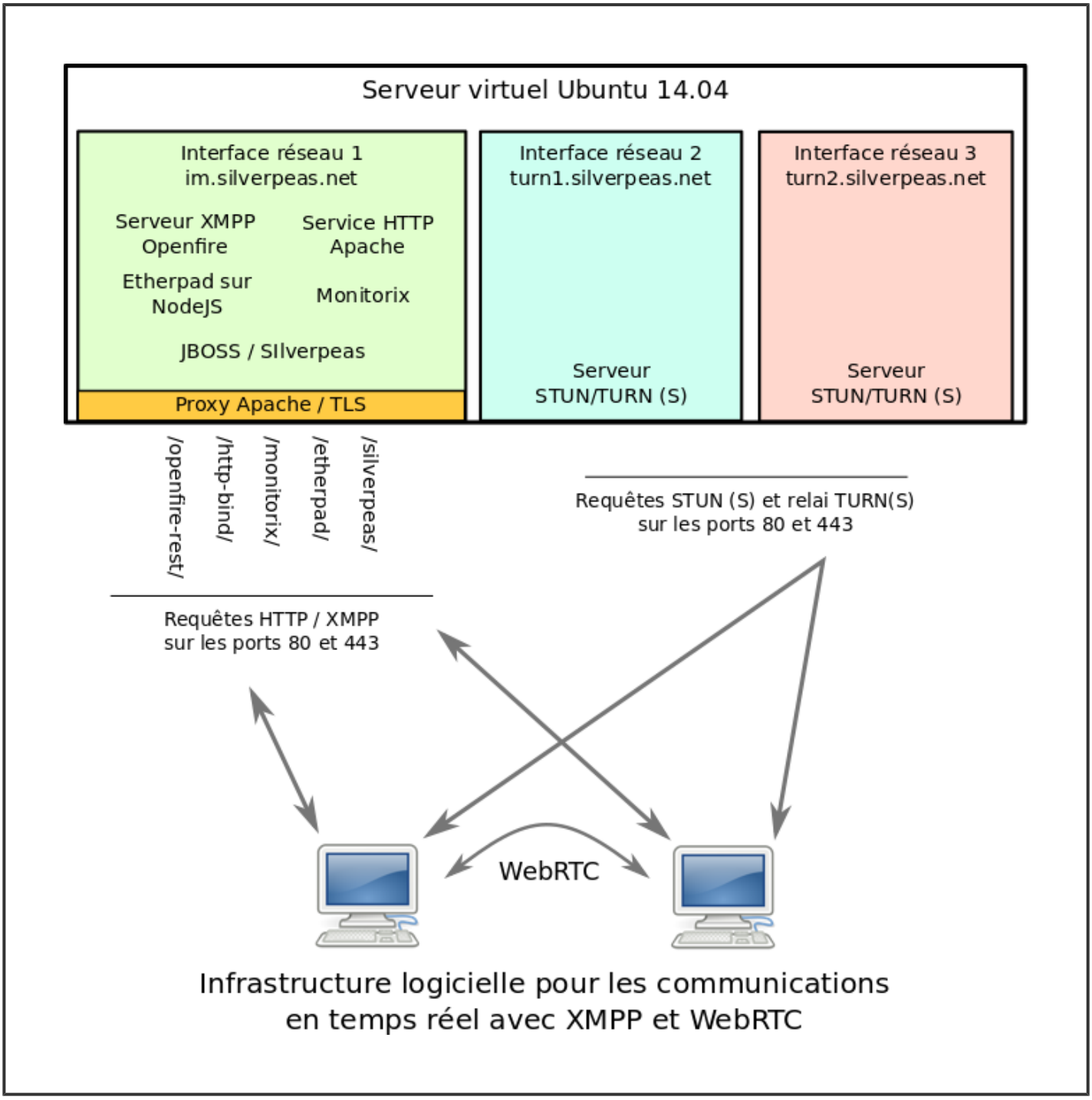
Infrastructure facilitée sous GNU/Linux

L'installation de cette infrastructure à été longue et a demandé beaucoup de configuration mais cette mise en place à heureusement grandement été simplifiée par l'utilisation d'Ubuntu 14.04.

