h e g

Haute école de gestion
Genève

Détection et analyse d'un problème de congestion réseau

Travail de Bachelor réalisé en vue de l'obtention du Bachelor HES

par :

Lorenzo Cortes

Conseiller au travail de Bachelor :

Gérard Ineichen

Genève, le 26 novembre 2015

Haute École de Gestion de Genève (HEG-GE)

Filière informatique de gestion

Hes·so GENÈVE
Haute Ecole Spécialisée de Suisse occidentale

Déclaration

Ce travail de Bachelor est réalisé dans le cadre de l'examen final de la Haute école de gestion de Genève, en vue de l'obtention du titre de Bachelor en Informatique de Gestion.

L'étudiant a envoyé ce document par email à l'adresse remise par son conseiller au travail de Bachelor pour analyse par le logiciel de détection de plagiat URKUND, selon la procédure détaillée à l'URL suivante : http://www.urkund.fr/student_gorsahar.asp.

L'étudiant accepte, le cas échéant, la clause de confidentialité. L'utilisation des conclusions et recommandations formulées dans le travail de Bachelor, sans préjuger de leur valeur, n'engage ni la responsabilité de l'auteur, ni celle du conseiller au travail de Bachelor, du juré et de la HEG.

« J'atteste avoir réalisé seul le présent travail, sans avoir utilisé des sources autres que celles citées dans la bibliographie. »

Fait à Genève, le 26 novembre 2015

Lorenzo Cortes

Remerciements

Tout d'abord, je tiens à remercier Monsieur Gérard Ineichen pour m'avoir accordé la possibilité de réaliser mon travail de Bachelor dans le domaine du réseau. Mais également pour sa sympathie, son aide, sa constante disponibilité tout au long du projet ainsi que pour la relecture de ce dossier.

Je remercie également l'entreprise Kyos de m’avoir permis de faire une première incursion professionnelle dans le monde des réseau et plus particulièrement Monsieur Maxime Feroul pour la mise en place initiale du projet et Monsieur Vincent Vuillemin pour son suivi.

Je tenais finalement à remercier Monsieur André Seydoux et Claire Hagry pour leur aide et conseils qu’ils m’ont apporté tout au long du projet.

Résumé

Ce travail a pour but de mettre en place une solution qui permette de détecter et alerter des problèmes de congestion réseau.

Il est divisé en deux étapes bien distinctes :

La première étape est une démarche de recherche/analyse sur les méthodes de détection de la congestion réseau dans laquelle on trouve trois parties distinctes :

1. Introduction à la problématique de la congestion réseau
2. Détection de la congestion
3. Transmission / alerte de la congestion

La deuxième étape (pratique) consiste au montage de trois schémas prototypes type afin de générer de la congestion et d'y appliquer la démarche de la première étape. Le trafic sera créé avec le générateur de trafic Iperf. La congestion sera détectée par l'équipement, signalée avec le protocole SNMP et analysée avec NetFlow.

La mise en œuvre et les différents outils utilisés pour la réalisation du montage seront décrits en détail dans ce rapport.

Table des matières

Déclaration i
Remerciements ii
Résumé iii
Liste des tableaux vii
Liste des figures vii
1. Introduction 1
2. La problématique de la congestion réseau 2
2.1 Les conséquences de la congestion réseau 2
2.2 Où se produit la congestion réseau 4
2.2.1 La couche transport 4
2.2.2 La couche réseau 4
2.2.3 La couche liaison de données 5
2.3 Eviter la congestion réseau 5
2.3.1 Le buffer (mémoire tampon) 5
2.3.2 Augmenter le débit 6
2.3.3 Bande passante variable 6
2.3.4 Bande passante asymétrique 6
2.3.5 Priorisation du trafic 7
2.3.6 La qualité de service (Quality of Service, QoS) 7
2.3.7 L'approche Flow control 8
2.4 Quelques différentes causes de congestion réseau 9
2.4.1 Les boucles de routage 9
2.4.2 Bande passante insuffisante 9
2.4.3 Les collisions 10
2.4.4 Mauvaise gestion des MTU 10
3. La détection de la congestion réseau 12
3.1 Les générateurs de trafic 12
3.1.1 Iperf 12
3.1.2 IP SLA 14
3.2 Les différents outils qui permettent de détecter la congestion 15
3.2.1 Le protocole SNMP 15
3.2.1.1 Caractéristiques et fonctionnement de SNMP 15
3.2.1.2 Fonctionnement détaillé de SNMP 16
3.2.1.3 La trappe SNMP (SNMP trap) 16
3.2.1.3.1 Le message inform 17
3.2.1.4 L'interface web Cacti 18
3.2.2 NetFlow 20
3.2.2.1 NfSen et NfDump 20

3.2.3 Le protocole Syslog 23
3.2.3.1 Configuration de Syslog 24
3.2.4 La méthode Storm control 24
3.2.4.1 Fonctionnement détaillé de Storm control 25
3.3 Sélection des outils les plus pertinents 26
4. Transmission, alerte de la congestion 27
4.1 Alerter avec Nfsen 27
4.1.1.1 Anomalie détectée avec les alertes Nfsen 30
4.1.2 L'envoi d'email avec Postfix 30
4.2 Alerter avec Syslog 31
4.3 Alerter avec prudence 32
5. Partie pratique, les schémas prototypes 33
5.1 Matériel utilisé 33
5.1.1 Les routeurs 33
5.1.1.1 Compatibilité 34
5.1.2 Les commutateurs (switch) 34
5.1.2.1 Compatibilité 34
5.1.3 Les ordinateurs et les serveurs 35
5.1.4 Le câblage 35
5.2 Configuration 35
5.3 Schéma n°1, chaine de deux routeurs avec lien série 36
5.3.1 Caractéristiques du schéma 36
5.4 Schéma n°2, chaine de trois commutateurs 37
5.4.1 Caractéristiques du schéma 37
5.5 Schéma n°3, chaine de deux routeurs avec câble coaxial 38
5.5.1 Caractéristiques du schéma 38
6. Phase de test 39
6.1 Schémas n°1, routeurs avec lien série, phases de test 40
6.1.1 Test n°1, schéma n°1, simulation d'une longue congestion 40
6.1.1.1 Trafic avec une boucle de routage 40
6.1.1.2 Simulation de la congestion avec du trafic TCP sur Iperf3 40
6.1.1.3 Simulation de la congestion avec du trafic UDP sur Iperf2 41
6.1.1.4 Mise en place et déclenchement des alertes sur Nfsen 42
6.1.1.5 Test TCP et UDP, déterminer le coupable avec Cacti et Nfsen 43
6.1.1.6 Test TCP et UDP, observation du trafic supplémentaire généré 46
6.1.1.7 Consultation du serveur Syslog 46
6.1.1.8 Conclusion du test n°1 47
6.1.2 Test n°2, schéma n°1, simulation d'un pic de trafic passager UDP 47
6.1.2.1 Simulation de la congestion avec du trafic UDP sur Iperf3 48
6.1.2.2 Mise en place et déclenchement des alertes sur Nfsen 48
6.1.2.3 Déterminer le coupable avec Cacti et Nfsen 50
6.1.2.4 Consultation du serveur Syslog 51
6.1.2.5 Conclusion du test n°2 51

6.2 Schémas n°2, suite de switch, phases de test 51
6.2.1 Test n°3, schéma n°2, simulation d'une longue congestion sans Storm control 52
6.2.1.1 Simulation de la congestion avec du trafic TCP sur Iperf3 52
6.2.1.2 Trafic supplémentaire 52
6.2.1.3 Mise en place et déclenchement des alertes sur Nfsen 52
6.2.1.4 Déterminer le coupable avec Cacti et Nfsen 54
6.2.1.5 Consultation du serveur Syslog 55
6.2.1.6 Conclusion du test n°3 55
6.2.2 Test n°4, schéma n°2, simulation d'une longue congestion avec Storm control qui va bloquer le port 55
6.2.2.1 Simulation de la congestion avec du trafic UDP sur Iperf2 56
6.2.2.2 Trafic supplémentaire avec Iperf version 2 56
6.2.2.3 Consultation du serveur Syslog 58
6.2.2.4 Mise en place et déclenchement des alertes sur Nfsen 59
6.2.2.5 Déterminer le coupable avec Cacti et Nfsen 59
6.2.2.6 Conclusion du test n°4 61
6.2.3 Test n°5, schéma n°2, simulation d'une longue congestion avec Storm control qui va mettre en shutdown le port 62
6.2.3.1 Nouvelle configuration du switch 62
6.2.3.2 Simulation de la congestion avec du trafic TCP sur Iperf3 63
6.2.3.3 Trafic supplémentaire avec Iperf version 2 63
6.2.3.4 Consultation du serveur Syslog 65
6.2.3.5 Mise en place et déclenchement des alertes sur Nfsen 65
6.2.3.6 Déterminer le coupable avec Cacti et Nfsen 66
6.2.3.7 Conclusion du test n°5 68
6.3 Schémas n°3, routeurs avec câble coaxial, phases de test 68
6.3.1 Test n°6, schéma n°3, simulation d'une longue congestion.....68
6.3.1.1 Simulation de la congestion avec du trafic TCP sur Iperf3 69
6.3.1.2 Simulation de la congestion avec du trafic UDP sur Iperf2 70
6.3.1.3 Consultation du serveur Syslog 70
6.3.1.4 Mise en place et déclenchement des alertes sur Nfsen 71
6.3.1.5 Test TCP, déterminer le coupable avec Cacti et Nfsen 72
6.3.1.6 Test UDP, déterminer le coupable avec Cacti et Nfsen 73
6.3.1.7 Conclusion du test n°6 73
6.3.2 Test n°7, schéma n°3, simulation de deux longue congestion parallèle .73
6.3.2.1 Simulation de la congestion avec Iperf2 entre le client et le serveur 74
6.3.2.2 Simulation de la congestion avec Iperf3 entre les deux PCs cisco-td-xx.... 75
6.3.2.3 Consultation du serveur Syslog 75
6.3.2.4 Mise en place et déclenchement des alertes sur Nfsen 76
6.3.2.5 Test UDP, Déterminer le coupable avec Cacti et Nfsen 76
6.3.2.6 Conclusion du test n°7 76
6.4 Conclusion des phases de test 77
7. Conclusion 80
7.1 Conclusion personnelle 81
Bibliographie 82

Liste des tableaux

Tableau 1 Récapitulatif de la phase de test 79

Liste des figures

Figure 1 Le monde connecté à internet 1
Figure 2 Test de bande passante chez Swisscom 7
Figure 3 Iperf3, client qui se connecte au serveur 13
Figure 4 Exemple d'une statistique IP SLA sur routeur Cisco – opération HTTP 14
Figure 5 Trappe envoyée avec succès au serveur (superviseur) 17
Figure 6 Requête Inform envoyée avec succès au superviseur 18
Figure 7 Exemple d'un graphique Cacti 18
Figure 8 Nfsen, exemple d'un graphique journalier 21
Figure 9 Nfsen, exemple de données récolté sur des équipements 21
Figure 10 Nfsen, exemple de flux récoltés 22
Figure 11 Nfsen, exemple d'un IP Lookup 22
Figure 12 Exemple de logs sur un routeur Cisco 23
Figure 13 Exemple de logs sur un serveur Syslog 24
Figure 14 Storm control, exemple de seuils 25
Figure 15 Nfsen, exemple d'alertes 27
Figure 16 Nfsen, Exemple page de création d'une alerte 28
Figure 17 Nfsen, Informations d'une alerte 29
Figure 18 Schéma n°1, routeurs avec lien série 36
Figure 19 Schéma n°2, chaîne de switch avec débit différent 37
Figure 20 Schéma n°3, routeurs avec câble coaxial 38

1. Introduction

Le développement d'Internet s'est fait très rapidement ces dernières années. En Suisse, plus de 85 %[1] de la population avait accès à internet en 2012. Tout le monde veut y partager des informations, s'informer et télécharger, sans forcément se soucier de ce que cela implique au niveau de leur réseau. Souvent, les personnes dépassent leur capacité de bande passante[2] ce qui implique un ralentissement de leur ordinateur. La congestion réseau est donc un sujet sensible qui peut concerner aussi bien une simple personne qui utilise son PC à domicile pour naviguer sur le web qu'une grande entreprise internationale qui utilise internet comme outil de travail.

Figure 1 Le monde connecté à internet

(Google image 2015)

De nos jours, toute personne qui utilise un ordinateur a été confrontée au moins une fois au problème de la congestion réseau sans forcément savoir de quoi il s'agit. Voyons donc une définition simple et précise de la congestion : La congestion réseau est la condition dans laquelle une augmentation du trafic (flux) provoque un ralentissement global de celui-ci. (Source Wikipédia 2015, Congestion réseau)

Mon travail consistera donc à bien comprendre comment se forment ces congestions réseau. Il s'agira ensuite de les déclencher et d'alerter afin de prendre les mesures correctives nécessaires. Pour cela, je ferai différents tests pratique grâce aux schémas type.

Le domaine du réseau m'intéresse tout particulièrement tant par sa complexité que par sa grande étendue. J'ai choisi ce sujet, car il concerne pratiquement tout individu utilisant un ordinateur connecté et sa rédaction me sera certainement utile plus tard. A travers ces trois années à la HEG, j'ai acquis de nombreuses connaissances dans ce domaine. Ce qui me permettra non seulement de les mettre en œuvre, mais aussi d'en découvrir de nouvelles.

[1] Source https://fr.wikipedia.org/wiki/Internet_en_Suisse

[2] La bande passante représente la quantité d'informations (en bits/s) qui peut être transmise sur une voie de transmission
Source http://www.marche-public.fr/Terminologie/Entrees/bande-passante-bandwidth.htm

2. La problématique de la congestion réseau

Avant de me lancer dans le vif du sujet avec des termes techniques et compliqués, je vais tâcher d'expliquer de manière simple et imagée, comment se produisent les congestions réseau pour que toute personne étrangère au domaine informatique puisse comprendre le problème.

Imaginez que vous jouez au ballon: une première personne vous passe un ballon, vous vous tourner pour le passer à une deuxième personne, puis, vous vous retourner pour recevoir un autre ballon de la première personne, et ainsi de suite.

Il y a alors quatre aspects qui rentrent en compte :

1. Les ballons représentent les données qui transitent à travers le réseau
2. La fréquence à laquelle la première personne vous envoie ces ballons représente le débit d'entrée
3. Le temps que vous mettez à faire la rotation pour passer le ballon à la deuxième personne, puis récupérer un nouveau ballon de la première personne représente le temps de traitement
4. La vitesse à laquelle vous remettez vos ballons à la deuxième personne représente le débit de sortie

On comprend alors facilement que plus vite les ballons seront envoyés par la première personne (débit d'entrée), plus vous aurez du mal à les remettre (temps de traitement) à la deuxième personne. Les ballons arriveront tellement vite que vous ne pourrez pas tous les récupérer et certains seront perdus. C'est ce qu'on appelle une congestion réseau en entrée, le débit d'entrée est supérieur à votre capacité de traitement et certains ballons (données) sont perdus ce qui provoque un ralentissement du réseau. Plus le nombre de ballons perdus sera grand, plus le réseau se verra ralenti.

Il peut également se produire une congestion réseau en sortie, c'est-à-dire les ballons reçus de la première personne sont tellement vite transmis à la deuxième personne que cette dernière se retrouve débordée. Votre temps de traitement est bon, mais, étant donné que le débit de sortie ne suit pas le rythme, certains ballons seront perdus.

2.1 Les conséquences de la congestion réseau

On sait que la congestion ralentit le réseau. La vitesse à laquelle transitent les informations de notre réseau correspond au débit[3] et celui-ci se trouve donc directement impacté par ce problème. Voyons de plus près comment la congestion affecte le débit.

[3] Quantité d'informations transmise via un canal de communication selon un intervalle de temps donné. Source http://www.futura-sciences.com/magazines/high-tech/infos/dico/d/internet-debit-1849/

Tout d'abord, il faut savoir que la bande passante est la taille du tuyau dans lequel vont transiter les informations, plus la bande passante est grande, plus le débit qui y transite peut être élevé. On dit en général que le débit maximum est égal à la taille de la bande passante.

Une diminution du débit maximum est provoquée par la bande passante. En effet, on ne peut pas transmettre les données à une vitesse supérieure à celle de la bande passante. A tout cela s'ajoute la latence[4] (temps nécessaire pour véhiculer un paquet au travers d'un réseau). Si la bande passante de notre réseau est saturée, une congestion a de fortes chances de se produire et la latence va alors augmenter. Cependant, si la bande passante n'est pas submergée, la latence restera alors la même. Exemple avec un réseau non saturé : Nous avons un réseau avec une bande passante de 100 mégas, un paquet met 10 ms (latence) pour traverser le réseau. Si on augmente la bande passante à 1000 mégas, le paquet mettra toujours 10 ms pour traverser le réseau, la seule différence est qu'un plus grand nombre de paquets pourront traverser le réseau en même temps, mais, ils n'iront pas plus vite.

S'il y a un engorgement du trafic, une forte latence n'aidera pas puisque les paquets mettent plus de temps pour traverser le réseau. Les buffers (mémoire tampon) vont alors se remplir et des paquets seront abandonnés si la surcharge de trafic persiste. Il faudra alors les retransmettre via divers protocoles pour que le réseau continue à fonctionner correctement. En résumé, il se produit un effet boule de neige puisque ces protocoles vont retransmettre les paquets abandonnés alors que le réseau est déjà saturé.

Il faut aussi tenir compte de la gigue[5] (jitter en anglais) qui va nous indiquer la variation du délai de transmission des paquets et permettre de savoir si la latence est stable. Si la gigue est élevée, alors la latence varie beaucoup, ce qui signifie que les paquets n'arrivent pas de façon constante. A l'inverse, si la gigue est faible, la latence est stable et les paquets arriveront donc de manière régulière.

Si la congestion n'est pas colmatée à temps et persiste, elle peut mener à la chute totale du réseau, c'est pourquoi il faut l'identifier au plus vite afin de la résoudre. Un réseau idéal et sans congestion est de ce fait composé d'un haut débit, d'une latence faible et d'une gigue faible.

---

[4] Source http://blog.securactive.net/les-liens-entre-debit-latence-et-perte-de-paquets/?lang=fr

[5] La gigue est la différence de délai de transmission de bout en bout entre des paquets choisis dans un même flux de paquets. Source Wikipédia

2.2 Où se produit la congestion réseau

Il faut savoir qu'une congestion peut survenir sur différentes couches du modèle OSI puisque chacune transmet des données selon son PDU[6] et que la congestion est justement provoquée par l'envoi d'un trop grand nombre d'informations à la fois.

Le principe est toujours le même, lors de l'envoi des données d'un point à un autre, le le tunnel dans lequel vont transiter les données n'arrive pas à gérer toute l'information en même temps. Le tunnel peut être trop petit, le flux de données trop important, le temps de traitement trop lent, etc. Ici on ne va pas s'intéresser à ce qui provoque la congestion mais dans quelles couches elle est susceptible de se produire.

2.2.1 La couche transport

La couche transport qui est la quatrième couche du modèle OSI peut subir de la congestion lors de l'envoi de segments entre deux hôtes. Par exemple un serveur très performant qui génère beaucoup plus de trafic que ne peut en supporter le réseau. La congestion peut survenir aussi bien avec du trafic TCP (Transmission Control Protocol, protocole de transport fiable) qu'avec UDP[7]. Il faudra alors garder un œil attentif sur la retransmission de paquets pour TCP qui veille à ce que l'intégrité des données arrivent à destination contrairement à UDP qui peut les abandonner (packet loss). La perte et retransmission de paquets peuvent être des indicateurs de congestion.

A noter que le protocole TCP de la couche transport dispose de plusieurs mécanismes pour contrôler ses flux, notamment plusieurs algorithmes comme Nagle ou Clark qui ne seront pas décrit dans ce travail.

2.2.2 La couche réseau

La couche réseau qui est la troisième couche du modèle OSI peut subir de la congestion lors de l'envoi de paquets entre deux équipements. Par exemple un routeur qui émet à 110 Mbits/s alors que la bande passante est fixée à 100 Mbits/s va générer plus de trafic que ne peut en supporter le réseau et va créer une congestion. Pour diminuer les symptômes des congestions au niveau de cette couche il est possible de mettre en place une Quality of Service (expliqué plus en détail au point 2.3.6) afin que les données circulent dans les meilleures conditions possibles.

---

[6] Le Protocol Data Unit est l'unité de mesure des informations échangées dans un réseau informatique. Source Wikipédia

[7] User Datagram Protocol est un protocole non orienté connexion de la couche transport du modèle TCP/IP et est très simple étant donné qu'il fonctionne sans négociation et ne fournit pas de contrôle d'erreurs. Source Wikipédia + commentçamarche.net

2.2.3 La couche liaison de données

La couche liaison de données qui est la troisième couche du modèle OSI peut subir de la congestion lors de l'envoi de trames entre deux équipements. Cependant, certaines techniques existent comme pour frame relay (protocole à commutation de paquets situé au niveau de la couche 2) où chaque trame dispose des deux bits BECN[8] (permet de faire remonter la connaissance de l'état de congestion d'un nœud à l'émetteur) et FECN[9](permet à un nœud congestionné de faire connaître son état au récepteur) qui permettent d'alerter que l'envoi des données passe par un nœud[10] congestionné (en mettant leur bit à 1). Le bit DE (Discard Eligibility) va alors indiquer un problème de congestion en se mettant à 1 également. Le problème sera ensuite traité dans les couches supérieures. (Source eBook Préparation à la certification CCNA CISCO Installation, configuration et maintenance de réseaux. Pages 381-383)

2.3 Eviter la congestion réseau

Les problèmes de congestion sont fréquents à différentes échelles, il n'est pas rare qu'un réseau soit ralenti quelques secondes, voir quelques minutes suite à un gros pic dans le trafic de données. Heureusement, il existe plusieurs solutions pour prévenir la congestion et la mettre hors d'état de nuire. Voici quelques pistes de méthodes[11] qui permettent de gérer une congestion réseau.

2.3.1 Le buffer (mémoire tampon)

Le buffer est la mémoire utilisée pour le stockage temporaire de données lors du transfert d'informations afin de compenser la différence de débit ou de vitesse de traitement entre les divers éléments de notre réseau. (Source journaldunet.com http://www.journaldunet.com/encyclopedie/definition/630/53/22/buffer.shtml)

Cette méthode est la plus utilisée puisque chaque équipement de transmission (PC, routeur, etc.) possède son propre buffer. Ce n'est pas vraiment une méthode en soit puisque ce mécanisme se déclenche automatiquement. L'équipement va accumuler la surcharge de trafic dans son buffer et « libérer » le surplus de données lorsque la surcharge va diminuer. Les buffers sont idéals pour absorber un pic de trafic passager, mais si la surcharge dure trop longtemps, le buffer sera à son tour surchargé et les

[8] Source http://www.createurdeconvergence.com/glossaire-telecom-reseaux/frame-relay/
[9] Source http://www.createurdeconvergence.com/glossaire-telecom-reseaux/frame-relay/
[10] Un nœud est l'extrémité d'une connexion, qui peut être une intersection de plusieurs connexions ou équipements (PC, routeur, switch, ...). Source Wikipédia
[11] Sources des points 2.3.1 à 2.3.5 : Le monde des réseaux, combattre la congestion + smartreport.fr saturation réseau – Détecter les congestions réseaux

données seront alors abandonnées. Mais comment le buffer supprime les données en trop ? Il existe plusieurs méthodes pour un buffer d'écraser un surplus de données. La plus connue est le buffer circulaire qui fonctionne comme la méthode FIFO (First In First Out). Lorsque le buffer est plein et que de nouveaux éléments arrivent, les données les plus anciennes sont supprimées au détriment des nouvelles.

2.3.2 Augmenter le débit

La solution idéale pour ne pas avoir de congestion réseau est d'augmenter le débit pour que le trafic ne soit jamais surchargé, on se retrouve alors avec un réseau surdimensionné. L'inconvénient de cette méthode est le coût financier du réseau, on payera cher pour un débit qui ne sera pas exploité dans son intégralité.

2.3.3 Bande passante variable

Diminuer le débit lorsque le trafic de notre réseau est calme, l'augmenter lorsqu'une pointe de trafic survient ? C'est l'idée de la bande passante variable. La mise en place d'une bande passante variable fonctionne selon les trois critères suivants :

- Le débit minimum qu'on est certain d'avoir en permanence
- Le débit maximum
- Le burst qui est la capacité à maintenir le débit maximum pour une certaine durée

Lorsqu'un pic de trafic survient, on va passer de l'utilisation du débit minimum au débit maximum afin de garantir le bon transfert des données. Néanmoins, la capacité à maintenir le débit maximum n'est pas illimité et des congestions peuvent donc toujours survenir.

Cette méthode est intéressante dans la mesure où elle va couter moins cher qu'une solution avec une bande passante garantie, cependant, la qualité de service sera moins intéressante puisqu'on ne pourra pas exploiter notre débit maximum (burst) tout le temps.

2.3.4 Bande passante asymétrique

C'est le fait d'avoir une bande passante de taille différente selon le sens dans lequel transitent les données, on va alors augmenter la bande passante là où la congestion est le plus susceptible de se produire. Cette technique est très utilisée par les fournisseurs d'accès à internet de nos jours. On sait que les requêtes des utilisateurs du réseau sont moins importantes que les réponses côté serveur, on va donc mettre une bande passante plus large côté serveur-client. Voici un exemple de la bande

passante montante et descendante que je dispose de la part de Swisscom sur mon ordinateur personnel.

Figure 2 Test de bande passante chez Swisscom

(speedtest.net, depuis mon PC)

On voit bien la grande différence de bande passante appliquée par Swisscom selon le débit descendant (serveur-client) ou montant (client-serveur). Avec cette méthode une congestion a moins de chance de se produire puisqu'on adapte la bande passante en conséquence.

2.3.5 Priorisation du trafic

Cette méthode ne permet pas vraiment de supprimer la congestion, mais va plutôt la rendre moins nuisible. En effet, lorsqu'une congestion va se produire on va simplement prioriser les flux de données à envoyer en premier et ceux à retarder. Il faut alors configurer l'équipement de transmission afin qu'il soit en mesure d'analyser les flux de données et y définir un ou plusieurs niveaux de priorités.

Cette méthode reste néanmoins difficile à réaliser de par sa complexité. Si les données prioritaires qui transitent consomment toute la bande passante, que faire avec les données non prioritaires, on les abandonne ? On peut alors partager la bande passante selon la priorité du trafic. Par exemple, on attribue 70 % de bande passante au trafic important et 30 % au trafic moins important. Que se passera-t-il si soudainement on ne reçoit que des données non prioritaires ? On aura 70 % de la bande passante sous-utilisée. Bref, c'est une solution qui existe, mais qui possède des algorithmes assez complexes et qui est difficile à mettre en œuvre.

Certains protocoles de routage (RIP, OSPF, EIGRP, etc) permettent de mettre en place un mécanisme un peu similaire en tenant compte de la charge. Lorsqu'il y a plusieurs chemins vers une même destination, on peut indiquer quel chemin sera emprunté le plus souvent

2.3.6 La qualité de service (Quality of Service, QoS)

La QoS représente la capacité à gérer le bon fonctionnement du réseau selon des critères bien définis. On veut optimiser le réseau et on va donc fixer des seuils pour être sûr que tout se passera bien. Par exemple lorsqu'on signe un contrat internet avec un FAI (fournisseur d'accès à internet), il nous garantit une certaine bande passante car ils disposent d'une qualité de service en conséquence.

La qualité de service est basée sur plusieurs critères qui vont permettre de bien la définir. Les critères principaux sont le débit, la latence, la gigue et la perte de paquets.

Il y a plusieurs niveaux de service dans la Qos[12] ;

1. Le best-effort (meilleur effort) qui assure de transmettre au mieux vos données, mais, sans aucune garantie de service.
2. Le soft QoS (service différencié) qui permet aux composants réseau de prioriser leur trafic sans garantie de performance.
3. Le hard Qos (service garanti) qui permet de réserver un peu de bande passante pour l'allouer à différents types de flux. Le bénéfice de la garantir est que si une congestion survient dans notre réseau, le trafic du service ne sera pas impacté.

Bien entendu, la QoS n'est pas une méthode de résolution de la congestion, c'est simplement un moyen de s'assurer qu'on gère au mieux son réseau, par conséquent, cela permettra de minimiser la formation des congestions.

2.3.7 L'approche Flow control

Le contrôle de flux en français, est un procédé qui va gérer les transmissions entre deux nœuds et empêcher une source d'envoyer plus de données que ne peut en recevoir le récepteur. Puisqu'il a un débit supérieur, l'émetteur va alors le diminuer pour ne pas submerger le récepteur qui lui, à une capacité de traitement inférieur. Cela va notamment permettre de diminuer le nombre de paquets perdus par le récepteur qui n'arrive pas à traiter un débit supérieur à sa capacité de traitement. Une congestion réseau aura alors moins de chance de se produire.

Pour ne citer que celle-ci, la méthode stop-and-wait est la façon la plus basique de faire du flow control, voyons en quoi elle consiste. L'émetteur va segmenter les données à envoyer au récepteur. Le fonctionnement de cet échange est simple : Selon la window size, l'émetteur va envoyer un ou plusieurs segments à la fois puis, attendre la réception de l'acquittement de ceux-ci pour pouvoir renvoyer de nouveaux segments. Exemple : Si on a une window size de 3, on pourra envoyer au maximum trois segments à la fois sans se soucier d'un quelconque acquittement. Pour envoyer un quatrième segment, on est obligé d'attendre l'acquittement au minimum du premier segment et ainsi de suite. L'inconvénient de cette méthode est que le nombre de segments qu'on peut envoyer est limité.

A noter que Flow control s'applique sur un schéma avant qu'une congestion soit présente dans le réseau. Ce n'est donc pas un moyen de « colmater » une congestion,

[12] Source commentcamarche.net QoS, Qualité de service

mais, qui va plutôt servir à l'empêcher de se produire. (Source Wikipédia, https://fr.wikipedia.org/wiki/Contr%C3%B4le_de_flux)

2.4 Quelques différentes causes de congestion réseau

Les congestions peuvent survenir sur un réseau de plusieurs façons. Elles peuvent aussi bien survenir à la suite d'une surcharge de trafic qu'à cause d'un équipement mal configuré ou encore un câble qui cesse de fonctionner. C'est pourquoi, on peut les regrouper en plusieurs catégories.

2.4.1 Les boucles de routage

Une boucle de routage se produit lorsqu'un paquet n'arrive plus à trouver sa destination. Elle va transiter à travers le réseau jusqu'à l'expiration de son TTL[13] qui peut aller de 1 à 255. A un certain moment, il y aura beaucoup trop de paquets ne trouvant pas de route de destination et une congestion peut survenir. Une boucle de routage peut se produire lorsque le réseau n'est pas convergé, lorsqu'un équipement tombe en panne, lorsqu'un câble est mal branché, etc. Il existe plusieurs techniques pour pallier aux boucles de routage comme par exemple la route poisoning qui va, lorsqu'une route tombe, avertir le réseau d'une métrique de distance infinie. Ainsi, les paquets seront directement abandonnés.

Une boucle de routage peut également se produire sur la couche 2 lorsqu'il y a plusieurs chemins possibles entre deux équipements du réseau. La résolution de ces boucles se fait par l'implémentation du Spanning Tree Protocol[14] sur les commutateurs concernés.

Les boucles de routage peuvent survenir suite à une mauvaise configuration de notre schéma réseau et seront donc détectables assez rapidement. Celles-ci peuvent aussi se produire lorsqu'un équipement tombe en panne (on débranche un câble accidentellement par exemple), les données ne sauront plus où transiter puisqu'elles ne trouvent plus leur chemin de destination.

