Création d'un serveur de Monitoring réseau

Technique de Développement LOGiciel

2012-2012

Projet

Étienne Gaillard de Saint Germain
Alexandre Sarrazin
Raphaël Sivera

Le club informatique de l'école des Ponts ParisTech (KI) est en charge d'une grande partie de la gestion des résidences. Parmi ses attributions, on peut compter la gestion de mailing-list, la modération des mails, le dépannage des ordinateurs personnels des élèves, et également la gestion et l'entretien de l'architecture réseau permettant de dispenser une connexion Internet dans les résidences. Dans ce but, le club informatique possède plusieurs serveurs situés dans le local (P401), ainsi que du matériel dispersé au sein des résidences (tant dans des armoires au sein des résidences que dans un local technique central au sous-sol de l'une de celle-ci). Lorsqu'un problème survient sur le réseau, il est nécessaire de pouvoir réagir au plus vite en identifiant le plus précisément possible l'origine du défaut, afin de réduire au minimum l'interruption de la connexion internet.

La gestion et le bon entretien du réseau et du matériel nécessitent donc une vue d'ensemble globale de l'infrastructure en place. Aussi, automatiser la recherche de l'origine du défaut et les tâches de diagnostic permettrait de gagner un temps précieux en investigations, et également de faciliter les interventions des membres du KI.

Dans le cadre du club informatique (KI) de l'école des ponts et chaussées, nous nous proposons de programmer un ensemble d'applications et de scripts permettant de faciliter le monitoring du réseau sous la charge du club (entendre le réseau informatique des résidences Meunier et Perronet), et d'y ajouter une interface web donnant une vue d'ensemble de l'état du réseau en temps réel.

REMARQUES PRÉALABLES

Ce projet visant à s'insérer dans une architecture complexe déjà en place, il convient de s'assurer du bon fonctionnement des différentes applications et manipulations avant de l'intégrer réellement aux serveurs dont le KI dispose actuellement. C'est pourquoi, dans l'état actuel des choses, les applications n'ont pas encore été intégrées au serveur en question, et pour faciliter la présentation du projet, les manipulations seront réalisées sur l'un des ordinateurs des membres du projet. Le groupe se réserve l'incorporation des fonctionnalités développées dans un futur proche.

