Département
Informatique
IUT Bordeaux 1

Société Y3S SAS
47, rue Fragonard
Bruges

Solution d'administration à distance d'objets connectés

Stage de DUT réalisé par

Lorian Corbel

du 4 avril au 11 juin 2016

Maître de stage	Sylvain LERIS
Enseignant responsable	Olivier LYS
Année universitaire	2015-2016

Résumé

Dans l'univers naissant de l'Internet des objets en plein developpement, ces objets hébergent de plus en plus de contenu multimédia et service. L'entreprise Y3S recherche une solution pour pouvoir accéder à n'importe quel objets connectés à travers internet, sans connaître sa configuration réseau actuel ou son adresse internet de manière générique.

Abstract

World of IOT !

Remerciements

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed non risus. Suspendisse lectus tortor, dignissim sit amet, adipiscing nec, ultricies sed, dolor. Cras elementum ultrices diam. Maecenas ligula massa, varius a, semper congue, euismod non, mi. Proin porttitor, orci nec nonummy molestie, enim est eleifend mi, non fermentum diam nisl sit amet erat. Duis semper. Duis arcu massa, scelerisque vitae, consequat in, pretium a, enim. Pellentesque congue. Ut in risus volutpat libero pharetra tempor. Cras vestibulum bibendum augue. Praesent egestas leo in pede. Praesent blandit odio eu enim. Pellentesque sed dui ut augue blandit sodales. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae ; Aliquam nibh. Mauris ac mauris sed pede pellentesque fermentum. Maecenas adipiscing ante non diam sodales hendrerit. Ut velit mauris, egestas sed, gravida nec, ornare ut, mi. Aenean ut orci vel massa suscipit pulvinar. Nulla sollicitudin. Fusce varius, ligula non tempus aliquam, nunc turpis ullamcorper nibh, in tempus sapien eros vitae ligula. Pellentesque rhoncus nunc et augue. Integer id felis.

Table des matières

Table des figures

Liste des sigles et acronymes

IP	*Internet Protocol*
TCP	*Transmission Control Protocol*
HTTP	*HyperText Transfer Protocol*
MIPS	*Microprocessor Without Interlocked Pipeline Stages*
RISC	*Reduced Instruction Set Computer*
CISC	*Complex Instruction Set Computer*
GCC	*GNU Compiler Collection*
VPN	*Virtual Private Network*
SSH	*Secure Shell*
SSL	*Secure Sockets Layer*
TLS	*Transport Layer Security*

Introduction

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed non risus. Suspendisse lectus tortor, dignissim sit amet, adipiscing nec, ultricies sed, dolor. Cras elementum ultrices diam. Maecenas ligula massa, varius a, semper congue, euismod non, mi. Proin porttitor, orci nec nonummy molestie, enim est eleifend mi, non fermentum diam nisl sit amet erat. Duis semper. Duis arcu massa, scelerisque vitae, consequat in, pretium a, enim. Pellentesque congue. Ut in risus volutpat libero pharetra tempor. Cras vestibulum bibendum augue. Praesent egestas leo in pede. Praesent blandit odio eu enim. Pellentesque sed dui ut augue blandit sodales. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae ; Aliquam nibh. Mauris ac mauris sed pede pellentesque fermentum. Maecenas adipiscing ante non diam sodales hendrerit. Ut velit mauris, egestas sed, gravida nec, ornare ut, mi. Aenean ut orci vel massa suscipit pulvinar. Nulla sollicitudin. Fusce varius, ligula non tempus aliquam, nunc turpis ullamcorper nibh, in tempus sapien eros vitae ligula. Pellentesque rhoncus nunc et augue. Integer id felis. Curabitur aliquet pellentesque diam. Integer quis metus vitae elit lobortis egestas. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Morbi vel erat non mauris convallis vehicula. Nulla et sapien. Integer tortor tellus, aliquam faucibus, convallis id, congue eu, quam. Mauris ullamcorper felis vitae erat. Proin feugiat, augue non elementum posuere, metus purus iaculis lectus, et tristique ligula justo vitae magna. Aliquam convallis sollicitudin purus. Praesent aliquam, enim at fermentum mollis, ligula massa adipiscing nisl, ac euismod nibh nisl eu lectus. Fusce vulputate sem at sapien. Vivamus leo. Aliquam euismod libero eu enim. Nulla nec felis sed leo placerat imperdiet. Aenean suscipit nulla in justo. Suspendisse cursus rutrum augue. Nulla tincidunt tincidunt mi. Curabitur iaculis, lorem vel rhoncus faucibus, felis magna fermentum augue, et ultricies lacus lorem varius purus. Curabitur eu amet.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed non risus. Suspendisse lectus tortor, dignissim sit amet, adipiscing nec, ultricies sed, dolor. Cras elementum ultrices diam. Maecenas ligula massa, varius a, semper congue, euismod non, mi. Proin porttitor, orci nec nonummy molestie, enim est eleifend mi, non fermentum diam nisl sit amet erat. Duis semper. Duis arcu massa, scelerisque vitae, consequat in, pretium a, enim. Pellentesque congue. Ut in risus volutpat libero pharetra tempor. Cras vestibulum bibendum augue. Praesent egestas leo in pede. Praesent blandit odio eu enim. Pellentesque sed dui ut augue blandit sodales. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae ; Aliquam nibh. Mauris ac mauris sed pede pellentesque fermentum. Maecenas adipiscing ante non diam sodales hendrerit.