2.4.2 Bande passante insuffisante

La bande passante est insuffisante, les données n'arrivent pas à joindre leur destination à temps, le réseau est saturé. Dans ce cas, c'est la façon la plus courante de déclencher une congestion. On fait appel à plus de ressources que notre réseau peut en fournir, ce qui provoque le ralentissement de celui-ci. Il n'y a pas de solution

[13] Time To Live est une donnée placée dans l'en-tête d'un paquet IP et indique le temps pendant lequel une information doit être conservée. Source Wikipédia

[14] Protocole réseau de niveau 2 permettant de déterminer une topologie réseau sans boucle (Source, Wikipédia https://fr.wikipedia.org/wiki/Spanning_Tree_Protocol)

miracle à ce problème, on peut augmenter la bande passante, mais, cela coûte de l'argent. L'autre solution est simplement de réduire sa consommation de trafic en diminuant sa vitesse de téléchargement par exemple. S'il y a plusieurs utilisateurs sur le même réseau local, il faudrait gérer la bande passante de la meilleure façon possible pour que chaque personne puisse en bénéficier au mieux.

2.4.3 Les collisions

On appelle généralement un domaine de collision une partie du réseau ou les données transmises peuvent entrer en collision. Les informations qui transitent proviennent d'hôtes différents, mais, font partie du même réseau. Si les données entrent en collision elles deviennent inutilisables, il faut alors les retransmettre. Pour pallier à ce problème, on va segmenter au maximum les domaines de collision de notre réseau grâce aux routeurs, commutateurs ou bridge. Ceci permettra d'une part l'isolement du trafic entre les différents domaines de collisions, d'autre part d'offrir une bande passante réservée à chaque segment du réseau. On peut également, utiliser le mode full-duplex (permet de transporter les paquets simultanément dans les deux sens) qui empêche les collisions de se produire s'il n'y a que deux équipements dans le domaine de collisions.

L'exemple typique pour illustrer ce problème est l'utilisation du talkie-walkie où deux personnes ne peuvent pas parler en même temps car le signal se brouille (il y a des collisions, on est en bidirectionnel alterné). A l'inverse, avec un téléphone, deux personnes peuvent parler simultanément (il n'y a pas de collisions, on est en bidirectionnel simultané).

2.4.4 Mauvaise gestion des MTU

Maximum transmission unit (MTU) est la taille maximale d'un paquet pouvant être transmis en une seule fois (sans fragmentation) sur une interface.
(Source Wikipédia https://fr.wikipedia.org/wiki/Maximum_transmission_unit)

Lorsque deux entités vont communiquer, par exemple un client et un serveur, on peut utiliser ce qu'on appelle le Path MTU discovery qui est une technique permettant de déterminer le bon MTU entre deux hôtes IP. Cette technique fonctionne en définissant un bit DF (Dont't Fragment) dans les paquets sortants. Chaque équipement ayant un MTU plus petit qu'un paquet reçu va alors l'abandonner et envoyer un message d'erreur de type ICMP (Internet Control Message Protocol) spécifiant que le paquet est trop grand. L'émetteur va alors retransmettre avec un MTU plus petit jusqu'à qu'il soit capable de traverser l'hôte en question sans avoir à fragmenter le paquet. Si on ne définit pas de bit DF, les paquets trop gros pourront alors être fragmentés avant d'être

envoyés. Cette technique n'est pas optimale car les paquets fragmentés seront alors plus nombreux, ce que va augmenter le nombre de bytes transmis. De plus, les paquets fragmentés peuvent potentiellement arriver à destination dans le désordre.

Le problème qui survient est que de nombreux équipements bloquent par défaut (entièrement ou partiellement) le trafic ICMP ce qui peut empêcher les redimensionnements du MTU de se produire. Si les messages ICMP sont bloqués, l'émetteur va continuellement envoyer des paquets trop gros et ils seront jetés. L'émetteur ne sera pas notifié des messages d'erreurs suit au blocage d'ICMP. Les décisions relatives à ICMP peuvent être également prises par des administrateurs, il faut donc être conscient du problème des MTU afin d'éviter une congestion réseau de se produire.

(Source Wikipédia https://fr.wikipedia.org/wiki/Path_MTU_discovery)

3. La détection de la congestion réseau

Dans cette partie, nous allons tenter de comprendre comment détecter une congestion réseau. Bien entendu, dans le cadre de ce travail, la congestion sera détectée en sachant à l'avance qu'elle va se produire puisque je vais devoir appliquer un générateur de trafic à mes schémas pour la simuler. Ces schémas sont décrits au point 5 de ce travail, on ne va pas s'y intéresser tout de suite, mais, il faut simplement garder en tête que tous les points abordés dans cette rubrique seront susceptibles d'être appliqués sur ces schémas lors des phases de test afin d'y détecter de la congestion.

Je vais m'intéresser à plusieurs points essentiels ;

- Un générateur de trafic qui permettra de générer de la congestion
- Identifier les différents outils ou techniques qui permettront de détecter la congestion (des protocoles, des applications, etc)
- Sélectionner les outils les plus pertinents d'une part pour permettre de récolter les données en vue de la prochaine étape qui sera d'alerter du problème de congestion, d'autre part pour mettre en œuvre ces outils dans la partie pratique

3.1 Les générateurs de trafic

Il existe beaucoup de générateurs de trafic différents. Dans le cadre de mon travail, il serait bien d'avoir un générateur qui peut générer du trafic TCP et UDP pour pouvoir créer de la congestion de plusieurs façons. J'ai donc sélectionné deux générateurs de trafic, le premier est Iperf, ce générateur m'a été conseillé par mon entreprise mandataire Kyos. Pour le deuxième, j'ai opté pour un générateur qui s'implémente directement sur l'équipement Cisco, c'est IP SLA (Service Level Agreement). Voyons leur fonctionnement afin de choisir le plus adapté.

3.1.1 Iperf

Iperf est une application qui fonctionne avec un client et un serveur positionné aux deux extrémités du réseau. Iperf s'utilise en ligne de commande et sert à analyser les performances du réseau en y générant du trafic TCP ou UDP. Pour effectuer mes tests, j'ai installé les deux dernières versions pour Linux qui sont Iperf version 2 et 3.

Pour démarrer un test Iperf, il faut tout d'abord l'installer sur les deux machines (client et serveur). Une fois installé, la première étape est de se connecter au serveur depuis cette même machine avec la commande suivante :

```
iperf3 –s
```

« -s » pour spécifier que ce PC est le serveur.

Ensuite, il faut se connecter au serveur depuis la machine client :

```
Iperf3 –c <ip_de_mon_serveur>
```

« -c » pour spécifier que ce PC est le client.

Ces deux lignes sont le moyen le plus basique d'établir une connexion entre les deux PCs avec Iperf et va, par défaut, afficher du trafic TCP uniquement. En guise d'exemple, voici le résultat généré sur mon PC du côté client.

Figure 3 Iperf3, client qui se connecte au serveur

```
cisco@ubuntu:~/lorenzo$ iperf3 -c 192.168.1.48
Connecting to host 192.168.1.48, port 5201
[  4] local 192.168.1.15 port 42285 connected to 192.168.1.48 port 5201
[ ID] Interval           Transfer     Bandwidth       Retr  Cwnd
[  4]   0.00-1.00   sec  11.7 MBytes  98.3 Mbits/sec    0    204 KBytes
[  4]   1.00-2.00   sec  11.2 MBytes  93.9 Mbits/sec    0    208 KBytes
[  4]   2.00-3.00   sec  11.2 MBytes  94.0 Mbits/sec    0    211 KBytes
[  4]   3.00-4.00   sec  11.2 MBytes  93.9 Mbits/sec    0    211 KBytes
[  4]   4.00-5.00   sec  11.4 MBytes  95.9 Mbits/sec    0    211 KBytes
[  4]   5.00-6.00   sec  11.2 MBytes  94.1 Mbits/sec    0    211 KBytes
[  4]   6.00-7.00   sec  11.2 MBytes  94.0 Mbits/sec    0    212 KBytes
[  4]   7.00-8.00   sec  11.2 MBytes  93.7 Mbits/sec    0    215 KBytes
[  4]   8.00-9.00   sec  11.1 MBytes  93.4 Mbits/sec    0    215 KBytes
[  4]   9.00-10.00  sec  11.4 MBytes  95.6 Mbits/sec    0    215 KBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Retr
[  4]   0.00-10.00  sec   113 MBytes  94.7 Mbits/sec    0             sender
[  4]   0.00-10.00  sec   112 MBytes  94.3 Mbits/sec                  receiver

iperf Done.
```

Cette figure représente l'affichage par défaut d'Iperf3. On y voit d'abord les informations sur la connexion établie (IP des PCs client et serveur, ports utilisés). Ensuite, les dix premières lignes représentent l'intervalle temps d'une seconde chacune avec les données qui transitent entre les deux entités. Finalement, un résumé qui nous montre le transfert et la bande passante utilisée pour le transfert des données dans un intervalle de dix secondes.

Il y a plusieurs réglages qui peuvent être effectués pour afficher l'information de manière différente. On peut par exemple générer du trafic UDP (-u) plutôt que TCP, ce qui nous permettra de récupérer des informations supplémentaires comme la gigue et le taux de perte de paquets qui seront très utiles pour détecter une congestion. Je ne vais pas entrer plus dans les détails pour le moment. Les commandes utilisées pour générer du trafic dans les différents schémas lors de la partie pratique seront détaillées en temps voulu. Pour plus d'information, je vous invite à consulter le manuel d'Iperf3 sur leur site : https://code.google.com/p/iperf/wiki/ManPage.

Après avoir testé Iperf3 sur Windows et Linux, il apparait que sur Windows, pour des tests TCP, n'affiche pas le nombre de retransmission contrairement à Linux (voir figure ci-dessus). Vous trouverez en annexe mon guide d'installation d'Iperf2 et Iperf3 pour Linux et Windows (**Annexe 1**). En effet, les versions 2 et 3 d'Iperf sont un peu différentes, c'est pourquoi j'ai jugé utile d'installer les deux selon les besoins.

3.1.2 IP SLA

IP SLA (Service Level Agreement) est une technologie inventée par Cisco qui permet de mesurer le trafic et les performances d'un réseau permettant ainsi d'assurer une bonne qualité de service. Son but premier est donc de vérifier la qualité de Service du réseau en créant du trafic et en analysant les éventuels incidents sur le réseau résultant du trafic généré. IP SLA permet de générer du trafic dans un réseau grâce à son système de « master » et « responder ». Il faut définir un seul équipement (de préférence un routeur) « master » et un ou plusieurs équipements « responder ». Le master va alors générer du trafic entre lui et ses différents responder. Le type de trafic peut être varié selon les tests à effectuer dans le réseau (HTTP, HTTPS, DNS, ICMP, Telnet, SSH). IP SLA va nous permettre d'analyser plusieurs indicateurs tel que le temps de latence, la gigue, le packet loss et le temps de réponse d'un serveur. (Source http://www-igm.univ-mlv.fr/~dr/XPOSE2010/IPSLA/presentation.html)

Un des avantages d'IP SLA et qu'il est supporté par Netflow et le protocole SNMP qui sont deux outils que je vais devoir utiliser. L'autre avantage est qu'il n'est pas nécessaire d'installer du matériel supplémentaire, il suffit simplement de le configurer sur un équipement Cisco.

Voici un exemple d'une statistique IP SLA récupérée sur un routeur Cisco après avoir configuré une opération HTTP qui va chercher un URL défini toutes les 60 secondes.

Figure 4 Exemple d'une statistique IP SLA sur routeur Cisco – opération HTTP

```
Rt-PAT#sh ip sla monitor statistics
Round trip time (RTT)   Index 1
        Latest RTT: 10 ms
Latest operation start time: .10:46:13.002 eet Fri Oct 23 2015
Latest operation return code: OK
Latest DNS RTT: 0 ms
Latest TCP Connection RTT: 5 ms
Latest HTTP Transaction RTT: 5 ms
Number of successes: 9
Number of failures: 0
Operation time to live: Forever
```

L'opération va se déclencher toutes les 60 seconde et va incrémenter le compteur « number of successes » ou « number of failures ». Ici le routeur va effectuer cette

opération à vie puisque le « time to live » est configuré avec la valeur « forever ». On génère donc dans cet exemple, une opération de type « HTTP GET » toutes les minutes. En configurant plusieurs types de trafics différents, on pourra simuler du trafic assez facilement.

3.2 Les différents outils qui permettent de détecter la congestion

Il existe plusieurs outils pour détecter la congestion réseau. Je vais en analyser plusieurs afin de voir lesquels seront les plus pertinents à mettre en œuvre en vue de la partie pratique.

3.2.1 Le protocole SNMP

Le protocole Simple Network Management Protocol (SNMP) est un protocole de communication qui permet aux administrateurs réseau de gérer les équipements du réseau, de superviser et de diagnostiquer des problèmes réseaux et matériels à distance. (Source Wikipédia 2015, Simple Network Management Protocol). C'est donc un protocole idéal pour surveiller un réseau et potentiellement y détecter une congestion réseau. SNMP en est aujourd'hui à sa troisième version, les versions une et deux étant assez peu sécurisées (mot de passe en clair), la dernière version dispose des options d'encryption et d'authentification qui n'existait pas dans les versions antérieures. Voyons de plus près grâce à quels éléments SNMP fonctionne.

3.2.1.1 Caractéristiques et fonctionnement de SNMP

Pour interroger le réseau, le protocole SNMP dispose de trois éléments fondamentaux ;

1. **Un superviseur** (ou **serveur**) qui représente un poste de travail dans lequel on va pouvoir y effectuer des commandes de gestion afin de superviser le réseau. Une simple console suffit pour réaliser ces différentes commandes.
2. **Des agents** qui représentent n'importe quel équipement (routeur, commutateur, PC, serveur, etc.) interconnecté au réseau qui sera susceptible d'être supervisé.
3. **La MIB** (Management Information Base) est un ensemble d'informations propres à un équipement. Structurée en arbre, chaque objet de la MIB est reconnu grâce à son OID (object identifier). Le protocole SNMP permet de récupérer les informations contenues dans la MIB. Les MIB peuvent être standards ou propriétaires (permet d'y ajouter des variables pour y modifier son fonctionnement avec les agents SNMP).

A cela, on peut également ajouter si nécessaire, d'autres outils qui représenteront une aide parallèle à SNMP utilisée par le serveur pour diagnostiquer le réseau.

Pour que le serveur puisse interroger le réseau, il va falloir installer les agents sur tous les équipements qui composent le réseau afin que ceux-ci puissent en tout temps enregistrer leurs informations relatives à leurs activités. Comme cela a été expliqué plus haut, ces informations sont enregistrées dans la MIB de l'équipement, ainsi, le serveur va pouvoir interroger n'importe quel équipement pour lui soustraire les informations souhaitées.

3.2.1.2 Fonctionnement détaillé de SNMP[15]

SNMP fonctionne principalement par l'envoi d'une requête SNMP qui va ensuite générer une réponse.

Tout d'abord, le superviseur va interroger un agent, on parle alors de requête SNMP.

Cette requête peut être effectuée avec l'un des quatre messages suivants ;

- « **GetRequest** » qui permet d'obtenir un objet détenu par l'agent
- « **GetNextRequest** » qui permet d'obtenir le prochain objet dans la hiérarchie en arbre de l'agent
- « **GetBulk** » qui permet d'obtenir un ensemble d'objets regroupés
- « **SetRequest** » qui permet de modifier un objet de l'agent

Une fois un de ces messages envoyé, SNMP va répondre avec le message suivant, on parle alors de réponse SNMP :

- « **GetResponse** » qui est utilisé par l'agent pour répondre à l'une des trois requête « GetRequest », « GetNextRequest » ou « GetBulk »

L'agent peut également interagir avec le superviseur pour lui informer qu'une « **trap** », une « **Notification** » ou un « **Inform** » est survenue. Ces messages signalent qu'un événement particulier s'est produit. (SNMP trap et inform seront décrit plus en détail au point suivant). Il peut également y avoir des messages entre les agents SNMP on parle alors de message de type « **Report** ».

Vous pouvez consulter en annexe la configuration du protocole SNMP à appliquer sur un équipement Cisco (**Annexe 2**) afin qu'il puisse interagir avec le superviseur (serveur) SNMP.

3.2.1.3 La trappe SNMP (SNMP trap)

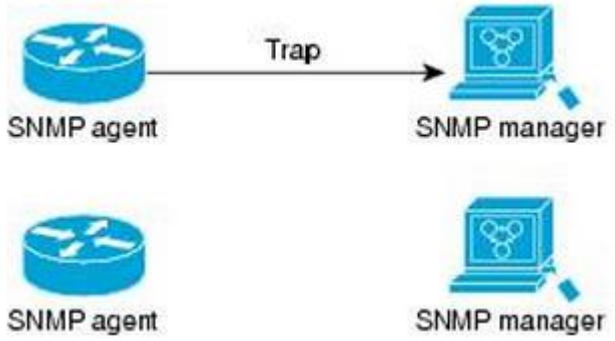
La trappe SNMP va être essentielle pour la détection réseau puisqu'elle se déclenche lorsqu'un agent repère un événement particulier dans le réseau. L'agent va envoyer un message de type trappe au serveur (superviseur) qui va servir de déclencheur afin de pouvoir réagir au problème et ainsi éviter la formation d'une éventuelle congestion

[15] Source frameip.com , SNMP + developpez.com

réseau (ou tout autre problème). Lorsqu'une trappe est envoyée au serveur, celui-ci n'envoie pas de réponse en retour pour indiquer s'il a bien reçu ou non la trappe. Ceci peut être problématique puisque le serveur ne sera alors jamais informé du problème si le message trappe est perdu en cours de route.

L'objectif à réaliser avec ce protocole sera donc de fixer et déterminer des seuils qui permettront de déclencher les trappes SNMP pour ainsi permettre d'alerter l'administrateur. La trappe SNMP est un outil qui permet de détecter la congestion et d'alerter du problème.

Figure 5 Trappe envoyée avec succès au serveur (superviseur)



(cisco.com, Configuring SNMP Support)

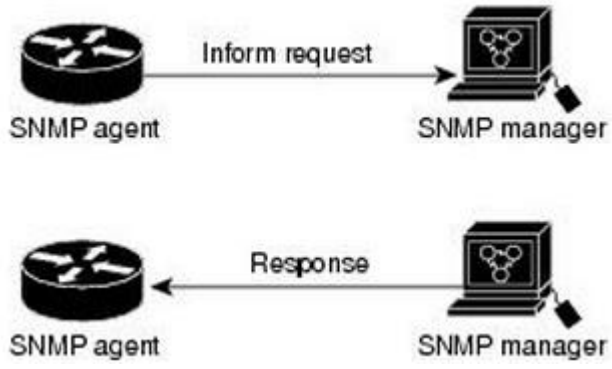
Une trappe SNMP est envoyée au serveur (SNMP manager), le serveur n'envoi aucun message en retour pour informer de la bonne réception de la trappe ou non.

Il est possible de configurer des seuils qui généreront des trappes SNMP qui seront récupérés sur le serveur d'administration. Ce serveur peut ensuite traiter de la manière souhaitée les informations reçues : envoi d'email, log dans un fichier...

3.2.1.3.1 Le message inform

Il existe un message de type « inform » qui fait le même travail que la trappe SNMP, sauf qu'en plus, le serveur doit obligatoirement lui envoyer une réponse pour lui indiquer que l'inform à bien été reçu. Tant qu'une requête inform n'arrive pas à destination, elle sera renvoyée continuellement au serveur jusqu'à réception de l'acquittement de ce dernier. Le message inform consomme bien entendu plus de ressource que la trappe car, un message de réponse (acquittement) doit être envoyé en retour par le serveur (SNMP manager). Il faut donc garder en tête que lorsqu'on a souvent des trappes SNMP qui n'arrive pas à joindre le serveur, on peut utiliser le message inform au détriment d'une consommation plus importante en ressources.

Figure 6 Requête Inform envoyée avec succès au superviseur



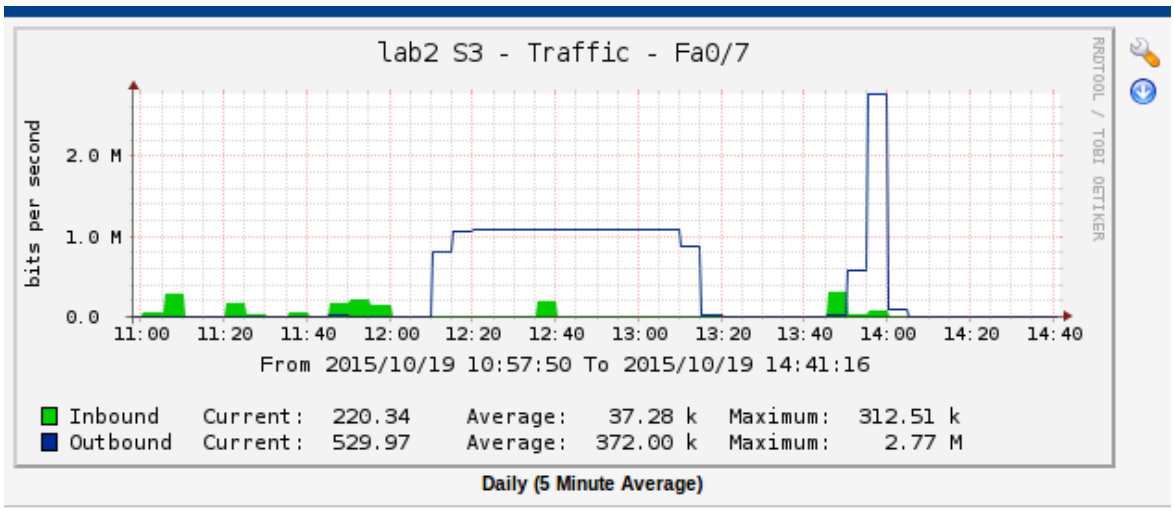
(cisco.com, Configuring SNMP Support)

Une requête inform est envoyée au serveur (SNMP manager), si la requête arrive, le serveur envoie une réponse pour indiquer à l'agent que l'inform à bien atteint sa destination. Tant que l'agent SNMP n'aura pas reçu de réponse, il continuera à envoyer la requête inform.

3.2.1.4 L'interface web Cacti

Cacti est un logiciel open source servant à mesurer les performances de réseaux grâce à la génération de graphiques. Cacti ne sert pas à alerter en cas de problème, mais, va simplement nous permettre d'avoir une vision globale des performances de notre réseau pour un laps de temps donné (jour, semaine, mois ou année). Les graphiques sont générés avec les informations récupérées par le protocole SNMP qui permet de capturer le débit des interfaces des équipements. Une fois la configuration SNMP effectuée sur l'équipement cisco, celui-ci sera reconnu par Cacti qui va alors pouvoir créer des graphiques pour chaque interface de l'équipement générant du trafic. (Source Wikipédia https://fr.wikipedia.org/wiki/Cacti)

Figure 7 Exemple d'un graphique Cacti



Cet exemple nous montre le trafic de l'interface fa0/7 d'un switch. On y voit le trafic entrant (inbound) et sortant (outbound) ainsi que le trafic courant, moyen et maximum. Par défaut, le graphique affiche le trafic d'une journée entière, ici, il a été zoomé, ce qui nous permet de voir plus en détail un point précis de la journée, ici de 11 :00 à 14 :40.

Néanmoins, comme on peut le constater au bas du graphique, il y a une fréquence d'échantillonnage de cinq minutes, ce qui signifie que si une pointe de trafic d'une minute survient, elle ne sera pas visible par Cacti puisque le graphique va calculer la moyenne sur cinq minutes de trafic avant d'afficher le résultat. Exemple, imaginons un trafic régulier de 100kb/s, soudainement, une pointe de trafic d'une durée d'une minute survient à 1000kb/s avant de repasser à 100kb/s. Cacti va alors afficher pour ce laps de temps un trafic moyen de 280kb/s [(100*4 +1000*1) / 5= 280] dans un intervalle de cinq minutes au lieu d'afficher 100kb/s, 100kb/s, 1000kb/s, 100kb/s et 100kb/s.

Tel qu'expliqué plus haut, Cacti ne sert pas à alerter, donc, même si dans ce graphique nous constatons un pic passager aux alentours de 14h00, Cacti ne va rien faire pour nous dire qu'il y a eu un souci. Ce dernier devra, en conséquence, être utilisé conjointement avec d'autres logiciels pour pouvoir à la fois observer le réseau et alerter lorsqu'une anomalie se produit. De plus, on peut également exporter les données du graphique (en cliquant sur le bouton bleu avec la flèche blanche à droite) en format csv afin de les réutiliser à notre guise. Finalement, on peut ajouter à Cacti une large gamme de plugins supplémentaires afin d'augmenter ses capacités. Pour plus d'information vous pouvez consulter la liste officielle des plugins Cacti : http://docs.Cacti.net/plugins.

Il faut également savoir que si on désire monitorer le serveur (Linux) de la même façon que les équipements, il faut installer les services SNMP et SNMPD sur la machine afin de pouvoir déclarer la community[16] et le réseau IP du serveur. Ainsi, on pourra avoir accès à beaucoup d'information concernant notre serveur (trafic des interfaces Eth0 et Lo, connaître la charge processeur, l'état des buffers, etc.). On peut également activer SNMP sur les PCs windows (qui seront les PCs clients dans mes schémas), pour cela il suffit simplement d'activer le service SNMP et de déclarer son superviseur (serveur) ainsi que la community, ces PCs deviendront alors des agents.

Vous trouverez en annexe le guide d'installation de SNMP et SNMPD qui est nécessaire pour monitorer le serveur et le guide d'installation du serveur Cacti (**Annexes 3 et 4**).

[16] La community *du protocole SNMP* fait office de mot de passe en clair pour autoriser la lecture ou l'écriture des OID de la MIB.

3.2.2 NetFlow

NetFlow est un protocole développé par Cisco qui capture les en-têtes des paquets du réseau et permet de récupérer les informations des flux IP. Il permet également de gérer la surveillance des réseaux en générant des statistiques comme par exemple l'analyse du nombre de paquets transitant d'un point A à un point B. Avec NetFlow, on sera capable de contrôler tous les flux de type TCP/IP de notre réseau et détecter d'éventuelles surcharge de trafic grâce aux graphiques que celui-ci va nous générer. Au fil des années, NetFlow n'a cessé de se développer et en est aujourd'hui à sa dixième version.

NetFlow est une référence essentielle pour la surveillance de réseaux et est de nos jours largement répandu dans le monde des réseaux. D'ailleurs certaines entreprises utilisent un autre nom pour le protocole NetFlow (NetStream chez HP et Huawei Technologies, Cflowd chez Alcatel-Lucent, Rflow chez Ericsson, etc). (Source Wikipédia https://fr.wikipedia.org/wiki/NetFlow).

Un flux NetFlow contient les sept champs suivants ; l'adresse IP source, l'adresse IP de destination, la source du port utilisé, le port de destination, la valeur du champ ToS[17], l'interface en entrée et le protocole IP. Vous trouverez en annexe la configuration Netflow à appliquer à un routeur pour que celui-ci puisse fonctionner (**Annexe 5**).

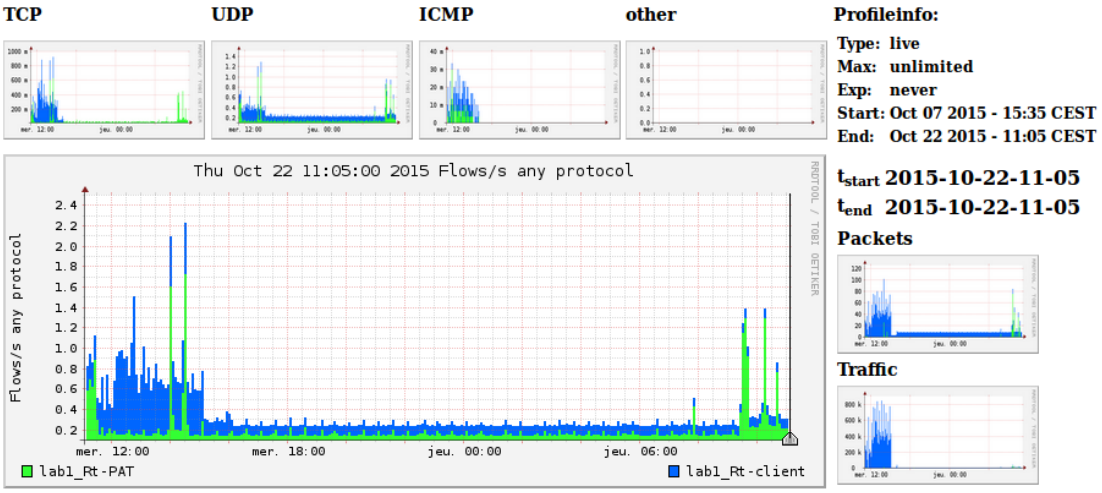
3.2.2.1 NfSen et NfDump

NfSen et NfDump sont une suite open source qui permettent l'affichage par interface web des données de NetFlow. NfDump s'occupe de récupérer les flux NetFlow et de les stocker alors que NfSen est la partie frontend et va permettre l'affichage de ces informations selon plusieurs paramétrages possibles. C'est grâce à ces outils qu'on va pouvoir visualiser les différentes statistiques et graphiques. Vous pouvez consulter en annexe le guide d'installation réalisé pour l'installation de ces deux outils (**Annexe 6**).

[17] Le champ Type of Service permet de distinguer différentes qualités de service différenciant la manière dont les paquets sont traités.

Voici plusieurs images de l'affichage que l'on peut obtenir de l'interface web Nfsen :

Figure 8 Nfsen, exemple d'un graphique journalier



Cette figure montre les flux des routeurs lab1_Rt-PAT et lab1_Rt-client. On constate que lorsqu'il n'y a pas d'activité, les flux restent stables alors que lorsque le réseau est utilisé, ils fluctuent. L'affichage de ces graphiques est assez flexible, comme nous le montre les petits graphiques aux alentours, on peut avoir plusieurs visualisations possibles, il suffit de cliquer dessus pour les afficher.

Figure 9 Nfsen, exemple de données récolté sur des équipements

Statistics timeslot Oct 22 2015 - 11:25

Channel:	Flows:					Packets:					Traffic:				
	all:	tcp:	udp:	icmp:	other:	all:	tcp:	udp:	icmp:	other:	all:	tcp:	udp:	icmp:	other:
lab3_Rt-client	0 /s	0 /s	0 /s	0 /s	0 /s	0 /s	0 /s	0 /s	0 /s	0 /s	0 b/s	0 b/s	0 b/s	0 b/s	0 b/s
lab3_Rt-PAT	0 /s	0 /s	0 /s	0 /s	0 /s	0 /s	0 /s	0 /s	0 /s	0 /s	0 b/s	0 b/s	0 b/s	0 b/s	0 b/s
lab2_R1	0 /s	0 /s	0 /s	0 /s	0 /s	0 /s	0 /s	0 /s	0 /s	0 /s	0 b/s	0 b/s	0 b/s	0 b/s	0 b/s
lab1_Rt-client	1.1 /s	0.7 /s	0.4 /s	0.0 /s	0 /s	8.9 /s	6.2 /s	2.7 /s	0.0 /s	0 /s	50.6 kb/s	48.6 kb/s	2.0 kb/s	3.0 b/s	0 b/s
lab1_Rt-PAT	0.4 /s	0.2 /s	0.3 /s	0 /s	0 /s	33.9 /s	33.4 /s	0.5 /s	0 /s	0 /s	16.2 kb/s	16.0 kb/s	277.8 b/s	0 b/s	0 b/s
	all:	tcp:	udp:	icmp:	other:	all:	tcp:	udp:	icmp:	other:	all:	tcp:	udp:	icmp:	other:
TOTAL	1.6 /s	0.9 /s	0.7 /s	0.0 /s	0 /s	42.9 /s	39.6 /s	3.2 /s	0.0 /s	0 /s	66.9 kb/s	64.5 kb/s	2.3 kb/s	3.0 b/s	0 b/s

All None Display: Sum Rate

Dans cette figure, on peut voir les informations qui transitent à travers les équipements avec trois vues différentes ; les flux, les paquets et le trafic. Cette image nous montre le type de données récoltées sur chaque équipement à un instant T. On peut également afficher le trafic des équipements pour un laps de temps défini, (heures, jours, mois, etc).

