

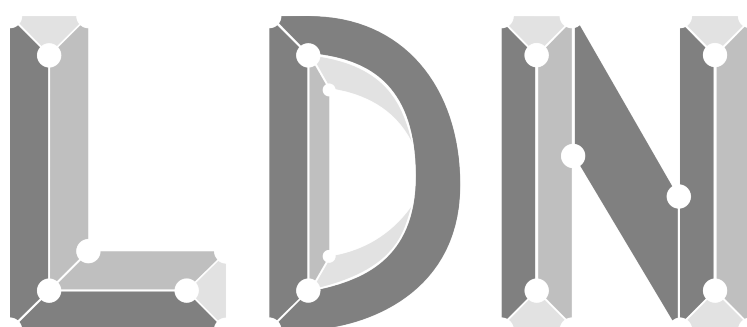
Projet Tuteuré

Marc REB, Romain BOUAJILA, Steffy FORT, Jugurta HENDEL

23 mars 2014

Supervision et contrôle de la consommation réseau des abonnés d'un FAI

Tuteurs : Sébastien JEAN et Julien VAUBOURG



LORRAINE DATA NETWORK



Table des matières

I	Présentation du projet	4
1	Introduction	5
2	Lorraine Data Network	6
2.1	Qu'est-ce que LDN et qui sont-ils ?	6
2.2	LDN face à la loi	6
2.3	Les services proposés	6
2.4	Wiki LDN	7
II	Partie technique	8
3	Reproduction réseau	9
3.1	Création des machines virtuelles	9
3.2	Mise en place des configurations réseau	9
3.2.1	Pour les machines virtuelles	9
3.2.2	Pour la machine Services	10
3.2.3	Pour la machine Router	10
3.2.4	Pour le PC en ASRALL	11
3.3	Évolution du schéma réseau	11
3.4	Automatisation	12
4	Zabbix	13
4.1	Présentation de la solution	13
4.2	Installation	13
4.2.1	Installation de la base de données MySQL	13
4.2.2	Installation des paquets	13
4.2.3	Édition de la configuration PHP de l'application web Zabbix	13
4.3	Installation de l'agent	14
4.4	Mise à jour Zabbix	14
4.5	Documentation pour LDN	15
5	95^{ème} centile	16
5.1	Présentation	16
5.2	Mise en place des graphiques automatiques pour le 95 ^{ème} centile	16
5.3	Récupération de la valeur	17
6	API Zabbix	18
6.1	Qu'est-ce que c'est ?	18
6.2	Utilisation de l'API	18
7	RRDTool	20
7.1	Qu'est-ce que c'est ?	20
7.2	Les vues utilisateur (Screen)	20
7.3	RRDTool	21
7.3.1	Requête MySQL	21
7.3.2	RRDTool fonctionnement	21
8	Limitation du débit	23
8.1	TC	23
8.2	Mise en place de script dans Zabbix	23
8.2.1	Prérequis	23
8.2.2	Création du script et lancement via Zabbix	24
8.3	Mise en place des déclencheurs automatiques	24
8.4	Les actions	26

9	Supervision du débit	28
9.1	Solution vnStat	28
9.1.1	Présentation de vnStat	28
9.1.2	Utilisation	28
9.1.3	Mise en place dans Zabbix	28
9.1.4	Les limites de vnStat	30
9.2	Solution de développement d'un sniffer	30
10	Icinga	31
10.1	Présentation de Icinga	31
10.2	Les différents composants d'Icinga	31
10.3	Installation d'Icinga-Core	31
10.3.1	Installation des pré-requis	31
10.3.2	Installation Icinga avec IDOUtils	31
10.3.3	Activation du module idomod	32
10.3.4	Autoriser les commandes externes (CGI)	32
10.3.5	Installation de Icinga Web	32
10.4	Addons officiels	33
10.4.1	NRPE :	33
10.4.2	NDOUtils :	33
10.5	Comparatif Icinga / Zabbix :	34
11	Conclusion	35
11.1	Conclusion générale	35
11.2	Expérience acquise	35
11.3	Remerciements	35
III	Annexes	36
12	Annexes	37
12.1	Webographie	37
13	Lexique	38
13.1	Schéma réseau simplifié	39
13.1.1	Premier schéma	39
13.1.2	Second schéma	40
13.2	Exemple xml de vm	41
13.3	Les différentes tables de routage	43
13.3.1	Vm0	43
13.3.2	Services	44
13.3.3	Router	45
13.3.4	PC ASRALL	46
13.4	Les nouvelles routes	47
13.4.1	Router	47
13.4.2	PC ASRALL	48
13.5	API Zabbix	49
13.5.1	Script Perl	49
13.5.2	Script PHP	51
13.6	RRDTool	51
13.6.1	Script	51
13.7	Documentation pour LDN	54
13.7.1	Installation de Zabbix serveur et agent	54
13.7.2	Configuration/création des graphiques prototypes	54
13.7.3	Configuration/création des déclencheurs prototypes	54
13.8	Image de configuration de l'action	56
13.9	Script Sniffer	57

