Du log centralisé des connexions au taux d'utilisation des salles pédagogiques : *QoQ-Cot*, une solution modeste mais libre...

Frédéric Bloise, Frédéric Giudicelli, Gérard Milhaud, Arnaud Salvucci
Université d'Aix-Marseille – Direction Opérationnelle du Système d'Information
Polytech' Marseille Bt A – Campus de Luminy – 163 Avenue de Luminy, 13288 Marseille Cedex 9

Résumé

*Nous présentons ici notre solution libre, baptisée QoQ-CoT, permettant d'une part de répondre aux obligations légales de journalisation des connexions sur les postes du parc informatique (QoQ : **Q**ui, **o**ù, **Q**uand) et d'autre part d'exploiter ces informations de connexion pour illustrer graphiquement le taux d'utilisation des machines (CoT : **Co**mbien de **T**emps) ou groupe de machines (salles pédagogiques, groupe quelconque de salles composant une entité, un thème...).*

Pour la journalisation, nous nous appuyons sur un serveur rsyslog centralisant dans une base de données (MySQL) les journaux de connexion issus de tous les clients. Nous détaillerons les configurations nécessaires à effectuer sur le serveur et les clients, puis donnerons un aperçu de l'interface PHP permettant d'interroger cette base de données selon de multiples critères.

Pour la visualisation du taux d'utilisation des salles (ou groupe de salles), nous avons enrichi l'interface PHP afin de permettre la génération de divers graphes dont nous donnerons le détail : taux d'utilisation en regard de l'utilisation maximale, maximum de machines utilisées par jour sur une période donnée, etc.

Nous listerons ensuite quelques exemples d'utilisation illustrant l'intérêt potentiel de la solution en tant qu'outil de pilotage et d'aide à la décision, puis indiquerons les prérequis nécessaires à l'installation d'une QoQ-CoT chez vous.

Nous terminerons par les pistes d'amélioration de la version actuellement disponible.

Mots-clefs

journalisation centralisée, taux d'utilisation, gestion de parc, salles pédagogiques, aide à la décision, rsyslog, MySQL, PHP, pChart.

1 Introduction

Dans notre société globalisée et fortement connectée, une université (ou un EPST quelconque, ou un cyber-café, etc.) est désormais assimilée par la loi à un *opérateur de communications électroniques* qui se doit, à ce titre, d'effacer ou rendre anonyme toute donnée relative au trafic (article L34-1[1], alinéa 2). Cependant, via l'alinéa 3, la loi s'auto-modère en indiquant que, pour certaines données techniques des communications que l'opérateur permet à ses usagers, cet effacement *doit* être différé d'une durée maximum d'une année, *à la seule fin* de fourniture d'informations aux autorités judiciaires. L'alinéa 6 précise que ces données conservées ne doivent porter que sur l'identification des usagers, les caractéristiques techniques des communications et la localisation des terminaux. C'est enfin un décret de 2006 qui précise quelles données sont à conserver[2]. Dans le cadre de la surveillance des connexions d'un parc de postes de travail informatiques, on en retiendra trois :

- *les informations permettant d'identifier l'utilisateur* ;
- *les données relatives aux équipements terminaux de communication utilisés ;*
- *les caractéristiques techniques ainsi que la date, l'horaire et la durée de chaque communication.*

Finalement et pour synthétiser, nous, responsables de parcs informatiques permettant l'accès à l'Internet à nos usagers, devons donc être à même de répondre, sur demande des autorités judiciaires compétentes, à une seule question : « *Qui était connecté sur cette machine à ce moment-là ?* »

1. Article L34-1 du Code de la Poste et des communication électroniques
2. Décret du 24/03/2006, qui crée l'article R10-13.

Une question que l'on pourra reformuler dans le slogan suivant : « Qui, où, quand ? », finalement abrégé dans le mnémotechnique, graphique et patriotique *QoQ*.

Sous couvert d'*optimisation des fonctions supports*, la tendance actuelle est à la création de services informatiques mutualisés par regroupement des anciens services de proximité : l'ordre de grandeur des parcs gérés est ainsi souvent passé de la centaine au millier. Cet accroissement très significatif rend plus critique et plus difficile la réponse à QoQ . D'une part, la probabilité de problèmes de sécurité a proportionnellement augmenté et ne pas savoir répondre à QoQ devient donc beaucoup plus risqué . On peut citer, à ce sujet, B. Parker : *un grand pouvoir implique de grandes responsabilités*[3]. D'autre part, la récolte des informations nécessaires à cette réponse devient plus difficile de par l'accroissement du nombre de machines (et donc potentiellement d'OS), l'inévitable complexification du réseau, etc.

Nous présentons donc ici tout d'abord notre méthode fondée sur le gestionnaire de journaux *rsyslog*, le système de gestion de bases de données relationnelles MySQL et une interface web PHP permettant de répondre très vite et très facilement à la question QoQ sur l'ensemble des machines du parc (Linux, MacOSX, Windows XP et Seven).