Figure 10 Nfsen, exemple de flux récoltés

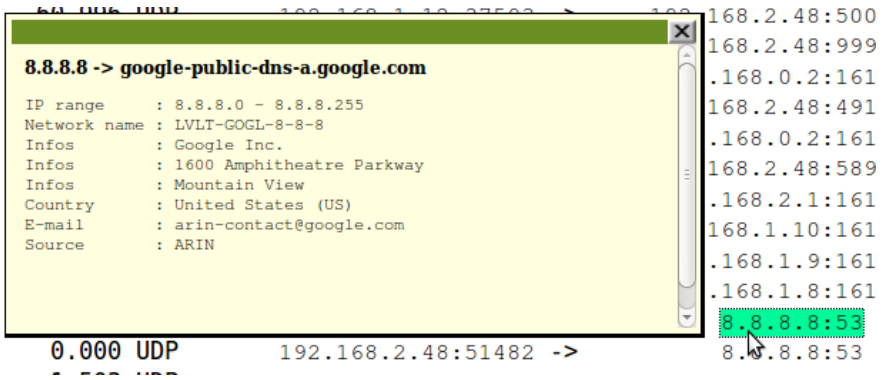
```
** nfdump -M /home/nfsen/profiles-data/live/lab1_Rt-client:lab1_Rt-PAT  -T  -r 2015/10/22/nfcapd.201510221125 -n 10 -s ip/flows
nfdump filter:
any
Top 10 IP Addr ordered by flows:
Date first seen          Duration Proto           IP Addr    Flows(%)     Packets(%)       Bytes(%)         pps      bps   bpp
2015-10-22 11:24:52.801   274.525 any        192.168.1.12      315(66.7)     1991(15.5)    1.9 M(73.8)        7    53918   929
2015-10-22 11:23:36.366   354.426 any        192.168.2.48      156(33.1)    10864(84.5)   656864(26.2)       30    14826    60
2015-10-22 11:24:42.859   281.555 any             8.8.8.8      140(29.7)      140( 1.1)    15713( 0.6)        0      446   112
2015-10-22 11:26:18.597   140.224 any      54.230.201.209       37( 7.8)      617( 4.8)   759394(30.3)        4    43324  1230
2015-10-22 11:26:22.889   184.437 any      104.86.150.168       32( 6.8)      154( 1.2)   121582( 4.8)        0     5273   789
2015-10-22 11:26:12.524   120.337 any       91.198.42.140       19( 4.0)      124( 1.0)   116513( 4.6)        1     7745   939
2015-10-22 11:25:01.562   240.875 any         192.168.0.2       14( 3.0)       24( 0.2)     1728( 0.1)        0       57    72
2015-10-22 11:24:42.595   277.375 any     108.160.172.238       12( 2.5)      269( 2.1)   296083(11.8)        0     8539  1100
2015-10-22 11:26:17.119    15.418 any      212.74.177.170       12( 2.5)      100( 0.8)    73136( 2.9)        6    37948   731
2015-10-22 11:26:12.536    20.005 any      212.147.53.194       11( 2.3)       61( 0.5)    40984( 1.6)        3    16389   671

Summary: total flows: 472, total bytes: 2507424, total packets: 12856, avg bps: 56596, avg pps: 36, avg bpp: 195
Time window: 2015-10-22 11:23:36 - 2015-10-22 11:29:30
Total flows processed: 472, Blocks skipped: 0, Bytes read: 26672
Sys: 0.000s flows/second: 510822.5   Wall: 0.000s flows/second: 1661971.8
```

Cet exemple nous montre un certain nombre d'informations qu'on peut capturer avec Netflow. Tout en bas, un résumé démontre des informations comme des moyennes ou le total de byte ou paquets qui ont véhiculé dans notre réseau. Ceci est l'affichage standard, il faut savoir qu'il y a une grande quantité de filtres applicables disponible dans une liste déroulante pour afficher les données à notre guise. Il y a également une petite fenêtre (pas présente sur cette figure) qui permet de filtrer manuellement les données, en y spécifiant une adresse IP, un protocole ou autre. Bref, si vous voulez en savoir plus sur les multiples possibilités de filtre, je vous renvoie à la documentation officielle Nfsen http://nfsen.sourceforge.net/, l'information est sous la rubrique « Netflow Processing ».

Chaque adresse IP listée peut être consultée plus en détail en cliquant simplement dessus pour afficher son « lookup ». Voici un exemple des informations supplémentaires fournies en consultant le lookup du DNS de Google (8.8.8.8).

Figure 11 Nfsen, exemple d'un IP Lookup



Une des autres possibilités de Nfsen est la création d'un ou plusieurs profiles. Par défaut, Nfsen crée le profil appelé « live » dans lequel on va monitorer notre réseau. Un profil est une vue spécifique des données Netflow. On a alors la possibilité de créer un nouveau profil où on va sélectionner plus en détail les éléments de notre réseau que nous voulons

contrôler. Par exemple, si je vérifie très souvent le trafic internet de mon réseau, je peux alors créer un nouveau profile avec un filtre (exemple de filtre : proto tcp and dst port 80) qui va m'afficher le trafic http exclusivement. Cette option est très utile lorsqu'on monitore un réseau avec plusieurs sources. Pour créer un profil il suffit de cliquer sur la liste déroulante des profils dans le menu Nfsen et cliquer sur « New profile ».

Nfsen permet également de gérer des alertes qu'on doit nous-mêmes définir. Par exemple, on peut programmer Nfsen pour nous envoyer un email d'alerte lorsque le réseau atteint un débit trop important. Un onglet « alert » permet de gérer le tout et sera décrit à la rubrique 4.1 de ce travail.

Il existe la possibilité d'ajouter divers plugins à Nfsen pour augmenter ses capacités, cependant je n'en utiliserai pas puisque la configuration par défaut est suffisante pour la réalisation de ce travail.

En résumé, l'outil Netflow qui fonctionne grâce à Nfsen et NfDump va s'avérer très utile pour la détection de congestion dans un réseau. De plus, il permet de visualiser un problème rapidement grâce à la génération flexible de ses graphiques et l'affichage de ses statistiques.

3.2.3 Le protocole Syslog

Syslog est un protocole qui permet de transmettre les logs générés par les équipements réseau vers un serveur Syslog à travers le port UDP 514. Son intérêt est la centralisation de tous les messages des équipements du réseau pour faciliter le dépannage. Lorsqu'un problème survient, on peut facilement retrouver sa source en consultant les logs du serveur Syslog sans avoir à analyser chaque équipement du réseau un par un.

Un message de type Syslog sur un équipement réseau permet d'afficher plusieurs informations comme la date exacte du jour de l'équipement, le processus déclencheur, le niveau de sévérité[18] et le corps du message.

Figure 12 Exemple de logs sur un routeur Cisco

```
Oct  1 09:13:11.752: %LINK-5-CHANGED: Interface Serial0/0, changed state to administratively down
Oct  1 09:13:12.753: %LINEPROTO-5-UPDOWN: Line protocol on Interface Serial0/0, changed state to down
Oct  1 09:13:19.556: %LINK-3-UPDOWN: Interface Serial0/0, changed state to up
Oct  1 09:13:20.558: %LINEPROTO-5-UPDOWN: Line protocol on Interface Serial0/0, changed state to up
Oct  1 09:14:42.207: %SYS-5-CONFIG_I: Configured from console by console
```

Si on consulte les logs sur un serveur Syslog, on a comme information supplémentaire le nom ou l'IP de l'équipement et la date du serveur.

[18] Le niveau de sévérité est à définir et peut contenir 8 valeurs : 0 emergency, 1 alert, 2 critical, 3 error, 4 warning, 5 notice, 6 informational, 7 debugging

Figure 13 Exemple de logs sur un serveur Syslog

```
Oct  1 16:20:56 192.168.2.1 67: .Oct  1 14:20:51.136: %LINK-5-CHANGED: Interface Serial0/0,
 changed state to administratively down
Oct  1 16:21:01 192.168.2.1 68: .Oct  1 14:20:52.138: %LINEPROTO-5-UPDOWN: Line protocol on
 Interface Serial0/0, changed state to down
Oct  1 16:21:06 192.168.2.1 69: .Oct  1 14:20:55.984: %LINK-3-UPDOWN: Interface Serial0/0,
changed state to up
Oct  1 16:21:11 192.168.2.1 70: .Oct  1 14:20:56.985: %LINEPROTO-5-UPDOWN: Line protocol on
 Interface Serial0/0, changed state to up
```

Il faut savoir que Syslog permettra également de récolter les données des outils Nfsen et Cacti décrits plus haut si on les configure correctement. Le fait de tout centraliser sur notre serveur Syslog peut être une bonne chose pour autant que l'on retrouve les informations souhaitées. En effet, Syslog génère énormément de log, il faudra donc trouver une technique pour filtrer les informations recherchées.

3.2.3.1 Configuration de Syslog

On va d'abord configurer Syslog sur un équipement Cisco pour qu'il puisse envoyer ses messages de journalisation au serveur Syslog. La première commande à effectuer est « logging ' *ip_serveur_syslog* ' » qui permettra d'indiquer à l'équipement, le serveur Syslog auquel il va devoir envoyer ses messages. La deuxième commande essentielle sera « logging trap '*severity_level*' » qui permettra d'indiquer le niveau de sévérité des messages à capturer. Voir la configuration complète en annexe (**Annexe 7**).

Ensuite, il ne reste plus qu'à installer et configurer le serveur Syslog, pour cela, merci de consulter en annexe le petit guide que j'ai conçu pour l'installation d'un serveur Syslog sur Ubuntu (**Annexe 8**).

(Source developpez.com http://ram-0000.developpez.com/tutoriels/reseau/Syslog/)

3.2.4 La méthode Storm control

Storm control sert à surveiller la surcharge de trafic en définissant des seuils qui vont permettre : D'alerter avec une trappe SNMP, de supprimer un surplus de trafic en bloquant temporairement l'interface ou de bloquer directement l'interface en la mettant « Shutdown ». Storm control permet de gérer tous les types de trafic ; broadcast, multicast et unicast. Il compare le trafic du réseau avec les seuils définis afin de pouvoir agir. Ces seuils sont aussi appelés « Storm control level » qui peuvent être représentés de trois façons différentes :

1. Pourcentage du total de la bande passante d'une interface physique.
2. Paquets par seconde (PPS)
3. Bits par seconde (BPS)

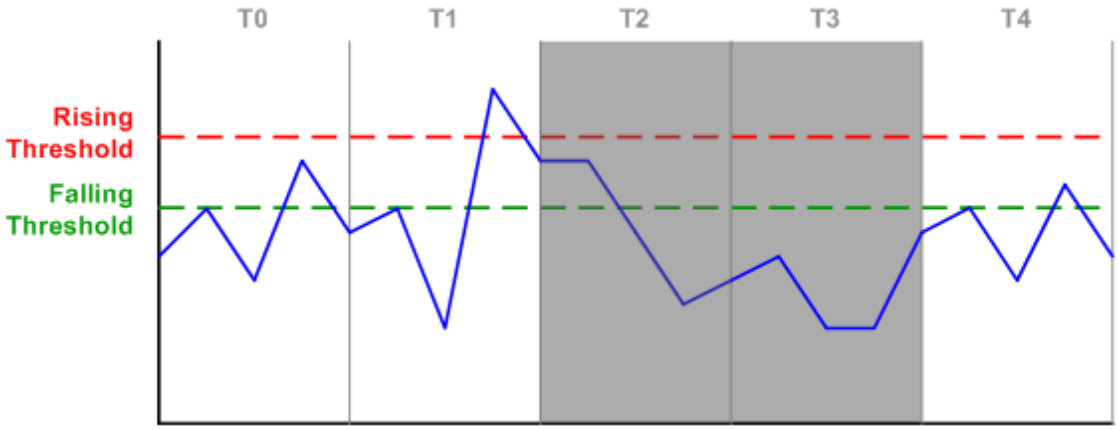
Une information très importante concernant Storm control est qu'il ne s'applique que sur le trafic entrant, il faudra alors veiller à bien le configurer sur les deux extrémités d'un équipement interconnecté pour être sûr de le déclencher.

Chaque port peut avoir sa propre règle de storm control. Dans le cadre de mon travail, je vais surtout m'intéresser à la partie qui permet de générer des trappes. La partie qui shutdown l'interface sera tout de même montrée dans la phase de test à la fin de ce travail étant donné qu'une des mesure correctives de la congestion réseau peut être la décision par un administrateur de bloquer le port subissant trop de trafic.

3.2.4.1 Fonctionnement détaillé de Storm control

Lorsqu'on va définir nos Storm control level (qu'on peut également appeler « seuil » ou encore « threshold » en anglais), voici ce qui va se passer :

Figure 14 Storm control, exemple de seuils



(packetlife.net, Storm Control)

Dans cette figure, on a configuré deux seuils ; rising threshold qu'il ne faudra pas dépasser sous peine de voir le trafic bloqué et falling threshold qui sera activé uniquement lorsque le « rising threshold » sera dépassé. Il faudra alors repasser au-dessous de ce seuil pour pouvoir ré-autoriser le trafic.

0. En T0 on dépasse le falling threshold mais, il ne se passe rien puisque ce seuil ne sert pas à bloquer le trafic.
1. En T1, le rising threshold est dépassé, on va alors bloquer le trafic pour T2.
2. En T2, on n'est toujours pas passé au-dessous du falling threshold, le trafic reste alors bloqué pour T3.
3. En T3 le trafic est plus bas que le falling threshold et sera réactivé pour T4.

3.3 Sélection des outils les plus pertinents

Tous les outils potentiellement utilisables pour la réalisation de mon travail ont été décrits plus haut. Même si la liste n'est pas exhaustive, je vais maintenant pouvoir choisir les plus adaptés. En effet, en ayant étudié tous ces outils je suis maintenant plus à même faire le bon choix.

Voici donc la liste des outils que je vais utiliser pour contrôler la congestion de mes schémas prototype :

- Le protocole SNMP et son interface web Cacti
- Netflow et son interface web Nfsen
- Un serveur Syslog
- La méthode Storm control

A cela, s'ajoute le générateur de trafic Iperf3 et Iperf2 pour pouvoir générer de la congestion. Ces outils sont selon moi les plus pertinents puisqu'en plus d'être très répandu dans le domaine du réseau, ils permettent de générer des graphiques, ce qui attirera l'attention lorsqu'on va observer le trafic du réseau.

On constate que presque tous les outils cités plus haut vont être utilisés pour mon travail. En effet, lors de la mise en place de ce travail de Bachelor les mandants m'avaient déjà aiguillé sur quelques technologies à exploiter, c'est pourquoi, grâce à leurs conseils avisés j'ai été capable de chercher l'information directement au bon endroit.

IP SLA ne sera pas utilisé simplement parce qu'Iperf est assez efficace à lui seul pour pouvoir générer du trafic.

Maintenant que les outils de détection de la congestion ont été passés en revue et sélectionnés, on va passer à l'étape suivante qui va être d'alerter du problème lorsque la congestion a été détectée.

4. Transmission, alerte de la congestion

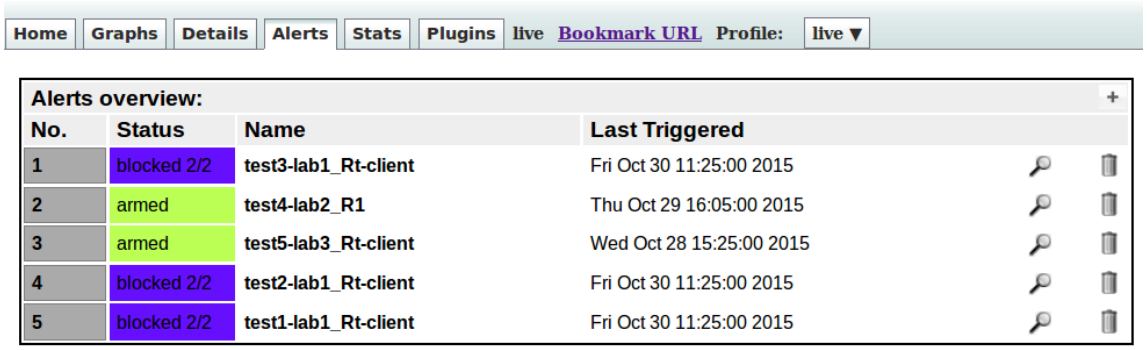
Cette étape est la suite logique à la détection de la congestion réseau. Maintenant que nous avons tous les outils en main pour analyser et détecter le trafic de notre réseau, il va falloir alerter l'administrateur (ou n'importe quelle personne responsable du réseau) pour qu'il puisse agir en conséquence lorsqu'une congestion se produit.

Dans cette rubrique, on va voir comment se déroulent les étapes d'alerte lorsque le réseau commence à être surchargé. Pour cela je vais simplement utiliser des outils décrits au point 3 qui disposent du nécessaire pour effectuer cette opération.

4.1 Alerter avec Nfsen

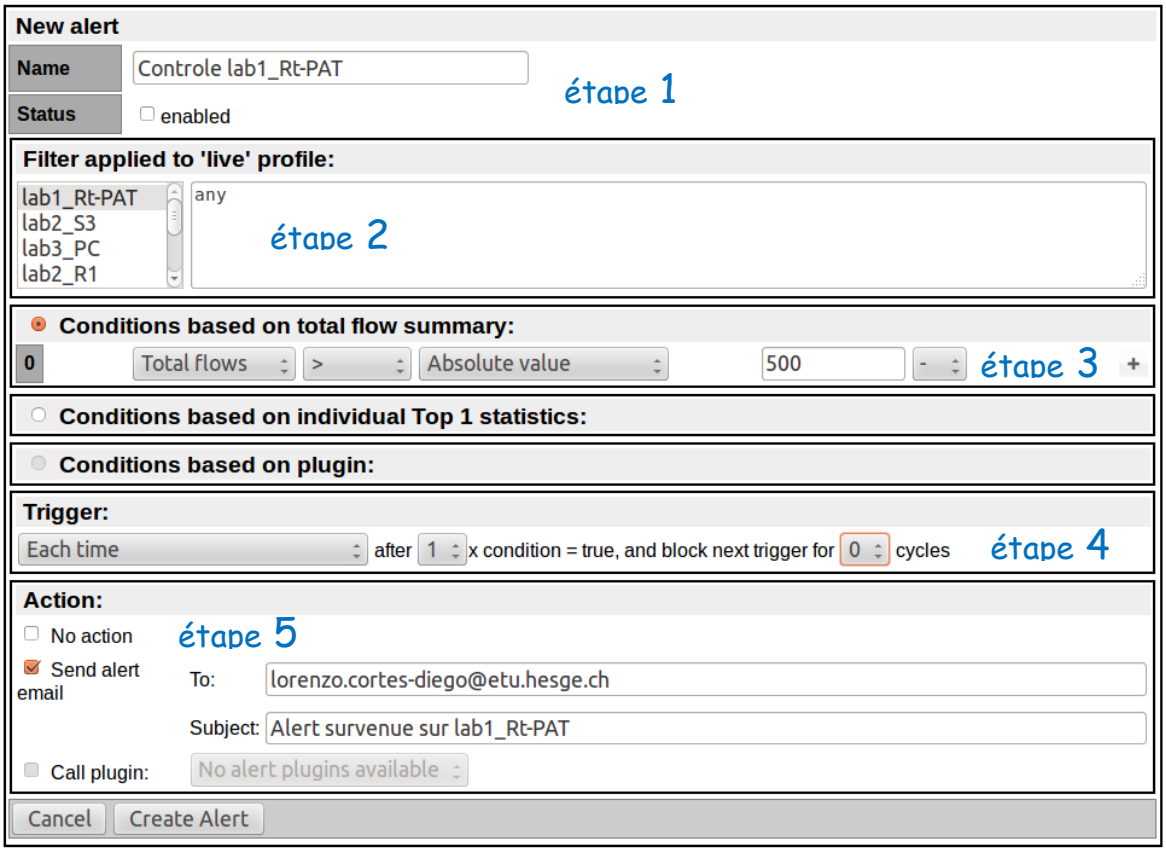
Nfsen dispose d'une partie spécialement dédiée aux alertes, on peut y définir le nombre d'alertes qu'on veut et envoyer un email à une personne lorsque celle-ci se déclenche. C'est un outil relativement puissant car, les seuils qu'on peut y définir vont nous permettre de détecter la congestion à plusieurs niveaux. On pourra déterminer une alerte lors d'un simple pic passager et tout aussi bien lors d'une longue période de surcharge de trafic. Voyons un peu plus en détail le fonctionnement des alertes de Nfsen.

Figure 15 Nfsen, exemple d'alertes



Voici la page de vue d'ensemble des alertes de Nfsen, on y voit, le statut de l'alerte, son nom et la dernière date à laquelle elle a été déclenchée. Le statut « armed » signifie simplement que l'alerte est active et blocked 2/2 signifie que l'alerte est active et qu'elle a été bloquée durant deux cycles. Pour créer une alerte, il faut cliquer sur le bouton en haut à droite.

Figure 16 Nfsen, Exemple page de création d'une alerte



Voici la page qui s'affiche lorsqu'on crée une nouvelle alerte.

La première étape consiste à donner un nom et à activer l'alerte. On peut d'ailleurs le désactiver à tout moment en décochant la case « enabled ».

La deuxième étape va être de filtrer l'équipement qu'on veut analyser en le sélectionnant à gauche et écrire « any » dans la zone de commentaire. Si on n'écrit pas « any », par défaut le texte « not any » va être configuré et notre alerte ne sera jamais déclenchée.

La troisième étape est la partie la plus importante, on va devoir définir nos seuils afin de décider quand l'alerte se déclenchera. Dans l'exemple ci-dessus, on va la déclencher lorsque le nombre total de flux sera supérieur à 500. Il y a d'autres options possibles, à la place du total de flux on peut choisir, le nombre total de paquet, le nombre total de bytes, les bits/s, etc. A la place de la valeur absolue on peut choisir la valeur moyenne selon un laps de temps (10 minutes, 1heures, 2heures, etc.). On peut également rajouter jusqu'à six conditions en cliquant sur le bouton « + » à droite. Attention, si on en ajoute plusieurs, c'est le critère de programmation « OU » qui s'applique et non le « ET ».

La quatrième étape va nous permettre de définir quand déclencher l'alerte (trigger). Les options possible sont: « each time », « once only » et « once only while condition=true ». On peut ensuite définir après combien de fois la condition devra être respectée pour se déclencher ainsi que le nombre de cycles durant lesquels l'alerte sera bloquée. Ces paramètres sont également très important puisqu'une alerte se déclenche par défaut chaque cinq minutes lorsque son seuil est atteint. Cela nous permet notamment de ne pas envoyer d'email lors de fausses alertes où un petit pic de trafic fait son apparition avant de disparaître aussitôt. Un des objectifs de ce travail va être de définir des seuils pertinents pour ne pas inonder l'administrateur d'alertes inutiles.

La cinquième étape permet de définir l'action à effectuer. Soit ne rien faire, dans ce cas on peut imaginer que l'administrateur est devant l'interface Nfsen et il consulte directement les alertes, soit envoyer un email à la personne souhaitée, ainsi, pas besoin d'être devant son écran à attendre qu'une alerte se déclenche.

Une fois ces étapes de création d'alerte terminées, on peut l'éditer si on souhaite modifier ou corriger sa configuration. On peut alors changer n'importe quel champ présent dans la figure 16 à l'exception du titre. On peut également la consulter. Voici les informations qui sont affichées :

Figure 17 Nfsen, Informations d'une alerte

		Last	Avg 10m	Avg 30m	Avg 1h	Avg 6h	Avg 12h	Avg 24h
	Flows	493.6 k	504.1 k	510.3 k	447.2 k	165.8 k	143.8 k	292.5 k
		1.6 k/s	1.7 k/s	1.7 k/s	1.5 k/s	552.6 /s	479.2 /s	974.9 /s
	Packets	7.3 M	7.6 M	7.2 M	6.7 M	3.4 M	3.7 M	5.8 M
		24.3 k/s	25.2 k/s	24.0 k/s	22.3 k/s	11.3 k/s	12.2 k/s	19.5 k/s
	Bytes	6.6 GB	6.8 GB	6.5 GB	6.1 GB	3.1 GB	3.4 GB	5.4 GB
		176.8 Mb/s	181.9 Mb/s	172.2 Mb/s	161.4 Mb/s	83.6 Mb/s	91.7 Mb/s	144.2 Mb/s

Sourceforge.net, Nfsen – Netflow sensor

Ce graphique permet de voir les valeurs moyennes du résultat à filtrer. Ainsi, on pourra ajuster facilement nos seuils au besoin. Les lignes verticales indiquent les moments auxquels les alertes se sont déclenchées. Les différentes lignes horizontales du graphique représentent les moyennes selon un temps donné, on peut voir à quoi elles correspondent en légende (par exemple avg10m = moyenne sur 10 minutes). Les valeurs des courbes vont permettre de nous aider à paramétrer correctement nos conditions d'alerte. Pour identifier une pointe de trafic, on va plutôt utiliser la moyenne sur 10 minutes (courbe bordeaux) plutôt que la moyenne sur 24 heures (courbe bleue).

En dessous, le grand tableau nous permet de changer la vue du graphique en affichant les flux, les paquets ou les bytes. Le petit tableau nous montre simplement l'état des conditions définies et si l'alerte a été déclenchée.

4.1.1.1 Anomalie détectée avec les alertes Nfsen

J'ai détecté un petit bug lorsqu'on donne un nom à une alerte. Il faut éviter d'utiliser des caractères spéciaux (>, <, +, -, *, etc) car cela peut empêcher le graphique ci-dessus (figure 16) de s'afficher. Après avoir testé, lorsque nous créons une alerte avec le caractère « > » nous recevons une notification qui nous empêche de créer l'alerte alors qu'avec le caractère « + », nous ne recevons pas d'erreur et le graphique n'arrive pas à s'afficher.

4.1.2 L'envoi d'email avec Postfix

Postfix est un serveur de messagerie électronique open source. Il se charge de la livraison de courriers électroniques (courriels) et a été conçu comme une alternative plus rapide, plus facile à administrer et plus sécurisée que l'historique Sendmail[19]. (Source Wikipédia https://fr.wikipedia.org/wiki/Postfix).

Pour sa configuration, j'ai décidé d'utiliser une adresse Gmail. J'ai dû spécifier au protocole SMTP (Simple Mail Transfer Protocol) le serveur Google à utiliser : smtp.gmail.com ainsi que mon adresse Gmail. De plus, Google ne lésine pas au niveau de la sécurité, il m'a donc aussi fallu générer un certificat SSL[20] pour permettre à Nfsen d'envoyer des courriers depuis mon compte Gmail.

La dernière étape est de spécifier dans la configuration Nfsen l'adresse email qui sera utilisée pour envoyer les alertes. On doit également indiquer le serveur SMTP à utiliser,

[19] Serveur de messagerie qui est critiqué pour sa lenteur, sa complexité et sa maintenance difficile en comparaison à Postfix (Source Wikipédia https://fr.wikipedia.org/wiki/Sendmail)

[20] Un certificat SSL atteste l'identité d'une entreprise et est utilisé afin de crypter les données échangées sur un réseau (Source Google https://www.google.ch/webhp?sourceid=chrome-instant&ion=1&espv=2&ie=UTF-8#q=certificat%20ssl)

il faut alors spécifier « localhost ». Ainsi, l'alerte sera envoyée à la machine qui va utiliser Postfix pour faire le relais et envoyer l'email au bon destinataire. Vous trouverez en annexe le guide d'installation et configuration de Postfix pour une adresse Gmail (**Annexe 10**).

4.2 Alerter avec Syslog

Dans cette partie on va s'intéresser à comment récupérer les informations sur le serveur Syslog. En principe, les trappes générées par les équipements Cisco vont se stocker dans les logs. Maintenant que nous avons vu comment alerter par email, une des prochaines étapes est de consulter les logs pour avoir plus d'information sur cette alerte.

Je vais vous expliquer comment filtrer nos logs sans avoir besoin d'une éventuelle application supplémentaire, c'est-à-dire en n'utilisant que les commandes de Linux. La méthode qui vous sera présentée ci-dessous est une idée parmis d'autres, il en existe probablement d'autres.

Avant de commencer, il est judicieux de configurer le logrotate[21] afin de toujours être certain de savoir où se trouve l'information que nous recherchons. Une configuration pour créer un fichier syslog par jour et les conserver durant sept jours est disponible en annexe (**Annexe 8**).

La première étape est de créer un nouveau fichier de log (sous /var/log par exemple) où vont aller s'afficher les informations qu'on souhaite récupérer.

La deuxième étape consiste à créer un script qui va :

- parcourir le fichier Syslog
- récupérer l'information souhaitée grâce à la commande « grep »
- rediriger le contenu trouvé avec la commande « > » dans notre fichier créé à la première étape

Cela parait simple à première vue, mais, un problème essentiel se pose. Comment récupérer les logs de manière continue puisque rappelons-le, le fichier Syslog se met à jour très souvent. Un parcours unique ne permettra pas à notre filtre de récupérer les informations qui arriveront après le lancement du script étant donné que la plupart des langages de programmation son prévus, lorsqu'on parcourt un fichier, pour s'arrêter à la fin de celui-ci (end of file). Une solution qui permet de contourner ce problème (certainement pas la plus efficace) est de créer une boucle infinie dans notre script qui

[21] logrotate est conçu pour faciliter l'administration des logs générant un grand nombre de contenu. Il automatise la permutation, la compression, la suppression et l'envoi des logs.

va analyser en continue le fichier Syslog afin d'y répertorier les éventuels nouveaux messages qui arriveront après coup. Certes le fait d'avoir une boucle infinie qui tourne en arrière-plan n'est certainement pas la meilleure des méthodes, mais ça marche.

La troisième et dernière étape est facultative, mais peut s'avérer utile. Elle consiste simplement à envoyer par email les fichiers de log créés. Vous trouverez en annexe les étapes pour réaliser des filtres Syslog via Linux ainsi que l'envoi par email de ceux-ci (**Annexe 11**).

Idéalement, le mieux serait d'avoir une interface web pour gérer les logs et ainsi pouvoir les filtrer ou les exporter directement. Il en existe plusieurs comme Kiwi qui dispose d'énormément de fonctionnalités en plus d'être très connu mais qui est malheureusement payant (240 euros). Kiwi dispose aussi d'une version light gratuite moins intéressante (uniquement cinq équipements peuvent envoyer des logs en même temps). Une autre alternative et gratuite cette fois ci est Graylog2 qui est une solution open source qui utilise une interface web écrite en Ruby (langage de programmation) qui permet entre autre de filtrer les logs via des cases à cocher.

Cependant, les messages Syslog sont souvent consultés « à la volée », d'où le fait d'avoir présenté une méthode automatisée ci-dessus. Pour filtrer les logs via une interface web, il sera plus difficile de réaliser automatiquement les tâches de filtrage et une personne physique devra probablement être présente.