Partie I

Présentation du projet

1 Introduction

Dans le cadre des projets tuteurés de la licence professionnelle ASRALL, nous avons eu pour objectifs de concevoir une interface permettant d'informer en temps réel les abonnés d'un FAI à propos de leur consommation de bande passante ainsi de pouvoir leur permettre de réduire cette dernière. L'association Lorraine Data Network est un FAI (fournisseur d'accès à Internet) mais aussi un FSI (fournisseur de services Internet). Le but de l'association est d'offrir des services aux utilisateurs pour défendre sa vision du réseau.

C'est en effet une association à but non-lucratif, pour la défense d'un Internet libre, neutre et décentralisé. La facturation du transitaire étant basée sur le débit, l'usage fait de leur ligne par les adhérents a un impact direct sur celle-ci. L'objectif est donc de pouvoir permettre aux usagers de prendre conscience de leur impact sur le réseau, en fonction des différents services souscrits.

Nous avons donc dû recréer le réseau en production de manière minimaliste, de façon à pouvoir l'utiliser comme plateforme de test. Pour ce faire nous avons installé Debian Wheezy sur 2 machines, la première servant de routeur, et la seconde hébergeant les différents services. Il fallait ensuite une solution pour calculer les débits différenciés à partir d'une interface réseau, d'un vhost ou d'un service pour les représenter sous forme de graphiques (RRD) et indiquer le 95^{ème} centile correspondant à l'abonné. Les tuteurs nous ont orientés sur la solution Zabbix. Les utilisateurs devront ensuite être en mesure de les consulter avec un accès limité (vues) sur le serveur Zabbix.

Le projet devra également être capable de lever des alertes à partir de seuils et éventuellement de restreindre le débit de l'interface / du vhost / du service.

2 Lorraine Data Network

2.1 Qu'est-ce que LDN et qui sont-ils ?

Lorraine Data Network (LDN) est une association de type loi 1901. Elle n'a donc aucun client, uniquement des adhérents, auxquels certains services peuvent être proposés (un accès ADSL, par exemple), contre compensation financière. Les statuts qui définissent l'association sont ainsi : *«L'association a pour but : la promotion, l'utilisation et le développement des réseaux Internet dans le respect de leur éthique en favorisant en particulier les utilisations à des fins de recherche ou d'éducation sans volonté commerciale.»*



En n'ayant aucune volonté commerciale, Lorraine Data Network propose une alternative, au modèle économique actuel des FAI classiques, en s'engageant à respecter les libertés fondamentales de l'Internet. En s'invitant dans des débats où seuls les principaux acteurs étaient jusqu'alors conviés, LDN milite aux côtés des autres associations (ALTERN, La Quadrature du Net, FDN, etc.) pour défendre nos libertés sur Internet. En s'invitant dans un maximum d'événements de la région, LDN a pour but de sensibiliser la population lorraine aux enjeux de la neutralité des réseaux et des conséquences de son abandon progressif.

Actuellement, l'association compte une soixantaine de passionnés et bénévoles qui prennent sur leur temps libre pour faire avancer l'association. Ces derniers sont aussi en majorité des passionnés d'Internet et d'informatique, mais pas uniquement. L'association compte énormément de soutiens de personnes qui n'ont aucun lien direct avec l'informatique, mais qui sont soucieuses de leurs libertés. Ils sont donc des informaticiens, des étudiants, des professeurs, etc. De tout âge et de tout horizon, mais partageant le même idéal.

2.2 LDN face à la loi

LDN n'accepte que les requêtes judiciaires, qui passent par les procédures telles qu'elles sont écrites dans la loi, sans aucune sorte de compromis possible. Ce strict respect des règles serait rarement une politique suivie par les gros opérateurs, qui auraient plutôt tendance à livrer des informations sur un simple coup de téléphone des autorités.