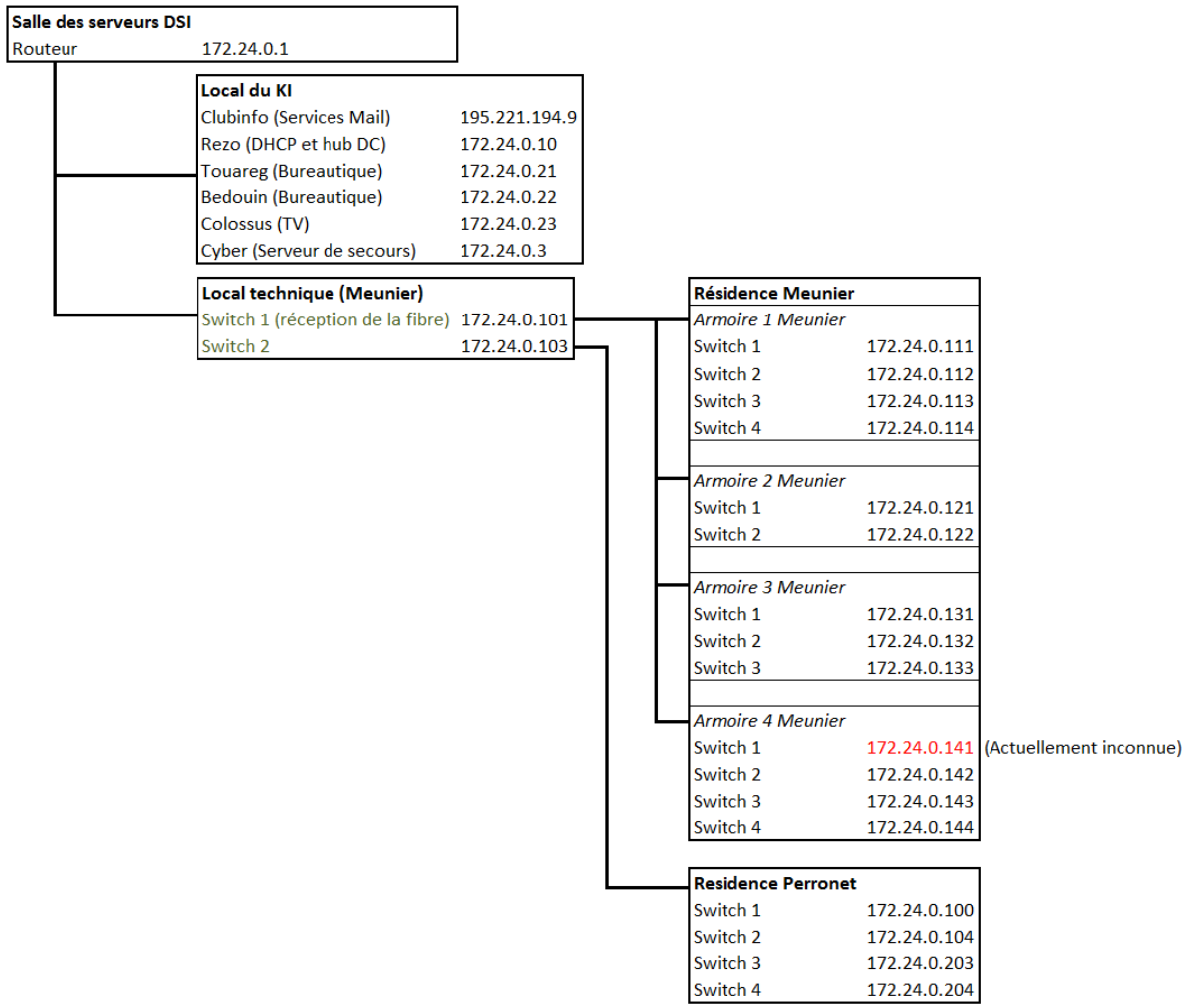
Dans l'optique d'un travail collectif et collaboratif, l'ensemble des fonctionnalités développées n'ont pour but que de favoriser le diagnostic du réseau. Aussi, il n'y a pas de "fin" de développement dans un axe de recherche, et on proposera des pistes pour des améliorations possibles par les futures générations de KIens/ennes.

L'ensemble des connaissances qu'ont réussi à regrouper les membres de ce projet, qu'elles aient été implémentées ou non, seront présentées ici, afin de faciliter le travail de recherche des prochaines personnes travaillant sur ce projet. Il ne faut donc pas s'étonner de voir certaines parties rédigées à la manières d'un mode d'emploi, ou comportant des explications non exhaustives. Tout le travail réalisé, sans distinction de réussite ou d'utilité immédiate, sera donc présenté ici.

Sommaire

I. PLAN DU RESEAU DES RESIDENCES

Il convient pour la pleine compréhension du lecteur de débuter par le plan de l'architecture actuelle du réseau.



L'ensemble des applications va être implémenté sur *Clubinfo*, afin de faciliter la gestion de l'affichage web.

Les plus anciens ont été installés par les '09 donc pas 2011-2012. Ages hétérogènes...

Les switchs numéro 100,101,103 et 104 sont des HP J4813A, et tous les autres sont des NETGEAR fs726t, changés dans le courant de l'année 2011-2012.

L'adresse IP du switch 141 n'est actuellement pas configurée en adresse IP fixe; il faudra donc aller physiquement la régler lors d'un passage dans l'armoire correspondante.

Sauf erreur, c'est le switch maudit avec une mise à jour de firmware foirée ; il y a un peu plus qu'un problème d'IP fixe donc

II. Gestion des donnees : Rezo et Clubinfo

A. Logs et Donnees disponibles

Notation pas standard. Sinon ça attribue réellement jusqu'à 172.24.255.255 ?

Actuellement, *Rezo* fait office de DHCP sur le réseau des résidences (il attribue les adresses IP), et attribue des adresses dans la plage 172.24.150+.* .Nous avons cherché à exploiter les logs présents sur ce serveur pour pouvoir récupérer de manière lisible les sessions d'attribution des adresses IP (leases).