Quelque temps après la mise au point de ce système, alors que nous profitions du doux sentiment de travail accompli, allongés dans les hamacs de la salle serveur, une évidence explosa soudain à notre conscience : nous avions dans notre base de données bien plus que la réponse à QoQ ! En effet, puisque nous connaissions avec précisions les dates de début et de fin de toutes les connexions, sur tout le parc, alors nous disposions de toute l'information nécessaire pour évaluer finement le taux d'utilisation de toute machine, et donc, par extension, de tout groupe de machines...

Et s'il est un chiffre après lequel courent désespérément tous les décideurs responsables (et propriétaires...) de parcs informatiques destinés aux usagers (directeurs de composantes universitaires, DSI, etc.), c'est bien **le taux d'utilisation *réel* de leurs machines**... Un chiffre-clé d'aide à la décision mais qui, à défaut d'une mesure précise difficile à réaliser, est plus souvent *estimé* que calculé : dans le meilleur des cas, on s'appuie sur les horaires de réservation (mais les machines ont-elles été utilisées pendant ces heures ? Et si oui, combien ? Et *quid* des heures de libre-service ?) ; dans le moins bon, c'est un combiné de *sensations* personnelles plus ou moins éclairées qui va faire foi (« il y a toujours du monde là-bas » ; « ça m'étonnerait bien qu'il y ait des étudiants après 19h... »).

Nous proposons donc une deuxième interface PHP exposant cette fois, de manière graphique, un grand nombre d'informations sur le taux d'utilisation *réel* des machines, salles et groupes de salles, afin de fournir une véritable aide à la décision pour l'évolution du parc informatique. Cette deuxième interface, liée à la première, répond cette fois à la question « *Combien de temps ?* », abrégée dans un souci de cohérence en *CoT*.

Partons donc maintenant ensemble à la découverte de l'ensemble de notre système, logiquement baptisé *QoQ-CoT*. Notons que nous parlerons plusieurs fois par la suite de l'*archive* de QoQ-CoT, pour faire référence à la version téléchargeable[4] sur le site communautaire SourceSup.

2 Répondre à la question QoQ : Qui, où, quand ?

2.1 Méthode

Pour pouvoir répondre à la question QoQ, nous allons faire en sorte que chaque poste géré envoie les informations de connexion/déconnexion à un seul serveur de journalisation, qui les stockera dans une base de données. Cette base de données sera ensuite interrogeable au travers d'une interface web, selon de multiples critères (nom d'utilisateur, nom de machine, date de début/fin de connexion, système d'exploitation).

Le choix de passer par un serveur de journalisation central plutôt que de faire écrire directement les clients dans la base correspond à une volonté de gestion unifiée, de minimisation des configurations clients et de protection d'accès à la base. De plus, le serveur central nous permet de filtrer finement les messages reçus des différents systèmes d'exploitation afin d'envoyer uniquement les informations d'ouverture/fermeture de session dans la base.

3. Benjamin Parker à son neveu. Spider Man. Columbia Pictures, 2002

4. « Chez QoQ-CoT », ouvert 24/24, 7/7 : https://sourcesup.renater.fr/projects/qoq-cot/

2.2 Centralisation des logs clients de connexion/déconnexion : rsyslog

Nous avons choisi le gestionnaire de journaux *rsyslog* pour le serveur unique qui recevra les informations de tous les clients et les inscrira dans la base de données. Les raisons de notre choix sont bien sûr liées à l'analyse de plusieurs articles et autres comparatifs, mais les deux principales sont sans doute d'une part qu'il est libre sans aucune limitation (par opposition à son concurrent *syslog-ng* qui propose une version libre et une « premium », non libre) et d'autre part qu'il est le gestionnaire par défaut de la distribution Debian, utilisée sur l'ensemble de nos serveurs.

Pour la lisibilité de ce qui suit, instancions quelques *variables* par des valeurs d'exemple :

- nom DNS du serveur rsyslog : *RSYSLOG.QoQ-CoT.ORG* ;
- nom de la base MySQL dans laquelle *rsyslog.qoq-cot.org* stocke les informations : *RSYSLOG_DB* ;
- nom de l'utilisateur MySQL qui écrit dans *RSYSLOG_DB* : *RSYSLOG_DB_USER* ;
- mot de passe MySQL de *RSYSLOG_DB_USER* : *RSYSLOG_DB_USER_PASSWD*.

2.2.1 Configuration des clients Linux

L'idée est de faire en sorte que le serveur de journalisation local envoie tous les messages d'authentification sur le serveur *rsyslog* central. La configuration dépend du type de serveur local (*syslog*, *syslog-ng*, *rsyslog*), mais reste très simple dans tous les cas. On en trouvera le détail dans répertoire *ressources* de l'archive QoQ-CoT.

2.2.2 Configuration des clients MacOSX

Première idée, logique : MaCOSX est un Unix et dispose d'un serveur de journalisation (*syslog* standard) local, utilisons donc exactement la même configuration que pour Linux. Mais Apple *pense différent*, c'est bien connu... : en effet, avec cette configuration, seules les ouvertures de session sont consignées, jamais les fermetures.