Quelles que soient les obligations légales auxquelles l'association est soumise en tant que fournisseur d'accès à Internet, elle ne peut garantir à terme qu'une totale transparence vis-à-vis des adhérents ainsi qu'une mise au vote de toutes les décisions qui peuvent être soumises à contestation. Si ces obligations posent un problème vis-à-vis de l'éthique ou de la neutralité ou des libertés du Net, LDN fera son possible pour les combattre, à vos côtés et aux côtés des autres entités militantes.

2.3 Les services proposés

LDN propose de nombreux services. Bien entendu en tant que FAI, l'association propose une connexion à l'Internet en fournissant l'accès à une dizaine de lignes ADSL. Certains services comme l'hébergement ou bien la fibre ne sont pas encore opérationnels mais sont en cours de développement. Voici globalement la liste des services disponibles :

- **VPN**, permet d'établir un tunnel chiffré entre votre ordinateur/téléphone/tablette et un serveur de LDN. Ainsi, quelle que soit la connexion Internet utilisée, vous récupérez vos IP habituelles et vous pouvez profiter d'un accès à Internet neutre, non filtré et sécurisé. Livré avec un bloc IPv6 /48 et une adresse IPv4 fixes.
- **Espace web mutualisé**, vous envoyez vos fichiers sur votre espace de stockage, et votre site web est directement accessible. Par exemple, avec des solutions comme WordPress, ce service est idéal pour proposer un blog en ayant peu de connaissances en informatique. Espace accessible en SFTP/FTPS/FTP avec une base de données MySQL et un vhost Apache avec PHP5. Espace de départ de 2Go. Utilisation d'un ou plusieurs noms de domaines personnels ou d'un ou plusieurs sous-domaines gratuits de acteurdu.net ou altu.fr.
- **Pack adhérent** (avec courriels), lot de services divers, de la solution e-mail complète et clé-en-main au DNS. Services basiques (courriels, jabber, etc.). Espace de départ pour les courriels de 1Go. Nombre d'adresses non limité, consultables via IMAP(S) et webmail, et utilisant un ou plusieurs noms de domaines personnels ou un ou plusieurs sous-domaines gratuits de acteurdu.net ou altu.fr.
- **Serveur virtuel dédié** (VPS), vous êtes maître de votre système GNU/Linux, tout en profitant du confort d'un hébergement en datacenter. Il n'y a quasiment aucune contrainte, mais vous êtes responsable de tout.

Système Debian. Installation personnalisée. Espace de départ à 30Go avec 256Mo de mémoire. Livré avec un bloc IPv6 /56 et une IPv4 fixes. Possibilité d'utiliser un ou plusieurs sous-domaines gratuits de acteurdu.net ou altu.fr pour les services accessibles dessus.

2.4 Wiki LDN

Pour nous aider à réaliser le projet, notamment pour la reproduction la plus proche possible du réseau que celui qui est mis en production, les tuteurs nous ont donné un accès complet à la documentation établie par l'association (<https://wiki.ldn-fai.net>)

Partie II

Partie technique

3 Reproduction réseau

Pour une meilleure compréhension du réseau et du travail fourni par LDN, ainsi qu'une mise en situation la plus proche possible de l'environnement actuellement en production dans l'association, les tuteurs nous ont proposé d'effectuer une reproduction simplifiée de leur infrastructure (voir schéma disponible au point 13.1.1). Bien entendu, nous avons tout de suite accepté car cela allait nous permettre de pouvoir travailler sur un aspect purement réseau que nous avons peu abordé au cours de la licence, ainsi que la découverte de l'IPv6 que nous voyons peu en cours ainsi que sur des réseaux en production.

Malheureusement, au départ nous n'avions pas compris l'importance de la mise en place des routes pour le reste du sujet, et face à de la création des machines virtuelles, surtout pour la partie de la configuration réseau, nous avons pris l'initiative de mettre en place du NAT sur le routeur. Après notre première réunion avec les tuteurs, qui nous ont expliqué l'intérêt de la mise en place des routes, nous avons donc fait le maximum pour récupérer l'erreur que nous avons commise.

3.1 Création des machines virtuelles

La création des machines virtuelles a été l'une des tâches les plus compliquées pour cette partie, notamment sur le plan réseau. En effet nous n'avions pas encore commencé les cours sur l'aspect réseau des machines virtuelles, énormément de choix qui s'offraient à nous lors de la création de ces dernières et nous ne savions laquelle de ces solutions pouvait répondre à nos attentes. Nous avons donc demandé aux tuteurs qui nous ont indiqué que LDN utilisait une technique un peu spéciale. Dans un premier temps, ils créent une machine virtuelle (grâce à KVM/QEMU) avec les configurations réseau par défaut, connexion en bridge, puis la sortent du bridge pour lui attribuer une adresse IPv6 manuellement.