Ut velit mauris, egestas sed, gravida nec, ornare ut, mi. Aenean ut orci vel massa suscipit pulvinar. Nulla sollicitudin. Fusce varius, ligula non tempus aliquam, nunc turpis ullamcorper nibh, in tempus sapien eros vitae ligula. Pellentesque rhoncus nunc et augue. Integer id felis. Curabitur aliquet pellentesque diam. Integer quis metus vitae elit lobortis egestas. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Morbi vel erat non mauris convallis vehicula. Nulla et sapien. Integer tortor tellus, aliquam faucibus, convallis id, congue eu, quam. Mauris ullamcorper felis vitae erat. Proin feugiat, augue non elementum posuere, metus purus iaculis lectus, et tristique ligula justo vitae magna. Aliquam convallis sollicitudin purus. Praesent aliquam, enim at fermentum mollis, ligula massa adipiscing nisl, ac euismod nibh nisl eu lectus. Fusce vulputate sem at sapien. Vivamus leo. Aliquam euismod libero eu enim. Nulla nec felis sed leo placerat imperdiet. Aenean suscipit nulla in justo. Suspendisse cursus rutrum augue. Nulla tincidunt tincidunt mi. Curabitur iaculis, lorem vel rhoncus faucibus, felis magna fermentum augue, et ultricies lacus lorem varius purus. Curabitur eu amet.

Chapitre 1

L’entreprise

1.1 Y3S

Chapitre 2

Solution d'administration à distance d'objets connectés

2.1 Nécéssité de cette solution

La société Y3S developpe toute sorte d'objets connectés comme des centrales d'alarmes, sondes de relevé de température et tous hébergent un serveur HTTP local pour sa gestion ou son utilisation, malheureusement pour accéder à son contenu local depuis internet il faut ouvrir les ports et configurer les routeurs sur lequel nous y avons connecté l'objet. Pour les grandes entreprises ou collectivités il n'est pas toujours possible d'effetuer ce genre de modification à leur architecture réseau à cause de problèmes de sécurité, de compétence ou tout simplement parce que le nombre d'objets connectés est trop important pour recevoir ces configurations manuellement pour chaque unité.

Une autre solution consiste à developper un CLOUD sur lequel se connectent tout les objets connectés et permettant aux utilisateurs d'y accéder par l'intermédiaire d'un frontend HTTP par exemple. Cette solution reste extrêmement couteuse, en effet elle nécessite de développer une solution CLOUD spécifique à chaque service ou type d'objet connecté.

C'est donc dans cette optique que l'entreprise recherche une solution qui permettrerai de remplacer le CLOUD par une solution plus générique qui donnerait directement accès aux services locaux hébergés directement sur l'objet connecté, réduissant ainsi les coût en developpement et installation.