La solution finalement choisie consiste à forger, à l'ouverture comme à la fermeture de session, un message qui sera envoyé au serveur *syslog* local du Mac l'aide de la commande *logger*. Pour cela, on utilise un procédé appelé *hook,* qui permet d'associer une action à un événement. Ensuite, il ne restera plus qu'à indiquer au serveur *syslog* local d'envoyer ces messages vers le serveur *rsyslog*.

Nous vous renvoyons au répertoire *ressources* de l'archive QoQ-CoT pour récupérer le détail de la configuration. Retenons simplement que deux *hooks* seront utilisés, l'un déclenché à la connexion, l'autre à la déconnexion.

2.2.3 Configuration des clients Windows XP et Seven

Le principe consiste cette fois à utiliser un logiciel tiers permettant l'envoi des messages des journaux d'événements vers le serveur *rsyslog*. Nous avons choisi le programme libre *eventlog-to-syslog*[5].

Ici encore, vous trouverez la procédure de configuration complète dans le répertoire *ressources* de l'archive QoQ-CoT.

2.2.4 Configuration du serveur *rsyslog*

L'installation de *rsyslog* avec le support MySQL est dépendante des distributions. Vous pouvez choisir d'installer le serveur MySQL localement (a priori plus performant) ou d'utiliser un serveur distant. Dans tous les cas, vous devez vous assurer que la base par défaut est créée, ainsi que l'utilisateur MySQL qui pourra y écrire depuis le serveur *rsyslog*.

Sous Debian avec le serveur MySQL installé en local, la commande suivante suffit[6] pour installer tout ce qui est nécessaire (en choisissant les valeurs par défaut pour le nom de la base l'utilisateur MySQL) :

```
aptitude install rsyslog-mysql
```

5. *eventlog-to-syslog* : https://code.google.com/p/eventlog-to-syslog/

6. Bon, avouons-le : selon les cas, il se peut qu'un *dpkg-reconfigure rsyslog-mysql* puisse aider...

Cette installation effectuée, vous devez disposer dans MySQL :

- de la base *RSYSLOG_DB* qui contient les tables par défaut *SystemEvents* et *SystemEventsProperties* avec les champs standards définis lors de la création automatique de la base ;
- d'un utilisateur *RSYSLOG_DB_USER*, mot de passe *RSYSLOG_DB_USER_PASSWD*, qui dispose des droits d'écriture sur *RSYSLOG_DB* depuis *RSYSLOG.QoQ-CoT.ORG*

Il reste à configurer le fichier *rsyslog.conf* correctement. Vous en trouverez une version complète et fonctionnelle dans le répertoire *ressources* de l'archive QoQ-CoT. L'idée principale est d'utiliser le mécanisme de conditions pour filtrer finement les messages reçus et n'envoyer que le nécessaire dans la base de données.

On peut ensuite ajouter une tâche planifiée qui vide la table *SystemEvents* selon un rythme à définir. Le fichier de commandes MySQL à exécuter pourra s'inspirer de ceci (*MONTH* peut par exemple être remplacé par *YEAR*...) :

```
connect RSYSLOG_DB;
DELETE FROM `SystemEvents` WHERE ReceivedAt< DATE_SUB(NOW(), INTERVAL 1 MONTH);
quit
```

Il conviendra également de s'assurer que tous les clients puissent communiquer en UDP et TCP vers le serveur *rsyslog*, sur le port choisi (par défaut 514).

Enfin, puisqu'il s'agit de pouvoir répondre à des injonctions juridiques, il est primordial de s'assurer que le serveur *rsyslog* dispose d'estampilles horaires fiables : la synchronisation avec un serveur NTP s'impose. De la même façon, les clients devraient également être synchronisés NTP : en effet, si le serveur est synchronisé et que le client envoie ses informations en temps réel, on peut à la rigueur accepter l'approximation consistant à utiliser l'heure du serveur, considérant que le temps de transfert client/serveur est négligeable. Mais dès lors que, pour quelque raison que ce soit, le dialogue client/serveur est rompu pendant un certain temps, les messages sont mis en attente sur le client et ne seront envoyés sur le serveur que lorsque le lien sera rétabli : prendre l'heure du serveur dans ce cas conduit à une erreur inacceptable. Et utiliser l'heure du client est également incorrect si ce dernier n'est pas synchronisé NTP... La synchronisation NTP des clients permet donc au système d'être résistant aux ruptures de lien, une qualité dont il ne serait pas raisonnable, à notre sens, de faire l'économie, dans ce contexte potentiellement juridique.

2.3 Interroger la base de données : interface PHP

2.3.1 Création d'une table intermédiaire dans la base de données

Nous avons choisi, pour des raisons de performance, de générer de façon asynchrone une nouvelle table *Connexions* dans la base de données *RSYSLOG_DB* plutôt que d'utiliser directement la table standard *SystemEvents*. Les entrées de *SystemEvents* correspondent à un objet général de type « évènement » (en particulier, une ouverture et une fermeture de session, soit une connexion, constituent deux entrées différentes), alors que nous avons besoin d'un objet de type « connexion », dont les éléments (durée, heure de début et de fin, etc.) ne sont pas *directement* présents dans *SystemEvents*. L'objet de la table *Connexions* est de permettre un accès direct à tous ces éléments, calculés en amont à partir de *SystemEvents*, afin les requêtes d'interrogation de l'interface PHP soient beaucoup plus efficaces.