Pour commencer il faut préalablement installer les paquets qui nous seront utiles, `apt-get install libvirt-bin virtinst qemu-kvm debootstrap`, ils serviront entre autres à la création de la machine virtuelle ainsi que sa gestion grâce à un terminal. Dans l'exemple qui suit nous allons créer une machine virtuelle appelée `vm1` fonctionnant sur une distribution Debian Wheezy. Dans l'ordre nous allons donc : créer un disque de stockage, ajouter un système de fichiers, installer la distribution et enfin ajouter la machine à libvirt pour la gestion (avec `virsh start` ou bien l'utilitaire `libvirt-manager`).

```
root@services $ cd /var/lib/libvirt/images/
root@services $ #création du disque
root@services $ dd if=/dev/zero of=vm1.img bs=1M seek=4000 count=1
root@services $ #mise en place d'un système de fichiers
root@services $ mkfs.ext3 vm1.img
root@services $ #on monte l'image puis on installe Debian
root@services $ mkdir /mnt/vm1; mount -o loop vm1.img /mnt/vm1
root@services $ debootstrap wheezy /mnt/vm1 http://ftp.debian.org/debian/
root@services $ #penser à mettre un mot de passe puis démonter le disque
root@services $ chroot /mnt/vm1 passwd
root@services $ umount vm1.img
root@services $ #installation de la machine dans libvirt avec l'ajout temporaire dans le bridge
root@services $ virt-install --import --disk path=/var/lib/libvirt/images/vm1.img --vcpus=1 --ram=256 --name=vm1 --hvm --vnc --network bridge:virbr0
```

3.2 Mise en place des configurations réseau

3.2.1 Pour les machines virtuelles

Suite à notre erreur de compréhension du schéma, expliquée précédemment, les machines virtuelles étaient en bridge avec des configurations réseau assez spéciales. Ne comprenant pas comment et de quelle façon les utiliser pour pouvoir mettre les cartes réseaux sur les VM, nous avons demandé aux tuteurs qui sont venus nous aider. Nous avons donc commencé par prendre un exemple des fichiers xml actuellement en production chez LDN (fichiers de configuration de machines virtuelles pour libvirt) pour l'adapter à nos machines virtuelles grâce à la commande `virsh edit <nomVm>` (voir exemple de xml dans le point 13.2).

Pour commencer nous devons faire sortir nos machines virtuelles de notre bridge grâce à la commande `ip link set dev vnetXX nomaster`. Pour vérifier qu'elles sont correctement sorties du bridge, utiliser la commande `brctlshow`.

Comme dit précédemment, les interfaces réseau des machines virtuelles sont sorties du bridge, elles apparaissent donc comme de nouvelles interfaces sans configuration réseau. Il faut donc commencer par leur en ajouter. Pour cela il faut bien distinguer les deux côtés des interfaces réseau des machines. Dans un premier temps, il y a les interfaces **vnetXX** disponibles sur le serveur, **services**, puis celles disponibles dans les VM, **eth0**. Les interfaces de type **vnetXX** n'ont pas besoin de configuration particulière, néanmoins il faut quand même leur ajouter une IP (de type IPv6 dans notre cas) pour que les paquets depuis la machine virtuelle puissent être routés sur ces interfaces. Pour cela, on ajoute l'adresse IPv6 **fe80::42/64**, pour un aspect purement pratique car tout les interfaces disposent déjà d'une adresse **fe80**, grâce à l'outil IP : `ip a a fe80::42/64 dev vnetXX`. En IPv6 le réseau **fe80::** est dédié aux liens locaux.

Ensuite, nous avons besoin d'ajouter les IP des machines virtuelles ainsi que les routes pour qu'elles puissent communiquer avec le faux internet. Celui-ci sera tout simplement une adresse IP qui représentera l'Internet mondial pour notre configuration. Les configurations qui suivent seront faites pour la **vm0** dont le hostname est **vps0** sur le schéma réseau (voir point 13.1.1). Pour communiquer avec le faux internet, nous devons faire router tous les paquets de **vps0** vers l'interface **eth1** de la machine **services**, qui se chargera par la suite de les router correctement vers le faux internet.

```
root@vm0 $ #Ajout des adresses ip
root@vm0 $ ip a a 172.16.91.1/32 dev eth0
root@vm0 $ ip a a fc01:42::1/64 dev eth0
root@vm0 $ #On force la route pour la destination de l'interface réseau eth1 de la machine services, puis
on met cette adresse en temps que route par défaut
root@vm0 $ ip r a 172.16.90.41/32 via eth0
root@vm0 $ ip r a default via 172.16.90.41
root@vm0 $ #On effectue ensuite la même opération pour l'IPv6
root@vm0 $ ip r a default via fe80::42 dev eth0
```