4.3 Alerter avec prudence

Nous venons de voir quelques moyens d'alerter lorsqu'une congestion se produit dans un réseau. Bien que cette étape soit très importante, il faut avoir en tête que trop d'alertes tuent les alertes. Si on en reçoit plusieurs par jour, on ne va plus y prêter attention. Il faudra donc bien faire attention à définir des seuils avec parcimonie afin de ne pas inonder les administrateurs de messages inutiles.

Par exemple, il sera important de bien faire la différence entre un pic de trafic de quelques secondes et un long trafic de plusieurs minutes. Pour cela plusieurs techniques existent. Avec Nfsen par exemple, on peut décider après combien d'occurrence se déclenchera une alerte et pendant combien de cycle nous voulons la bloquer, ainsi, on pourra facilement ignorer une petite pointe de trafic sans importance.

5. Partie pratique, les schémas prototypes

Maintenant que toute la partie préliminaire sur la congestion réseau est terminée, il va falloir mettre en œuvre le tout sur plusieurs schémas réseau afin de pouvoir faire des tests en tirer des conclusions.

Il a été décidé de mettre en œuvre trois schémas différents qui devront permettre de créer des cas de congestion que l'on peut rencontrer dans un réseau. Ces trois schémas ont pour but de visualiser un problème de congestion dans sa totalité et sous différents aspects.

Le premier schéma visera à simuler une bande passante de petite taille grâce à un lien série. Le deuxième schéma sera constitué de trois commutateurs avec un petit débit à un certain endroit pour permettre la formation de congestion. Le troisième aura pour objectif de créer des collisions qui montent en flèche grâce à un câble coaxial 10base2[22].

Le but de ces différents montages sera de générer du trafic avec l'application Iperf afin de créer de la congestion réseau. Cette étape permettra de mettre en œuvre la démarche de détection de congestion, à savoir; détecter la congestion lors de sa formation, transmettre ou alerter du problème de congestion et éventuellement prendre les mesures correctives nécessaires.

5.1 Matériel utilisé

Les montages ont été élaborés dans un laboratoire Cisco de la HEG. Le matériel comprend des équipements réseau Cisco ainsi que des PCs avec une configuration standard.

5.1.1 Les routeurs

Sur le schéma n°1, il y a deux routeurs Cisco 2651XM avec les versions (C2600-ADVENTERPRISEK9-M), Version 12.4(25d) et (C2600-IS-M), Version 12.2(15)T17
Sur le schéma n°2, il y a un routeur Cisco 2621 avec la version (C2600-ADVENTERPRISEK9-M), Version 12.4(15)T11
Sur le schéma n°3, il y a un routeur Cisco 2651XM et un routeur 2621 avec les versions (C2600-ADVSECURITYK9-M), Version 12.3(21) et (C2600-ADVSECURITYK9-M), Version 12.4(12).

[22] Standard Ethernet de la couche physique dans le modèle OSI utilisant un câble coaxial fin. Celui-ci permet le transfert de données à des débits jusqu'à 10 Mbits/s (Source Wikipédia https://fr.wikipedia.org/wiki/10BASE2)

5.1.1.1 Compatibilité

Certain de ces routeurs ne sont pas compatibles pour faire du IP SLA qui est compatible uniquement avec des IOS de version 12.4 ou plus. De plus, les commandes de configuration d'IP SLA sont disponibles sous deux versions différentes selon la modernité de l'appareil utilisé. Pour voir ces quelques différences, vous pouvez consulter cet article qui explique la différence entre les deux manières de faire : https://www.plixer.com/blog/netflow/cisco-ip-sla-monitor-or-just-ip-sla/

5.1.2 Les commutateurs (switch)

Utilisés uniquement sur le schéma n°2, les commutateurs Cisco sont un Catalyst 2950 avec la version (C2950-I6K2L2Q4-M), Version 12.1(22)EA8a, un Catalyst 2960 avec la version (Cisco IOS Software, C2960 Software (C2960-LANBASEK9-M), Version 15.0(2)SE7, RE) ainsi qu'un Catalyst 3550 avec la version C3550 Software (C3550-IPSERVICESK9-M), Version 12.2(50)SE3.

5.1.2.1 Compatibilité

Les trois Switchs à ma disposition cités ci-dessus sont incompatibles avec Netflow. Pour pouvoir programmer Netflow sur des Switchs il faut utiliser des Catalyst 45xx, 55xx ou 6xxx qui sont des switchs plus récents, plus imposants et beaucoup plus performantes, ce qui implique un prix relativement plus cher. Pour voir toutes les compatibilités relatives à Netflow vous pouvez visiter le site ci-après : https://www.manageengine.com/products/netflow/help/cisco-netflow/netflow-ios-versions.html

Concernant storm control, après avoir fait plusieurs tests sur mes trois types de switch, voici ce qu'il en ressort : Le Catalyst 2960 est totalement compatible avec storm control (cependant, la commande « snmp-server enable traps storm-control trap-rate *value* » qui est censé permettre de définir combien de fois par minute autoriser une trappe Storm control ne fonctionne pas). Le Catalyst 3550 peut également faire du Netflow mais il lui manque quelques commandes[23] qui ne gênent pas la configuration du storm-control. Les Catalyst 2950 sont à moitié compatible, on peut « shutdown » le port après avoir atteint un seuil fixé, mais, il n'est pas possible d'envoyer des trappes et ce, malgré le fait que la commande soit disponible. De plus, si on souhaite spécifier un seuil avec des bits par seconde, il faudra utiliser des switch plus récent (fonctionne avec le Catalyst 2960) puisque ceux-ci permettent uniquement de définir des seuils en fonction du pourcentage de bande passante et des paquets par secondes.

[23] On ne peut pas définir un seuil avec les bits par seconde commande : *storm-control unicast level **bps** 10m*

5.1.3 Les ordinateurs et les serveurs

Les ordinateurs utilisés sont ceux de la HEG et disposent des systèmes d'exploitation Windows 7 et Ubuntu 12.04 LTS.

Les serveurs des différents schémas sont simulés par un PC de la HEG avec un serveur Apache2 que j'ai configuré sous Linux (Ubuntu).

5.1.4 Le câblage

L'utilisation de câbles droits et croisés est présent sur tous les schémas.

Sur le schéma n°1, il y a un câble série pour relier les routeurs et sur le schéma n°3 il y a des câbles coaxiaux pour relier les routeurs et deux PCs.

Les câbles coaxiaux sont en 10base2 et utilisent donc le protocole CSMA/CD (Carrier Sense Multiple Access / Collision Detection) pour gérer ses collisions. Quand CSMA/CD veut transmettre des informations, elle va vérifier qu'aucune transmission ne soit en cours, si c'est le cas, les trames seront transmises au récepteur, sinon, CSMA/CD va attendre la fin de la communication en cours pour envoyer ses données. En résumé, on fait du half-duplex, une seule communication à la fois.

5.2 Configuration

Chaque schéma à bien évidemment sa propre configuration, cependant certains standards ont été appliqués partout. Chaque équipement a été configuré avec les règles standards de base, à savoir la mise en place d'un mot de passe, la protection des lignes vty pour gérer les accès à distance et l'encryption des mots de passe. Toutes les interfaces non utilisées sont shutdown.

Les trois montages sont fait avec du PAT[24]en sortie vers internet. Le-s PC.s client-s récupèrent leurs adresses privées en DHCP dans une plage de plusieurs adresses disponibles.

La liaison entre le routeur et le réseau externe dans les différents schémas est toujours configurée en DHCP et va ainsi récupérer automatiquement une adresse IP sur son interface afin d'avoir accès à internet sur les trois schémas.

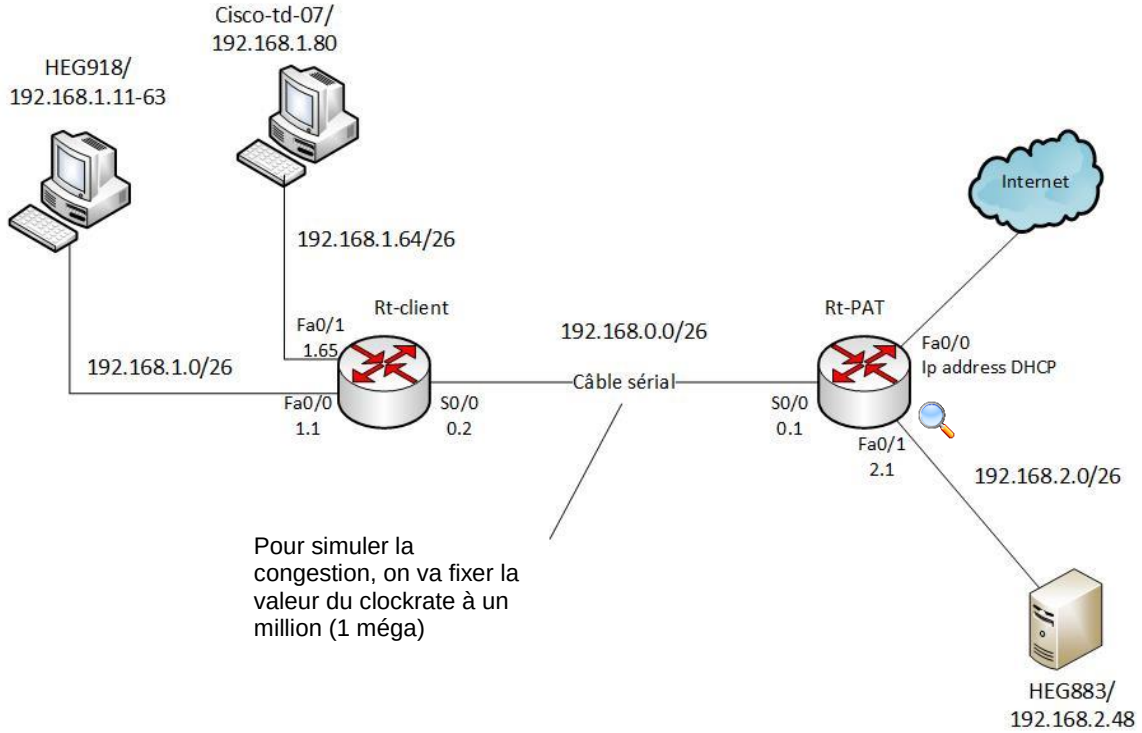
Le serveur sert à monitorer le réseau grâce à Syslog, Cacti et Nfsen qui sont décrits plus haut. Il servira également à générer du trafic sur les différents schémas grâce à Iperf. Pour plus d'information, vous pouvez consulter les différentes configurations des équipements en annexe mentionnées dans les **points 5.3 à 5.5.**

[24] Le Port Adress Translation sert à effectuer une translation des ports IP entre l'intérieur d'un réseau privé (adresse privé) et une adresse IP (adresse de sortie) sur internet

5.3 Schéma n°1, chaine de deux routeurs avec lien série

Ce schéma comprend deux PCs client dans des sous réseaux différents, un serveur et deux routeurs.

Figure 18 Schéma n°1, routeurs avec lien série



5.3.1 Caractéristiques du schéma

On veut créer une congestion réseau principalement entre les deux routeurs, c'est pourquoi ils sont interconnectés avec un lien série simulant 1méga de bande passante. A l'exception du lien série, le reste du réseau est interconnecté par une bande passante de 100méga. Il sera ainsi très facile de créer une congestion. La petite loupe représente l'interface qui sera utilisée pour le monitoring.

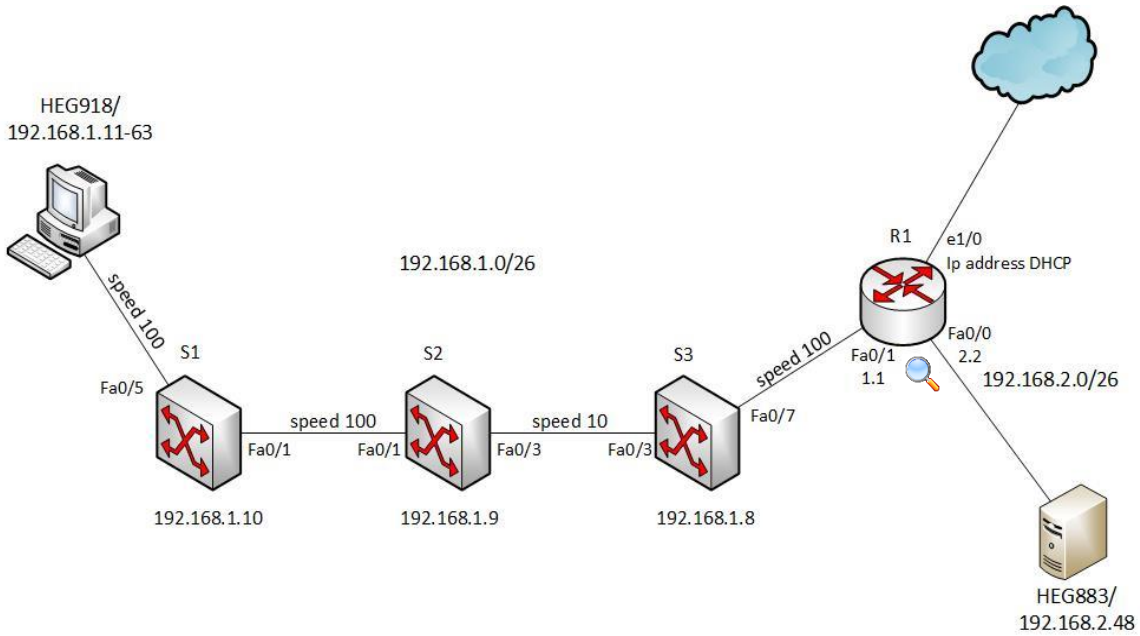
Dans ce schéma, on va simuler une boucle de routage de la façon suivante : Sur le routeur Rt-PAT, on va faire un résumé de route pour les deux sous réseaux clients en une ligne : « ip route 192.168.1.0 0.0.0.255 ». Depuis le serveur, on va pinger une adresse IP qui est résumé par la route, mais, qui n'est dans aucun des deux sous-réseaux (192.168.1.201 par exemple). Une boucle de routage va se créer puisque le paquet connait sa route, mais ne trouvera pas sa destination.

Pour plus d'information, voir la configuration complète des équipements du schéma en annexe (**Annexe 12**).

5.4 Schéma n°2, chaine de trois commutateurs

Ce schéma comprend un PC client, un serveur, trois commutateurs et un routeur.

Figure 19 Schéma n°2, chaîne de switch avec débit différent



5.4.1 Caractéristiques du schéma

Dans ce schéma on veut créer une congestion réseau principalement entre les deux commutateurs étant interconnectés avec une bande passante de 10mégas (speed 10). Les commutateurs ne disposent pas des commandes de configuration de Netflow et ne pourront donc pas déclencher d'alertes avec Nfsen. Cependant, on aura tout de même un moyen d'alerter en cas de problème grâce à S2 (Catalyst 3550) et S3 (Catalyst 2960) via storm control. Il faudra alors aller récupérer les messages dans les logs du serveur ou du switch lui-même directement si on décide d'envoyer une trappe. Si on décide de bloquer le port, il faudra simplement constater que le trafic ne passe plus. On va faire du storm control uniquement du côté où on a fixé le speed à 10. La petite loupe représente l'interface qui sera utilisée pour le monitoring.

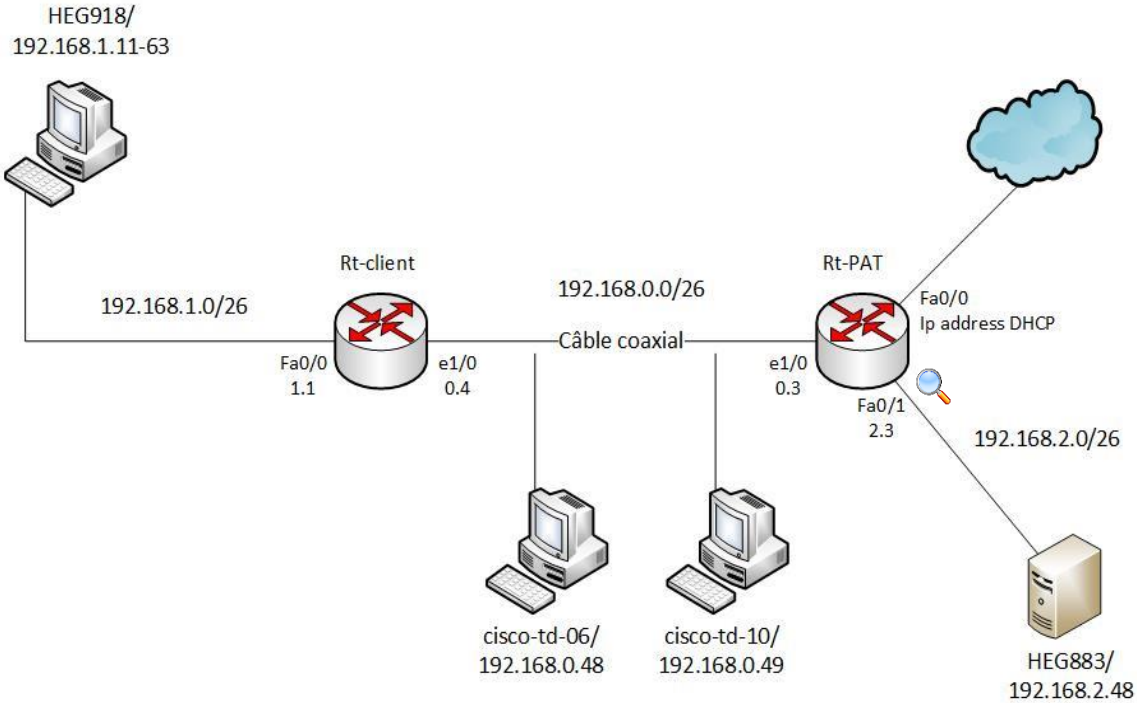
La bande passante entre R1 et le serveur et de 100mégas, celle entre R1 et internet est de 10mégas. Ce schéma ne contient pas de VLAN (à l'exception du VLAN1) puisqu'ils n'ont aucun intérêt dans le cadre de ce travail.

Pour plus d'information, voir la configuration complète des équipements du schéma en annexe (**Annexe 13**).

5.5 Schéma n°3, chaine de deux routeurs avec câble coaxial

Ce schéma comprend un PC client, un serveur, deux routeurs ainsi que deux autres PC situé entre les routeurs qui vont servir à générer du trafic.

Figure 20 Schéma n°3, routeurs avec câble coaxial



5.5.1 Caractéristiques du schéma

Dans ce schéma on veut créer une congestion réseau principalement entre les deux routeurs qui sont reliés par des câbles coaxiaux 10base2 sur l'interface Ethernet1/0 (10méga). Entre eux, nous avons placé deux PCs qui vont servir à générer du trafic. On va alors s'intéresser au domaine de collision situé entre les deux routeurs. Afin de créer de la congestion au niveau du câble coaxial, on va générer du trafic avec Iperf entre les deux PCs cisco-td-06 et cisco-td-10 en plus de générer du trafic entre le PC client et le serveur. La petite loupe représente l'interface qui sera utilisée pour le monitoring.

A l'exception du câble coaxial entre les routeurs, le reste du réseau est interconnecté par une bande passante de 100mégas.

Pour plus d'information, voir la configuration complète des équipements du schéma en annexe (**Annexe 14**).

6. Phase de test

Chaque schéma aura sa propre phase de test spécifique, cependant, certains tests seront généralisés afin de comparer le tout à la fin.

Les phases de test se dérouleront en cinq étapes :

- Générer du trafic avec Iperf pour créer une congestion et observer les performances du réseau (retransmission ou perte de paquets, gigue, etc.)
- Créer plusieurs alertes sur Nfsen afin d'alerter en cas de congestion avérée
- Observer les graphiques Cacti et les statistiques de Nfsen afin de voir les éventuelles traces de la congestion et identifier le coupable
- Consulter les logs afin de voir les éventuels messages apparus suite à la simulation
- Une petite conclusion pour chaque test réalisé

Pour chaque schéma, il y aura entre deux et trois tests : Les simulations d'une longue ou petite période de trafic se feront avec les protocoles TCP ou UDP. Si nécessaire, un test sera subdivisé en deux parties, une fois pour TCP et une fois UDP pour pouvoir les comparer directement.

Chaque test générera du trafic selon la bande passante disponible du schéma concerné. Les simulations de congestion avec Iperf utiliseront entre 75 % et 110 % de la bande passante du cas de figure. Les seuils des alertes seront également définis en fonction de la bande passante. Pour distinguer les longues congestions des petits pics de trafic passager, nous allons mettre en place différents types d'alertes capables de les différencier.

Les tests simulant le protocole TCP seront réalisés avec Iperf version 3 alors que les tests UDP avec Iperf version 2. Il est apparu lors de ma phase de test que la simulation UDP provoquait certaines anomalies sur Nfsen en utilisant Iperf3 (le trafic n'est pas pris en compte). Je n'ai pas réussi à comprendre pourquoi et ce, même après avoir cherché sur le net. C'est pourquoi, j'utiliserai Iperf version 2 pour les tests UDP où tout fonctionne correctement.

Par souci de lecture, toutes les images apparaissant dans les tests ci-dessous ne seront pas associées aux figures de mon travail.

6.1 Schéma n°1, routeurs avec lien série, phases de test

Pour les tests du schéma n°1, le clockrate entre le lien série sera toujours d'un million et fait office d'une bande passante de 1méga. Cette partie comprendra deux tests :

- Un test qui simule une longue congestion utilisant une forte bande passante (90 %)
- Un test qui simule un petit pic de trafic consommant plus que la bande passante disponible (110 %).

Le premier test sera subdivisé en deux parties (TCP et UDP).

Les simulations seront faites entre le PC client et serveur. A la fin de chaque test, il faudra être capable d'identifier et alerter du problème ainsi que trouver le perturbateur du réseau ayant provoqué la congestion, soit le PC client.

6.1.1 Test n°1, schéma n°1, simulation d'une longue congestion

Ce test simulera deux charges de trafic TCP et UDP d'une heure (3600 secondes) chacune. La bande passante sera fixée à 0,9 méga, soit 90 % d'utilisation. Tout sera expliqué en détail pour chaque étape effectuée.

6.1.1.1 Trafic avec une boucle de routage

Avant de voir les tests TCP et UDP, on va en parallèle rajouter un peu de trafic en bruit de fond en créant une boucle de routage comme décrit lors de la présentation du schéma n°1 avec la commande suivante :

```
ping 192.168.1.200 –l1500 –i0.1
```

On va envoyer une requête ping de 1'500 octets (-l1500) chaque 0.1ms (-i0.1) à une adresse qui n'existe pas (192.168.1.200).

Depuis le serveur, on va effectuer ce ping pendant toute la durée du test afin de générer une charge de trafic supplémentaire dans le réseau. Nous verrons plus tard avec Nfsen, à quelle hauteur il a perturbé le trafic.

6.1.1.2 Simulation de la congestion avec du trafic TCP sur Iperf3

Pour simuler du trafic TCP avec Iperf3, voici la ligne de commande à entrer du côté client et ses explications :

```
iperf3 –c 192.168.2.48 –b900000 –t3600 –i60
```

On lance le test depuis notre machine client (-c) en lui spécifiant le serveur auquel on veut envoyer nos données (192.168.2.48). On va générer du trafic TCP à hauteur de

90 % de notre bande passante (-b900000 puisque nous avons un clockrate d'un million) pour une durée d'une heure (-t3600). On va recevoir des données intermédiaires chaque intervalle de 60 secondes (-i60) afin de pouvoir évaluer les résultats tout au long du test.

```
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Retr
[  5]   0.00-3600.03 sec   386 MBytes   900 Kbits/sec  2826             sender
[  5]   0.00-3600.03 sec   386 MBytes   900 Kbits/sec                  receiver
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
```

Voici le résultat obtenu SANS la boucle de routage, on constate que tout s'est bien passé et que nous avons réussi à transmettre à 900Kbits/sec sans problème.

Observons maintenant le même test avec la boucle de routage en plus.

```
Interval           Transfer     Bandwidth       Retr
  0.00-3600.00 sec   338 MBytes   787 Kbits/sec  3480             sender
  0.00-3600.00 sec   338 MBytes   787 Kbits/sec                  receiver
```

Le test d'une heure a transféré 338 MBytes avec une bande passante de 787Kbits/s. Il y a eu un total de 3'480 retransmissions. Les résultats sont nettement moins bons ! La boucle de routage a donc bel et bien réussi à perturber notre transmission et créer de la congestion. Nous avons transmis 338 Mbytes contre 386 pour le test qui n'avait pas de boucle de routage, ce qui représente une diminution du trafic transféré de -13.44%. Observons maintenant les résultats du test UDP avant d'aller plus loin dans l'analyse.

6.1.1.3 Simulation de la congestion avec du trafic UDP sur Iperf2

Pour simuler du trafic UDP avec Iperf2, voici la ligne de commande à entrer côté client :

```
iperf –c 192.168.2.48 –b900000 –t3600 –i60 -u
```

Par rapport au test précédent, on rajoute le paramètre « –u » pour indiquer qu'on veut du trafic UDP. Attention, sur Iperf version 2, il faut également préciser côté serveur qu'on est en UDP (-u).

```
Sent 275526 datagrams
Server Report:
 0.0-3600.0 sec   386 MBytes   900 Kbits/sec   0.067 ms    0/275526 (0%)
```

Voici le résultat obtenu également SANS la boucle de routage, nous constatons que tous s'est bien déroulé pour ce test et nous n'avons rencontré aucun problème.

Voici maintenant le résultat du même test avec la boucle de routage en parallèle :

```
] 0.0-3600.3 sec   352 MBytes   819 Kbits/sec   2.132 ms 24662/275526 (9%)
```

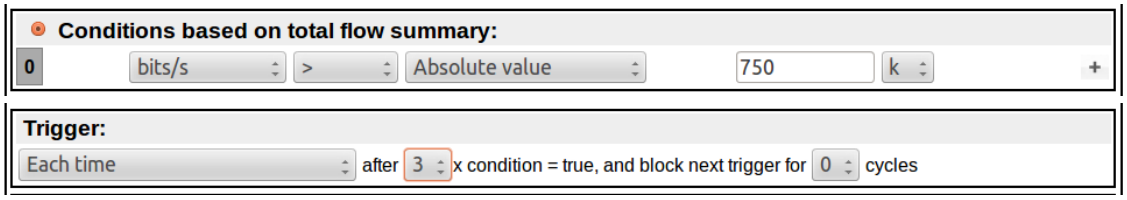
Ce test est un peu moins bon. De plus, les mêmes conclusions que pour le test TCP ressortent ; la boucle de routage à aussi perturbée le trafic de la simulation.

Nous avons transmis 352 Mbytes contre 386 au test qui n'avait pas de boucle de routage, ce qui représente une diminution du trafic transféré de -8.6%. Il y a eu moins de perte que pour TCP probablement due à la différence de ces deux protocoles.

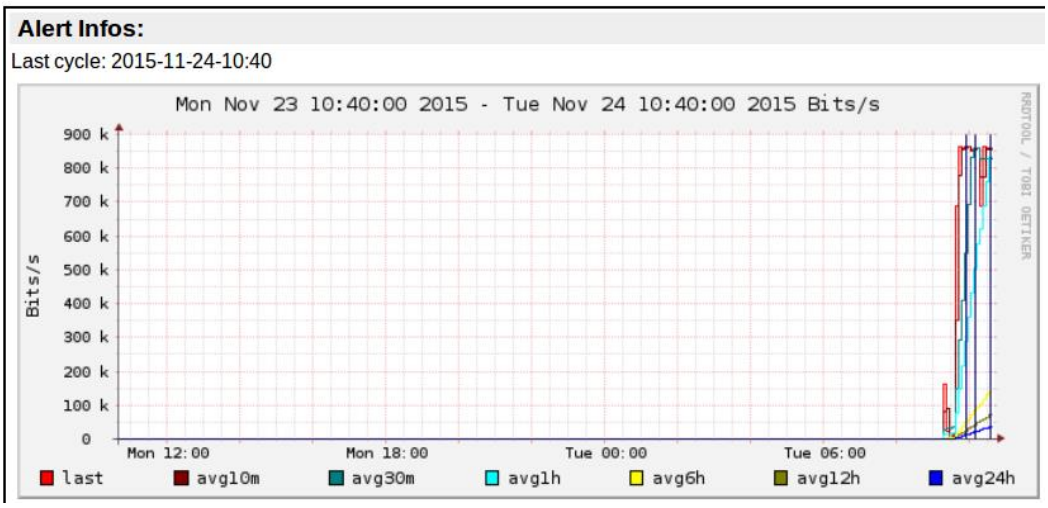
Dans le cadre de ce test, nous savons que c'est la boucle qui a diminué notre débit et créé de la congestion. Nous allons donc tenter de l'identifier avec nos différents outils à disposition.

6.1.1.4 Mise en place et déclenchement des alertes sur Nfsen

On va créer une alerte (la même pour les deux tests) qui va avertir l'administrateur lorsque la bande passante dépassera 75 % d'utilisation. On va alors créer une alerte qui se déclenchera lorsque les bits par seconde dépasseront 750kbits. C'est un seuil raisonnable puisque l'idée est de détecter la congestion avant qu'elle devienne incontrôlable, c'est pourquoi, il ne serait pas judicieux de créer une alerte qui se déclenche à 95 % d'utilisation de la bande passante puisque la marge pour réagir serait plus faible.



Cette alerte a été définie avant de démarrer le test Iperf. Pour éviter qu'elle ne se déclenche sans bonne raison, on va construire une alerte qui s'activera uniquement trois fois après avoir dépassé son seuil fixé, soit un débit supérieur ou égal à 750 kbit/s. Ainsi, si un pic de trafic de quelques secondes survient, on ne sera pas alerté inutilement. Pour que le test soit concluant, on génère du trafic pendant une longue période pour ainsi être sûr que l'alerte sera atteinte en tout cas trois fois et se déclenchera.



Voici les informations relatives à l'alerte. Nous constatons qu'elle s'est déclenchée trois fois (lignes verticales bleues). Un email automatique va alors être envoyé au responsable du réseau indiquant qu'une congestion est en cours.

Alert triggered on lab1, trafic750k - Alert 'lab1_Rt-PAT_bps750bps' triggered at timeslot 201511181105 (YYYY-MM	18 nov.
Alert triggered on lab1, trafic750k - Alert 'lab1_Rt-PAT_bps750bps' triggered at timeslot 201511181050 (YYYY-MM	18 nov.
Alert triggered on lab1, trafic750k - Alert 'lab1_Rt-PAT_bps750bps' triggered at timeslot 201511181035 (YYYY-MM	18 nov.