Plus précisément, les champs de la table *Connexions* sont : *Login, Date début initiale, Date début, Heure début, Date fin, Heure fin, Durée, Jour, Jour semaine, Machine, OS, Id Process.*

Le peuplement de la table se fait à la demande de l'utilisateur au travers d'un menu de l'interface web : tous les événements de la table *SystemEvents* ultérieurs à la dernière demande de peuplement sont alors traités et insérés dans la table *Connexions*.

On ne rentrera pas ici dans une description détaillée de chaque champ, mais le lecteur attentif s'interrogera peut-être sur la redondance de certains d'entre eux, comme *Heure début* et *Heure fin* par exemple, qui pourraient être déduits de *Date début* et *Date fin*. Nous avons fait le choix de cette redondance à des fins de gain de performance lors des requêtes, tous les nécessaires calculs de découpe de chaîne (*substr*) étant ainsi déjà effectués.

Néanmoins, le champ *Id Process* nécessite quelque précisions : c'est par ce numéro de *process* que sont appairés les évènements d'ouverture et de fermeture d'une même session. En fait lorsqu'on lance un peuplement, chaque nouvel enregistrement d'ouverture dans la table *SystemEvents* provoque un *INSERT* dans la table *connexions,* tandis que chaque événement de fermeture provoque un *UPDATE* de celui des enregistrements insérés auparavant qui a les mêmes champs *Login*, *Machine* et *Id Process*. C'est lors de cet *UPDATE* que les champs *Date Fin*, *Heure fin* et *Durée* sont renseignés.

2.3.2 L'interface PHP

Au vu de la simplicité de l'interface, une copie d'écran devrait suffire à en donner une idée assez précise.

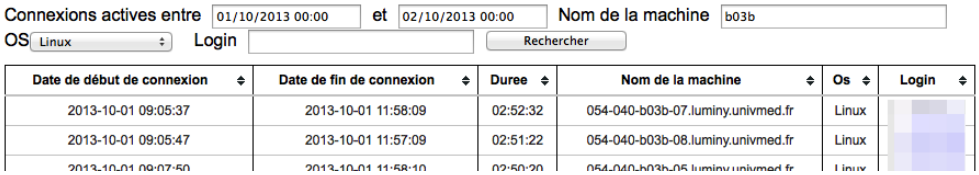

Figure 1 : Suivi des connexions

Dans l'exemple présenté ci-dessus, on cherche toutes les connexions actives entre le 01/10/2013 à 00h00 et le 02/10/2013 à 00h00 (même si la date de début ou la date de fin est en dehors de cet intervalle) sous Linux, sur les machines dont le nom contient *b03b* (ce qui, dans notre environnement, correspond à une salle). La colonne floutée indique, sous forme d'hyperliens vers la fiche annuaire correspondante, les noms d'utilisateurs (*logins*). Les colonnes peuvent être triées.

2.3.3 Filtrage de l'accès à l'application

L'interface donne accès à des données nominatives qui n'ont évidemment pas à être connues de tous. Néanmoins, il est intéressant de pouvoir donner un accès large à l'application, à des fins de pilotage et d'information, en particulier pour la partie *CoT* et ses graphes de taux d'utilisation.

C'est pourquoi une notion de rôle a été implémentée, liée à une authentification (actuellement seulement auprès d'un serveur LDAP). Seuls les usagers bénéficiant du rôle « admin » ont accès à l'intégralité des données. Pour les autres, tous les accès aux informations nominatives (suivi de connexions, options de certains graphes...) sont désactivés.

2.3.4 Conservation des données au-delà de la limite légale d'un an

Nous l'avons indiqué en introduction, les données de connexions doivent être supprimées ou rendues anonymes après une durée maximum d'un an. Afin d'avoir une vue statistique la pertinente possible sur l'évolution des taux d'utilisation, la meilleure option dans notre cas est de conserver les données indéfiniment, en les anonymisant au bout d'un an.

La table *SystemEvents* peut être simplement vidée régulièrement, puisque nous n'utilisons finalement que les données de la table intermédiaire *Connexions*. C'est dans cette dernière qu'il convient d'anonymiser les données, à l'aide d'une tâche planifiée chaque jour, exécutant par exemple la commande suivante qui remplace tous les noms d'utilisateurs par *old* :

```
UPDATE `Connexions` SET login='old' WHERE JOUR<DATE_SUB(CURDATE(),INTERVAL 1 YEAR)
```

3 Répondre à la question CoT : Combien de temps ? Taux d'utilisation des machines et informations connexes

3.1 Méthode

Pour pouvoir répondre à la question CoT et fournir des informations fiables et lisibles sur le taux d'utilisation des machines et des salles, nous avons développé une seconde interface PHP (un second onglet de l'application complète

QoQ-CoT). Elle utilise les données de la table *Connexions* pour tracer – à l'aide de la librairie libre (licence GPL pour les applications non commerciales) et très riche en fonctionnalités *pChart*[7] – divers graphes permettant de cerner finement le taux d'utilisation.