Une fois les routes en place on pourra les vérifier grâce à la commande `route -n`, pour IPv4, et `route -n -6`, pour IPv6. Vous trouverez un exemple des tables de routage de **vm0** sur le point 13.3.

3.2.2 Pour la machine Services

La machine **services** est la machine qui dispose du plus de modifications réseau à effectuer du fait qu'elle héberge les machines virtuelles. De plus, nous lui avons ajouté, pour le bon déroulement du TP, une interface **eth0** qui nous permet une communication constante avec le réseau de la classe et l'Internet. Pour la bonne communication entre tout ces réseaux, nous devons : ajouter les routes pour répondre aux requêtes à destination des VM, ajouter les routes pour le faux internet, mettre en place le forwarding. Par défaut, le forwarding des interfaces réseau ne s'effectue pas dans le noyau Linux, mais il nous sera utile pour transférer les requêtes à destination des VM qui arriveront sur **eth1**. Pour cela :

```
root@services $ #Activation du forwarding IPv4 et IPv6
root@services $ sysctl -w net.ipv4.conf.all.forwarding=1 && sysctl -w net.ipv6.conf.all.forwarding=1
root@services $ #Mise en place des configurations réseau IPv4 et IPv6 et l'interface eth1
root@services $ ip a a 172.16.90.41 dev eth1
root@services $ ip a a fc00::1:2/112 dev eth1
root@services $ #Ajout des routes pour la vm0 puis pour la vm1
root@services $ ip r a 172.16.91.1/32 dev vnet0
root@services $ ip r a fc01:42::/64 dev vnet0
root@services $ ip r a 172.16.91.2/32 dev vnet1
root@services $ ip r a fc01:1337::/64 dev vnet1
root@services $ #Ajout de la route pour le faux internet
root@services $ ip r a 10.200.0.0/16 via 172.16.90.40
root@services $ ip r a fcee::/16 via fc00::1:1
```

Les configurations réseau pour **eth0** (route par défaut et adresse IP) seront automatiquement mises en place par le DHCP de la classe. Une fois les routes en place, on pourra les vérifier grâce à la commande `route -n`, pour IPv4, et `route -n -6`, pour IPv6. Vous trouvez un exemple des tables de routage sur le point 13.3.2.

3.2.3 Pour la machine Router

La machine **router** est celle qui va servir pour effectuer le routage des paquets entre le réseau des VM et le faux internet. L'ajout des adresses réseau permettra de mettre automatiquement la majorité des routes qui nous seront nécessaires. Il suffira donc d'ajouter les routes pour le réseau de VM et une pour le faux internet, qui se servira d'une des IP du PC en ASRALL, ainsi que le forwarding.

```
root@router $ #Ajout des IP IPv4 et IPv6 sur eth0
root@router $ ip a a 10.42.0.3/31 dev eth0
root@router $ ip a a fcab::2/112 dev eth0
root@router $ #Même chose sur eth1
root@router $ ip a a fc00::1:1/112 dev eth1
root@router $ ip a a 172.16.90.40/31 dev eth1
root@router $ #Activation du forward IPv4 IPv6
root@router $ sysctl -w net.ipv4.conf.all.forwarding=1 && sysctl -w net.ipv6.conf.all.forwarding=1
root@router $ #Ajout des routes pour le réseau de vm
root@router $ ip r a fc01::/16 via fc00::1:2
root@router $ ip r a 172.16.91.0/24 via 172.16.90.41
root@router $ #Ajout des routes pour le faux internet
root@router $ ip r a fcee:dead:babe::/64 via fcab::1
root@router $ ip r a 10.200.42.0/24 via 10.42.0.2
```

Une fois les routes en place, on pourra les vérifier grâce à la commande `route -n`, pour IPv4, et `route -n -6`, pour IPv6. Vous trouverez un exemple des tables de routage sur le point 13.3.3.