2.2 Spécifications

2.2.1 Spécifications fonctionnelles

Il est necessaire de développer la solution en deux partie, le client a installé sur l'objet et le serveur backend en charge de rendre l'objet connecté accesible sur internet. La solution doit être capable de fournir un client ou un SDK à installer sur l'objet connecté, qui permet permet de se connecter au serveur backend de la solution. Les principales fonctionnalités de la solution sont :

— Générer un nom de domaine unique par objet connecté qui hébergent un serveur HTTP.
— Gérer les déconnexion, reconnexion de l'objet connecté, pointer sur une page HTML d'erreur si on tente d'accéder au nom de domaine d'un objet déconnecté et founir un nouveau nom de domaine unique si l'objet est resté deconnecté jusqu'à l'écoulement d'un délais prédéfini, sinon lui resservir le même nom de domaine.
— Pouvoir se reconnecter automatiquement à un des serveurs s'il perd la connexion.
— La solution ne doit pas se contenter d'un seul seule protocole tel que le HTTP, mais doit pouvoir relayer n'importe quel protocole utilisant le TCP.

2.2.2 Contraintes

Le monde des objets connectés utilise des architectures de processeur spécifiques et variés. La solution cliente doit donc être multi-plateforme, elle doit fonctionner sur UNIX, Windows, Android sur des architectures x86/x64, ARM, MIPS. On doit pouvoir aussi réimplémenter le client sur microcontrôleur possèdant une stack IP parce qu'il sont très utilisés dans le monde de l'industrie et donc des objets connectés.

Certaine entreprise ont des pare-feux très restrictifs ne permettant la sortie que de certain protocole comme le HTTP et HTTPS respectivement sur le port 80 et 443.

2.2.3 Spécifications techniques

Pour répondre aux besoins de la solution, des choix techniques ont étés fait tout le long du developpement du prototype de la solution. Au début Y3S m'a demandé de comparer les solutions existantes de Reverse Tunneling, qui leur semblait être la technique la plus pertinente, particulièrement si la solution utilise les websockets.

J'ai personnellement choisi de développer le client en C++, un langage bas niveau orienté objet qui reste facile à compiler sur plusieurs architectures. Le serveur a été développé en nodejs pour sa simplicité de mise en place et de developpement, sa scabilité et sa maintenance.

Pour résumer, voici les technologies techniques utilisés actuellement sur le prototype à la fin de son développement, leur justification d'utilisation suivra dans le rapport :

— Reverse Tunneling en websocket : transfert de protocole TCP à travers n'importe quel routeur, pare-feu qui accepte le HTTP/HTTPS.
— C++ et socket natif : développement client.
— Node.js : développement serveur.
— Redis : base de données.
— OpenResty : proxy HTTP dynamique.

2.3 Principe du Reverse Tunneling

Pour illustrer le principe du reverse tunneling je vais me servir d'un scénario. Imaginons que nous souhaitons atteindre le serveur HTTP d'Alice, mais Alice est derrière un NAT qui bloque toute les connexions entrantes sur son réseau. Malheureusement elle n'a pas la main sur son routeur, ce qui nous empêche naturellement toute modification du réseau. Par contre Bob à le contrôle de son réseau, qui accepte les connexion entrante sur sa machine, ce qui va nous permettre de procéder en sens inverse. C'est Alice que nous souhaitons joindre qui va créer une connexion vers Bob que nous appelerons tunnel. En effet il suffit à Bob d'écouter sur le port de son choix qu'Alice connait, il attend qu'Alice se connecte dessus et lui transmet sa requête HTTP. De son côte Alice va recevoir la requête HTTP de Bob, qu'elle relaye à son serveur HTTP et renvoit la réponse HTTP par cette même connexion. C'est pour cela que cela s'appelle du reverse tunneling, c'est notre cible qui est à l'initiative de la connexion, autrement dis du tunnel.

2.4 Comparaison de solution de reverse tunnel

Après quelque jours de recherche, j'ai retenu trois solutions de reverse tunnel :

— OpenSSH
— Etherws
— Node Reverse Wstunnel

2.4.1 OpenSSH

OpenSSH est une suite d'outils SSH libre mettant à disposition un client et server SSH très complet. Hors le SSH permet d'initialiser un tunnel directement avec une commande SSH. Pour en revenir sur notre scénario précédent. Bob installe un serveur SSH sur sa machine (en 22), toujours accessible de l'extérieur par le nom de domaine bob.fr et rajoute sur son serveur SSH l'utilisateur alice. Elle va s'y connecter en précisant qu'elle ouvre un tunnel inverse de son port 80 sur lequel écoute son serveur HTTP, jusqu'au port de son choix par exemple le 8080. Bob peut

desormais accéder au serveur d'Alice en remontant le tunnel par l'adresse http ://localhost :8080.