Le lecteur attentif remarquera qu'à ce point du récit, une information manque cruellement pour pouvoir produire ces graphes... En effet, la localisation (salle, immeuble, etc.), c'est-à-dire plus généralement l'appartenance à un groupe de machines, cruciale pour obtenir des taux d'utilisation cumulés, n'est pas présente dans la base, tout simplement car cette donnée n'existe pas dans les messages de connexion/déconnexion.

Pour ajouter cette connaissance à la base, l'interface dispose d'un module d'importation des groupes de machines, soit automatiquement pour les (heureux) utilisateurs du logiciel de déploiement JeDDLaJ[8], soit manuellement, à partir d'un fichier csv, pour les plus démunis... On peut ainsi définir tout groupe de machines (salle, bâtiment, groupe de salles, etc.) sur lequel on pourra ensuite observer le taux d'utilisation à l'aide des différents graphes.

Avant de nous lancer dans ce développement, nous avons évidemment cherché si un produit équivalent n'était pas disponible. Il semble ne pas en exister. On peut néanmoins citer le logiciel propriétaire et payant *Userlock*[9], puissant et très utilisé dans le monde entier mais... limité au monde Windows, ce qui le disqualifie dans notre contexte.

3.2 Les graphes disponibles

3.2.1 Utilisation détaillée par machine

L'idée générale est d'obtenir des informations détaillées de l'utilisation de chaque machine d'un groupe, avec la possibilité de choisir une fenêtre horaire d'observation quotidienne (ex. 12h-14h).

Cas 1 : si l'on ne renseigne que la date de début, on peut aussi définir une fréquence d'échantillonnage (ex. 15 min, 30 min, 1h, etc.). On obtient un graphe sur la journée avec, en abscisse, les machines du groupe et en ordonnée, les heures découpées en créneaux dont la taille est fonction de l'échantillonnage choisi. Si la machine était occupée pendant un créneau, il est « allumé » et réagit au passage de la souris en indiquant le nom d'utilisateur, le système d'exploitation et les heures de début et de fin de connexion.

Cas 2 : si la date de fin est également renseignée, on obtient alors, pour chaque machine, un histogramme indiquant la durée d'utilisation totale entre la date de début (à 00h00) et la date de fin (à 23h59), sur la fenêtre horaire spécifiée[10].

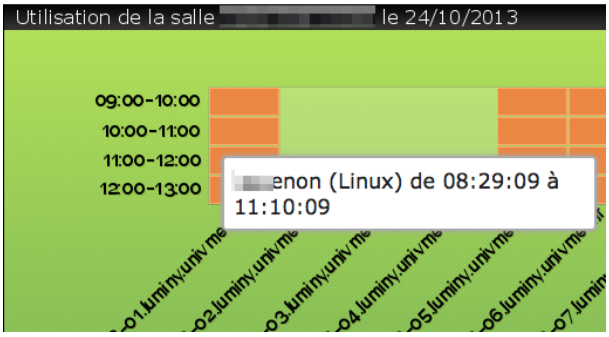

Figure 2 : Utilisation détaillée par machine, cas 1

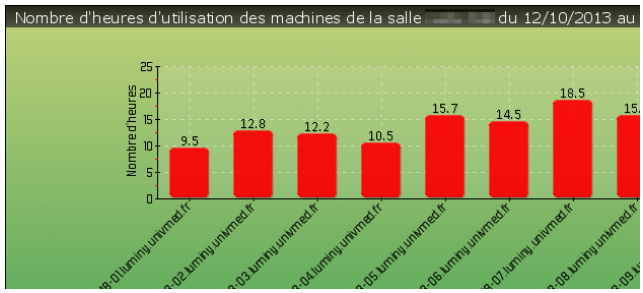

Figure 3 : Utilisation détaillée par machine, cas 2

3.2.2 Taux d'utilisation annuel

Ici, on veut observer le taux d'utilisation des salles sur une année glissante, dans une fenêtre horaire d'observation quotidienne (ex. 8h-18h) choisie[10]. À nouveau, plusieurs possibilités :

Cas 1 : on choisit une *salle*. On obtient alors un graphe avec les mois en abscisse et le taux d'utilisation en ordonnée, calculé par rapport à l'utilisation maximum possible U_{max}, obtenue comme suit : U_{max} = nb machines de la salle * nb heures de la fenêtre d'observation choisie en paramètre * nb jours ouvrés dans l'année

Une utilisation de U_{max} heures correspond à un taux de 100 %. La courbe trace le taux d'utilisation de la salle sur un an.