En effet, cette distribution GNU/Linux dérivée de Debian met à disposition des utilisateurs le gestionnaire de paquet APT permettant l'installation de logiciels serveurs avec une grande simplicité.

Schéma de l'infrastructure

Le client et son infrastructure sont disponibles en ligne à l'adresse: https://im.silverpeas.net



Détail de l'infrastructure

Un serveur Apache : Pour servir le site de démonstration de la messagerie, mais surtout pour servir de proxy TLS pour toutes les applications. Grâce à l'utilisation du proxy Apache les requêtes sont centralisés et tous les serveurs peuvent écouter simultanément les ports 443 et 80 ce qui est essentiel pour permettre une accessibilité maximale vis à vis des pare-feux. De plus le proxy Apache facilite la gestion des clefs TLS.

Un serveur XMPP Openfire : pour gérer les communications textuelles mais aussi comme serveur de

signalement multimédia grâce à la XEP 0166 Jingle.

Un serveur STUN/TURN : pour permettre aux clients de découvrir leurs adresses IP publiques et pour faire relais entre deux clients dans les cas les plus restrictifs.

Monitorix: Application de surveillance de l'activité d'un système. Monitorix a été installé dans le but de surveiller la consommation de ressources du relais audio/vidéo TURN. Malheureusement ses mesures sont faussées car l'application n'est pas adaptée aux machines virtuelles (les mesures concernent le serveurs entier et non la machine virtuelle).

Une base de donnée Postgres : pour stocker les informations d'Openfire, du portail Silverpeas et d'Etherpad.

NodeJS4 et Etherpad : pour permettre l'édition de texte collaborative.

Les contraintes

Utilisation de TLS : L'utilisation de communications sécurisées est indispensable pour que les applications puissent s'exécuter sans problèmes. En effet les navigateurs empêchent le chargement de ressources non sécurisées dans certains cas, et certains empêchent l'accès aux périphériques d'acquisition multimédia.

Un domaine séparé et deux interfaces réseaux supplémentaires pour TURN : En effet, pour que le serveur STUN/TURN puisse découvrir au mieux le type de réseau utilisé par ses clients, les clients doivent contacter deux adresses IP distinctes sur les ports 80 et 443.

Configuration du proxy

Portion du fichier de configuration du serveur Apache :

```
SSLEngine on
SSLCertificateFile /etc/ssl/srvstage.crt
SSLCertificateKeyFile /etc/ssl/srvstage.key
SSLCertificateChainFile  /etc/ssl/GandiStandardSSLCA2.pem
SSLVerifyClient None

ProxyVia On
ProxyRequests Off
ProxyPreserveHost On
RewriteEngine On

# Silverpeas portal
ProxyPass /weblib http://127.0.0.1:8000/weblib
ProxyPassReverse /weblib http://127.0.0.1:8000/weblib
ProxyPass /silverpeas http://127.0.0.1:8000/silverpeas
ProxyPassReverse /silverpeas http://127.0.0.1:8000/silverpeas
ProxyPass /website http://127.0.0.1:8000/website
ProxyPassReverse /website http://127.0.0.1:8000/website
ProxyPass /help_fr http://127.0.0.1:8000/help_fr
ProxyPassReverse /help_fr http://127.0.0.1:8000/help_fr

# Etherpad
ProxyPass /etherpad/p/ http://localhost:9001/p/
ProxyPassReverse /etherpad/p/ http://localhost:9001/p/

ProxyPass /etherpad/ http://localhost:9001/
ProxyPassReverse /etherpad/ http://localhost:9001/

# Openfire http-bind
# Beware ! Don't use another method for proxying. Only RewriteRule works with Openfire.
RewriteRule ^/http-bind/$ http://localhost:7070/http-bind/ [P,L]
ProxyPassReverse /http-bind/ http://localhost:7070/
```

```
# Openfire REST API
RewriteRule ^/openfire-rest/(.*) http://localhost:9090/plugins/restapi/v1/$1 [P,L]
ProxyPassReverse /openfire-rest/  http://localhost:9090/plugins/restapi/v1/ [P,L]

# Monitorix
ProxyPass /monitorix/ http://212.83.142.6:8080/monitorix/
ProxyPassReverse /monitorix/ http://212.83.142.6:8080/monitorix/

ProxyPass /monitorix-cgi/ http://212.83.142.6:8080/monitorix-cgi/
ProxyPassReverse /monitorix-cgi/ http://212.83.142.6:8080/monitorix-cgi/

DocumentRoot /var/www/djoe/
```