L'action se résume par cette unique ligne de commande qu'Alice exécute sur sa machine :

```
$ ssh -NR 8080:localhost:80 alice@bob.fr
```

Bien évidemment cela fonctionne pour n'importe quel service en TCP, il suffit juste de choisir le bon port.

Avantages :
- Facile à mettre en place.
- Automatiquement sécurisé.

Inconvénients :
- Les microcontrôleur ne possède pas de client SSH, en développer un serait périlleux.
- Pour chaque port ouvert il faut une nouvelle connexion SSH.
- Un serveur SSH n'est pas optimiser pour maintenir une centaine de connexion permanente.
- Le port d'entrée et sortie du tunnel est choisi par le client.

2.4.2 Etherws

Etherws est un mini VPN utilisant des websockets comme tunnel, il est entièrement développé en python. La configuration se fait en deux étapes, d'abords la création de l'interface virtuel en TUN/TAP avec l'adressage de l'IP virtuel, puis la connexion au serveur (toujours celui de Bob qui est le seul à disposé d'un nom de domaine public). Le fait d'utiliser une interface virtuel permet d'utiliser tout les ports à la fois sans en préciser un en particulier qui doit être relié à un autre distant sur la machine de Bob.

Dans notre cas Alice va choisir d'appeler son interface virtuel « etherws0 »avec l'IP « 10.0.2.8 »et se connecter sur le serveur etherws de Bob (en 80). Pour accéder au serveur HTTP d'Alice, Bob doit remonter le tunnel en passant par l'IP virtuel d'Alice, soit comme cela http ://10.0.2.8 :80.

Commande nécessaire à la création du tunnel :

```
# etherws sw
# etherws ctl addport tap ethws0
# etherws ctl setif --address 10.0.2.8 --netmask 255.255.255.0 1
# etherws ctl addport client ws://bob.fr/
```

Avantages :
- Utilise les websockets.
- Peut être sécurisé en SSL/TLS.

Inconvénients :
- Monte une interface virtuel.
- L'IP virtuel est choisi par le client.

— Tout les clients peuvent communiquer ensemble par le réseau virtuel.
— Portage difficile, voir impossible sur microcontrôleur.

2.4.3 Node Reverse Wstunnel

Node Reverse Wstunnel est un tunnel inversé développé en javascript avec Node.js, utilisant les websockets. Il s'utilise sur le même principe que le tunnel inversé SSH mais en passant par un tunnel websocket au lieu d'un tunnel SSH. Bob lance sur sa machine le serveur tunnel inversé en 80 (avec node) et Alice s'y connecte avec le client fourni en précisant les deux ports qu'elle veut relier.

Si on reprend la même configuration que pour le tunnel SSH, la commande qu'Alice doit exécuter est :

```
$ ./wstt.js -r 8080:localhost:80 ws://bob.fr/
```

Bob peut donc accéder au serveur HTTP d'Alice en se connectant en http ://localhost :8080.

Avantages :
— Utilise les websockets.
— Peut être sécurisé en SSL/TLS.
Inconvénients :
— Le port d'entrée et sortie du tunnel est choisi par le client.

2.4.4 Choix final

Finalement nous avons décidé de choisir la solution de reverse tunneling en Node.js en websocket, mais en reprogrammant le client en C++ parce que le Node.js n'est pas portable sur toute les versions d'ARM. Le serveur restera en Node.js mais sera re-écris pour mieux répondre à nos besoins, c'est en effet une technologie facilement maintenable pour développer un serveur websocket évolutif. L'architecture de base de Node Reverse Wstunnel est la seule chose conservé dans son intégrité.