7. *Pchart 2.0 – A PHP charting library* : http://www.pchart.net/ ; Page wikipedia en français : http://fr.wikipedia.org/wiki/PChart

8. *JeDDLaJ pour **Je** **D**éploie **D**ans **La** **J**oie* : https://www.projet-plume.org/fiche/jeddlaj ; https://sourcesup.renater.fr/projects/jeddlaj/

9. *Userlock* : http://www.isdecisions.fr/logiciels/userlock/

10. Tous les taux et durée d'utilisation sont par défaut calculés en se restreignant aux jours ouvrés définis dans le fichier de configuration. Cependant, l'interface permet de modifier ponctuellement les jours à considérer avant de lancer la génération d'un graphe.

Cas 2 : on choisit un *groupe de salles*. Trois graphes sont alors générés :

1. le même que dans le cas 1, mais pour l'ensemble des machines du groupe
2. un graphe à histogrammes avec, en abscisse, les salles du groupe et, en ordonnée, le nombre d'heures d'utilisation cumulées des machines de la salle sur une année.
3. un second graphe à histogramme, avec les salles du groupe en abscisse et le taux d'utilisation en ordonnée, toujours calculé selon U_{max} comme vu *supra*. Pour chaque salle, l'histogramme bicolore représente à la fois le taux d'utilisation (partie inférieure) et son complémentaire le taux de vacance (partie supérieure).

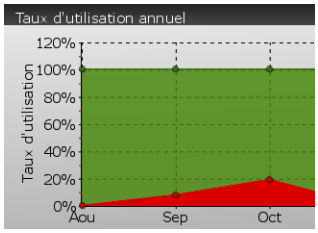

Figure 4 : Taux utilisation annuel, Cas 1

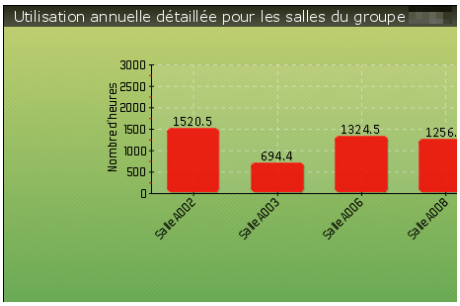

Figure 5 : Taux utilisation annuel, Cas 2, graphe 2

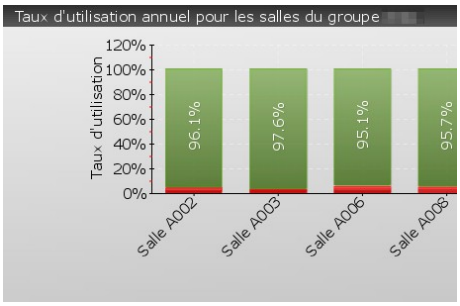

Figure 6 : Taux utilisation annuel, Cas 2, graphe 3

3.2.3 Nombre de machines utilisées et pic de connexions simultanées

Cette famille de graphes se veut illustrer la « charge humaine » des salles observées, en fonction du temps. On a, en paramètres, la fenêtre horaire d'observation quotidienne (ex. : 8h-18h), la date de début et celle de fin, la fréquence d'échantillonnage (ex. : 15 min, 30 min, 1h, etc.), la salle ou le groupe de salles. Encore une fois, il est possible de générer plusieurs types de graphes selon le choix des données d'entrée.

Cas 1 : on choisit *une date de début sans préciser de date de fin*. On obtient alors, sur la journée indiquée, l'évolution du *nombre de machines utilisées* dans la salle ou le groupe de salles : les créneaux horaires, dont la taille dépend de l'échantillonnage choisi, sont en abscisse, le nombre de machines est en ordonnée et une ligne fixe de couleur indique le nombre total de machines de la salle (ou du groupe de salles). Chaque point du graphe indique le nombre de machines *distinctes* qui ont été utilisées, *à un moment quelconque* du créneau horaire : notons qu'elles n'ont donc pas toutes été forcément utilisées *simultanément*.

Cas 2 : on choisit *une date de début et une date de fin*. On aura en abscisse les jours de l'intervalle choisi et le graphe indiquera cette fois, pour chaque jour, *le nombre maximum de connexions simultanées* survenu pendant la journée, même si ce pic n'a été atteint qu'un instant.

L'interface permet dans ce cas de choisir de faire figurer (*a*) une seule connexion par machine, ou (*b*) toutes les connexions. Avec le choix (*a*), on obtient le pic de *machines* utilisées *simultanément* : un graphe intéressant pour mettre en évidence le delta entre l'offre de machines et la demande maximale... L'option (*b*) fournira par contre le pic de *connexions simultanées,* qui pourra dépasser le nombre de machines, par le jeu des accès *ssh*, *su*, *console*, etc. C'est alors l' « intensité » de l'utilisation des machines qui est illustrée, plus que le nombre de postes occupés.