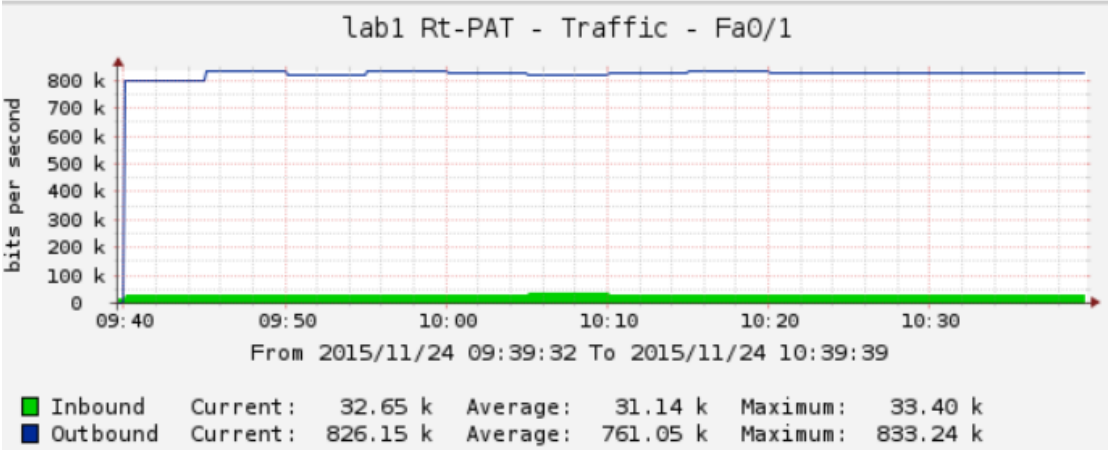
Comme une alerte se produit par défaut chaque cinq minutes et que nous avons spécifié d'attendre trois fois avant d'alerter, nous les avons reçues par tranche de 15 minutes pendant toute la durée du test. Nous avons reçu trois emails puisque l'alerte s'est déclenchée trois fois. Les images ci-dessus sont le résultat du test TCP, pour test UDP les résultats sont les mêmes, je ne montrerai donc pas les images du test UDP qui sont répétitives.

Pourquoi ne pas avoir reçu quatres alertes si on les reçoit par tranche de quinze minutes et que le test a duré une heure ? Les alertes sont vérifiées toutes les cinq minutes précises (00h00, 00h05, 00h15 etc), ce test a été lancé à 9h41, et ne prendra donc pas en compte l'alerte de 09h45 à cause de la fréquence d'échantillonnage de cinq minutes. C'est pourquoi uniquement trois alertes ont été déclenchées et non quatre. Maintenant que nous avons constaté la congestion et alerté du problème, il va falloir chercher le coupable.

6.1.1.5 Test TCP et UDP, déterminer le coupable avec Cacti et Nfsen

Cette partie sera analysée avec le test ayant généré du trafic TCP uniquement. Le test UDP est très similaire et les mêmes conclusions pourront être tirées.

Le responsable a été alerté, maintenant il va falloir chercher plus d'information sur la congestion en cours. On va alors se rendre sur Cacti afin de voir le trafic de nos équipements et ainsi comprendre d'où vient le problème. Cacti va nous être très utile puisqu'en jetant un simple coup d'œil aux graphiques, on va savoir si le problème provient de trafic sortant ou entrant et on peut zoomer sur le graphique pour connaître l'heure et la durée de la congestion. Voici le graphique pour le test TCP :

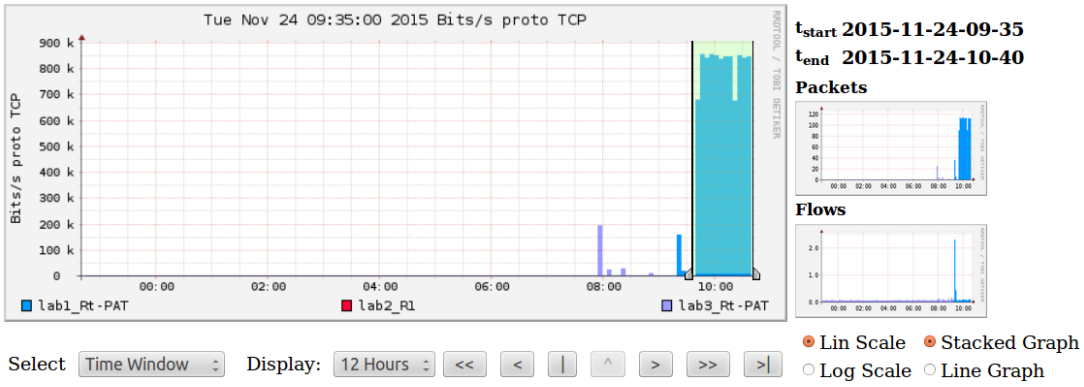


En consultant notre interface de monitoring et en sélectionnant la bonne tranche horaire, on arrive à retrouver pratiquement les mêmes résultats qu'Iperf, ce qui nous confirme que nous sommes au bon endroit. Cacti nous permet de voir en plus, le trafic maximum atteint, ici 833.24k ainsi que la valeur moyenne, 761.05k. Par ailleurs, nous savons que la congestion a été provoquée par du trafic sortant. Si on ne trouve pas exactement la même moyenne de trafic qu'Iperf, c'est simplement à cause de l'échantillonnage de Cacti qui est de cinq minutes.

Nous constatons également un peu de trafic sortant d'environ 30kbit/s qui est probablemnt notre boucle de routage.

Nous en savons maintenant un peu plus, mais, le coupable n'est toujours pas identifié. On sait seulement la durée pendant laquelle il a agi, le type de trafic qui est sortant et nous connaissons son pic maximum et sa moyenne de trafic.

On va maintenant se tourner vers Nfsen pour une analyse plus approfondie.



Statistics timeslot Nov 24 2015 - 09:35 - Nov 24 2015 - 10:40

Channel:	Flows:					Packets:					Traffic:			
	all:	tcp:	udp:	icmp:	other:	all:	tcp:	udp:	icmp:	other:	all:	tcp:	udp:	icmp:
lab3_Rt-PAT	0 /s	0 /s	0 /s	0 /s	0 /s	0 /s	0 /s	0 /s	0 /s	0 /s	0 b/s	0 b/s	0 b/s	0 b/s
lab2_R1	0 /s	0 /s	0 /s	0 /s	0 /s	0 /s	0 /s	0 /s	0 /s	0 /s	0 b/s	0 b/s	0 b/s	0 b/s
lab1_Rt-PAT	0.5 /s	0.1 /s	0.4 /s	0.0 /s	0 /s	113.1 /s	95.2 /s	1.0 /s	17.0 /s	0 /s	726.1 kb/s	715.5 kb/s	658.8 b/s	9.9 kb/s
	all:	tcp:	udp:	icmp:	other:	all:	tcp:	udp:	icmp:	other:	all:	tcp:	udp:	icmp:
TOTAL	0.5 /s	0.1 /s	0.4 /s	0.0 /s	0 /s	113.1 /s	95.2 /s	1.0 /s	17.0 /s	0 /s	726.1 kb/s	715.5 kb/s	658.8 b/s	9.9 kb/s

Il faut sélectionner le laps de temps identifié précédemment dans Cacti. Dans un premier temps, pas besoin d'être très précis puisque Nfsen va nous indiquer l'heure à laquelle la congestion a débutée. Nous voyons à ce stade que le débit pour ce laps de temps est de 715,3kb/s pour TCP, ce qui est assez proche de notre test Iperf, mais tout de même inférieur puisqu'il y a probablement eu d'autres sources de trafic dans ce laps de temps. Nous allons donc devoir analyser plus en détail le tout.

Voyons ce que l'affichage des statistiques nous montre. Dans l'image ci-dessous, les résultats sont directement filtrés par « ip 192.168.1.12 and proto TCP » puisque cette adresse affichait un nombre de bytes très élevé, c'est probablement la source du problème. De plus, dans le tableau nous pouvons voir qu'environ 99 % du trafic provient de TCP, d'où l'idée de filtrer également par ce protocole.

```
Date first seen          Duration Proto          IP Addr    Flows(%)      Packets(%)       Bytes(%)          pps
2015-11-24 09:40:13.540  3600.453 any        192.168.1.12     124(100.0)   398756(100.0)  375.3 M(100.0)      110
2015-11-24 09:40:13.540  3600.453 any        192.168.2.48     124(100.0)   398756(100.0)  375.3 M(100.0)      110

Summary: total flows: 124, total bytes: 375292022, total packets: 398756, avg bps: 833877, avg pps: 110, avg bpp: 9
Time window: 2015-11-24 09:34:35 - 2015-11-24 10:45:00
Total flows processed: 2279, Blocks skipped: 0, Bytes read: 148768
Sys: 0.002s flows/second: 911964.8   Wall: 0.000s flows/second: 3232624.1
```

Nous constatons que l'adresse IP identifiée a provoqué la congestion qui a débuté à 09h40m13s pour une durée de 3600 secondes en générant 124 flux avec notre serveur (192.168.2.48). En jetant un coup d'œil sur le Schéma n°1 dans lequel nous avons effectué ce test, nous pouvons constater que le problème vient d'un client faisant partie de notre pool DHCP ayant comme IP 192.168.1.12. Si la surcharge de

trafic provient de plusieurs endroits, on pourra l'identifier facilement en triant par bytes les statistiques affichées par Nfsen.

Cependant, on constate que nos données ne correspondent pas tout à fait avec le test Iperf TCP, on a généré 338 Mbyte alors qu'on en constate 375.32 ci-dessus, soit une différence de 11 %. Comment est-ce possible ? Il apparait que les statistiques Netflow prennent en compte les en-têtes de paquets TCP, ce qui justifierait cette différence. Ce phénomène à par ailleurs également un impact sur les bits/s transférés qui se trouvent majorés. Pour le trafic UDP, on constate le même problème, mais, en moins important puisqu'un en-tête UDP est plus petit qu'un en-tête TCP. Cette majoration sera présente dans tous les tests qui vont suivre et je n'y ferai plus allusion.

Nous avons identifié le coupable, si nous voulons pousser l'analyse un peu plus loin, il faudra afficher les flux pour avoir plus de détails sur ce surplus de trafic. Etant donné que c'est une simulation de trafic avec Iperf, cette étape pas montrée.

6.1.1.6 Test TCP et UDP, observation du trafic supplémentaire généré

Qu'en est-il du trafic généré par les boucles de routage présentent pendant le test ? Apparemment, elles ont eu une incidence sur notre test Iperf.

Sur Cacti, nous avons déjà répéré la boucle de routage qui représentait le petit trafic sortant. On va donc directement rechercher l'information sur Nfsen.

Date first seen	Duration	Proto	IP Addr	Flows(%)	Packets(%)	Bytes(%)	pps
2015-11-24 09:34:35.934	3887.244	any	192.168.2.48	2134(100.0)	462305(100.0)	374.2 M(100.0)	118
2015-11-24 09:40:13.540	3537.452	any	192.168.1.12	129(6.0)	391787(84.7)	368.7 M(98.5)	110
2015-11-24 09:37:15.630	3717.285	any	192.168.1.200	59(2.8)	40599(8.8)	3.4 M(0.9)	10
2015-11-24 09:34:35.934	3871.977	any	192.168.0.2	367(17.2)	25819(5.6)	1.5 M(0.4)	6
2015-11-24 09:35:13.692	3834.779	any	108.160.172.238	170(8.0)	540(0.1)	247455(0.1)	0
2015-11-24 09:35:17.001	3846.177	any	192.168.1.11	208(9.7)	786(0.2)	54354(0.0)	0
2015-11-24 10:02:54.079	0.384	any	91.189.89.22	4(0.2)	57(0.0)	21180(0.0)	148
2015-11-24 10:30:25.393	58.970	any	212.40.1.44	4(0.2)	70(0.0)	20769(0.0)	1
2015-11-24 09:34:40.738	3867.569	any	8.8.8.8	317(14.9)	317(0.1)	20345(0.0)	0
2015-11-24 09:35:08.992	3842.172	any	192.168.0.4	65(3.0)	260(0.1)	17940(0.0)	0

En triant les statistiques Nfsen par nombre de bytes, nous pouvons constater qu'après la simulation de trafic déjà identifiée vient l'adresse IP 192.168.1.200 ayant généré 3.4MBytes de trafic, puis l'adresse 192.168.0.2 avec un total de 1.5Mbyte. En analysant le schéma, nous nous rendons compte que c'est le trafic qui a été généré par notre boucle de routage. La boucle à donc généré à peine un peu plus de 1 % de trafic supplémentaire par rapport au débit généré pendant les tests, mais à tout de même réussi à causer pas mal de dégats.

6.1.1.7 Consultation du serveur Syslog

La dernière étape consiste à consulter les logs et chercher d'éventuelles informations supplémentaires. Après avoir effectué quelques recherches en filtrant les équipements

de mon réseau, l'anomalie causée par la boucle de routage n'a pas été détectée durant ces tests.

Les logs nous permettent par contre de voir quand un email est envoyé suite au déclenchement d'une alerte :

```
Nov 18 16:20:17 ubuntu nfsen[11117]: Process alert 'lab1_Rt-PAT_bps750bps'
Nov 18 16:20:17 ubuntu nfsen[11117]: alert 'lab1_Rt-PAT_bps750bps': conditions based on total flow summary
Nov 18 16:20:17 ubuntu nfsen[11118]: comm child[3837] terminated with no exit value
Nov 18 16:20:17 ubuntu nfsen[11117]: condition 0: evaluated to True
Nov 18 16:20:17 ubuntu nfsen[11117]: Resulted condition: True
Nov 18 16:20:17 ubuntu nfsen[11117]: Alert 'lab1_Rt-PAT_bps750bps' condition == true, condition counter: 3
Nov 18 16:20:17 ubuntu nfsen[11117]: Alert 'lab1_Rt-PAT_bps750bps' execute action
Nov 18 16:20:17 ubuntu nfsen[11117]: alert 'lab1_Rt-PAT_bps750bps' Send email to: lorenzocortes1991@gmail.com
```

6.1.1.8 Conclusion du test n°1

Aucune grande différence entre TCP et UDP n'a pu être observée pour ces tests. Tout s'est bien passé lorsqu'ils génèrent du trafic seul. Les dégats causés par la boucle de routage ont été pratiquement les mêmes pour chaque protocole.

La congestion n'a eu lieu que lorsque nous avons simulé la boucle de routage. Pourtant, elle n'a générée que 1.3 % de trafic supplémentaire, montant le total de notre bande passante utilisée à 91.3 %. Comment expliquer que nous ayons perdu plus de 10% de trafic par rapport aux tests ne simulant pas de boucle de routage ? Mon hypothèse est que, même si la boucle a généré relativement peu de trafic par rapport à notre simulation, elle a beaucoup perturbé le canal de transmission par son envoi très fréquent de ping. Pour rappel, pendant toute la durée du test, il y a eu chaque 0,1ms, un ping de 1'500 octets qui s'est transformé en boucle de routage perturbant le canal de transmission.

Ce que nous révèle ce premier test est que même avec une utilisation de la bande passante fortement élevée, une congestion ne se produira pas forcément dans la mesure où le trafic présent ne sera pas perturbé par une autre source. Lorsqu'une nouvelle source entre en compte, le trafic peut être rapidement perturbé. La mise en place d'alertes est donc indispensable pour pouvoir prévenir avant que la congestion ne se produise.

6.1.2 Test n°2, schéma n°1, simulation d'un pic de trafic passager UDP

Ce test simulera une petite charge de trafic UDP uniquement d'une durée de cinq minutes. La bande passante sera fixée à 1,1 méga, soit 110 % d'utilisation pour être certain de créer une congestion. Tout sera expliqué au fur et à mesure des différentes étapes.

6.1.2.1 Simulation de la congestion avec du trafic UDP sur Iperf3

Pour simuler du trafic avec UDP, voici la ligne de commande à entrer et ce qu'elle signifie :

```
Iperf –c 192.168.2.48 –b1100000 –t300 –i2 -u
```

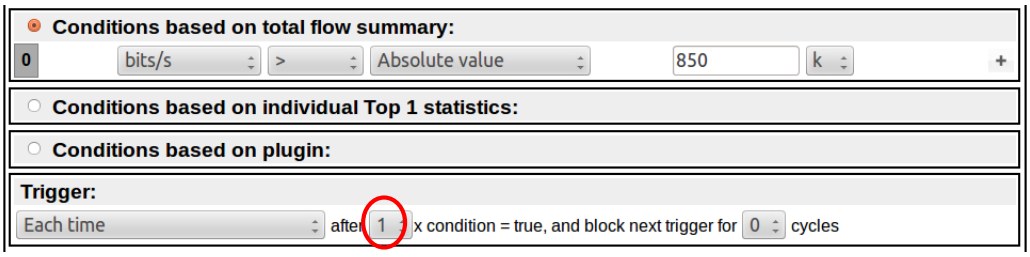
On lance le test depuis notre machine client (-c) en lui spécifiant le serveur auquel on veut envoyer nos données. Nous allons générer du trafic UDP (-u) à hauteur de 110 % de notre bande passante (-b1100000) d'une durée de cinq minutes (-t300). On demande à recevoir par intervalle (-i) de deux secondes des informations intermédiaires au cas où.

```
Server Report:
 0.0-300.7 sec   35.3 MBytes    986 Kbits/sec    1.920 ms 2863/28065 (10%)
```

Nous constatons ici que la bande passante utilisée a été de 986kbits/s alors que nous lui avions spécifié d'utiliser 1100kbits/s. En fait, puisque le débit était trop important, UDP a abandonné certains paquets (10 %) comme on peut le voir sur le rapport ci-dessus. Ces paquets été abandonnés de manière régulière tout au long du test afin de jeter le surplus de débit.

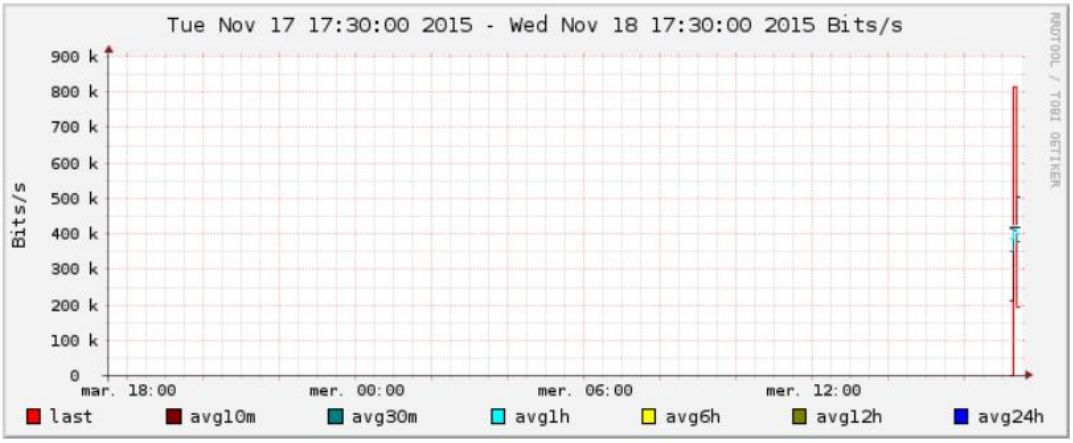
6.1.2.2 Mise en place et déclenchement des alertes sur Nfsen

Nous allons créer une alerte qui cette fois, va avertir l'administrateur lorsque la bande passante dépasse 85 % d'utilisation, soit 850kbit/s pour signaler un pic de trafic. Nous avons augmenté le seuil d'alerte à 850kbit/s (avant 750kbit/s) pour éviter que l'alerte ne se produise trop souvent puisqu'on cherche à détecter un pic et que ceux-ci peuvent se produire assez fréquemment.



Nous avons affaire à une petite congestion pas très importante, il va donc être nécessaire de la différencier d'une longue congestion. Cette alerte a donc été construite de telle sorte à être directement déclenchée après avoir été atteinte. Ainsi, un pic de trafic aussi court soit-il permettra de mettre en garde l'administrateur. Cette alerte sera à considérer avec beaucoup de précaution puisque les pics passagers sont souvent nombreux. On peut par exemple décider d'envoyer cette alerte à une boite

email spéciale qui ne servirait qu'à consulter les petits pics de trafic pour information. Une autre solution est de ne pas alerter, on verrait alors sur Nfsen qu'un pic passager s'est produit, mais, dans les paramétrages de l'alerte, on décide de ne rien faire.



Ce graphique nous montre que l'alerte ne s'est pas déclenchée car il n'y a pas de ligne verticale bleue. Le débit dépassant à peine les 800 kbit/s, alors que le test Iperf nous indique un débit de 986 kbits/s, comment est-ce possible ? Cette explication est due à la fréquence d'échantillonnage qui a déjà été abordée plus haut dans ce travail (rubrique 3.2.1.4 , 3ème paragraphe). Pour Nfsen cette fréquence d'échantillonnage est de cinq minutes, tout comme Cacti. Il faut avoir conscience de ce problème qui est présent dans Nfsen. Un petit pic de trafic sera difficilement détectable s'il dure moins de cinq minutes.

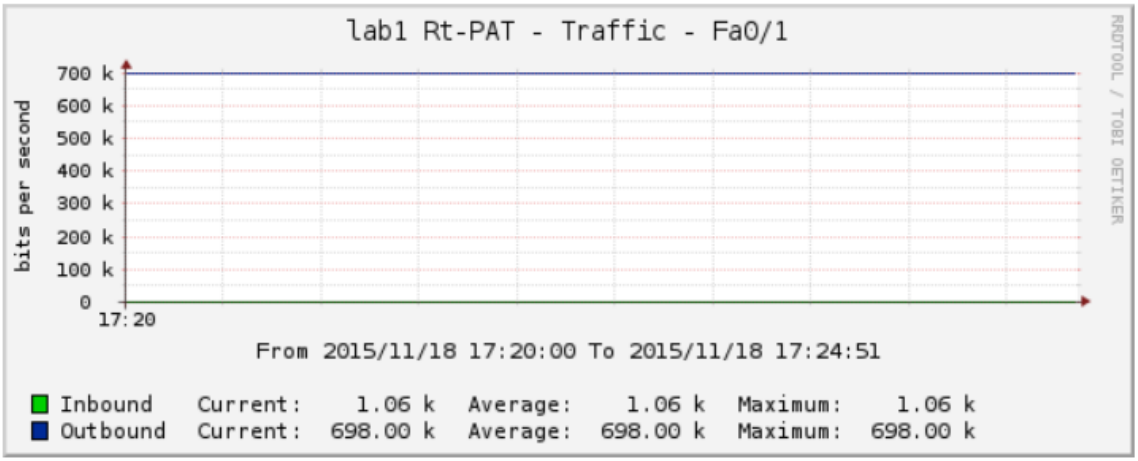
Une première solution est alors de mettre des seuils d'alerte plus bas qu'à l'accoutumé, prenant cependant le risque de déclencher beaucoup d'alertes et de polluer l'email de l'administrateur. Cette solution n'est pas bonne puisque les graphiques Nfsen seront toujours faussés lorsque de petits trafics de moins de cinq minutes surviendront.

La deuxième solution, qui est bien meilleure, est de configurer le taux d'échantillonnage de Netflow sur les équipements Cisco ou directement dans le fichier de configuration de Nfsen pour avoir des données qui tiennent un peu plus la route. Vous trouverez comment configurer le taux d'échantillonnage en annexe (**Annexe 5**) qui parle de la configuration de Netflow sur un équipement Cisco.

A ce stade, on ne sait donc pas qu'une congestion a eu lieu. Allons tout de même regarder les résultats obtenus sur Cacti et Nfsen.

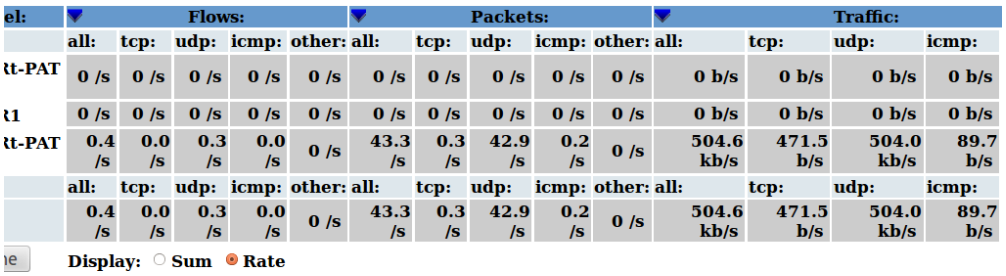
6.1.2.3 Déterminer le coupable avec Cacti et Nfsen

Voyons ce que Cacti va nous montrer sachant que lui aussi, a un taux d'échantillonnage de cinq minutes.



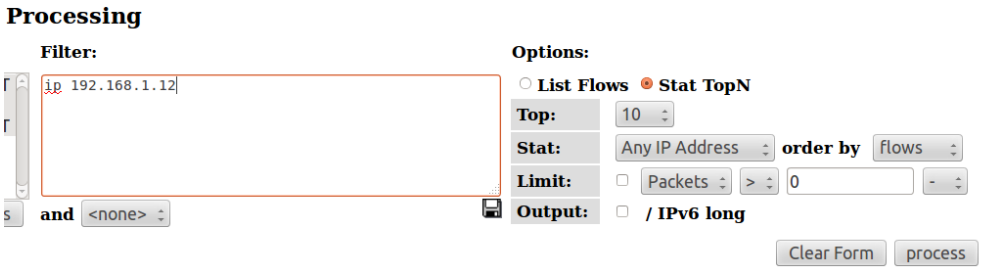
Comme on pouvait s'y attendre, ici aussi le résultat a été échantillonné et les données sont faussées. Cacti ne peut donc pas nous aider pour identifier précisément un petit pic de trafic, mais il arrive tout de même à le détecter.

Observons maintenant les résultats obtenus par Nfsen :



el:	Flows:					Packets:					Traffic:			
	all:	tcp:	udp:	icmp:	other:	all:	tcp:	udp:	icmp:	other:	all:	tcp:	udp:	icmp:
tt-PAT	0 /s	0 /s	0 /s	0 /s	0 /s	0 /s	0 /s	0 /s	0 /s	0 /s	0 b/s	0 b/s	0 b/s	0 b/s
t1	0 /s	0 /s	0 /s	0 /s	0 /s	0 /s	0 /s	0 /s	0 /s	0 /s	0 b/s	0 b/s	0 b/s	0 b/s
tt-PAT	0.4 /s	0.0 /s	0.3 /s	0.0 /s	0 /s	43.3 /s	0.3 /s	42.9 /s	0.2 /s	0 /s	504.6 kb/s	471.5 b/s	504.0 kb/s	89.7 b/s
	all:	tcp:	udp:	icmp:	other:	all:	tcp:	udp:	icmp:	other:	all:	tcp:	udp:	icmp:
	0.4 /s	0.0 /s	0.3 /s	0.0 /s	0 /s	43.3 /s	0.3 /s	42.9 /s	0.2 /s	0 /s	504.6 kb/s	471.5 b/s	504.0 kb/s	89.7 b/s

e Display: Sum Rate




```
-M /home/nfsen/profiles-data/live/lab3_Rt-PAT:lab2_R1:lab1_Rt-PAT  -T  -R 2015/11/18/nfcapd.2015111817
ter:
.1.12
Addr ordered by flows:
 seen          Duration Proto          IP Addr    Flows(%)     Packets(%)       Bytes(%)         pps
 17:20:07.953   300.744 any        192.168.1.12      7(100.0)    25210(100.0)   37.8 M(100.0)
 17:20:07.953   300.744 any        192.168.2.48      7(100.0)    25210(100.0)   37.8 M(100.0)

otal flows: 7, total bytes: 37762736, total packets: 25210, avg bps: 1004515, avg pps: 83, avg bpp: 14
w: 2015-11-18 17:19:26 - 2015-11-18 17:30:00
s processed: 220, Blocks skipped: 0, Bytes read: 14672
s flows/second: 67775.7    Wall: 0.000s flows/second: 1349693.3
```

Malgré le fait que Nfsen ait échantillonné nos données et ne parvienne pas à déclencher notre alerte, on arrive à retrouver l'intégralité du trafic généré et l'heure à laquelle le pic a commencé. On voit ci-dessus un total de 25'210 paquets, si on les compare avec le résultat d'Iperf qui a généré 25'202 paquets – 2'863 abandons on arrive à 25'202 paquets, une précision presque parfaite. Cependant, à cause de la fréquence d'échantillonnage on n'arrivera pratiquement pas à l'observer sur les graphiques de Nfsen.

6.1.2.4 Consultation du serveur Syslog

Aucunes données supplémentaires n'ont été trouvées dans les logs pour ce test.

6.1.2.5 Conclusion du test n°2

Nous avons constaté une congestion grâce à Iperf qui nous a montré que 10 % des paquets ont été abandonnés. Malgré l'alerte fixée et le trafic supérieur généré, celle-ci ne s'est pas déclenchée à cause de la fréquence d'échantillonnage de Nfsen. Il est important avant de se lancer dans n'importe quelle étape de monitoring de congestion réseau, de bien connaître les limites des outils que nous utilisons.

Ce test a été bien évidemment mis en place de telle sorte à ce qu'on voie les effets de l'échantillonnage. Chaque outil de capture de trafic possède sa propre fréquence d'échantillonnage (souvent cinq minutes). On peut sur les équipements Cisco faisant du Netflow ainsi que sur Nfsen, configurer le taux d'échantillonnage. Pour Cacti je n'ai pas fait de recherche, mais il doit probablement exister un moyen de remédier au problème.

6.2 Schéma n°2, suite de switch, phases de test

Pour les tests du schéma n°2, on va particulièrement s'intéresser au Storm control. Un speed de 10 (10mega) a été fixé entre les switch S2 et S3, c'est donc entre eux qu'on va créer de la congestion. Cette partie comprendra trois tests :

- Un test sans Storm control
- Un test avec Storm control qui bloque le port lorsqu'on atteint notre seuil et le débloque seulement après être repassé au-dessous du seuil.
- Un test avec Storm control qui va shutdown le port durant un certain temps lorsqu'on atteint notre seuil.

La simulation sera faite entre le PC client et serveur. Il faudra à la fin de chaque test être capable d'identifier la source du problème.

6.2.1 Test n°3, schéma n°2, simulation d'une longue congestion sans Storm control

Ce test simulera une charge de trafic TCP d'une heure avec une bande passante de 9mégas, soit 90 % d'utilisation.

6.2.1.1 Simulation de la congestion avec du trafic TCP sur Iperf3

Le trafic TCP sera simulé avec la commande suivante :

```
iperf3 –c 192.168.2.48 –b9000000 –t3600 –i60
```

Nous lançons le test depuis notre machine client (-c) en lui spécifiant le serveur avec lequel on va communiquer. La bande passante sera de 9méga (-b9000000 puisque nous avons défini un speed de 10 entre S2 et S3) pour une durée d'une heure (-t3600). Chaque minute, nous allons recevoir des résultats intermediaires (-i60).

```
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Retr
[  4]   0.00-3600.00 sec  3.77 GBytes  9.00 Mbits/sec  1941             sender
[  4]   0.00-3600.00 sec  3.77 GBytes  9.00 Mbits/sec                  receiver
```

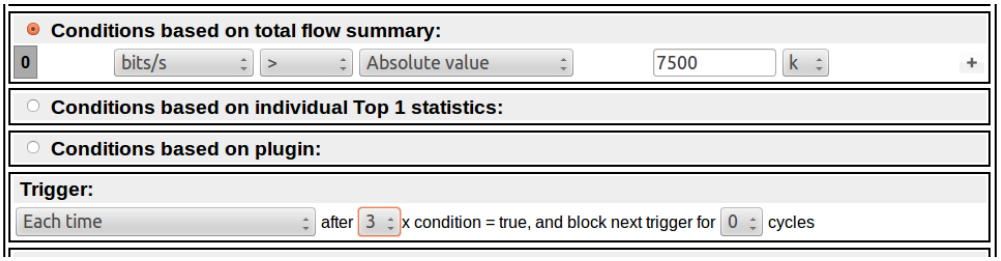
Voici le résultat obtenu, le test d'une heure a transféré 3.77 GBytes avec une bande passante de 9.0Mbits/s. Il y a eu un total de 1'941 retransmissions pour un total de 4'276'487 paquets envoyés (j'ai récupéré le total de paquets avec Nfsen), ce qui représente un taux de retransmission d'environ 0.04 %. Pour l'instant les résultats observés sont plutôt bons. Le taux de retransmission de paquets étant quasi-nul, nous n'observerons pas son détail pour ce test

6.2.1.2 Trafic supplémentaire

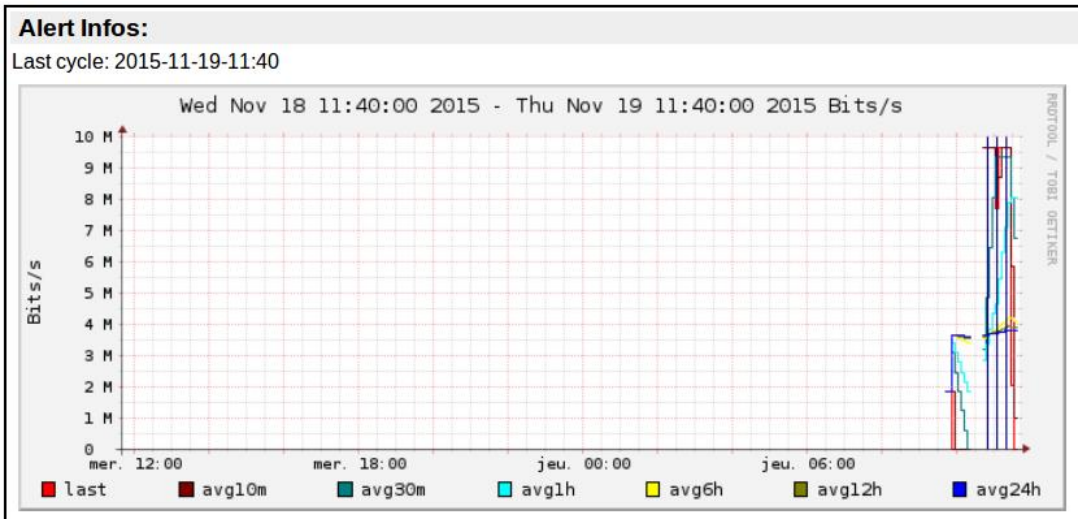
Pour ce test, on n'ajoutera pas de trafic supplémentaire. On veut simplement observer l'état du réseau subissant un fort débit pendant une longue période SANS y avoir intégré le Storm control et par la suite pouvoir comparer les prochains tests à celui-ci.