3.2.4 Pour le PC en ASRALL

Le PC en ASRALL est la station qui va représenter le faux internet. Pour cela nous avons tout simplement ajouté une IP qui n'appartenait à aucun des réseaux utilisés dans notre infrastructure. Il suffisait tout simplement d'ajouter 4 nouvelles adresses IP, deux IPv4/IPv6 pour la connexion avec le routeur et deux autres pour le faux internet, à l'interface eth1 de notre station, deux routes pour la connexion au VM et enfin l'activation du forwarding.

```
root@asrall $ #Ajout des ip pour la connexion au routeur
root@asrall $ ip a a 10.42.0.2/31 dev eth1
root@asrall $ ip a a fcab::1/112 dev eth1
root@asrall $ #Ajout des ip du faux internet
root@asrall $ ip a a 10.200.42.1/32 dev eth1
root@asrall $ ip a a fcee:dead:babe::1/128
root@asrall $ #Ajout de route pour les vm
root@asrall $ ip route a 172.16.0.0/16 via 10.42.0.3
root@asrall $ ip route a fc00::/12 via fcab::2
```

Une fois les routes en place, on pourra les vérifier grâce à la commande `route -n`, pour IPv4, et `route -n -6`, pour IPv6. Vous trouvez un exemple des tables de routage sur le point 13.3.4.

3.3 Évolution du schéma réseau

Pour pouvoir effectuer plus en profondeur les objectifs, ainsi que la ressemblance maximale par rapport au réseau en production, les tuteurs nous ont fait évoluer notre schéma réseau (voir nouveau schéma point 13.1.2).

Le peering et le transit représentent deux types de routage utilisés par les FAI en général, mais leur différence est uniquement commerciale. En effet l'interrogation de routes n'est pas gratuite en règle générale, car le transitaire payera le coût de la demande, au niveau du transfert de paquets de la demande, de cette dernière dont il ne dispose pas, cela passe donc sur sa consommation au 95^{ème} centile, c'est ce que l'on appelle le transit. Néanmoins il existe des groupes de peers, des associations dans la plupart du temps, qui mettent à disposition leurs propres routes et prennent à leur charge le coût de l'interrogation en étant reliés directement entre eux, c'est ce qu'on appelle le peering.

De ce fait nous avons donc mis en place deux vlan, eth0.800 pour représenter le transit et eth0.267 pour représenter le peering, et une connexion directe entre **Router** et le PC en **ASRALL**. Pour des moyens pratiques nous avons changé la station qui était auparavant le PC **ASRALL**, donc dans notre cas nous repartons de zéro pour cette dernière, dans le cas contraire il aurait fallu penser à supprimer toutes les adresses ajoutées préalablement avec la commande `ip`. Aucune modification ne sera à faire sur la machine **services**, puisque tous les paquets à destination du réseau 10.200.0.0\16 seront routés en direction de **Router** via **eth1**, idem pour **vps0** et **vps1** qui redirigeront tout vers **Services**. Petite particularité à savoir, puisqu'il n'est pas possible au PC **ASRALL** de disposer de deux routes avec une destination identique, il faut rediriger toutes les réponses en direction des VM uniquement par une seule interface vlan. En l'absence de vnet, qui pourrait contourner le problème, nous devons permuter les routes à la main pour les simulations.

```
root@asrall $ #on stop l'interface eth0
root@asrall $ ifdown eth0
root@asrall $ #On ajoute les vlan dans le fichier interfaces et on commente eth0
root@asrall $ vim /etc/network/interfaces
12 auto eth0.800
13 iface eth0.800 inet static
14 address 10.42.0.2
15 netmask 255.255.255.254
16
17 auto eth0.267
18 iface eth0.267 inet static
19 address 10.43.0.2
20 netmask 255.255.255.254

root@asrall $ #On up les deux interfaces vlan
root@asrall $ ifup eth0.800 && ifup eth0.267
root@asrall $ #Plus qu'à donner les bonnes configurations réseau pour eth0.800 pour commencer
root@asrall $ ip a a 10.200.42.1/32 dev eth0.800
root@asrall $ ip a a fcab::1/112 dev eth0.800
root@asrall $ ip a a fcee:dead:babe::1/128 dev eth0.800
root@asrall $ #Idem pour eth0.267
root@asrall $ ip a a 10.200.43.1/32 dev eth0.267
root@asrall $ ip a a fcac::1/112 dev eth0.267
root@asrall $ ip a a fcff:dead:beef::1/128 dev eth0.267
root@asrall $ #Ajout des routes pour communiquer avec les VM. Dans ce cas ci tout passera par eth0.800
root@asrall $ ip r a fc00::/12 via fcab::2
root@asrall $ ip r a 172.16.0.0/16 via 10.42.0.3
root@router $ #On stop eth0 sur router maintenant
root@router $ ifdown eth0
root@router $ #On ajoute les vlan dans le fichier interfaces et on commente eth0
root@router $ vim /etc/network/interfaces
12 auto eth0.800
13 iface eth0.800 inet static
14 address 10.42.0.3
15 netmask 255.255.255.254
16
17 auto eth0.267
18 iface eth0.267 inet static
19 address 10.43.0.3
20 netmask 255.255.255.254

root@router $ #On ajoute l'IPv6 sur eth0.800 et eth0.267
root@router $ ip a a fcab::2/112 dev eth0.800
root@router $ ip a a fcac::2/112 dev eth0.267
root@router $ #Ajout des routes pour le transit
root@router $ ip r a fcee:dead:babe::/64 via fcab::1
root@router $ ip r a 10.200.42.0/24 via 10.42.0.2
root@router $ #Idem pour le peering
root@router $ ip r a fcff:dead:beef::/64 via fcac::1
root@router $ ip r a 10.200.43.0/24 via 10.43.0.2
```