Figure 7: Nombre de machines utilisées (cas 1)

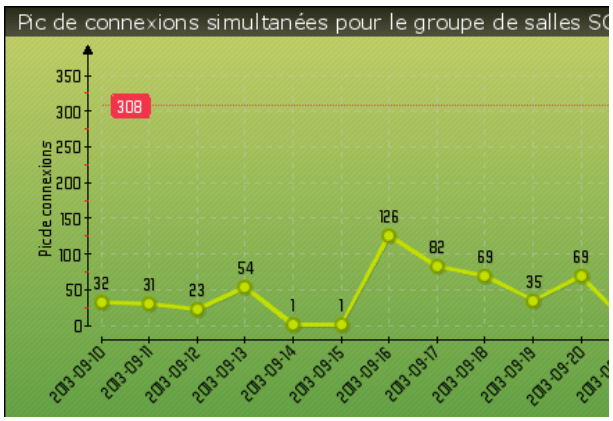

Figure 8 : Pic de connexions simultanées (cas 2)

3.2.4 Répartition des connexions par système d'exploitation

Ce dernier graphe simple, de type secteur, illustre la distribution des systèmes d'exploitation utilisés sur l'ensemble des connexions sur une salle (ou un groupe de salles), pendant la totalité de la période d'observation disponible en base.

4 Exemples d'utilisation : ce que QoQ-CoT est toute disposée à vous offrir...

C'est sûr, tracer de beaux graphes en couleur, c'est déjà en soi assez satisfaisant et ça peut même remplir honnêtement une journée de travail selon votre place dans l'organigramme... Néanmoins, QoQ-CoT peut vous apporter davantage :

- répondre facilement aux injonctions du RSSI (parfois tatillon), voire de la maréchaussée (parfois nerveuse) ;
- en utilisant de façon pertinente les différents graphes sur une période représentative, optimisation du nombre de machines d'une salle, du nombre de salles nécessaires ;
- en observant les taux d'utilisation sur des créneaux bien choisis, optimisation des horaires d'ouverture ;
- en observant la répartition des connexions par système d'exploitation, intérêt du maintien du mode « multi-boot » dans une salle ;
- en observant un groupe de salles, détection des possibilités de mutualisation inter-entités ;
- etc.

Tout ceci concerne l'aspect pilotage de QoQ-CoT, vue comme outil d'aide à la décision. Mais QoQ-CoT peut en faire encore plus pour vous. Car les graphes s'avèrent finalement être des outils donnant plus d'informations que ce pour quoi ils ont été initialement conçus. Ils peuvent vous permettre d'affiner la surveillance de votre parc, de façon *proactive*, en détectant des problèmes inaccessibles aux meilleurs logiciels de monitoring.

En effet, vous pourrez observer, par exemple, qu'une machine particulière dans une salle n'est jamais utilisée – alors même qu'elle apparaît opérationnelle selon les logiciels de monitoring. Et en vous déplaçant, vous allez découvrir à cette place un clavier maculé de mayonnaise, un courant d'air glacial, ou encore une fuite d'eau usée localisée...

5 Comment faire pour que QoQ-Cot vienne s'installer chez vous?

Il faut tout d'abord satisfaire quelques prérequis :

- être (si possible l'unique) administrateur de son parc ;
- disposer d'un outil de déploiement efficace car tous les clients vont devoir être adaptés. En effet, outre les réglages vus *supra* pour les envois de journaux sur le serveur *rsyslog*, il faudra, pour chaque poste :
  - ajouter et configurer une solution de fermeture automatique de session[11] au bout de X minutes d'inactivité. Ceci est fondamental pour ne pas fausser les statistiques avec des sessions ouvertes oubliées. $X = 60$ est un bon compromis entre le confort d'utilisation et la cohérence des statistiques.
  - s'assurer qu'il est synchronisé *NTP* ;
- disposer d'un serveur *rsyslog* synchronisé *NTP,* accessible depuis tous les clients par les ports 514/tcp et 514/udp, et configuré comme indiqué *supra* ;
- disposer d'un serveur MySQL accessible depuis le serveur *rsyslog* qui hébergera la base de données ;
- disposer d'un serveur web, par exemple Apache, disposant de PHP avec la librairie pChart version 2 installée et capable d'accéder au serveur MySQL ;
- disposer d'un serveur LDAP pour l'authentification des utilisateurs de l'application, afin de pouvoir différencier les administrateurs (accès données nominatives) des autres[12].

À ce moment-là, il ne vous reste plus qu'à télécharger l'archive QoQ-CoT, renseigner le fichier de configuration et lancer le *setup* comme indiqué dans la documentation (répertoire *ressources*) et ça y est : elle s'est installée chez vous...

11. Pour Windows, le gratuiciel *toff* est une bonne option : http://dennisbabkin.com/toff/. On trouvera dans le répertoire « ressources » de l'archive de QoQ-CoT la procédure pour le configurer. Pour Linux, un script maison est fourni dans ce même répertoire. Pour MacOS, l'option de déconnexion automatique est directement disponible : Préférences système → Sécurité et confidentialité → Avancé

12. L'utilisation sans serveur LDAP est possible mais un brin fastidieuse pour l'instant : il faut intervenir sur la fonction d'authentification dans le code pour instancier les rôles... Plus de précisions dans le répertoire « ressources » de l'archive de QoQ-CoT.