6.2.1.3 Mise en place et déclenchement des alertes sur Nfsen

Comme pour le test du schéma n°1, notre alerte ne se déclenchera que lorsque la bande passante aura atteint 75 % d'utilisation. Notre seuil sera alors de 7500 kbit/s.



Cette alerte s'activera trois fois après avoir atteint le seuil de 7500kbit/s. On évite ainsi les problèmes de petites pointes de trafic qui ne nous perturberons pas.



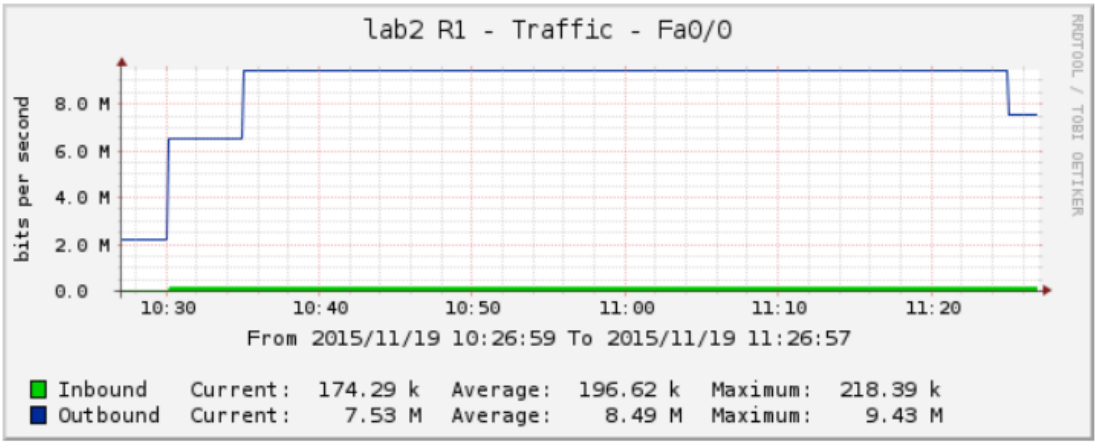
Nous pouvons observer sur le graphique ci-dessus que notre alerte s'est déclenchée plusieurs fois en observant les lignes verticales bleues. Le responsable du réseau va alors être notifié par email :

Alert triggered on lab2, trafic7500k - Alert 'lab1_Rt-PAT_bps7500bps' triggered at timeslot 201511191120 (YYY	**11:25**
Alert triggered on lab2, trafic7500k - Alert 'lab1_Rt-PAT_bps7500bps' triggered at timeslot 201511191105 (YYY	**11:10**
Alert triggered on lab2, trafic7500k - Alert 'lab1_Rt-PAT_bps7500bps' triggered at timeslot 201511191050 (YYY	**10:55**

Nous retrouvons bien nos trois emails qui correspondent aux trois lignes verticales bleues du graphique. Maintenant que nous avons constaté et alerté de la surcharge de trafic, il va falloir trouver le coupable.

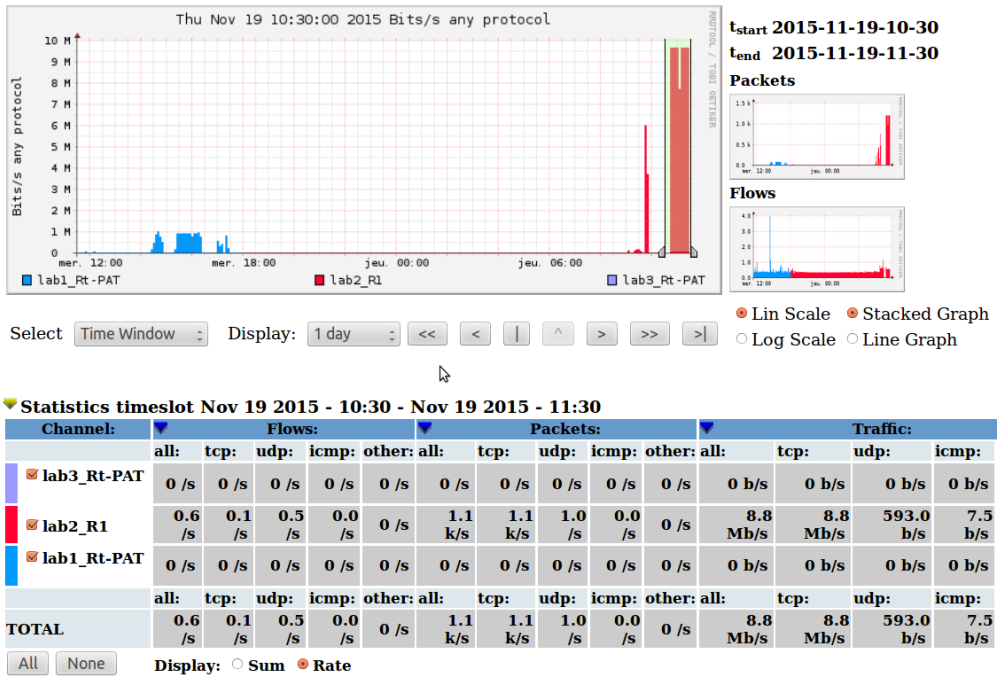
6.2.1.4 Déterminer le coupable avec Cacti et Nfsen

Le responsable ayant été alerté, il va maintenant falloir chercher plus d'information sur la congestion en cours. On va alors se rendre sur Cacti et ainsi comprendre d'où vient le problème.



En se plaçant sur notre interface de monitoring de notre routeur, on s'aperçoit rapidement que le trafic généré est très similaire avec le test Iperf. Cependant, à cause de l'échantillonnage de Cacti, on ne trouve pas la même moyenne que sur Iperf.

Maintenant que nous avons réussi à obtenir plus d'informations, nous allons nous tourner vers Nfsen afin de poursuivre l'analyse.



Nous arrivons grâce au graphique, à identifier visuellement le surplus de trafic. En le comparant avec les résultats de Cacti, on constate que nous sommes bien au bon

endroit. Le débit de 8.8mb/s n'est pour l'instant pas précis car, il prend en compte tous les échanges de données ayant eu lieu durant ce laps de temps et pas seulement notre test Iperf.

Nous allons donc afficher les statistiques nfsen en filtrant nos données par « ip 192.168.1.12 and proto TCP » comme cela a été fait pour les précédents tests.

```
Top 10 IP Addr ordered by flows:
Date first seen          Duration Proto          IP Addr    Flows(%)     Packets(%)       Bytes(%)
2015-11-19 10:26:06.804  3594.261 any        192.168.1.12     124(100.0)    4.3 M(100.0)    4.3 G(100.0)
2015-11-19 10:26:06.804  3594.261 any        192.168.2.48     124(100.0)    4.3 M(100.0)    4.3 G(100.0)

Summary: total flows: 124, total bytes: 4273328410, total packets: 4276487, avg bps: 9511448, avg pps: 1189,
Time window: 2015-11-19 10:25:24 - 2015-11-19 11:35:00
Total flows processed: 2368, Blocks skipped: 0, Bytes read: 155400
Sys: 0.003s flows/second: 696265.8   Wall: 0.000s flows/second: 2587978.1
```

En observant le Schéma n°2 dans lequel ce test s'est déroulé, nous arrivons facilement à identifier la source de notre problème.

6.2.1.5 Consultation du serveur Syslog

Après avoir effectué quelques recherches en filtrant les équipements de mon réseau, aucune anomalie n'a été détectée durant ces deux tests. On peut cependant voir que des emails ont bien été envoyés suite à une alerte qui s'est déclenchée comme cela a été montré au test n°1.

6.2.1.6 Conclusion du test n°3

La congestion n'a pas eu lieu non plus pour ce test. Nous avons simulé du trafic utilisant 90 % de notre bande passante, mais comme le test Iperf était pratiquement le seul trafic présent, il n'y a pas eu de congestion et aucun ralentissement du réseau n'a été signalé et cela, même après avoir atteint nos alertes plusieurs fois. En résumé, on a obtenu les mêmes résultats que pour le test n°1.

Ce qui va être intéressant maintenant, c'est d'activer Storm control sur les switch et voir quelles seront les incidences par rapport à ce test. Même si une congestion n'a pas été détectée, il est risqué d'avoir une consommation de bande passante de 90 %, c'est là que va entrer en compte le Storm control.

6.2.2 Test n°4, schéma n°2, simulation d'une longue congestion avec Storm control qui va bloquer le port

Ce test va simuler une charge de trafic UDP de trente minutes et simulera un débit à hauteur de 70% d'utilisation de la bande passante, soit 7 mégas. Un seuil Storm control a été défini et s'activera à 80 % d'utilisation de bande passante. Afin de déclencher Storm control, chaque dix minute durant le test, nous générerons un flux supplémentaire 1mbit/s durant vingt secondes.

Deux seuils Storm control ont été définis ; le premier se déclenchera à 80 % d'utilisation de bande passante afin de bloquer le trafic et ne le réactivera que lorsque l'utilisation de bande passante sera passée sous le seuil des 75 % d'utilisation.

6.2.2.1 Simulation de la congestion avec du trafic UDP sur Iperf2

Cette fois, afin de simuler notre trafic, nous allons utiliser Iperf2 qui est très similaire à la version 3 :

```
iperf –c 192.168.2.48 –b7m –t1800 –i60 -u
```

On va créer du trafic UDP (-u) depuis notre PC client (-c) en lui spécifiant le serveur auquel on veut envoyer nos données. Le test durera 30 minutes (-t1800) et utilisera 70 % de la bande bande passante (-b7m). En cas de besoin, on spécifie de recevoir les résultats intermédiaire du test en cours par intervalle de 60 secondes (-i60).

```
[  3] Sent 1071430 datagrams
[  3] Server Report:
[  3]  0.0-1800.0 sec  1.44 GBytes  6.88 Mbits/sec   0.035 ms 17951/1071429 (1.7
%)
```

Ce test de 30 minutes a transféré pour 1.44 GBytes de données avec une bande passante de 6.88Mbits/s et une gigue faible. Il y a eu 17'951 paquets perdus pour un total de 1'071'429 paquets envoyés. Ce test a taux de perte de 1.7% alors qu'il a généré moins de trafic que le test n°3, comment l'expliquer ? C'est probablement le Storm control qui s'est déclenché. Nous verrons cela plus en détail lors de l'analyse du trafic supplémentaire ci-dessous.

6.2.2.2 Trafic supplémentaire avec Iperf version 2

Chaque 10 minutes, du trafic supplémentaire de 1mbit/s sera généré durant vingt secondes. Voici la ligne de commande à entrer :

```
Iperf3 –c 192.168.2.48 –t20 –b100000
```

On génère du trafic côté client (-c) durant 20 secondes (-t20) avec une bande passante d'un méga (-b1m).

Voici les résultats causés par ce surplus de trafic :

```
 4]   0.00-1.00   sec   130 KBytes  1.07 Mbits/sec    0
 4]   1.00-2.00   sec   124 KBytes  1.02 Mbits/sec    1
 4]   2.00-3.00   sec  0.00 Bytes  0.00 bits/sec    1
 4]   3.00-4.00   sec   226 KBytes  1.85 Mbits/sec   11
 4]   4.00-5.00   sec   133 KBytes  1.09 Mbits/sec    0
 4]   5.00-6.00   sec  0.00 Bytes  0.00 bits/sec    2
 4]   6.00-7.00   sec  93.3 KBytes   765 Kbits/sec   20
 4]   7.00-8.00   sec   209 KBytes  1.71 Mbits/sec    1
 4]   8.00-9.00   sec  0.00 Bytes  0.00 bits/sec    1
 4]   9.00-10.00  sec   144 KBytes  1.18 Mbits/sec   22
 4]  10.00-11.00  sec   209 KBytes  1.71 Mbits/sec    1
 4]  11.00-12.00  sec  0.00 Bytes  0.00 bits/sec    1
 4]  12.00-13.00  sec   139 KBytes  1.14 Mbits/sec   26
 4]  13.00-14.00  sec   204 KBytes  1.67 Mbits/sec    1
 4]  14.00-15.00  sec  0.00 Bytes  0.00 bits/sec    1
 4]  15.00-16.00  sec   110 KBytes   904 Kbits/sec   28
 4]  16.00-17.00  sec   260 KBytes  2.13 Mbits/sec    1
 4]  17.00-18.00  sec  0.00 Bytes  0.00 bits/sec    1
 4]  18.00-19.00  sec  42.4 KBytes   348 Kbits/sec   30
 4]  19.00-20.00  sec   277 KBytes  2.27 Mbits/sec    1
- - - - - - - - - - - - - - - - - - - - - - - - -
ID] Interval           Transfer     Bandwidth       Retr
 4]   0.00-20.00  sec  2.25 MBytes   943 Kbits/sec  150
 4]   0.00-20.00  sec  2.25 MBytes   943 Kbits/sec
```

Comme on peut le constater, il y a eu du trafic supprimé à plusieurs reprises. Ceci est provoqué par le déclenchement du Storm control qui bloque le trafic lorsque son seuil est atteint.

Rappelez-vous du test précédent (test n°3), où l'on avait généré pour 9mbit/s de trafic durant une heure et aucun incident n'avait été signalé. En mettant en place le Storm control, on arrive à limiter des flux de trafic trop important.

Observons maintenant ce qu'il s'est passé au même moment pour notre principale simulation de trafic :

```
 3] 582.0-584.0 sec  1.67 MBytes  7.00 Mbits/sec   0.034 ms    0/ 1190 (0%)
 3] 584.0-586.0 sec  1.67 MBytes  7.00 Mbits/sec   0.048 ms    0/ 1191 (0%)
 3] 586.0-588.0 sec  1.67 MBytes  7.00 Mbits/sec   0.030 ms    0/ 1190 (0%)
 3] 588.0-590.0 sec   848 KBytes  3.48 Mbits/sec   0.036 ms  600/ 1191 (50%)
 3] 590.0-592.0 sec  1.27 MBytes  5.33 Mbits/sec   0.452 ms    0/  906 (0%)
 3] 592.0-594.0 sec   347 KBytes  1.42 Mbits/sec   0.937 ms 1205/ 1447 (83%)
 3] 594.0-596.0 sec   446 KBytes  1.83 Mbits/sec   0.841 ms    0/  311 (0%)
 3] 596.0-598.0 sec   777 KBytes  3.18 Mbits/sec   0.792 ms 1245/ 1786 (70%)
 3] 598.0-600.0 sec   297 KBytes  1.22 Mbits/sec   0.929 ms 1257/ 1464 (86%)
 3] 600.0-602.0 sec   471 KBytes  1.93 Mbits/sec   0.890 ms    0/  328 (0%)
 3] 602.0-604.0 sec   768 KBytes  3.15 Mbits/sec   0.813 ms 1262/ 1797 (70%)
 3] 604.0-606.0 sec   280 KBytes  1.15 Mbits/sec   1.325 ms 1263/ 1458 (87%)
 3] 606.0-608.0 sec   484 KBytes  1.98 Mbits/sec   0.826 ms    0/  337 (0%)
 3] 608.0-610.0 sec   863 KBytes  3.53 Mbits/sec   0.072 ms 1265/ 1866 (68%)
 3] 610.0-612.0 sec  1.12 MBytes  4.69 Mbits/sec   0.036 ms  598/ 1395 (43%)
 3] 612.0-614.0 sec  1.67 MBytes  7.00 Mbits/sec   0.042 ms    0/ 1191 (0%)
 3] 614.0-616.0 sec  1.67 MBytes  7.00 Mbits/sec   0.049 ms    0/ 1190 (0%)
 3] 616.0-618.0 sec  1.67 MBytes  7.00 Mbits/sec   0.032 ms    0/ 1190 (0%)
```

On remarque que lorsque le flux de vingt secondes est apparu, cela a provoqué une diminution du débit très net provoquée par le déclenchement du Storm control qui bloque le trafic dès qu'il dépasse 8mbit/s. On voit que de nombreux paquets ont été

abandonnés durant ce laps de temps. Le trafic sera alors réactivé uniquement lorsque le débit total sera inférieur à 7.5mbit/s. A cela s'ajoute le fait de devoir partager la bande passante avec un perturbateur. Lorsque les vingt secondes de trafic supplémentaire sont passées, le trafic redevient normal et il n'y a plus de congestion et tout revient à la normal.

La suite logique de cette étape après avoir identifié ce flux de trafic incohérent est d'aller consulter notre serveur Syslog afin de voir si on peut y trouver des informations concernant l'anomalie qu'on vient de détecter.

6.2.2.3 Consultation du serveur Syslog

Après avoir filtré les équipements de mon réseau, nous avons la confirmation que c'est bien Storm control qui a bloqué le surplus de trafic. Voici ce qui a été répertorié :

```
Nov 19 15:35:25 192.168.1.8 612: Nov 19 15:35:17: %STORM_CONTROL-3-FILTERED: A Unicast storm detected on Fa0/3. A
packet filter action has been applied on the interface.
Nov 19 15:35:30 192.168.1.8 613: Nov 19 15:35:20: %STORM_CONTROL-3-FILTERED: A Unicast storm detected on Fa0/3. A
packet filter action has been applied on the interface.
Nov 19 15:35:35 192.168.1.8 614: Nov 19 15:35:23: %STORM_CONTROL-3-FILTERED: A Unicast storm detected on Fa0/3. A
packet filter action has been applied on the interface.
Nov 19 15:35:40 192.168.1.8 615: Nov 19 15:35:26: %STORM_CONTROL-3-FILTERED: A Unicast storm detected on Fa0/3. A
packet filter action has been applied on the interface.
Nov 19 15:35:45 192.168.1.8 616: Nov 19 15:35:29: %STORM_CONTROL-3-FILTERED: A Unicast storm detected on Fa0/3. A
packet filter action has been applied on the interface.
Nov 19 15:35:50 192.168.1.8 617: Nov 19 15:35:32: %STORM_CONTROL-3-FILTERED: A Unicast storm detected on Fa0/3. A
packet filter action has been applied on the interface.
Nov 19 15:35:55 192.168.1.8 618: Nov 19 15:35:35: %STORM_CONTROL-3-FILTERED: A Unicast storm detected on Fa0/3. A
packet filter action has been applied on the interface.
Nov 19 15:45:21 192.168.1.8 620: Nov 19 15:45:15: %STORM_CONTROL-3-FILTERED: A Unicast storm detected on Fa0/3. A
packet filter action has been applied on the interface.
Nov 19 15:45:26 192.168.1.8 621: Nov 19 15:45:18: %STORM_CONTROL-3-FILTERED: A Unicast storm detected on Fa0/3. A
packet filter action has been applied on the interface.
Nov 19 15:45:31 192.168.1.8 622: Nov 19 15:45:21: %STORM_CONTROL-3-FILTERED: A Unicast storm detected on Fa0/3. A
packet filter action has been applied on the interface.
Nov 19 15:45:36 192.168.1.8 623: Nov 19 15:45:26: %STORM_CONTROL-3-FILTERED: A Unicast storm detected on Fa0/3. A
packet filter action has been applied on the interface.
Nov 19 15:45:41 192.168.1.8 624: Nov 19 15:45:29: %STORM_CONTROL-3-FILTERED: A Unicast storm detected on Fa0/3. A
packet filter action has been applied on the interface.
Nov 19 15:45:46 192.168.1.8 625: Nov 19 15:45:32: %STORM_CONTROL-3-FILTERED: A Unicast storm detected on Fa0/3. A
packet filter action has been applied on the interface.
Nov 19 15:45:51 192.168.1.8 626: Nov 19 15:45:35: %STORM_CONTROL-3-FILTERED: A Unicast storm detected on Fa0/3. A
packet filter action has been applied on the interface.
```

Le switch ayant comme IP 192.168.1.8 a activé Storm control suite à du trafic unicast trop important et ce, quatorze fois durant tout le test. La bande passante a donc dépassée 80 % d'utilisation quatorze fois.

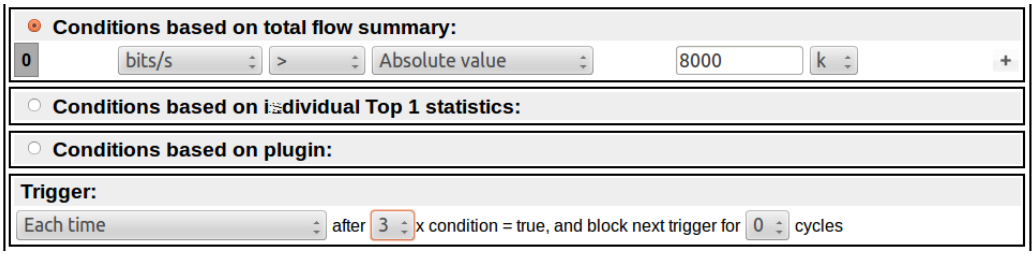
Logiquement, l'alerte n'aurait dû se déclencher que deux fois, (une fois par pic de trafic de vingt secondes). Ce problème peut être résolu par la commande suivante dans le switch : « snmp-server enable traps storm-control trap-rate *value* » où « value » correspond au nombre de fois qu'on autorise une trappe Storm control par minute.

On peut se demander pourquoi Storm control n'a été détecté que sur un switch alors qu'il a été également configuré sur son switch voisin ? Cela s'explique simplement par le fait que Storm control ne s'applique que sur le trafic entrant, c'est pourquoi il faudra

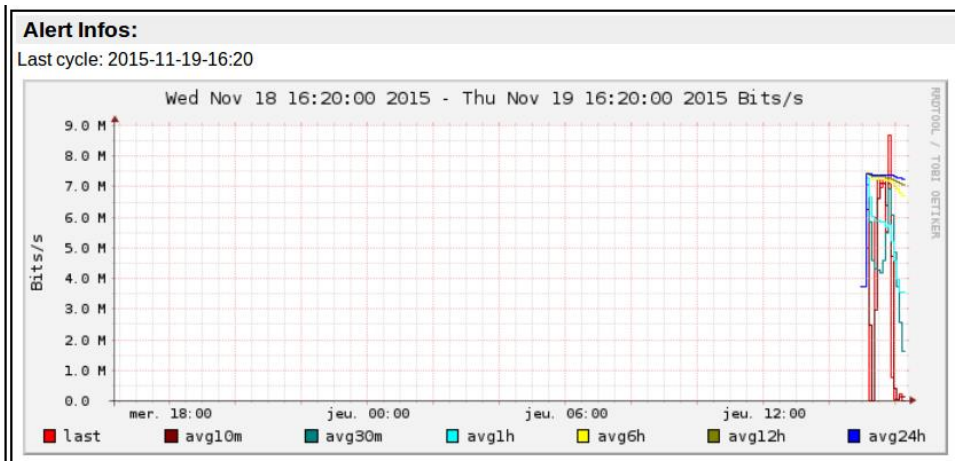
veiller à bien le configurer sur les deux extrémités d'un équipement interconnecté pour être sûr de le déclencher.

6.2.2.4 Mise en place et déclenchement des alertes sur Nfsen

Pour ce test, l'alerte que nous allons créer va se déclencher au même moment que Storm control, soit lorsque le débit dépassera 80 % d'utilisation de la bande passante. Nous aurons ainsi deux moyens de détection ; Nfsen et Storm control. C'est une idée qui semble intéressante, ainsi, si un des deux outils n'est plus disponible, on utilisera l'autre.



Voyons ce que les informations de l'alerte nous montrent.

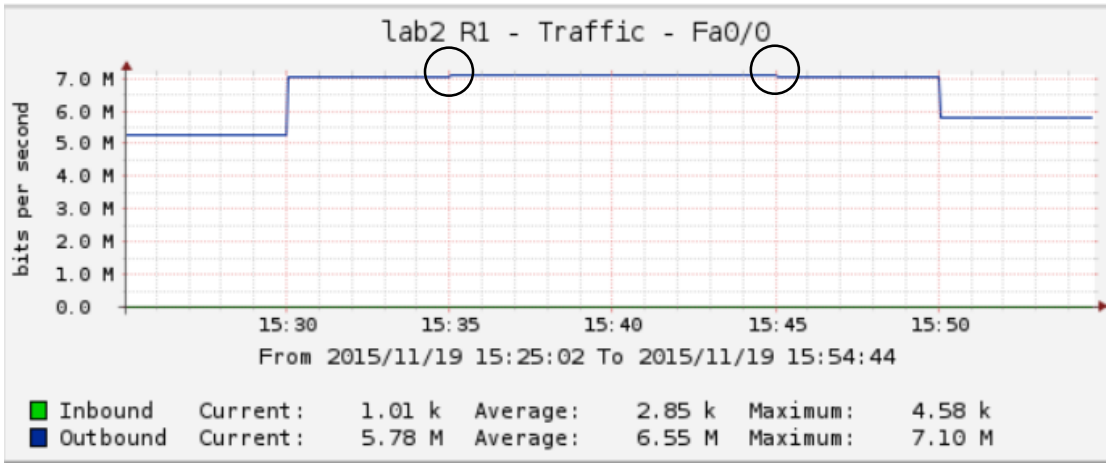


Nous constatons que notre alerte ne s'est pas déclenchée car il n'y a pas de ligne verticale bleue. Comment peut-on expliquer cela sachant que deux fois durant le test, le seuil de 8méga a été atteint ? Cela est encore une fois dû à la fréquence d'échantillonnage de cinq minutes. Nfsen n'a pas réussi à identifier les petites pointes de trafic de quelques secondes à 8mbit/s. On peut alors remercier Storm control d'être intervenu à sa place.

6.2.2.5 Déterminer le coupable avec Cacti et Nfsen

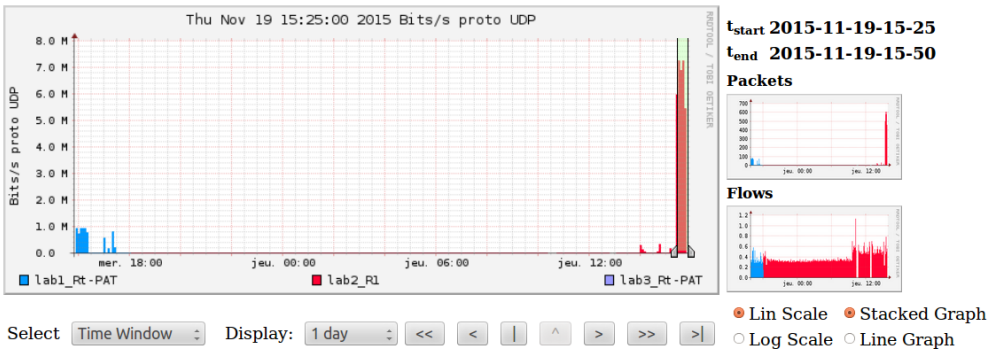
Le responsable du réseau a pu prendre connaissance du problème grâce au serveur Syslog.

Essayons d'en savoir plus en consultant Cacti.



On arrive comme d'habitude à identifier la zone de trafic mais impossible de détecter qu'il y a eu plusieurs pics de trafic dépassant 8mbit/s. Cependant, on arrive à distinguer deux minuscules pixels (sortez votre loupe !) un peu plus élevé qui ont été ensevelis, encore une fois, par la fréquence d'échantillonnage. Ces deux pixels devraient bien concorder avec le surplus de trafic généré car ceux-ci ont été déclenchés par intervalle de dix minutes, ce qui semble correspondre.

Voyons ce que Nfsen nous montre.



Statistics timeslot Nov 19 2015 - 15:25 - Nov 19 2015 - 15:50

Channel:	Flows:					Packets:					Traffic:			
	all:	tcp:	udp:	icmp:	other:	all:	tcp:	udp:	icmp:	other:	all:	tcp:	udp:	icmp:
lab3_Rt-PAT	0 /s	0 /s	0 /s	0 /s	0 /s	0 /s	0 /s	0 /s	0 /s	0 /s	0 b/s	0 b/s	0 b/s	0 b/s
lab2_R1	0.7 /s	0.1 /s	0.6 /s	0.0 /s	0 /s	583.2 /s	4.8 /s	578.3 /s	0.0 /s	0 /s	6.9 Mb/s	30.9 kb/s	6.9 Mb/s	57.3 b/s
lab1_Rt-PAT	0 /s	0 /s	0 /s	0 /s	0 /s	0 /s	0 /s	0 /s	0 /s	0 /s	0 b/s	0 b/s	0 b/s	0 b/s
	all:	tcp:	udp:	icmp:	other:	all:	tcp:	udp:	icmp:	other:	all:	tcp:	udp:	icmp:
TOTAL	0.7 /s	0.1 /s	0.6 /s	0.0 /s	0 /s	583.2 /s	4.8 /s	578.3 /s	0.0 /s	0 /s	6.9 Mb/s	30.9 kb/s	6.9 Mb/s	57.3 b/s

Grace au graphique, nous arrivons facilement à trouver la charge de trafic et constatons que le débit identifié correspond fortement à notre test Iperf.

En consultant les statistiques préalablement filtrées par « ip 192.168.1.12 and proto UDP » nous arrivons à identifier l'adresse ayant généré le plus de trafic.

```
ip 192.168.1.12 and proto UDP
Top 10 IP Addr ordered by flows:
Date first seen          Duration Proto          IP Addr    Flows(%)     Packets(%)       Bytes(%)         pps
2015-11-19 15:20:19.978  1788.021 any        192.168.1.12      32(100.0)    1.0 M(100.0)    1.6 G(100.0)      581
2015-11-19 15:20:19.978  1788.021 any        192.168.2.48      32(100.0)    1.0 M(100.0)    1.6 G(100.0)      581

Summary: total flows: 32, total bytes: 1556562855, total packets: 1039095, avg bps: 6964405, avg pps: 581, avg bpp:
Time window: 2015-11-19 15:20:03 - 2015-11-19 15:55:00
Total flows processed: 1257, Blocks skipped: 0, Bytes read: 82224
Sys: 0.002s flows/second: 452646.7   Wall: 0.000s flows/second: 3142500.0
```

En se penchant sur le Schéma n°2 on identifie facilement le premier coupable, mais à lui seul il n'a pas pu déclencher les alertes de Storm control puisqu'il a un débit inférieur à 8méga. Il va alors falloir trouver le deuxième responsable.