2.5 Proxy HTTP dynamique

Nous avons résolu comment accéder à une machine distante sans modifier son NAT ou connaître son IP. Cependant notre objectif est d'accéder à plusieurs machines, objets connectés à travers un nom de domaine unique pour chaque. Actullement si je reprends l'exemple précédent d'Alice et Bob. Alice fait tourner son serveur HTTP sur le port 80, en se connectant au serveur wstunnel de Bob, elle a demandé à relier son port 80 avec le port 8080 de Bob. Ainsi depuis internet, si on se connecte en http ://bob.fr :8080 on accéde au serveur HTTP d'Alice, car Bob

possède un nom de domaine qui pointe sur l'IP fixe de sa machine. Cela fonctionne parfaitement sauf qu'on voulait un nom de domaine sans devoir préciser le port, certes on pourrait déplacer le serveur wstunnel sur le port 8080 et remettre le tunnel du port 80 d'Alice au port 80 de Bob comme cela le nom de domaine serait bob.fr tout simplement. Mais qu'arriverait il si nous souhaitons rajouter une ou dix autre machine dans le même cas qu'Alice ? Il nous faudra bien utiliser des ports autre que le 80.

C'est pour résoudre ce problème que j'ai dû installer un proxy HTTP. Celui-ci est en fait un serveur HTTP qui va rediriger une requête ou une réponse HTTP sur un autre port en fonction du nom de domaine. Quand nous rentrons un nom de domaine dans un navigateur, celui-ci va résoudre le nom de domaine en IP et envoyer une requête HTTP sur cette IP, port 80 par défaut. C'est grâce à la requête HTTP que le proxy peut connaître le serveur cible sur lequel rediriger la requête. En effet une requête HTTP contient un champs « Host »qui contient le nom de domaine que l'utilisateur a saisi dans le navigateur, le proxy redirige donc en fonction de ses configurations sur tel ou tel port la requête et la réponse HTTP.

Dans notre cas Bob peut configurer son proxy HTTP qui écoute sur le port 80 de rediriger le nom de domaine « alice.bob.fr »en « localhost :8080 »(le serveur wstunnel est sur un autre port disponible).

Exemple de requête HTTP :

```
GET / HTTP/1.1
Host: alice.bob.fr
```

Malheureusement la pluspart des proxy HTTP

Conclusion et perspectives

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed non risus. Suspendisse lectus tortor, dignissim sit amet, adipiscing nec, ultricies sed, dolor. Cras elementum ultrices diam. Maecenas ligula massa, varius a, semper congue, euismod non, mi. Proin porttitor, orci nec nonummy molestie, enim est eleifend mi, non fermentum diam nisl sit amet erat. Duis semper. Duis arcu massa, scelerisque vitae, consequat in, pretium a, enim. Pellentesque congue. Ut in risus volutpat libero pharetra tempor. Cras vestibulum bibendum augue. Praesent egestas leo in pede. Praesent blandit odio eu enim. Pellentesque sed dui ut augue blandit sodales. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae ; Aliquam nibh. Mauris ac mauris sed pede pellentesque fermentum. Maecenas adipiscing ante non diam sodales hendrerit. Ut velit mauris, egestas sed, gravida nec, ornare ut, mi. Aenean ut orci vel massa suscipit pulvinar. Nulla sollicitudin. Fusce varius, ligula non tempus aliquam, nunc turpis ullamcorper nibh, in tempus sapien eros vitae ligula. Pellentesque rhoncus nunc et augue. Integer id felis. Curabitur aliquet pellentesque diam. Integer quis metus vitae elit lobortis egestas. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Morbi vel erat non mauris convallis vehicula. Nulla et sapien. Integer tortor tellus, aliquam faucibus, convallis id, congue eu, quam. Mauris ullamcorper felis vitae erat. Proin feugiat, augue non elementum posuere, metus purus iaculis lectus, et tristique ligula justo vitae magna. Aliquam convallis sollicitudin purus. Praesent aliquam, enim at fermentum mollis, ligula massa adipiscing nisl, ac euismod nibh nisl eu lectus. Fusce vulputate sem at sapien. Vivamus leo. Aliquam euismod libero eu enim. Nulla nec felis sed leo placerat imperdiet. Aenean suscipit nulla in justo. Suspendisse cursus rutrum augue. Nulla tincidunt tincidunt mi. Curabitur iaculis, lorem vel rhoncus faucibus, felis magna fermentum augue, et ultricies lacus lorem varius purus. Curabitur eu amet.