6 Vers une QoQ-Cot encore plus désirable : limitations et améliorations à envisager

6.1 Aspect sécurité

La solution de récolte des journaux décrite *infra* est fonctionnelle. Néanmoins, elle présente deux défauts principaux de sécurité. Le premier porte sur le transfert non sécurisé des journaux entre les clients et le serveurs. Il peut être réglé en utilisant le collecteur de log *nxlog*[13] qui permet, aussi bien pour Linux qu'OSX ou Windows, d'une part de chiffrer les journaux lors de leur transmission et d'autre part d'assurer un transfert fiable grâce à l'utilisation de TCP.

Mais le second problème reste malheureusement entier : nous ne rentrerons pas ici dans une explication en profondeur, mais il est possible pour un usager de faire en sorte que sa connexion n'apparaisse pas dans la base, en forgeant sur le client de faux messages à l'attention du système de journalisation local, qui les transmettra au *rsyslog* central[14]...

Ce risque doit toutefois être relativisé. L'usager susceptible de frauder répond à un profil relativement rare, car il doit, d'une part, disposer de bonnes connaissances techniques et d'autre part, mettre en œuvre les mécanismes de rétro-ingénierie nécessaires pour savoir quelle solution de journalisation est en production, et enfin la comprendre suffisamment pour déterminer comment la contourner...

Pour éliminer complètement ce risque, le meilleur moyen est de ne plus reposer sur les systèmes de journalisation locaux des clients, que l'utilisateur peut éventuellement abuser. Une solution consiste à faire en sorte que les clients dépendent pour l'authentification d'un unique serveur capable d'enregistrer les informations de connexion/déconnexion (par exemple en fédérant tous les clients dans un domaine de type Samba4 ou Active Directory[15]) puis de restreindre à ce seul serveur la possibilité de communiquer avec le serveur *rsyslog*. Les informations transmises par les clients au serveur d'authentification lors des ouvertures/fermetures de sessions reposent alors sur des secrets (par exemple mot de passe de la machine) rendant l'élaboration de faux messages quasi impossible.

Il reste encore une option pour échapper aux griffes de QoQ-CoT : commettre l'action répréhensible, puis faire en sorte que la déconnexion ne soit pas enregistrée, par exemple en arrachant la prise de courant. Il est alors impossible de prouver que le dernier usager connecté l'était toujours au moment des faits, puisqu'aucune heure de fin de connexion n'a été transmise au serveur. Le suspect peut donc nier, arguant un dysfonctionnement ponctuel du serveur qui aurait perdu le message de sa déconnexion, survenue avant les faits : audacieux et risqué certes, mais plaidable... avec un bon avocat.

Une méthode pour éliminer ce problème est de configurer les clients de façon à ce qu'ils consignent en permanence (par exemple chaque seconde), dans un fichier local, le (ou les) utilisateur(s) actuellement connecté(s). La consultation de ce fichier local en complément de QoQ-CoT, dans les cas de connexions non fermées, permet de dissiper tout doute.

D'autres moyens restent toutefois envisageables pour mettre en défaut l'identification des usagers connectés, comme par exemple la substitution d'une machine du parc par celle du fraudeur, mais l'élimination de ce type de risques dépasse le champ spécifique de ce papier.

6.2 Améliorations générales

Citons ici quelques pistes intéressantes, en dehors des aspects sécurité évoqués supra :

- la possibilité de choisir le mode d'authentification des utilisateurs de l'application, au lieu du seul LDAP actuel, serait bienvenue pour une adaptabilité à un plus grand nombre de contexte ;
- la prise en charge de Windows 8 devrait très vite s'avérer incontournable ;
- savoir prendre en compte les connexions de type *SSH* et *su* sur plateformes OSX constituerait une avancée primordiale pour les parcs à dominante *Apple*.

13. http://nxlog.org/

14. Pour éviter ce problème, il faut donc parvenir à interdire à l'utilisateur l'écriture dans les journaux locaux d'authentification. Sous Windows, pas de problème, le journal d'évènements est protégé en écriture. On peut parvenir à une situation similaire sous Linux en jouant sur les permissions de */dev/log*. Mais pour OSX, nous n'avons pas à ce jour de solution satisfaisante...

15. Notons qu'une authentification centralisée sur un serveur LDAP – la solution en production chez nous – ne répond pas au besoin : le serveur n'est pas sollicité lors des déconnexions et n'en a donc pas connaissance.

7 Conclusion

Au final, la solution gallinacée détaillée dans cet article dispose, malgré ses lacunes, de plusieurs caractéristiques qui peuvent, à notre sens, intéresser largement notre communauté :

- répond aux exigences légales en matière de journalisation des connexions ;
- fournit, au travers de graphes, de très nombreuses informations liées aux taux d'utilisation de tout groupe de machines identifié, par exemple une ou plusieurs salles de formation ;
- améliore la surveillance du parc en permettant, au travers de ces mêmes graphes, de mettre au jour des problèmes non détectables par les logiciels de supervision classiques ;
- est entièrement libre, et repose exclusivement sur des briques logicielles ouvertes : il vous sera ainsi très facile et peu coûteux d'ajouter une ou plusieurs familles de graphes adaptées à vos besoins.

Essayez-là donc et profitez de sa liberté pour l'accommoder finement à votre sauce : au pot, au vin, ou même tandoori, QoQ-CoT ne vous décevra pas...