Vous trouverez un exemple des nouvelles tables de routage en annexes point 13.4.

3.4 Automatisation

Bien que les configurations soient correctement mises et opérationnelles, elles n'en sont pas pour autant persistantes. Pour cela, il suffit de créer un script sur chaque machine pour remettre les configurations réseau, et le démarrage des machines virtuelles pour **services**, au redémarrage. Nous avons donc juste besoin de recopier les commandes effectuées sur chaque machine pour les stocker dans un script bash que nous avons appelé `/etc/init.d/route.sh`. Ensuite nous avons ajouté l'exécution du script dans `/etc/init.d/rc.local`.

4 Zabbix

4.1 Présentation de la solution

Zabbix est un logiciel de monitoring open source sous licence GPL créé par Alexei Vladishev. Ce logiciel permet de superviser des réseaux, et de surveiller les statuts de différents services, systèmes et réseaux.

Un serveur Zabbix peut être décomposé en trois parties. Tout d'abord, l'application est composée d'une partie données, avec notamment l'utilisation d'un serveur de base de données (MySQL dans notre cas), permettant de stocker les informations sur les paramètres des hôtes, des événements. Ensuite, il y a un serveur de traitement Zabbix Server, gérant les différents outils de supervision et de surveillance. Et pour finir, l'interface web pour configurer et administrer Zabbix.

4.2 Installation

4.2.1 Installation de la base de données MySQL

Pour commencer, nous devons installer la base de données. Pour cela il suffit d'installer le paquet `mysql-server` mais attention pas besoin de créer de base de données. En effet l'installation des paquets Zabbix créeront eux-mêmes la bonne base, néanmoins pensez à démarrer votre service si ce n'est pas le cas avec la commande `service mysql start`.

Vous pouvez aussi modifier le paramètre dans le fichier `/etc/mysql/my.conf` pour mettre l'encodage des caractères en UTF-8 :

```
[mysqld]
default-character-set=utf8
```

4.2.2 Installation des paquets

L'installation d'un serveur Zabbix sur Debian reste très simple. En effet la communauté Zabbix a simplifié au maximum en créant un fichier `.deb` disponible sur leur dépôt. Pour déployer le service il nous suffit donc de télécharger le paquet `.deb` puis de l'installer avec un `dpkg -i`. Ce paquet va tout simplement installer les sources, ensuite nous pourrons lancer l'installation des paquets `zabbix-server-mysql` `zabbix-frontend-php` avec un simple `apt-get`.

```
root@services $ wget http://repo.zabbix.com/zabbix/2.0/debian/pool/main/z/zabbix-release/zabbix-
release_2.0-1wheezy_all.deb
root@services $ dpkg -i zabbix-release_2.0-1wheezy_all.deb
root@services $ apt-get update
root@services $ apt-get install -y zabbix-server-mysql zabbix-frontend-php
```

4.2.3 Édition de la configuration PHP de l'application web Zabbix

Le fichier de configuration Apache pour l'application web Zabbix est dans le répertoire `/etc/apache2/conf.d/zabbix`. Certains paramètres PHP sont déjà configurés.

```
php_value max_execution_time 300
php_value memory_limit 128M
php_value post_max_size 16M
php_value upload_max_filesize 2M
php_value max_input_time 300
# php_value date.timezone Europe/Riga
```