De plus, *Rezo* fait également office de hub direct connect, un service de partage de fichiers pourvu par le club informatique. Les utilisateurs s'identifient et peuvent s'ils le souhaitent ajouter un descriptif associé à leur compte (leur numéro de chambre par exemple). Cela permet d'associer adresse mail (requise pour l'inscription au service DC) avec un numéro de chambre, et donc de contacter plus facilement les utilisateurs.

B. Mise en forme

L'idée principale est de créer un outil permettant de regrouper toutes les informations disponibles sur les différents serveurs du KI : adresses IP des personnes connectées, adresses MAC associées, nom de la machine, dates de début et de fin d'attribution des adresses IP.

La solution retenue a été de créer une base de données utilisable simplement pour afficher ces informations. Après avoir sélectionné les données à garder et choisi les formats de stockage, les premiers problèmes ont été, au delà de la gestion de multiples fichiers et de la recherche d'information, la manipulation et la conversion en entier de dates et d'adresses (IP et MAC). Il existe pour cela en python des bibliothèques adaptées à la résolution de ces problèmes : *time*, *datetime* et *netaddr*.

Deux scripts ont été écrits, *traitement_dhcp_leases.py* et *traitement_dcpp.py, qui* prennent en entrée les logs bruts et renvoient chacun une base de donnée avec les données recquises.

C. Mise a disposition

Attention, l'accès extérieur n'a rien à voir avec les entrées DNS ; par exemple : clubinfo2.enpc.fr => 172.24.0.3, et peut-être toujours rezo.enpc.fr => 172.24.0.10 ; ce qui compte, c'est d'avoir une IP publique. De plus, pas d'accès à clubinfo en HTTP...

Outre la difficulté d'exporter les logs depuis *Rezo*, il est préférable que le projet soit hébergé par un même serveur, *Clubinfo*. En effet, celui-ci dispose d'une entrée DNS fournie par la DSI et donc un accès extérieur, et gère l'hébergement du site web du KI, http://clubinfo.enpc.org/. Le problème est que l'hébergement en question est réalisé par Apache, et, en l'état actuel de nos connaissances, il nous était plus efficace de mettre en place un hébergement en python, à l'aide de la librairie *web.py*. Il faudra donc reprendre dans un futur proche la méthode de génération des pages html, afin de rester totalement cohérent à l'intérieur du site internet.

Non : clubinfo n'héberge pas le site du KI, qui est sur un serveur OVH ; vous allez vous faire allumer s'il tape "dig a clubinfo.enoc.org" dans un terminal :)

Nous avons donc mis en place un système automatique de connexion ssh depuis *Clubinfo* vers *Rezo*, permettant de récupérer les logs sur ce dernier. Une tache cron s'effectue quotidiennement sur *Rezo* pour

Attention au vocabulaire à propos de l'hébergement en python : Apache supporte parfaitement le python via un module dédié ; d'ailleurs, mailman est en python, interface web comprise ; bref, il y a tout ce qu'il faut sur clubinfo pour avoir du HTTP by python.

préparer les fichiers (notamment pour le droit de lecture), et une deuxième dix minutes plus tard sur *Clubinfo* pour établir une connexion ssh avec *Rezo* grâce à une clé RSA et récupérer les dits fichiers. S'ensuit ensuite l'exécution des deux scipts décrits précédemment pour la mise en forme de la base de donnée.

D. AMELIORATIONS POSSIBLES

Une des optiques d'amélioration possible est de recouper directement les informations disponibles. Cela commencerait par une bonne jointure des bases de données disponibles. Il serait également possible de recouper les données avec d'autres sources d'informations, notamment si ces données sont statiques (par exemple, il est peu probable qu'une adresse MAC change de chambre et de nom d'utilisateur, ou qu'un changement dans le plan du réseau soit effectué), et d'optimiser ainsi la mise à jour des données aux seules variables dans le réseau.

Il pourrait également être utile de récupérer certaines données en temps réel, comme l'association MAC-IP, grâce aux tables de routage des switchs, pour ceux dont elle est facilement récupérable (ce qui n'est pas le cas actuellement, comme nous le verrons par la suite).

Déjà, on ne parle de table de routage (couche 3) que pour les routeurs. Pour les switchs, il faut préférer "table de commutation".

Ensuite, les tables de commutation ne contiennent que des associations MAC/port, donc pas l'IP.