Évaluation technique approfondie

Afin d'être sur du choix des outils et de leur adaptabilité, une phase de modification du projet JSXC à été convenue. Subdivisée en trois temps, cette phase a permit de mieux appréhender les possibilités de XMPP et de JSXC.

Essais autour du protocole XMPP

Afin de valider l'utilisation du XMPP trois modifications ont été réalisées. Seule la première modification sera détaillée à titre d'exemple.

Implémentation de la XEP 0085 : Notifications d'état de messagerie

Implémentation d'une partie de cette XEP dans le but d'obtenir un indicateur de composition dans la fenêtre de saisie de message. Par exemple, si Pierre écrit à Paul, alors Paul verra s'afficher « Pierre est en train d'écrire... » en bas de sa fenêtre de discussion.

La XEP 0085 (http://xmpp.org/extensions/xep-0085.html) spécifie une procédure d'échange de donnée qualifiant l'état du client d'un utilisateur :

- En train d'écrire
- Client en activité
- Client inactif temporairement
- Client inactif sur une longue durée
- …

La XEP spécifie le moyen de découvrir si un composant supporte cette fonctionnalité :

```
<!-- Envoi d'un message de découverte de capacités -->
<iq from='romeo@shakespeare.lit/orchard'
    id='disco1'
    to='juliet@capulet.com/balcony'
    type='get'>
  <query xmlns='http://jabber.org/protocol/disco#info'/>
</iq>

<!-- Réponse simplifiée -->
<iq from='juliet@capulet.com/balcony'
    id='disco1'
    to='romeo@shakespeare.lit/orchard'
    type='result'>
  <query xmlns='http://jabber.org/protocol/disco#info'>
    <feature var='http://jabber.org/protocol/chatstates'/>
  </query>
```

```
</iq>
```

A la différence du système classique d'échange d'informations de présence, la XEP 0085 recommande l'utilisation de la stance <message> pour signaler ces informations, sous la forme suivante :

```
<message
    from='bernardo@shakespeare.lit/pda'
    to='francisco@shakespeare.lit/elsinore'
    type='chat'>
  <composing xmlns='http://jabber.org/protocol/chatstates'/>
</message>
```

Dans le but de simplifier la terminologie, le terme « *présence de composition* » sera utilisé pour désigner la stance ci-dessus.

Afin de n'implémenter que le signalement de composition dans un premier temps, une chronologie de signalement et de réception de présence à été établie comme suit :

- Un utilisateur ouvre sa messagerie, choisit un utilisateur ou un groupe et commence la rédaction d'un message à son intention
- A chaque frappe de l'utilisateur :
  - Si la différence entre le temps présent et le dernier envoi de présence de composition est supérieure à une période déterminée (800ms), alors envoi d'une présence de composition
  - Puis enregistrement de l'instant du dernier envoi de message
- A la réception d'une présence de composition :
  - Enregistrement du nom du rédacteur
  - Affichage ou modification de la notification en bas de fenêtre de composition