Cacti a réussi à nous indiquer (avec peine) où ont été détectés les pointes de trafic. Essayons de les retrouver avec Nfsen.

```
Top 10 IP Addr ordered by flows:
Date first seen          Duration Proto          IP Addr    Flows(%)     Packets(%)       Bytes(%)         pps
2015-11-19 15:30:27.430    22.369 any        192.168.1.12       6(100.0)     2608(100.0)    2.5 M(100.0)      116
2015-11-19 15:30:27.430    22.369 any        192.168.2.48       6(100.0)     2608(100.0)    2.5 M(100.0)      116

Summary: total flows: 6, total bytes: 2495538, total packets: 2608, avg bps: 892498, avg pps: 116, avg bpp: 956
Time window: 2015-11-19 15:29:49 - 2015-11-19 15:40:00
Total flows processed: 313, Blocks skipped: 0, Bytes read: 20328
Sys: 0.000s flows/second: 325026.0   Wall: 0.000s flows/second: 1920245.4


Top 10 IP Addr ordered by flows:
Date first seen          Duration Proto          IP Addr    Flows(%)     Packets(%)       Bytes(%)         pps
2015-11-19 15:40:27.278    21.199 any        192.168.1.12       6(100.0)     2289(100.0)    2.2 M(100.0)      107
2015-11-19 15:40:27.278    21.199 any        192.168.2.48       6(100.0)     2289(100.0)    2.2 M(100.0)      107

Summary: total flows: 6, total bytes: 2193067, total packets: 2289, avg bps: 827611, avg pps: 107, avg bpp: 958
Time window: 2015-11-19 15:39:43 - 2015-11-19 15:50:00
Total flows processed: 192, Blocks skipped: 0, Bytes read: 12584
Sys: 0.000s flows/second: 236744.8   Wall: 0.000s flows/second: 1729729.7
```

Nfsen a réussi à identifier nos deux petits pics de trafic de 20 secondes. Sur Nfsen, ils se sont produits respectivement à 15h30 et 15h40 alors que sur cacti on a pu observer un décalage de cinq minutes. A mon avis, c'est encore un problème dû à la fréquence d'échantillonage qui fait que les données ont été indiquées cinq minutes trop tôt ou trop tard.

6.2.2.6 Conclusion du test n°4

Deux congestions se sont produites à la suite du déclenchement de Storm control lorsque le trafic a dépassé 8mbit/s. La consultation du serveur Syslog à cette fois été beaucoup plus utile que dans les tests précédent. Les outils cacti et Nfsen ont été un peu moins performants que d'habitude. A cause de sa fréquence d'échantillonnage, Cacti a eu beaucoup de peine à identifier les deux pointes de trafic et Nfsen n'a pas été capable de déclencher son alerte ayant comme seuil 8mégas.De plus, si nous comparons ce test au précédent (test n°3), nous avons constaté de la congestion alors

qu'il y en a pas eu pour le test n°3 qui a généré pourtant plus de trafic. Ceci est dû à la mise en place du Storm control.

Ce test a été conçu pour montrer les problèmes qu'on peut rencontrer en implémentant deux outils qui font le même travail (alerter avec Nfsen et alerter avec Storm control). Encore une fois, la fréquence d'échantillonnage a posé problème et n'a pas permis à Nfsen d'agir comme il aurait dû alors que pour Storm control, tout s'est déroulé correctement.

Ce qu'il faut retenir de ce test et que Storm control est un outil très intéressant, il permet non seulement d'alerter en cas de dépassement d'un seuil, mais en plus, il implémente directement une mesure corrective qui est de bloquer temporairement le trafic. Cependant, il faut garder en tête que la congestion a été créée par Storm control en bloquant son port, le trafic supprimé aurait pu être important. Si cette méthode n'avait pas été implémentée durant ce test, tout ce serait bien passé et il n'y aurait pas eu de congestion.

6.2.3 Test n°5, schéma n°2, simulation d'une longue congestion avec Storm control qui va mettre en shutdown le port

Ce test simulera une charge de trafic TCP de trente minutes et générera du trafic à hauteur de 7Mbits/s, soit 70 % d'utilisation de la bande passante. Comme pour le test précédent, Storm control a été activé sur les switch et se déclenchera lorsque l'utilisation de la bande passante sera supérieure à 80%. Cependant, cette fois nous allons utiliser l'option « shutdown » du Storm control. A la fin du test, nous générerons un flux supplémentaire de 1Mbits/s durant dix secondes. Ce qui aura pour conséquence de bloquer le port. Une rubrique contenant la nouvelle configuration du switch vous sera expliquée en détail.

6.2.3.1 Nouvelle configuration du switch

Le seuil de Storm control se déclenchera à 80 % d'utilisation de bande passante et va shutdown le port et ainsi empêcher tout trafic de passer. Celui-ci sera réactivé 300 secondes après avoir été fermé, soit cinq minutes.

Les nouvelles commandes à entrer dans le switch pour ce test sont les suivantes :

```
S2(config-if)# storm-control broadcast level 80 75
S2(config-if)# storm-control multicast level 80 75
S2(config-if)# storm-control unicast level 80 75
S2(config-if)# storm-control action trap
S2(config-if)# storm-control action shutdown
S2(config)#errdisable recovery cause all
S2(config)#errdisable recovery interval 300
```

Le deuxième seuil de 75 % qui réactivait l'interface lorsque le trafic était bloqué n'a pas d'effet pour la commande « shutdown » puisque le port sera désactivé (mettre le deuxième seuil ou non ne changera rien). Cependant, il faut laisser le premier seuil de 80 pour pouvoir dire à quel moment on va arrêter le port.

On doit évidemment rajouter l'action « shutdown » et laisser l'action « trap » pour continuer à envoyer des trappes.

On configure finalement l'option « errdisable » qui va se charger de remettre sur pied l'interface chaque 300 secondes après qu'elle ait été mise en shutdown. Sans cette option, on aurait dû rentrer nous-mêmes dans la configuration du switch pour y faire un « no shutdown » manuellement, ce qui aurait été fastidieux.

6.2.3.2 Simulation de la congestion avec du trafic TCP sur Iperf3

La ligne de commande qui permet de simuler ce test est la suivante :

```
Iperf3 –c 192.168.2.48 –b7000000 –t1800 –i60
```

On génère du trafic TCP depuis notre PC client avec une bande passante de 7mégas durant 30 minutes.

```
ID] Interval           Transfer     Bandwidth       Retr
 4]   0.00-1581.39 sec  1.22 GBytes  6.61 Mbits/sec  220             sender
```

Voici le résultat obtenu, le test a permis le transfert de 1.22 GBytes avec une bande passante de 6.61Mbits/s. Nous pouvons également observer que le test n'a pas duré exactement 30 minutes (1800 secondes) puisque le port a été shutdown vers la fin.

6.2.3.3 Trafic supplémentaire avec Iperf version 2

Vers la fin du test, on va générer 10 secondes de trafic UDP supplémentaire à 1mbit/s pour faire tomber le port puisqu'on va atteindre le seuil des 80 % de consommation de bande passante fixée par Storm control.

Voici la ligne de commande à entrer :

```
Iperf –c 192.168.2.48 –t10 –b1m -u
```

Si je déclenche ce test en fin de trafic et non en plein milieu, ce qui nous aurait permis d'observer ce qui se serait passé lors de la réouverture du port est simplement dû au fait que le flux Iperf tombe également et n'arrive pas à renvoyer du trafic, même lorsque la connexion est rétablie. Il serait intéressant cependant d'observer un cas réel où lorsque le port est à nouveau opérationnel, comment le trafic recommence à circuler.

Voici les résultats causés par ce surplus de trafic :

```
1487.00-1488.00 sec   851 KBytes  6.97 Mbits/sec
1488.00-1489.00 sec   883 KBytes  7.24 Mbits/sec
1489.00-1490.00 sec   825 KBytes  6.76 Mbits/sec
1490.00-1491.00 sec   851 KBytes  6.98 Mbits/sec
1491.00-1492.00 sec   883 KBytes  7.24 Mbits/sec
1492.00-1493.00 sec   825 KBytes  6.76 Mbits/sec
1493.00-1494.00 sec   180 KBytes  1.48 Mbits/sec
1494.00-1495.00 sec  0.00 Bytes  0.00 bits/sec
1495.00-1496.00 sec  0.00 Bytes  0.00 bits/sec
1496.00-1497.00 sec  0.00 Bytes  0.00 bits/sec
1497.00-1498.00 sec  0.00 Bytes  0.00 bits/sec
1498.00-1499.00 sec  0.00 Bytes  0.00 bits/sec
1499.00-1500.00 sec  0.00 Bytes  0.00 bits/sec
1500.00-1501.00 sec  0.00 Bytes  0.00 bits/sec
1501.00-1502.00 sec  0.00 Bytes  0.00 bits/sec
```

Comme nous pouvons le constater, plus aucun trafic ne passe. Le flux supplémentaire est venu perturber notre test et a déclenché notre seuil Storm control et a mis en « shutdown » le port.

Rappelez-vous du test précédent (test n°4), la réaction n'a pas été la même, le trafic n'avait pas été stoppé net comme ici mais fortement ralenti avant d'être totalement rétabli une fois la surcharge de trafic terminée.

Dans le cadre de ce test, nous savons que c'est Storm control qui a bloqué le port. Dans un cas réel, on n'aurait pas pu le deviner puisqu'il peut y avoir plusieurs raisons à ce que le trafic ne passe plus (câble débranché, maintenance en cours, routeur éteint, etc.). Nous allons donc directement aller consulter le serveur Syslog afin de savoir ce qui a interrompu notre trafic.

6.2.3.4 Consultation du serveur Syslog

Après avoir appliqué un filtre aux équipements du réseau, nous avons la confirmation que c'est bien Storm control qui a bloqué le port. Voici ce qui a été signalé :

```
Nov 20 10:10:14 192.168.1.8 71: Nov 20 10:10:09: %PM-4-ERR_DISABLE: storm-control error detected on Fa0/3,
putting Fa0/3 in err-disable state
Nov 20 10:10:19 192.168.1.8 72: Nov 20 10:10:09: %STORM_CONTROL-3-SHUTDOWN: A packet storm was detected on Fa0/3.
The interface has been disabled.
Nov 20 10:10:24 192.168.1.8 73: Nov 20 10:10:10: %LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/3,
changed state to down
Nov 20 10:10:29 192.168.1.8 74: Nov 20 10:10:11: %LINK-3-UPDOWN: Interface FastEthernet0/3, changed state to down

Nov 20 10:15:15 192.168.1.8 75: Nov 20 10:15:09: %PM-4-ERR_RECOVER: Attempting to recover from storm-control err-
disable state on Fa0/3
Nov 20 10:15:20 192.168.1.8 76: Nov 20 10:15:13: %LINK-3-UPDOWN: Interface FastEthernet0/3, changed state to up
Nov 20 10:15:25 192.168.1.8 77: Nov 20 10:15:16: %LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/3,
changed state to up
```

Le switch ayant comme IP 192.168.1.8 a activé Storm control après avoir dépassée 80 % d'utilisation de la bande passante.

Nous avons ici été alerté en premier lieu par l'option errdisable configuré sur la switch (%PM-4-ERR_DISABLE et %PM-4-ERR_RECOVER) qui nous indique à quelle heure le port a été fermé, puis réactivé. On remarque l'importance de cette option qui nous indiquera pourquoi le port se désactive (Storm-control error) et se réactive (Attempting to recover from storm-control err-)

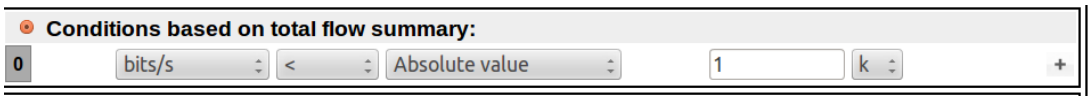
Il y a également eu une alerte Storm control (%STORM_CONTROL-3-SHUTDOWN) indiquant que l'interface a été désactivée. Cependant, Storm control ne nous indiquera pas quand l'interface sera réactivée.

Finalement, nous avons été alertés par les logs classiques (UPDOWN) qui s'activent lorsqu'il y a un « shutdown » ou un « no shutdown » qui survient sur une interface.

6.2.3.5 Mise en place et déclenchement des alertes sur Nfsen

Aucune alerte ne sera créée pour ce test puisque, comme nous l'avons vu au test précédent, lorsque le seuil fixé du Storm control est atteint, Nfsen ne parvient pas à détecter l'alerte à cause de la fréquence d'échantillonnage de cinq minutes. Ici le port sera directement shutdown lorsque le trafic atteindra un débit de 8mbit/s et une éventuelle alerte avec le même seuil ne sera pas déclenchée.

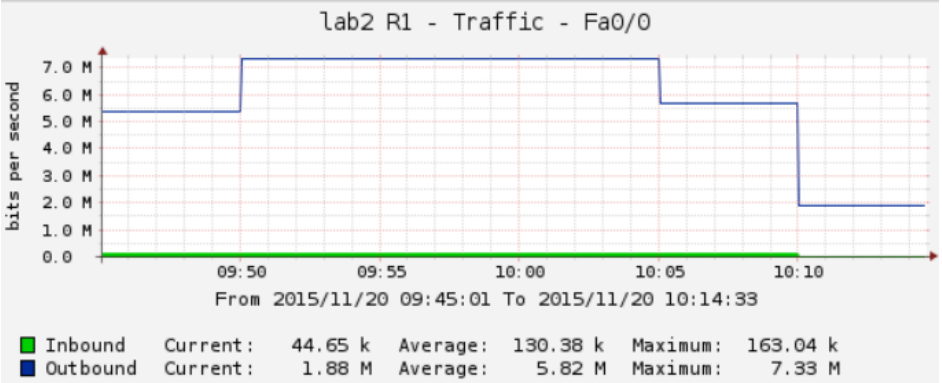
Néanmoins, nous pourrions imaginer une alerte qui détecte lorsqu'il n'y a plus de trafic. Voici un exemple :



Cette alerte se déclenche lorsque les bits/s sont inférieurs à 1k, on ne peut pas mettre 0k puisqu'il faudrait alors du trafic négatif pour la déclencher. Malheureusement, Nfsen ne dispose que des conditions « < » et « > ». Avec la condition « = », on aurait pu facilement créer une alerte qui s'active seulement lorsque les bits/s sont égals à zéro.

6.2.3.6 Déterminer le coupable avec Cacti et Nfsen

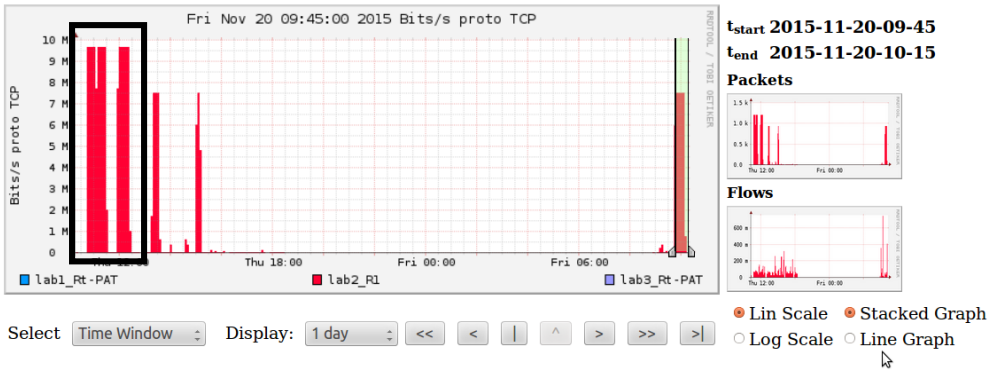
Le responsable du réseau a pu prendre connaissance du problème grâce au serveur Syslog. Essayons de déterminer ce qui a déclenché le seuil Storm control.



Cacti nous permet de savoir que, comme d'habitude, le problème provient de trafic sortant mais, celui-ci ne pourra pas nous donner plus d'information compte tenu de la fréquence d'échantillonnage. On distingue cependant une fréquence d'échantillonnage légèrement plus basse à la fin du graphique, ce qui laisse envisager que le trafic a diminué.

Nous allons nous tourner vers Nfsen pour identifier la personne ayant causé le plus de trafic sur le réseau et ainsi pouvoir la réprimander.

Voyons ce que Nfsen nous montre.



Statistics timeslot Nov 20 2015 - 09:45 - Nov 20 2015 - 10:15

Channel:	Flows:					Packets:					Traffic:			
	all:	tcp:	udp:	icmp:	other:	all:	tcp:	udp:	icmp:	other:	all:	tcp:	udp:	icmp:
lab3_Rt-PAT	0 /s	0 /s	0 /s	0 /s	0 /s	0 /s	0 /s	0 /s	0 /s	0 /s	0 b/s	0 b/s	0 b/s	0 b/s
lab2_R1	0.6 /s	0.1 /s	0.5 /s	0.0 /s	0 /s	652.6 /s	651.6 /s	1.0 /s	0.0 /s	0 /s	5.3 Mb/s	5.3 Mb/s	1.0 kb/s	20.4 b/s
lab1_Rt-PAT	0 /s	0 /s	0 /s	0 /s	0 /s	0 /s	0 /s	0 /s	0 /s	0 /s	0 b/s	0 b/s	0 b/s	0 b/s
	all:	tcp:	udp:	icmp:	other:	all:	tcp:	udp:	icmp:	other:	all:	tcp:	udp:	icmp:
TOTAL	0.6 /s	0.1 /s	0.5 /s	0.0 /s	0 /s	652.6 /s	651.6 /s	1.0 /s	0.0 /s	0 /s	5.3 Mb/s	5.3 Mb/s	1.0 kb/s	20.4 b/s

En sélectionnant le bon laps de temps (à droite), nous constatons que le débit identifié ne correspond pas vraiment à notre test Iperf. C'est normal puisque le test a été interrompu et n'a pas duré les 30 minutes initialement prévues.

Etant donné que c'est un labo de test, le trafic apparaissant ci-dessus provient uniquement de Nfsen. Dans un cas réel, nous aurions du trafic en continu. Observez l'encadré noir (ces données n'ont rien à voir avec ce test) qui colle parfaitement avec ce qu'aurait été un cas réel d'un port fermé suite à l'activation de Storm control, puis réactivé après un certain laps de temps. On a du trafic, une coupure puis du trafic à nouveau.

Comme pour chaque test, on va analyser les statistiques de Nfsen.

```
Date first seen          Duration Proto          IP Addr    Flows(%)     Packets(%)       Bytes(%)         pps
2015-11-20 09:45:02.776  1634.744 any        192.168.1.12      70(100.0)    1.4 M(100.0)    1.4 G(100.0)      835
2015-11-20 09:45:02.776  1634.744 any        192.168.2.48      70(100.0)    1.4 M(100.0)    1.4 G(100.0)      835

Summary: total flows: 70, total bytes: 1377529037, total packets: 1366084, avg bps: 6741258, avg pps: 835, avg bpp:
Time window: 2015-11-20 09:44:56 - 2015-11-20 10:20:00
Total flows processed: 1197, Blocks skipped: 0, Bytes read: 77420
Sys: 0.001s flows/second: 708284.0   Wall: 0.000s flows/second: 2375000.0
```

Nous arrivons comme d'habitude à identifier le coupable et à retrouver le trafic qui a été généré avant la fermeture du port. En observant cette statistique, nous arrivons à identifier l'élément le plus perturbateur de notre réseau qui à lui seul, a utilisé 6.7mbit/s

de la bande passante, ce qui n'a pas laissé beaucoup de marges aux autres utilisateurs du réseau et a eu comme conséquence le déclenchement de Strom control.

6.2.3.7 Conclusion du test n°5

Nous avons généré du trafic et tout se passait bien jusqu'à ce que le débit dépasse 8mbits/s. Storm control a alors été activé et à shutdown le port rendant toute communication impossible durant cinq minutes. On ne peut pas vraiment dire qu'il y a eu une congestion durant ce test puisque le trafic n'a pas été ralenti mais directement bloqué suite à un débit trop important. Nfsen et Cacti ont été un peu moins performant pour ce test, puisqu'ils servent à analyser du trafic et quand il n'y en a pas, il est forcément difficile de l'analyser. En revanche, le serveur Syslog nous a permis de comprendre rapidement d'où venait le problème et s'est révélé indispensable.

Ce qu'il faut retenir de ce test est que l'option shutdown de Storm control est vraiment une solution radicale. On ne laisse même pas la congestion se produire, on bloque directement tout le trafic. Il faut vraiment bien connaître son réseau avant d'activer une telle option. Dans la plupart des cas, la solution montrée au test précédent (test n°4) qui jetait temporairement le surplus de trafic dépassant un seuil plutôt que de fermer le port semble plus judicieuse. Mais comme on dit, à chaque problème sa solution.

6.3 Schéma n°3, routeurs avec câble coaxial, phases de test

Pour les tests du schéma n°3, une suite de câbles coaxiaux ont été reliés entre les deux routeurs permettant ainsi d'y joindre deux PCs. C'est donc entre eux qu'on observera la congestion. Cette partie comprendra deux tests :

- Un test qui simule une longue congestion utilisant une forte bande passante (90 %)
- Double test Iperf en allouant 45 % de la bande passante pour le test entre le client et le serveur et 45 % pour le test Iperf entre les deux autres PC

Si besoin, un test pourra être subdivisé en deux parties (TCP et UDP).

La simulation sera faite entre le PC client et serveur. A la fin de chaque test, il faudra être capable d'avoir identifié et alerté du problème ainsi que trouver le perturbateur du réseau ayant provoqué la congestion, soit le PC client.

6.3.1 Test n°6, schéma n°3, simulation d'une longue congestion

Cette simulation montrera deux charges de trafic TCP et UDP d'une heure chacune. Nous fixerons la bande passante à 0,9 méga pour qu'elle représente ainsi 90 % d'utilisation.

6.3.1.1 Simulation de la congestion avec du trafic TCP sur Iperf3

Pour réaliser ce test, nous entrerons la ligne de commande suivante :

```
iperf3 –c 192.168.2.48 –b9000000 –t3600 –i60
```

On lance le test TCP depuis notre machine client avec une bande passante de 9mégas (puisque le câble coaxial utilisé est du 10base2 et fait du 10mégas) pour une durée d'une heure.

```
Interval           Transfer     Bandwidth       Retr
  0.00-3600.00 sec  3.30 GBytes  7.87 Mbits/sec  2999             sender
  0.00-3600.00 sec  3.30 GBytes  7.87 Mbits/sec                  receiver
```

Pour résumer le tout, ce test a transféré 3.30 GBytes avec une bande passante de 7.87Mbits/s. Il y a eu un total de 2'999 retransmissions pour un total de 3'719'890 paquets envoyés (donnée récupérée avec Nfsen), ce qui représente un taux de retransmission relativement faible**.**

Nous constatons que contrairement aux précédents tests où on a également simulé du trafic sur 90 % de la bande passante, les résultats obtenus ne sont pas les mêmes. En effet, lors des tests des schémas un et deux, la bande passante était souvent de 9Mbits/s alors qu'ici elle n'est que de 7.87Mbits/s ! Qu'est ce qui a provoqué cette congestion ? Nous ne pouvons pour l'instant pas vraiment le savoir, mais, on peut émettre l'hypothèse que cela vient de la technologie 10base2. Continuons l'analyse pour en avoir le cœur net.

```
2040.00-2100.00 sec  56.5 MBytes  7.90 Mbits/sec   41
2100.00-2160.00 sec  56.0 MBytes  7.83 Mbits/sec   44
2160.00-2220.00 sec  57.4 MBytes  8.02 Mbits/sec   59
2220.00-2280.00 sec  56.9 MBytes  7.95 Mbits/sec   52
2280.00-2340.00 sec  56.1 MBytes  7.85 Mbits/sec   44
2340.00-2400.00 sec  56.2 MBytes  7.86 Mbits/sec   48
2400.00-2460.00 sec  55.8 MBytes  7.79 Mbits/sec   40
2460.00-2520.00 sec  56.4 MBytes  7.88 Mbits/sec   67
2520.00-2580.00 sec  56.1 MBytes  7.85 Mbits/sec   71
2580.00-2640.00 sec  55.9 MBytes  7.81 Mbits/sec   37
2640.00-2700.00 sec  56.0 MBytes  7.83 Mbits/sec   45
2700.00-2760.00 sec  55.9 MBytes  7.81 Mbits/sec   46
2760.00-2820.00 sec  56.2 MBytes  7.86 Mbits/sec   46
2820.00-2880.00 sec  55.6 MBytes  7.78 Mbits/sec   52
```

En observent les résultats intermédiaires, nous constatons que le débit n'a jamais atteint les 9Mbits/s, c'est étrange. Nfsen et Cacti ne pourront certainement pas nous aider à comprendre pourquoi le débit a été plus bas en comparaison aux précédents tests. Afin d'en savoir davantage, allons consulter le serveur Syslog qui pourra peut-être nous renseigner, mais avant, regardons ce qu'a donné le même test avec UDP.

6.3.1.2 Simulation de la congestion avec du trafic UDP sur Iperf2

Afin de simuler ce test, voici la commande à entrer :

```
iperf –c 192.168.2.48 –b9000000 –t3600 –u –i60
```

Par rapport au test précédent, nous spécifions que le type de trafic doit être UDP (–u).

```
Sent 2756510 datagrams
Server Report:
 0.0-3600.0 sec  3.77 GBytes  9.00 Mbits/sec   0.119 ms    0/2756509 (0%)
```

Nous constatons qu'UDP a envoyé plus de données qu'avec TCP alors que cela n'avait pas été le cas pour le test n°1 lorsque nous avions comparé les deux protocoles. Comment expliquer cela ? Pour nous mettre sur la piste, nous allons devoir comparer ce qui a été capturé par notre serveur Syslog pour chacun des deux tests afin de voir s'il y a eu des différences.

6.3.1.3 Consultation du serveur Syslog

Voici ce que retourne le serveur Syslog pour le test ayant généré du trafic TCP :

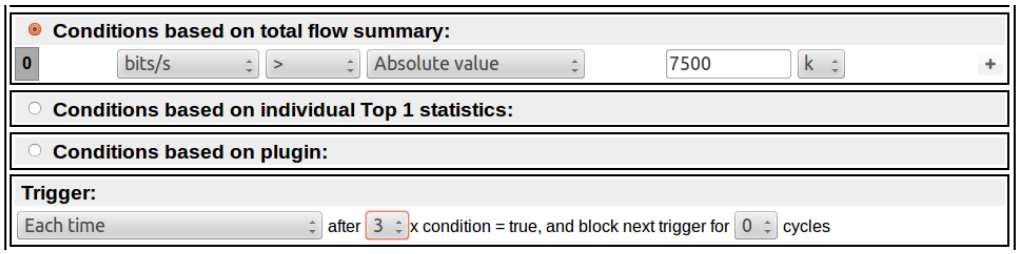
```
Nov 20 16:17:21 192.168.2.3 254: .Nov 20 16:16:57: %AMDP2_FE-6-EXCESSCOLL: Ethernet1/0 TDR=2, TRC=0
Nov 20 16:18:40 192.168.2.3 256: .Nov 20 16:18:17: %AMDP2_FE-6-EXCESSCOLL: Ethernet1/0 TDR=3, TRC=0
```

J'ai tronqué les résultats obtenus mais, voici ce qui est apparu très souvent tout au long du test. Cette erreur indique qu'un nombre excessif de collision est en train de se produire, ce qui explique pourquoi notre test Iperf a généré moins de trafic que ce qu'il aurait dû. Pour plus d'information sur les collisions, je vous renvoie à la rubrique 2.4.3 qui parle de ce phénomène.

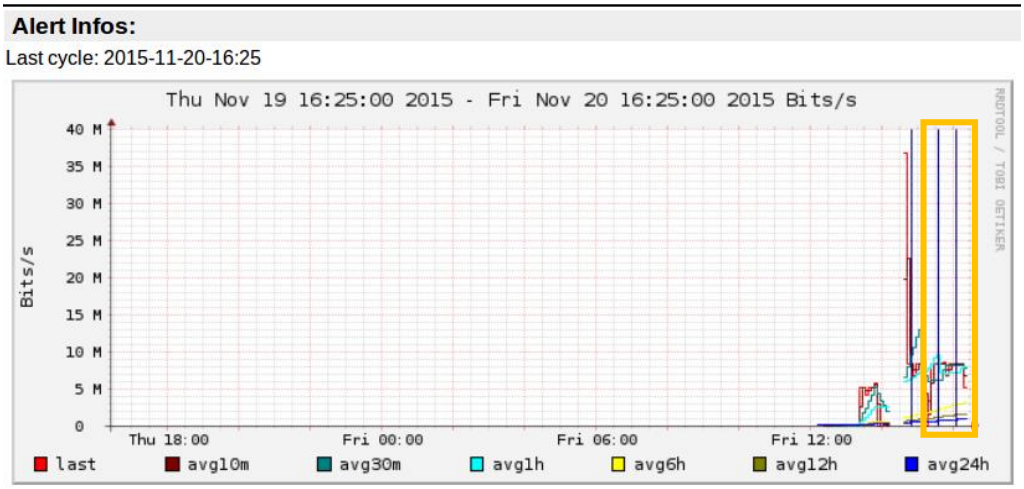
Concernant le test qui a généré du trafic UDP, aucun message n'a été trouvé dans les logs puisque tout s'est bien passé. Ceci peut nous amener à la conclusion suivante : Ce sont bien les collisions générées au test avec du trafic TCP qui ont diminué le débit maximum du test. TCP doit attendre son acquittement (ACK) pour continuer à transmettre, ce qui crée une communication dans les deux sens, le mécanisme CSMA/CD du câble coaxial se déclenche et créé des collisions. Le protocole UDP n'ayant pas de système d'acquittement, il n'a pas eu ce problème.

6.3.1.4 Mise en place et déclenchement des alertes sur Nfsen

Notre alerte se déclencher également lorsque l'utilisation de la bande passante dépassera 75 % et aura été atteinte trois fois.



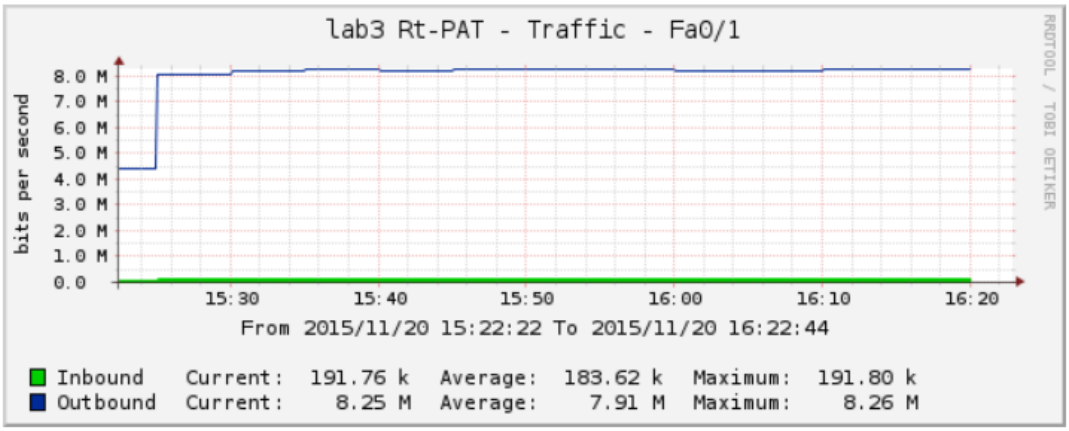
Voyons les informations de l'alerte après avoir finalisé le test TCP :



Le rectangle orange représente le trafic analysé, on y voit des lignes verticales bleues, ce qui indique que notre alerte s'est bien déclenchée durant le test. L'administrateur a été informé et poura intervenir. Pour UDP l'alerte a également été déclenchée, inutile de vous montrer deux fois un graphique similaire.