Enfin, on pense quand même beaucoup plus à l'ARP pour l'association MAC/IP :)

III. Outils de monitoring live

A. Introduction

Sauriez vous justifier comment vous avez utilisé netcat ^^ ? Accessoirement, simple check, vous n'utilisez/ mentionnez pas le arpwatch sur touareg ? De même pour le dhcpdump/loc ou je ne sais plus quoi sur rezo ?

L'idée est de fournir des outils permettant de simplifier la surveillance de l'état du réseau, ainsi que d'accélérer le diagnostic en cas de défaut. On utilise un mélange d'outils réseau élémentaires, comme *ping*, *ifconfig*, *arp*, *arping*, snmpwalk, *snmpget*, *snmpset*, *netcat*, *telnet*, *ssh*..., ainsi qu'une bibliothèque python très pratique *scapy*. Cette bibliothèque permet entre autre de forger des paquets, de les afficher, de les envoyer et de recevoir des réponses.

Scapy est un mot magique ; c'est /la/ référence, un vrai couteau suisse, et c'est cool que vous ayez regardé.

On peut citer deux situations problématiques habituelles, qui figurent parmi les problèmes les plus récurrents sur le réseau des ponts.

- une partie du réseau a été déconnectée, ou a des latences extrêmes, avec une impossibilité de contacter le serveur proxy

- la présence d'un DHCP étranger a été détectée (DHCP rogue).

B. Protocole SNMP (Simple Network Management Protocol)

Il y a un distingo à faire ici : les FS726T sont parfaitement compatibles avec le protocole SNMP ; de fait, la RFC 1157 SNMP v1 doit plus ou moins être la seule qu'ils implémentent correctement. Seulement le SNMP définit le protocole de communication, et pas ce qui doit être récupérable. La bonne formulation serait plutôt "n'implémentent que partiellement les RFC 1643 (Ethernet interface MIB) et 1493 (Bridge MIB) de sorte que seules quelques données [...] sont récupérables par SNMP"

Après de nombreuses recherches, il s'avère que les switchs Netgear principalement utilisés sur le réseau (FS726t) ne sont pas compatibles avec le protocole SNMP (ou, pour être plus précis, les fonctionnalités disponibles grâce à ce protocole sont moindres). Il n'est par exemple pas possible de récupérer le taux d'utilisation du CPU, la température ou la vitesse de rotation des ventilateurs, mais il est possble de récupérer le nom du switch, son emplacement, et d'autres informations moins utiles.

En revanche, cela est possible sur les quatre switch HP (100, 101, 103 et 104), et notamment sur les switchs centraux du réseau, à savoir ceux du local technique de Meunier. En utilisant le protocole SNMP, il est possible de récupérer par exemple, en connaissant simplement l'adresse IP du switch à interroger, et en utilisant la commande

C'est secondaire (et je suis partiellement coupable) mais ça ne fait pas super sérieux le "-c public" ; c'est exactement comme laisser un

```
snmpwalk -v1 -c public Adresse_du_switch_à_interroger OID
```

mot de passe par défaut ; peut-être changer le nom de communauté ou une note de bdp pour dire qu'il y a filtrage sur les IP qui peuvent se connecter...

L'adresse MAC du switch OID : .1.3.6.1.2.1.2.2.1.6.1
Le pourcentage d'utilisation de son CPU OID : .1.3.6.1.4.1.11.2.14.11.5.1.9.6.1
La mémoire totale du switch OID : .1.3.6.1.4.1.11.2.14.11.5.1.1.2.1.1.1.5.1
La mémoire utilisée du switch OID : .1.3.6.1.4.1.11.2.14.11.5.1.1.2.1.1.1.6.1
La table de routage du switch OID : . 1.3.6.1.2.1.17.4.3.1.2

L'utilisation avisée de ces commandes permettra d'accélérer le diagnostic en cas de défaut sur les switchs concernés. Par exemple, un tauc d'utilisation du CPU d'un switch proche de 90% après une maintenance et un changement de matériel pourrait diagnostiquer une tempête de broadcast, beaucoup plus rapidement qu'en étant obligé d'aller physiquement travailler sur le réseau.