Portion d'implémentation Javascript / JQuery :

```
/**
* Triggered on incoming messages, whatever the type of
* @param stanza
* @returns {boolean}
*/
onReceived: function (stanza) {

      // check if it is a composing presence
      var composing = $(stanza).
            find("composing[xmlns='http://jabber.org/protocol/chatstates']");

      if(composing.length > 0){

            var type = $(stanza).attr("type");
            var from = $(stanza).attr("from");

            // ignore own notifications in groupchat
            if(type === "groupchat" && Strophe.getResourceFromJid(from) ===
                        jsxc.xmpp.getCurrentNode()){
                  return true;
            }

            jsxc.gui.window.showComposingPresence(from, type);
```

```
            // stop but keep handler
            return true;
      }
}

/**
* Show a composing presence from jid specified in argument.
* JID can be a room jid or a person jid
* @param from
*/
showComposingPresence: function (from, type) {

var bid = Strophe.getBareJidFromJid(from);
var user = type === "chat" ? Strophe.getNodeFromJid(from) :
      Strophe.getResourceFromJid(from);

// iterate window list
$('#jsxc_windowList .jsxc_windowItem').each(function () {

      // the window element, where are stored informations
      var self = $(this);

      var winBid = self.data("bid");

      // check conversation
      if (winBid === bid) {

            // add user in array if necessary
            var usersComposing = self.data("usersComposing") || [];
            if (usersComposing.indexOf(user) === -1) {
                  usersComposing.push(user);
                  self.data("usersComposing", usersComposing);
            }

            var textarea = self.find(".jsxc_textarea");
            var composingNotif = textarea.find(".jsxc_userComposing");

            // add notification if necessary
            if (composingNotif.length < 1) {
                  textarea.append("<div class='jsxc_userComposing
            jsxc_chatmessage jsxc_sys'></div>");
                  composingNotif = textarea.find(".jsxc_userComposing");
            }

            // change text
            var msg = usersComposing.length > 1 ? " sont en train "
                        + "d'écrire ..." : " est en train d'écrire ...";
            composingNotif.html(usersComposing.join(", ") + msg);

            // scroll to bottom
            jsxc.gui.window.scrollDown(winBid);

            // hide notification after delay
            if ($(this).data("composingTimeout")) {
                  clearTimeout($(this).data("composingTimeout"));
            }

            $(this).data("composingTimeout",

                  setTimeout(function () {

                        textarea.find(".jsxc_userComposing").remove();

                        // empty user list
                        self.data("usersComposing", []);

                  }, jsxc.gui.window.hideComposingNotifDelay)
            );

            // show only one presence
            return false;
```

```
        }
});
```

L'implémentation s'est déroulée sans difficultés . Bien entendu elle mérite d'être testée plus avant et internationalisée.

Implémentation de la XEP 0055 : Recherche d'utilisateur

Implémentation en Javascript coté client de l'extension de XMPP permettant de rechercher des utilisateurs inscrits sur un serveur. Utilisation de StropheJS et JQuery, et intégration au module JSXC existant de gestion des communications XMPP.

La modification a été réalisée sans difficultés.

Implémentation de la XEP 0249 : Invitations directe à une session de discussion en groupe

Implémentation en Javascript de la fonctionnalité d'invitation d'utilisateurs à participer à une conversation de groupe. Gestion des notifications et de l'acceptation d'invitation. Utilisation du système de discussion de groupe de JSXC, StropheJS, JQuery.

La modification a été réalisée sans difficultés.

Essais autour de la structure du code

Afin de valider la structure du projet JSXC, plusieurs modifications de sa chaîne de compilation ont été effectuées.

JSXC utilise une chaîne de compilation Grunt pour :

- concaténer les différents fichiers Javascript du code source en un seul
- concaténer les différentes dépendances du client en un seul fichier
- transpiler le style SASS
- effectuer d'autres opération diverses : génération de fichiers .map, production de documentation, application de licence, …

Dans le but d'en apprendre plus sur la structure du code, et afin d'en améliorer la forme globale plusieurs modifications ont été apportées :

- Ajout de dépendances dans la chaîne de compilation
- Réduction du format du projet fini (concaténation de plusieurs fichiers)
- Mise à jour des dépendances (JQuery, Strophe, ...)
- Ajout de styles
- Un peu plus tard dans la phase d'intégration, un conflit avec la plateforme cible a contraint l'encapsulation du système d'internationalisation du client JSXC avec Webpack (un utilitaire de gestion de module plus récent que la technique de concaténation)

Toutes ces modifications ont été réalisées sans difficultés.