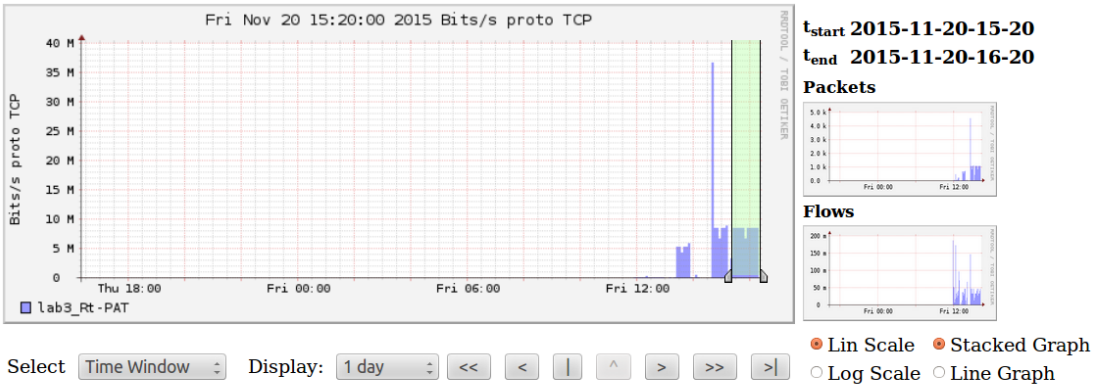
6.3.1.5 Test TCP, déterminer le coupable avec Cacti et Nfsen

Voyons ce que nous montre Cacti pour TCP :



Nous constatons une moyenne de trafic sortant de 7.91Mbits/s et un pic maximum à 8.26Mbit/s, ce qui semble correspondre avec le test TCP d'Iperf.

On va maintenant se tourner vers Nfsen pour savoir d'où provient le trafic.



Statistics timeslot Nov 20 2015 - 15:20 - Nov 20 2015 - 16:20

Channel:	Flows:					Packets:					Traffic:			
	all:	tcp:	udp:	icmp:	other:	all:	tcp:	udp:	icmp:	other:	all:	tcp:	udp:	icmp:
lab3_Rt-PAT	0.4 /s	0.0 /s	0.3 /s	0.0 /s	0 /s	954.8 /s	953.9 /s	0.8 /s	0.0 /s	0 /s	7.7 Mb/s	7.7 Mb/s	587.7 b/s	56.7 b/s
lab2_R1	0 /s	0 /s	0 /s	0 /s	0 /s	0 /s	0 /s	0 /s	0 /s	0 /s	0 b/s	0 b/s	0 b/s	0 b/s
lab1_Rt-PAT	0 /s	0 /s	0 /s	0 /s	0 /s	0 /s	0 /s	0 /s	0 /s	0 /s	0 b/s	0 b/s	0 b/s	0 b/s
	all:	tcp:	udp:	icmp:	other:	all:	tcp:	udp:	icmp:	other:	all:	tcp:	udp:	icmp:
TOTAL	0.4 /s	0.0 /s	0.3 /s	0.0 /s	0 /s	954.8 /s	953.9 /s	0.8 /s	0.0 /s	0 /s	7.7 Mb/s	7.7 Mb/s	587.7 b/s	56.7 b/s

En sélectionnant le laps de temps identifié précédemment dans Cacti, nous arrivons à obtenir plus ou moins les mêmes résultats, voyons de plus près l'affichage des flux.

L'image ci-dessous, comme pour les autres tests, est déjà filtré par « ip 192.168.1.12 and proto TCP ».

```
Date first seen          Duration Proto          IP Addr    Flows(%)     Packets(%)       Bytes(%)         pps
2015-11-20 15:22:10.605  3594.261 any        192.168.1.12     124(100.0)    3.7 M(100.0)    3.7 G(100.0)     1034
2015-11-20 15:22:10.605  3594.261 any        192.168.2.48     124(100.0)    3.7 M(100.0)    3.7 G(100.0)     1034

Summary: total flows: 124, total bytes: 3736891255, total packets: 3719890, avg bps: 8317462, avg pps: 1034, avg bp
Time window: 2015-11-20 15:19:49 - 2015-11-20 16:25:00
Total flows processed: 1556, Blocks skipped: 0, Bytes read: 103588
Sys: 0.002s flows/second: 571428.6   Wall: 0.000s flows/second: 2146206.9
```

Nous avons réussi à identifier le coupable de la surcharge de trafic débutée à 15h22m10s pour une durée de 3594 secondes. En jetant un coup d'œil sur le Schéma n°3 nous arrivons encore une fois à identifier le PC client comme étant le coupable.

6.3.1.6 Test UDP, déterminer le coupable avec Cacti et Nfsen

Les mêmes étapes que pour le test TCP ont été réalisées et les résultats sont assez similaires, on peut donc passer directement à la prochaine rubrique.

6.3.1.7 Conclusion du test n°6

Notre simulation ne s'est pas déroulée comme elle aurait dû. En effet, elle a provoqué un nombre important de collisions pour le test Iperf TCP. Nous avons cependant réussi à identifier le problème grâce aux logs. Pour le test UDP, tout s'est bien passé et les collisions n'ont pas eu lieu. Cela peut s'expliquer par la différence entre le protocole TCP qui est plus lourd mais, assure une bonne qualité de service et UDP qui va plus vite mais, ne peut garantir la qualité de son service.

Nous avons pu observer ici notre première différence entre le trafic TCP qui a généré des collisions et le trafic UDP pour qui tout s'est bien déroulé. Sachant que ce problème s'est produit uniquement dans le schéma n°3, nous en avons déduis que c'est notre câble coaxial 10base2 qui produit les collisions et que ce problème survient uniquement lorsqu'il y a plus d'une communication en même temps.

Une chose à retenir pour ce test est que notre serveur Syslog peut nous donner de précieuses informations. Sans nos logs, nous aurions mis beaucoup plus de temps à comprendre que le problème était dû à un trop grand nombre de collisions.

6.3.2 Test n°7, schéma n°3, simulation de deux longue congestion parallèle

Ce test simulera deux charges de trafic UDP d'une heure chacune et se fera cette fois-ci, de manière simultanée. Le test Iperf allant du client au serveur simulera 4.5mégas de bande passante. Le test simulant les deux PCs entre les routeurs simulera 4.5mégas de bande passante également.

6.3.2.1 Simulation de la congestion avec Iperf2 entre le client et le serveur

Voici la ligne de commande Iperf à entrer afin de démarrer ce test.

```
iperf –c 192.168.2.48 –b4500000 –t3600 –i60 –u –l 8192
```

On précise que nous voulons du trafic UDP lancé depuis notre machine client avec une bande passante de 4.5mégas pour une durée d'une heure. Par intervalle de 60 secondes, on reçoit des informations intermédiaires. On spécifie finalement le la taille du buffer UDP à 8192 bytes (-l 8192) pour qu'elle ait la même valeur que le test Iperf3 qui aura lieu en parallèle.

```
[ ID] Interval           Transfer     Bandwidth       Jitter    Lost/Total Datagra
ms
[  5]   0.00-3600.16 sec  1.89 GBytes  4.50 Mbits/sec  25.928 ms  24313/247161 (9.
8%)
```

Avec ce premier test, nous avons transferé 1.89GBytes avec une bande passante de 4.31Mbits/s (le test a buggé et affiche ci-dessus 4.5Mbit/s, j'ai effectué quelques calculs pour trouver le débit réel). Il y a eu en tout 24'313 paquets perdus, ce qui représente 9.8 % du total. La gigue est également un peu plus élevée que d'habitude. Pour l'instant, les résultats observés sont plus mauvais que le test précédent (test n°6). Pour rappel, nous avons ici simulé deux fois 4.5Mbits de trafic ce qui fait 9Mbits au total, soit le même montant que le test n°6 pour lequel aucun paquet n'avait été perdu. Il va nous falloir consulter les résultats intermédiaires afin d'en savoir un peu plus.

```
3468.0-3469.0 sec   496 KBytes  4.06 Mbits/sec  11.967 ms    4/   66 (6.1%)
3469.0-3470.0 sec   512 KBytes  4.19 Mbits/sec  11.029 ms    1/   65 (1.5%)
3470.0-3471.0 sec   536 KBytes  4.39 Mbits/sec  10.871 ms    2/   69 (2.9%)
3471.0-3472.0 sec   536 KBytes  4.39 Mbits/sec  13.499 ms    4/   71 (5.6%)
3472.0-3473.0 sec   416 KBytes  3.41 Mbits/sec  19.602 ms    1/   53 (1.9%)
3473.0-3474.0 sec   504 KBytes  4.13 Mbits/sec  17.760 ms   10/   73 (14%)
3474.0-3475.0 sec   456 KBytes  3.74 Mbits/sec  22.377 ms    7/   64 (11%)
3475.0-3476.0 sec   240 KBytes  1.97 Mbits/sec  19.638 ms   15/   45 (33%)
3476.0-3477.0 sec  0.00 Bytes  0.00 bits/sec  19.638 ms    0/    0 (-nan%)
3477.0-3478.0 sec   584 KBytes  4.78 Mbits/sec  11.771 ms  107/  180 (59%)
3478.0-3479.0 sec   424 KBytes  3.47 Mbits/sec  11.745 ms    1/   54 (1.9%)
3479.0-3480.0 sec   640 KBytes  5.24 Mbits/sec  12.565 ms    1/   81 (1.2%)
3480.0-3481.0 sec   512 KBytes  4.19 Mbits/sec  16.105 ms    1/   65 (1.5%)
3481.0-3482.0 sec   592 KBytes  4.85 Mbits/sec   9.627 ms    2/   76 (2.6%)
3482.0-3483.0 sec   528 KBytes  4.33 Mbits/sec   8.870 ms    1/   67 (1.5%)
3483.0-3484.0 sec   520 KBytes  4.26 Mbits/sec   7.251 ms    1/   66 (1.5%)
3484.0-3485.0 sec   552 KBytes  4.52 Mbits/sec   9.433 ms    3/   72 (4.2%)
3485.0-3486.0 sec   408 KBytes  3.34 Mbits/sec  15.420 ms   10/   61 (16%)
3486.0-3487.0 sec   536 KBytes  4.39 Mbits/sec  16.357 ms    2/   69 (2.9%)
```

Nous pouvons observer que la perte de paquets est plus ou moins régulière, mais pas toujours. En effet, à un moment la perte de paquets devient plus importante, puis, le trafic se coupe pour une courte durée. Qu'est-ce qui peut expliquer ce phénomène ? Je soupçonne les buffers des routeurs qui sont pleins et abandonnent le surplus de trafic qu'ils n'arrivent pas à gérer. De plus, nous constatons une perte régulière de

paquets, ce qui provient probablement des collisions générées à cause du câble coaxial. Voyons les résultats de la deuxième simulation avant d'aller consulter les logs pour vérifier si mon hypothèse est bonne.

6.3.2.2 Simulation de la congestion avec Iperf3 entre les deux PCs cisco-td-xx

Ce test sera exactement le même que celui présenté ci-dessus, mais avec Iperf3. On ne précise pas la taille du buffer UDP car sa valeur par défaut est déjà de 8192 bytes.

```
Iperf3 –c 192.168.2.48 –b4500000 -u –t3600 –i60
```

Il sera également lancé depuis Linux puisque les deux PCs cisco-td-06 et cisco-td-10 sont des machines qui tournent sous Ubuntu 12.04.

```
0.0-3600.2 sec  1.83 GBytes  4.37 Mbits/sec  24.059 ms 3026/243018 (1.2%)
```

En comparaison avec le test du dessus, les résultats sont un peu mieux. La bande passante affichée est de 4.37bits/sec, ce qui est à peine meilleur. Par contre, le taux de perte de paquets est nettement plus bas ici.

Comment expliquer pour cette simulation une perte de paquets de 1.2% alors qu'il y en a eu 9.8% pour l'autre ? Nous pouvons probablement assimiler cette différence au hasard, cela aurait très bien pu être l'inverse. Le câble coaxial en half duplex utilise CSMA/CD et va attendre la fin de la communication en cours pour pouvoir continuer à envoyer des données. C'est alors le premier équipement qui prendra la parôle qui pourra envoyer sa charge de trafic. Pour ce test, on peut en déduire que les équipements du test simulant du trafic entre les deux Pcs cisco-td-xx ont eu plus souvent la parole en premier. Ceci est peut-être dû au hasard ou alors, les équipements sont simplement plus performants. Il faudrait refaire le même test plusieurs fois pour en être sur.

6.3.2.3 Consultation du serveur Syslog

Notre Syslog va nous permettre de comprendre pourquoi la communication s'est mal passée.

```
Nov 23 19:31:14 192.168.2.3 589: .Nov 23 19:30:04: %IP_VFR-4-FRAG_TABLE_OVERFLOW: Ethernet1/0: the fragment table
has reached its maximum threshold 16
Nov 23 19:31:19 192.168.0.4 413: Nov 23 19:31:11: %AMDP2_FE-6-EXCESSCOLL: Ethernet1/0 TDR=2, TRC=0
Nov 23 19:31:47 192.168.0.4 414: Nov 23 19:31:41: %AMDP2_FE-6-EXCESSCOLL: Ethernet1/0 TDR=0, TRC=0
Nov 23 19:32:11 192.168.2.3 590: .Nov 23 19:31:00: %AMDP2_FE-6-EXCESSCOLL: Ethernet1/0 TDR=2, TRC=0
Nov 23 19:32:18 192.168.0.4 415: Nov 23 19:32:11: %AMDP2_FE-6-EXCESSCOLL: Ethernet1/0 TDR=2, TRC=0
Nov 23 19:32:23 192.168.2.3 591: .Nov 23 19:31:11: %IP_VFR-4-FRAG_TABLE_OVERFLOW: Ethernet1/0: the fragment table
has reached its maximum threshold 16
Nov 23 19:32:48 192.168.0.4 416: Nov 23 19:32:42: %AMDP2_FE-6-EXCESSCOLL: Ethernet1/0 TDR=3, TRC=0
Nov 23 19:32:54 192.168.2.3 592: .Nov 23 19:31:43: %AMDP2_FE-6-EXCESSCOLL: Ethernet1/0 TDR=3, TRC=0
Nov 23 19:33:09 192.168.2.3 593: .Nov 23 19:31:59: %IP_VFR-4-FRAG_TABLE_OVERFLOW: Ethernet1/0: the fragment table
has reached its maximum threshold 16
Nov 23 19:33:18 192.168.0.4 417: Nov 23 19:33:12: %AMDP2_FE-6-EXCESSCOLL: Ethernet1/0 TDR=3, TRC=0
Nov 23 19:33:37 192.168.2.3 594: .Nov 23 19:32:27: %AMDP2_FE-6-EXCESSCOLL: Ethernet1/0 TDR=1, TRC=0
```

Mon intuition était bonne ! L'erreur « IP_VFT-4-FRAG_TABLE_OVERFLOW » nous indique que le buffer du routeur est plein. Il va donc abandonner le surplus de trafic jusqu'à être à nouveau opérationnel. Dans cette capture des logs, on voit que les buffers se sont remplis environt toutes les minutes. Nous pouvons aussi observer que les collisions se produisent à la fois dans le routeur « Rt-Client » et « Rt-PAT ».

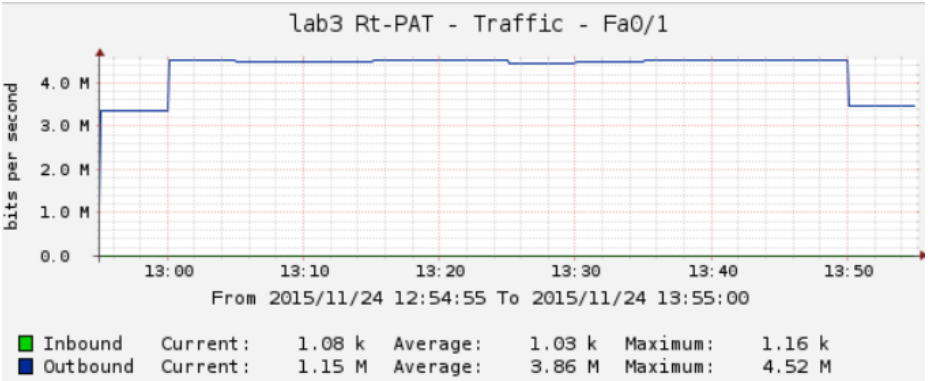
La question à se poser maintenant et à laquelle nous pouvons directement répondre est : Pourquoi les buffers se sont remplis avec deux charges de 4.5Mbits et non avec une de 9Mbits ? Comme nous le montre les données du log, le fait d'avoir eu deux charges de trafic en même temps au lieu d'une a créé des collisions. Par conséquent, la génération des collisions ont forcé le routeur à remplir son buffer plusieurs fois, ce qui l'a obligé à abandonner un nombre de paquets importants.

6.3.2.4 Mise en place et déclenchement des alertes sur Nfsen

Nous n'allons pas nous intéresser au déclenchement d'alerte ici puisqu'elle a déjà été abordée plusieurs fois dans les autres tests et cela serait répétitif.

6.3.2.5 Test UDP, Déterminer le coupable avec Cacti et Nfsen

Passons rapidement à l'analyse du graphique de Cacti :



On ne voit que la moitié du trafic généré, celle qui va du PC au serveur. C'est normal puisque le trafic généré entre les deux PCs cisco-td-xx ne passe pas par notre point de monitoring qui est le routeur « Rt-PAT ». Il nous faudra alors demander des droits d'accès pour contrôler l'entiereté de notre réseau, nous pourrons ainsi contrôler tout ou une partie des agents SNMP de notre réseau.

Nous n'allons pas procéder à l'analyse Nfsen étant donné que la conclusion à en tirer est la même que pour les autres tests.

6.3.2.6 Conclusion du test n°7

Nous avons simulé le même trafic que pour le test n°6 et pourtant, les résultats ne sont pas pareils. Nous avons créé de la congestion en simulant deux trafics parallèles de

4.5Mbits/s et une congestion s'est produite alors que nous n'avons pas utilisé l'intégralité de notre bande passante. Cette congestion est due à la création de collisions qui ont provoqué la perte de paquets réguliers et ont obligé les équipements à remplir leur buffer, puis à abandonner le surplus de trafic. Ici, nous avons uniquement simulé deux communications parallèles, le nombre d'équipements qui on dû prendre la parole pour transmettre leur données suite à une collision était donc de deux également. Une simulation de plus grande envergure avec par exemple, 40 communications parallèles auraient probablement créé énormément de collisions et la bande passante aurrait été beaucoup plus saturée et les résultats plus mauvais.

Ce qu'il faut retenir de ce test est que le réseau va agir différement selon le nombre de communications qui y transitent. Ce n'est pas le fait d'utiliser 90% de la bande passante qui pose problème puisque si une personne l'utilise exclusivement, il n'y aura probablement aucun risque de congestion. Par contre, si neuf personnes utilisent chacune 10 % de la bande passante, de nouveaux facteurs entre en compte comme on a pu le constater ici avec la création de collisions et une congestion aura plus de chance de se produire.

6.4 Conclusion des phases de test

Maintenant que la phase de test est terminée, il est temps de faire un résumé du tout. A travers ces tests, nous avons observé différents outils et certains se sont montrés plus intéressant que d'autres.

Nfsen s'est revélé être un outil indispensable puisque c'est le seul qui est capable d'identifier la source du problème, à savoir, trouver l'adresse IP de l'individu ayant généré le surplus de trafic. De plus, il permet également de définir des alertes, ce qui est essentiel pour pouvoir prévenir l'administrateur du réseau. Cacti à été moins important, mais tout de même intéressant. Il nous a permis d'un simple coup d'œil de visualiser des informations utiles comme le type de trafic (entrant ou sortant), la moyenne et le pic de trafic. Ces informations peuvent également être retrouvées sur Nfsen, mais, il faut y faire plusieurs manipulations, ce qui peut être fastidieux. Le serveur Syslog est un outil indispensable également. Lorsque tout fonctionne, il n'y a pratiquement pas besoin de le consulter, cependant, au moindre souci, il saura nous indiquer avec précision d'où vient le problème. Storm control s'est révélé être une méthode très intéressante qui permet de gérer le réseau en jetant le surplus de trafic, mais, il faudra tout de même l'utiliser avec prudence. Finalement, Iperf peut également s'avérer utile pour analyser les performances de notre réseau et nous indiquer si celui-ci fonctionne correctement.

Concernant les tests à proprement parler, vous avez sûrement constaté qu'ils sont parfois un peu répétitifs. Cependant, nous avons tout de même réussi à identifier différentes démarches à adopter selon le type de test réalisé. Ils ont été conçus de telle manière à relever les différents problèmes de congestion qu'on peut rencontrer selon les schémas. On a par exemple pu identifier le câble coaxial comme un outil produisant des collisions. Ces tests ont également montré les faiblesses des outils utilisés. La fréquence d'échantillonage nous a posé problème sur Cacti et Nfsen lorsque nous avons voulu analyser un petit pic de trafic passager. Le fait de jongler entre les seuils de Storm control a également posé problème avec la fréquence d'échantillonnage qui a empêché des alertes de se déclencher lorsqu'elles l'auraient dû. Un des autres problèmes rencontrés à été la différence des résultats obtenus entre les tests Iperf et Nfsen qui prend en compte les en-têtes des paquets ce qui fausse légèrement les résultats à la hausse. Finalement, la congestion a certaines fois elle-même empêchée l'alerte de se déclencher puisqu'en engorgeant le trafic, les seuils d'alertes n'ont pas toujours réussi à être atteints.

Pour terminer cette conclusion, voici un petit récapitulatif des tests réalisés. Afin d'avoir un tableau comparatif pertinant, nous allons retirer les simulations qui ont testé le Storm control (test n°4 et 5) ainsi que les tests qui ne durent pas exactement une heure (test n°2).

Tableau 1 Récapitulatif de la phase de test

	Type de trafic	Nombre de tests simultan-nés	Bande passante totale	Bande passante souhaitée pour le test	Bande passante réellement utilisée	Perte de paquets ou retransmis-sions
Schéma n°1, test n°1 sans et avec boucle	TCP	1	1 Mbits	900 Kbits	900 Kbits	0.06 %
	UDP	1	1 Mbits	900 Kbits	900 Kbits	0 %
	TCP	2	1 Mbits	913 Kbits	787 Kbits	0.08 %
	UDP	2	1 Mbits	913 Kbits	819 Kbits	9 %
Schéma n°2, test n°3	TCP	1	10 Mbits	9 Mbits	9 Mbits	0.04 %
	UDP	1	10 Mbits	9 Mbits	9 Mbits	0 %
Schéma n°3, test n°6	TCP	1	10 Mbits	9 Mbits	7.87 Mbits	0.08 %
	UDP	1	10 Mbits	9 Mbits	9 Mbits	0 %
Schéma n°3, test n°7	UDP	2	10 Mbits	9 Mbits	8.68 Mbits	5.57 %

En rouge, ce sont les tests pour lesquels il y a eu de la congestion, en noir, ceux pour lesquels il n'y en a pas eu. Nous remarquons que lorsque nous lançons un seul test à la fois, c'est-à-dire qu'une seule source de trafic est présente dans le réseau, une congestion a peu de chance de se produire même si nous sommes à 90 % d'utilisation de la bande passante. Cependant, il y a une exception. En effet, l'utilisation de câbles coaxiaux génère facilement des collisions et peut ralentir le réseau lorsqu'il y a une unique communication TCP ou plusieurs communications simultanées. Si nous prenons comme exemple le test n°6 réalisé sur le schéma n°3, nous pouvons déduire que le câble coaxial déclenche son mécanisme de CSMA/CD lorsque TCP envoie ses acquittements (ACK) et créé des collisions, ce qui diminue le débit par seconde du test.

Ces tests nous montrent une chose essentielle, le débit maximum n'a pas besoin d'être atteint pour créer une congestion. Il y a beauocup de facteurs à prendre en compte comme le nombre de communications simultanées ou la technologie utilisée dans le réseau par exemple.

7. Conclusion

Ce travail étant maintenant terminé, il est temps d'en tirer des conclusions.

Tout d'abord, il faut savoir que tous les outils de monitoring que j'ai utilisés dans ce travail sont gratuits et parfois même, open source. Il devrait être relativement aisé de produire une solution plus poussée avec des logiciels payants (coûtant relativement cher), disposant d'outils plus avancés et de plus de fonctionnalités. L'idée générale de ce travail est surtout de voir l'approche à adopter pour résoudre un problème de congestion réseau sans y aborder un quelconque aspect financier. De plus, même si tous les outils sont gratuits, ils sont parfaitement adaptés pour le monitoring de petites ou moyennes entreprises.

Comme cela a été démontré dans la phase de test, il est important de procéder par étape. D'abord, détecter la congestion, puis, alerter du problème en cours afin de pouvoir l'analyser en vue d'y remédier. Il est également important de bien avoir connaissances des outils utilisés, le monde des réseaux étant un domaine très vaste, une confusion peut très vite arriver.

Concernant la phase de tests qui représente tout de même une grande partie de ce travail, j'ai essayé de montrer au mieux les différentes façons de détecter et d'alerter d'un problème de congestion selon les différentes technologies utilisées dans les différents schémas (lien série, switchs et câble coaxial). Cependant, les tests ne sont de loins pas exhaustifs, plusieurs cas n'ont pas pu être testés comme la simulation de trafic multicast ou broadcast. Il m'a également été impossible de simuler une bande passante asymétrique ou variable à cause des limites de l'outil Iperf.

La suite logique de ce travail sera bien entendu de définir les mesures correctives nécessaires afin de régler la congestion. J'ai déjà donné quelques pistes au début de ce travail à la rubrique 2.3. Il convient à tout un chacun d'adopter sa propre stratégie afin de combattre la congestion réseau. Ajouter des filtres de trafic, brider la bande passante, augmenter le débit, bref, il existe plusieurs mesures correctives qui peuvent être mises en place de manière automatique. Par exemple, les fournisseurs d'accès à Internet tels que Swisscom ou Sunrise utilisent la bande passante asymétrique pour diminuer les problèmes de congestion de leurs clients et au passage, minimiser leurs coûts. Vous voulez LA solution révolutionnaire ? Nous sommes en 2015, installez la fibre optique et ainsi, votre réseau sera à des années lumières de rencontrer à nouveau un problème de congestion !

7.1 Conclusion personnelle

J'ai eu vraiment beaucoup de plaisir à réaliser ce travail de Bachelor tout au long de ces trois derniers mois. Le fait d'avoir eu mon propre laboratoire personnel et un si grand nombre d'outils à disposition n'a fait qu'enrichir cette expérience.

Lorsque j'ai débuté ce travail, je ne connaissais aucun des outils qu'il m'a fallu utiliser. Mes seules connaissances étaient la configuration basique d'équipements Cisco ainsi que des notions moyennes sur les lignes de commandes Linux. Même si je me suis renseigné avant de commencer mon travail de Bachelor sur les technologies que j'ai dues utiliser, j'étais complètement perdu au début. Cependant, la soif d'apprendre a rapidement pris le dessus et c'est avec joie que je me suis mis à poser des questions et fouiller un peu partout sur le net et afin d'acquérir plus de connaissances pour aboutir à ma solution finale.

J'ai également pu développer un certain sens de la débrouillardise face aux diverses difficultés rencontrés que j'ai finalement réussi à surmonter. En dépit des heures passées à découvrir comment fonctionnait tel ou tel outil, je tire un bilan satisfaisant de ce travail qui m'a permis d'acquérir de nouvelles compétences dans ce domaine.

Bibliographie

Pages web

Le monde des réseaux. Combattre la congestion [en ligne]. http://www.gatoux.com/SECTION4/p5.php (Consulté du 7 au 18 septembre 2015)

Smart Report. Saturation réseau – Détecter les congestions réseau [en ligne]. http://www.smartreport.fr/supervision-reseau/livre-blanc/detecter-congestions-reseaux (Consulté du 7 au 18 septembre 2015)

Commentcamarche.net Encyclopédie. Qos Qualité de Service [en ligne]. http://www.commentcamarche.net/contents/532-qos-qualite-de-service (Consulté le 16 septembre 2015)

FrameIP. SNMP [en ligne]. http://www.frameip.com/snmp/ (Consulté du 23 au 25 septembre 2015)

Developpez.com. Présentation du protocole SNMP [en ligne]. http://ram-0000.developpez.com/tutoriels/reseau/SNMP/ (Consulté du 23 au 25 septembre 2015)

Developpez.com. Présentation du protocole Syslog [en ligne]. http://ram-0000.developpez.com/tutoriels/reseau/Syslog/ (Consulté le 1 octobre 2015)

igm.univ-mlv.fr. Analyse des performances d'un réseau grâce à IPSLA [en ligne]. http://www-igm.univ-mlv.fr/~dr/XPOSE2010/IPSLA/presentation.html (Consulté le 23 octobre 2015)

Sourceforge.net. Nfsen – Netflow sensor [en ligne]. http://nfsen.sourceforge.net/#mozTocId368170 (Consulté d'octobre à novembre 2015)

Pages Wikipédia

Network congestion. Wikipedia : the free encyclopedia [en ligne]. Dernière modification de la page le 3 Septembre 2015 à 13:26. [Consulté du 8 au 11 septembre 2015]. Disponible à l'adresse : https://en.wikipedia.org/wiki/Network_congestion

NetFlow. Wikipédia : L'encyclopédie libre [en ligne]. Dernière modification de la page le 8 Mars 2015 à 02:44. [Consulté le 2 octobre 2015]. Disponible à l'adresse : https://fr.wikipedia.org/wiki/NetFlow

Contrôle de flux. Wikipédia : L'encyclopédie libre [en ligne]. Dernière modification de la page le 27 Décembre 2014 à 16:11. [Consulté le 12 et 13 octobre 2015]. Disponible à l'adresse : https://fr.wikipedia.org/wiki/Contr%C3%B4le_de_flux

Cacti. Wikipédia : L'encyclopédie libre [en ligne]. Dernière modification de la page le 8 Décembre 2014 à 12:53. [Consulté le 22 et 23 octobre 2015]. Disponible à l'adresse : https://fr.wikipedia.org/wiki/Cacti

Path MTU discovery. Wikipédia : L'encyclopédie libre [en ligne]. Dernière modification de la page le 12 Juin 2015 à 08:58. [Consulté le 10 Novembre 2015]. Disponible à l'adresse : https://fr.wikipedia.org/wiki/Path_MTU_discovery

E-book

Djillali Seba, 2003. Préparation à la certification CCNA CISCO Installation, configuration et maintenance de réseaux [en ligne]. Edition ENI, France, Chapitre 18 pp. 381-383. [Consulté du 7 au 11 septembre 2015]. Disponible à l'adresse :

https://books.google.ch/books?id=pQBFN89ADYcC&pg=PA381&lpg=PA381&dq=congestion+de+trames&source=bl&ots=vdcDOIGcfy&sig=6G6EcNW7efhKX1tZbn9B64u_WrY&hl=fr&sa=X&ved=0CEMQ6AEwBmoVChMI8Om3rsjuxwIVirQaCh1j3AAr#v=onepage&q=congestion%20de%20trames&f=false [accès gratuit]