Note purement technique : je ne suis pas sur qu'une tempête de broadcast soit particulièrement visible au niveau CPU ; je n'y connais rien en architecture switch, mais quand même, les opérations standard sont en hardware et ne remontent pas au CPU. Pour le cas du broadcast donc, aucune idée de si ça remonte ou pas.

Distingo à apporter : la numérotation est standardisée par RFC, et une partie des OID est commune à tout le monde ; simplement, la RFC a prévue des sous arbres de numéros qui seraient réservés constructeurs (qui y feraient ce qu'ils voudrait) juste pour que des fonctionalités spécifiques à un constructeur puissent être également accessibles.

La difficulté de l'utilisation du protocole SNMP reste la recherche des OID correspondant aux informations souhaitées. En effet, il ne s'agit pas de standards, et ils diffèrent même sur deux matériels provenant du même constructeur. Une amélioration possible serait de récupérer en détail la MIB relative aux switchs HP concernés, et regarder si d'autres informations pourraient être utiles. Il ne semble pas y avoir de capteurs de température sur les modèles de switchs que nous possédons, et il faudrait vérifier que ne pouvons effectivement pas avoir accès à la température du CPU, intérieure, extérieure, ni à la vitesse du ventilateur.

C. Scan general

Hum, personnellement je ne laisserai pas passer un "plus ou moins" sans appuyer là où ça fait mal. C'est un threshold lié au contrôle du broadcast qui fait que certains paquets passent et d'autres pas ? Êtes-vous sur que c'est les switchs, où n'est-ce pas certaines machines qui refusent de répondre aux pings en broadcast (mauvaise réputation à cause d'attaques ddos) ?

Nous cherchons ici à réaliser un un ping en broadcast et à observer la réponse. La commande *ping -b -i 2 -c 2 172.24.255.255* donne déjà de bons résultats, mais a de nombreuses limites (les réponses se bousculent et deviennent illisibles, certains switchs filtrent plus ou moins les paquets, et l'on obtient une trentaine de réponses au maximum alors qu'il y a plus de 130 personnes sur le réseau). Par ailleurs, le protocole ICMP (niveau 4 OSI) est limité au cas où l'IP est accessible et ne nous apprend pas grand chose. En particulier un broadcast arping (requête ARP, donc entre le niveau 2 et 3, de type who-as envoyé vers une mac de broadcast 'ff:ff:ff:ff:ff:ff') permet d'obtenir le couple MAC/IP de tous les hôtes. Pour éviter les chevauchement de réponses et l'utilisation maximale du CPU du switch principal (172.24.0.101), on étale la procédure.Ceci nous permet, en quelques minutes, de parcourir l'intégralité de la plage 172.24.0.0/16 et d'obtenir des réposes d'*a priori* toutes les machines présentes sur notre réseau.

Non, ICMP = niveau 3 OSI

Là je ne vous suis pas du tout. Une requête ARP se fait effectivement en broadcast, mais n'est censée générer qu'une unique réponse en unicast de la part de la machine qui possède l'IP demandée dans le paquet d'origine. Vous voulez dire que vous balayez la plage ? Parce que tel quel, on comprend qu'un unique arping génère les réponses de tout le monde... Et sinon, soyez prêts à justifier un balayage en arping plutôt qu'en ping par le fait que les machines refusent beaucoup plus rarement de réponre à ARP qu'à ICMP.

D. Recherche d'un individu

En théorie, au niveau 2, il n'est pas nécessaire de connaître l'adresse IP du destinataire. On pourrait donc en théorie envoyer un paquet en se basant uniquement sur une adresse MAC. Le problème peut paraître accessoire, mais est très utile dans la problématique de recherche de DHCP rogues. En effet, la détection de DHCP est une problématique de premier plan pour le club informatique: lorsque qu'un étudiant a connecté innocemment dans sa chambre un routeur (principalement wi-fi) mal configuré (qui cherche à faire DHCP sur l'ensemble du réseau), son adresse IP est souvent hors domaine (donc injoignable).

Ceci nous permet de connaître l'ip (par ex: 192.168.0.1) et la mac (00:11:22:33:44:55) du nouveau routeur. L'idée est d'envoyer des paquets mal formés qui déclenchent des erreurs aux niveaux des switches traversés, permettant ainsi de localiser le destinataire et par conséquent le DHCP. Par exemple, actuellement, nous savons qu'un switch mal configuré possède l'ip 192.168.0.239 (il s'agit manifestement du switch 141).