Essais autour de l'interface

Afin de valider l'utilisation de JSXC en tant que client une modification a été effectuée pour vérifier que le code de l'interface est structuré et modifiable.

Modification du menu principal de l'application

L'interface proposée par défaut par JSXC est assez rudimentaire. En effet elle n'offre que trop peu d'espace pour ajouter des fonctionnalités, et son code est réparti dans tout le code de l'interface.

Un nouveau menu à été conçu puis ajouter à JSXC en remplacement de l'ancien. Le nouveau menu utilise le plugin JQuery Accordion (menu en accordéon avec onglets se recouvrant successivement) ce qui permet de gagner de la surface utilisable. Une fonction de recherche à été ajoutée pour permettre aux utilisateurs de rechercher dans les commandes mais aussi dans les commentaires.

Le menu a été encapsulé dans un espace de nom à part entière, dans un fichier unique, pour permettre une meilleure lisibilité du code.

La modification a été réalisée sans difficultés particulières.

Modification du menu de JSXC pour analyse approfondie de la structure du code de son interface

Première intégration à la plateforme

Afin de vérifier que le projet est compatible avec le portail Silverpeas, une première intégration à été effectuée.

Le portail Silverpeas fonctionne avec des Java Server Page, des classes Java exécutées coté serveur et dont la sortie et redirigée vers le client.

Préparation

Dans un premier temps la chaîne de compilation du client JSXC a été modifiée pour embarquer un maximum des dépendances qui lui sont propres. Ce qui permet aujourd'hui l'inclusion de deux scripts et d'une feuille de style seulement.

Intégration

Le client JSXC à été intégré dans le portail Silverpeas en respectant les conventions de codage en cours.

L'arborescence du client à été placée dans le répertoire de ressources web ou sont placés les scripts Javascript.

Un plugin Java d'inclusion de ressources Javascript a été développé spécifiquement pour le portail Silverpeas. Cet outil à été utilisé pour intégré le client JSXC comme suit.

Plugin d'inclusion Javascript pour JSP :

```
public static final String CHAT_CLIENT_DIR = URLUtil.getApplicationURL() +
                                "/chatclient/";
public static final String CHAT_CLIENT_STYLES = CHAT_CLIENT_DIR +
                                "css/jsxc.css";
public static final String CHAT_CLIENT_DEPENDENCIES = CHAT_CLIENT_DIR +
                                "lib/jsxc.dep.js";
public static final String CHAT_CLIENT_MAIN = CHAT_CLIENT_DIR + "jsxc.js";
public static final String CHAT_CLIENT_INIT = CHAT_CLIENT_DIR +
                                "jsxc_init.js";

public static ElementContainer includeChatClient(final ElementContainer
                                                    xhtml) {

      xhtml.addElement(new link().setType(STYLESHEET_TYPE)
                                .setRel(STYLESHEET_REL)
                                .setHref(CHAT_CLIENT_STYLES));

      xhtml.addElement(new script().setType(JAVASCRIPT_TYPE)
                                .setSrc(CHAT_CLIENT_DEPENDENCIES));

      xhtml.addElement(new script().setType(JAVASCRIPT_TYPE)
                                .setSrc(CHAT_CLIENT_MAIN));

      xhtml.addElement(new script().setType(JAVASCRIPT_TYPE)
                                .setSrc(CHAT_CLIENT_INIT));

      return xhtml;
}
```

Ce qui permet au code final d'inclusion dans une JSP d'être très réduit :

```
<view:includePlugin name="chatClient"/>
```

Création automatique de comptes

Les comptes de la messagerie XMPP sont gérés au niveau du serveur XMPP, qui ne partage ses informations avec le portail Silverpeas. Or, les utilisateurs s'identifient sur le portail Silverpeas et ne doivent pas s'identifier une deuxième fois.

Avant d'envisager la création d'une API de gestion du serveur XMPP Openfire à partir de la plateforme Silverpeas, une stratégie de simplification a été adoptée pour permettre les premiers tests.