1) En théorie

L'idée est de s'inspirer directement des attaques ping flood ou ARP poisoning en utilisant un paquet ICMP truqué comme suit :

1) En couche 3, le paquet est un paquet de ping broadcast adressé à 255.255.255.255.
2) En couche 2, la mac destinataire a été modifiée pour ne pas être la mac recherchée : 00:11:22:33:44:55

pour être (on veut que ça arrive dessus). La faute est de moi ?

Il faudrait rechercher le nom du compteur/de l'erreur, sinon ça fait un peu tiré de derrière les fagots... Me redemander le cas échéant là je n'ai pas ça en tête

Le paquet va bien arriver jusqu'à la machine parasite (bonne adresse MAC, adresse IP compatible), mais la machine ne va pas y répondre (l'adresse IP émettrice n'est pas dans son domaine).Dès lors, il y a génération d'erreurs au niveau du port du switch, puisque la machine recherchée refuse de répondre à des paquets ICMP. Se comportant ainsi de manière non standard il y a activation très discriminante d'un compteur spécifique à un port. Il suffirait donc au préalable de mettre à zéro les compteurs d'erreurs du switch à surveiller (par exemple grâce à une requête snmpset pour les switchs disposant de cette fonctionnalité) puis de regarder quel port a été activé. On répète cette opération de père en fils dans l'arborescence de réseau pour descendre au port du switch armoire associé, et on utilise le plan physique du câblage pour connaître la chambre de la machine.

2) En pratique

Heu, ça veut dire quoi "ne donne pas beaucoup de résultats" :p ?

Malheureusement que ce soit avec arping ou avec scapy, nous nous heurtons à un problème de routage, et ne parvenons pas à envoyer un tel paquet. La modification des routes par défaut ne donne pas beaucoup de résultat, mais devrait toutefois pouvoir se faire facilement, et même appliquer les changements uniquement dans la session scapy. Pour l'instant, la méthode utilisée est de changer sa propre adresse IP (par exemple en 192.168.0.2) pour mette à jour les tables et envoyer un paquet truqué en modifiant en plus l'adresse émetteur pour la mettre dans un autre domaine.Le paquet est alors envoyé, et il peut ne pas y avoir de réponse. Le problème majeur de cette méthode est que nous sommes alors hors domaine ; il donc difficile de surveiller les compteurs d'erreurs à partir de la même machine. La méthode n'est donc pas optimale, et n'a pu être utilisé en pratique. Actuellement, nous sommes limités à l'heuristique qui consiste à générer brutalement beaucoup de trafic vers une IP qui est dans le bon domaine (utilisable pour vérifier l'arborescence).

Toujours une question de distingo : tous implémentent SNMP ; mais tous n'implément pas les objets qui sont nécessaires pour tel ou tel objectif

Il existe une autre limitation à plus long terme : la plupart des switchs ne sont pas administrables par SNMP, il faut donc se connecter manuellement à leur interface webpour contrôler les compteurs d'erreurs. On ne pourrait donc pas automatiser entièrement le processus avec le matériel à disposition.

E. DHCP Rogue

Cette méthode peut-être contestée : elle revient un peut à dosser le DHCP qui répond le plus rapidement ; on a plutôt envie d'émettre un unique DHCP request, mais d'attendre longtemps les réponses de tout le monde

Le principe le plus simple est d'émettre un grand nombre de DHCP request et de regarder les machines qui répondent. Cette méthode a le mérite d'être facile à exécuter rapidement, mais notre DHCP est normalement configuré pour qu'il nous prévienne automatiquement (par mail) lorsqu'il détecte un autre serveur DHCP. On obtient donc les adresses (MAC et IP) de l'intrus, qu'il faut alors localiser sur le réseau par les méthodes citées précédemment.

IV. INTERFACE DE L'APPLICATION

Les informations ainsi collectées seront affichées sur le site du club informatique, en générant des pages html grâce à la librairie *web.py*. Comme dit précédemment, il pourra être judicieux de se passer de cette librairie pour homogénéiser le site.

Actuellement, pour lancer l'interface, il suffit d'exécuter le fichier *interface.py*,, et de se rendre http://localhost:8080/index.html afin d'utiliser ses fonctionnalités (lecture et tri des logs, visualisation de l'arborescence du réseau et des nœuds défectueux, exécution de quelques commandes). Nous nous concentrerons ici donc sur son implémentation.

A. ORGANISATION DES DOSSIERS

Le dossier Base_de_donnees

Ce dossier contient en particulier la base de données contenant les logs nommée leases.db (adresse IP, adresse MAC, date de début d'attribution de l'adresse IP, date de fin d'attribution et éventuellement le nom d'utilisateur lorsque celui-ci est connu). En recroisant avec la base de données utilisateur.db, nous pourrions probablement compléter davantage les noms d'utilisateur.

Le dossier static

Ce dossier contient le CSS et les images que l'on retrouve sur les différentes pages du site.

Le dossier codeHTML

Ce dossier contient les codes HTML des différentes pages du site. Ces codes sont incomplets car certaines parties doivent être calculées en fonction des entrées sur la page d'index. Ces zones sont matérialisées par une ligne de la forme « |***partie_a_calculer***| » et calculés par la classe correspondante dans le fichier interface.py.

B. LA PAGE D'INDEX

La page d'index permet d'afficher plusieurs informations :

- l'arborescence des switchs
- les logs de connexion
- la présence d'un DHCP rogue sur le réseau. (Celle-ci se fait par un message d'alerte en haut de la page lorsque c'est le cas. Pour cela, la classe page_index regarde la valeur stockée dans le fichier dhcp_rogue.txt dans le dossier Scripts. La valeur 1 indique la présence d'autre DHCP et la valeur 0 indique leur absence.)

En outre, la page d'index permet de relancer manuellement certains scripts dont le lancement est périodique. On notera :

- *Test_IP_connues* qui effectue des ping sur l'ensemble des adresses IP des switchs afin de vérifier que ceux-ci fonctionnent toujours. Les résultats sont notés dans un fichier texte (*switchs_connus_reponse_ping.txt*), qui est consulté à chaque affichage de l'arborescence afin d'afficher en rouge les switchs ne répondant pas.

- *ping_IP* qui permet à l'aide d'une fenêtre d'effectuer un ping sur n'importe quelle adresse IP. Le résultat est stocké dans *pingIP.txt* puis affiché.

C. AFFICHAGE DES SWITCHS

L'arborescence des switchs pouvant évoluer au fil des années, elle est recalculée à chaque nouvel affichage. On notera que cela se fait par un parcours en profondeur de l'arbre des swicths, qui reste très rapide au vu de la taille du réseau. Cet affichage dynamique permet de modifier l'affichage en ne modifiant que l'arbre des switchs dans le fichier *switch.py*. En outre, on peut au fur et à mesure des tests modifier l'affichage pour faire ressortir les switchs défectueux.

D. L'AFFICHAGE DES LOGS

L'affichage des logs se fait à l'aide de la librairie *sqlite3*. Une requête SQL est envoyée à la base de donnée *leases.db* puis la classe *page_logs* se charge de générer le tableau correspondant afin de faciliter la lecture des logs. Pour former cette requête SQL, on utilise un formulaire demandant les informations à afficher et éventuellement l'ordre de tri. On peut également envisager d'afficher une fenêtre dans laquelle l'utilisateur entrerait intégralement sa requête SQL.

CONCLUSION

Ce projet nous a permis de nous confronter à de nombreuses problématiques liées à la gestion du réseau, et de trouver des méthodes pour les surmonter. Le sujet étant extrêmement large, la façon d'appréhender la construction de tant d'outils était nouvelle pour nous. Nous n'avions pas un but unique et précis, comme par exemple coder un logiciel bien particulier, mais réussir à récupérer toutes les informations, toutes les méthodes et tous les outils qui seraient nécessaires et utiles à la création du serveur de monitoring réseau. Il s'agit donc bien d'une expérience bien particulière, qui peut être déstabilisante au premier abord, face à la sensation de ne pas avancer dans un sujet bien trop large. Mais nous avons réussi à nous concentrer sur des aspects particuliers, dont certains n'ont malheureusement pas aboutis, mais qui au final n'ont fait que nous faire acquérir de nouvelles compétences et connaissances.