Lorsque qu'un utilisateur se connecte au portail Silverpeas :

1. le client JSXC est chargé
2. une analyse des cookies du navigateur est effectuée pour extraire l'identifiant de l'utilisateur
3. un compte XMPP est créé via l'API REST d'Openfire si besoin

Cette méthode simple permet l'utilisation rapide du client JSXC dans la plate-forme. Cependant, à terme l'objectif est de disposer d'une véritable API de contrôle de serveur XMPP, si possible indépendante de l'implémentation du serveur.

Conclusion de l'évaluation technique et de l'intégration

Infrastructure serveur

Bien que contraignante, l'infrastructure serveur est bien moins complexe à installer prévue. La vidéo conférence fonctionne, même à travers le réseau téléphonique 4G réputé pour être contraignant. A terme, une fois les différents test effectués et une fois la charge serveur quantifiée il sera certainement possible de proposer un service multimédia performant.

Client JSXC

Le client JSXC fonctionne bien. Son code est structuré, modifiable et assez facile à lire. Des modifications sont à prévoir pour le fiabiliser mais rien d'inaccessible ou de particulièrement coûteux en temps. Grâce au temps gagné sur la première implémentation, il sera possible de pousser le développement vers de nouvelles fonctionnalités innovantes.

Améliorations supplémentaires effectuées

Messagerie sans salons

La « XEP 0045 Conversation de groupe » spécifie que les utilisateurs doivent s'abonner ou se désabonner à des salons pour participer à des conversations de groupe.

Or, l'utilisation explicite de salons, bien que puissante, peut compliquer l'expérience utilisateur. En effet lorsqu'un utilisateur souhaite discuter en groupe avec d'autres utilisateurs il doit suivre cette chronologie :

1. « Pierre » créé un salon
2. Pierre invite « Paul » et « Jean » à le rejoindre, et leur communique le nom du salon
3. Paul et Jean cherche le salon
4. Puis Paul et Jean s'abonnent au salon

En faisant abstraction de la notion de salons dans les communications instantanées de groupe l'objectif et de simplifier l'expérience des utilisateurs :

- Pierre créé un salon, et invite Paul et Jean à y participer
- Paul et Jean acceptent de participer à la conversations

Dans cet objectif des modifications ont été nécessaires sur le client JSXC :

- **Permettre aux utilisateurs d'inviter d'autres utilisateurs à une conversation**

  En implémentant la XEP 0249 : Invitations directe à une session de discussion en groupe. Ainsi Pierre peut envoyer un message à Paul avec l'identifiant du salon de discussion.

- **Permettre la création de salons avec identifiant générés automatiquement**

  En implémentant une méthode de génération de salon automatique identifiées sous la forme proprietaire-date-sel@service-xmpp.net

- **Permettre le stockage des références des salons créés pour pouvoir les retrouver facilement**

  En utilisant la « XEP 0048 Favoris » permettant de stocker des utilisateurs ou des salons de conversation en favoris.

Intégration d'Etherpad

Afin de permettre l'édition simultanée de texte en ligne, la solution Etherpad à été intégrée à la messagerie instantanée.

Etherpad est une solution libre d'édition de documents texte en temps réel. Une fois un serveur installé, plusieurs utilisateurs peuvent saisir du texte simultanément sur un même document. Les documents sont nommés des « pad ».

L'utilisation d'Etherpad est pour l'instant envisagée uniquement comme espace de brouillon disponible en lien avec une conversation existante.

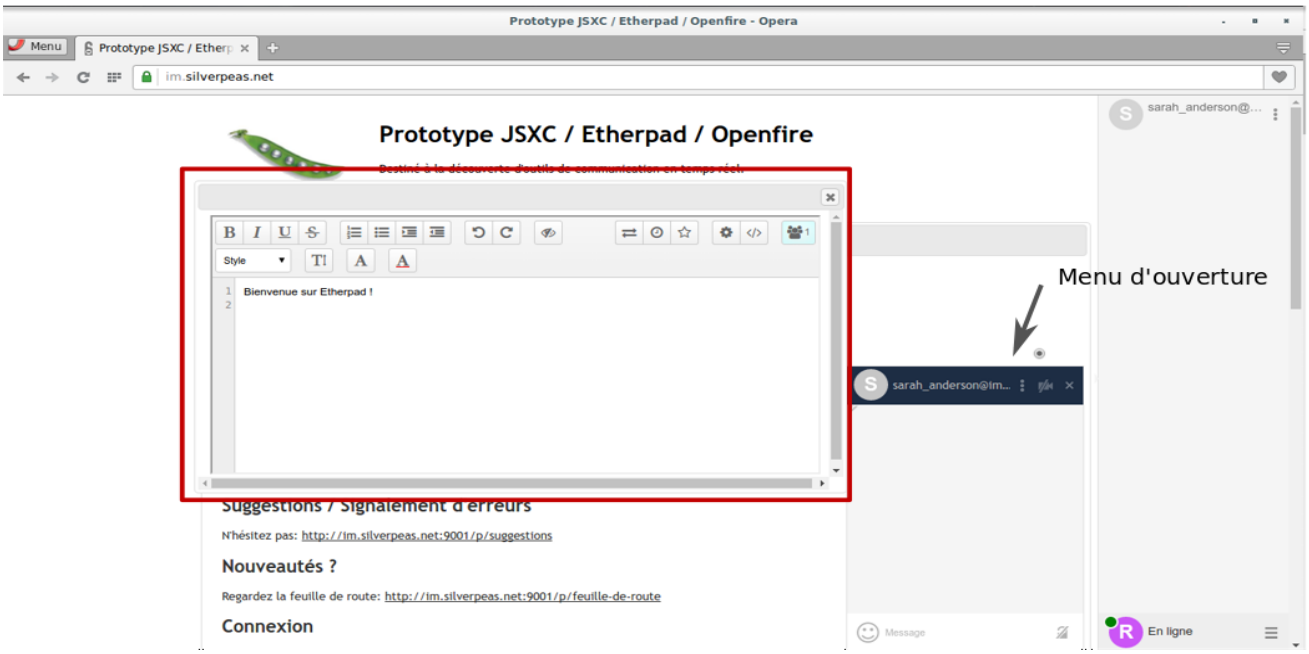
Infrastructure

Etherpad fonctionne avec NodeJS et le framework Express. Afin d'optimiser l'organisation du serveur Etherpad à été placé derrière un proxy Apache / HTTPS

Organisation

Une ouverture de document peut être déclenchée pendant une conversation de groupe. Chaque document est nommé par l'identifiant unique de la conversation de groupe qui lui est liée. L'utilisation de cet identifiant permet à tous les participants d'une même conversation d'ouvrir le même document.

Ouverture d'un pad assoicé à un salon de conversation



Modifications à venir

La suite des travaux est prévue pour l'instant comme suit :

- Travail sur la création d'une API Java EE permettant d'administrer automatiquement un serveur Openfire à partir du portail Silverpeas
- Travail de fiabilisation du client JSXC
- Visioconférence à plusieurs
- Essais sur les possibilités de partage d'écran

Conclusion

Le respect d'une première phase de veille technologique a permis la découverte de protocoles ouverts adaptés à une messagerie d'entreprise. Les essais qui ont suivi ont ensuite permis de déterminer les possibilités réelles offertes par les technologies modernes ainsi que leur complexité de mise en œuvre. Désormais s'ouvre une phase d'intégration et de développement plus poussée, qui fera l'objet d'un deuxième rapport.

Si tout ce travail a été rendu accessible à une seule personne en un temps raisonnable, c'est grâce au travail de nombreux acteurs du monde numérique qui choisissent de publier le résultat de leurs travaux sous licence libre.

Le choix de publier des travaux sous licence libre n'est pas qu'un choix éthique engagé, c'est avant tout une méthode de travail efficace. Publier sous licence libre, c'est un moyen d'amélioration de la qualité des logiciels produits, un excellent moyen de promouvoir le travail d'une personne ou d'une organisation, et surtout c'est donner à toute une communauté l'opportunité de s'améliorer et de proposer des services toujours plus modernes.