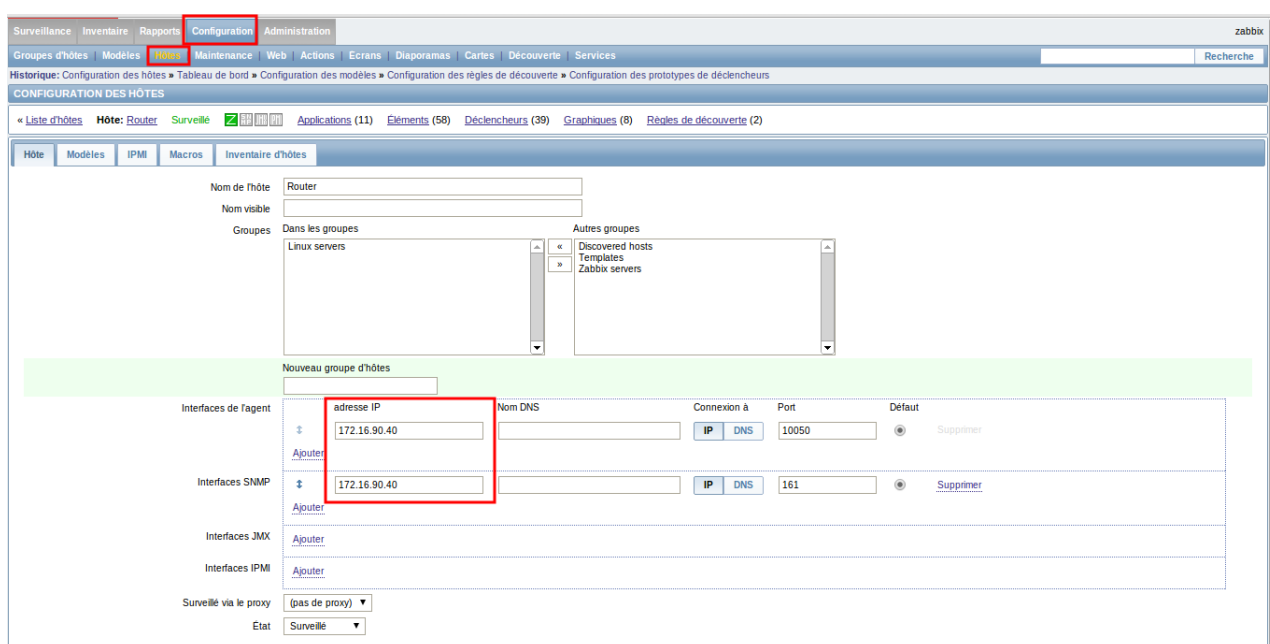
On peut ensuite passer à la configuration de Zabbix via un navigateur à l'adresse `http://<ipDuServeur>/zabbix`, les mots de passe par défaut étant `admin/zabbix`.

4.3 Installation de l'agent

Comme pour Nagios, Zabbix dispose d'un agent qui permet de remonter de multiples informations. Son installation reste très simplifiée, elle aussi, et est semblable à celle du serveur. Il suffit de télécharger le fichier `.deb` sur la machine cliente, de le l'installer avec `dpkg`, pour l'ajout des bonnes sources, puis de lancer un `apt-get upgrade` suivi de l'installation du paquet `zabbix-agent`. Enfin il faut indiquer à l'agent l'adresse IP du serveur pour l'autoriser à remonter des informations dans le fichier `/etc/zabbix/zabbix_agentd.conf` vers la ligne 86. Dans notre cas nous avons choisi l'installation sur la machine routeur :

```
root@router $ wget http://repo.zabbix.com/zabbix/2.0/debian/pool/main/z/zabbix-release/zabbix-release 2.0-1wheezy all.deb
root@router $ dpkg -i zabbix-release 2.0-1wheezy all.deb
root@router $ apt-get update
root@router $ apt-get install zabbix-agent
root@router $ vim /etc/zabbix/zabbix_agentd.conf +86
84 # Server=
85
86 Server=172.16.90.41
87
88 ### Option: ListenPort
```

Une fois l'installation finie, il faut déclarer le client à votre serveur Zabbix. Le plus simple reste de cloner votre serveur Zabbix, déjà présent dans `configuration - hôtes`, et de simplement changer le nom de l'hôte ainsi que son adresse IP, puis de sauvegarder. Exemple dans l'image qui suit :



4.4 Mise à jour Zabbix

Pour pouvoir effectuer des tests avec certaines solutions, nous avons dû passer notre serveur Zabbix de la version 2.0 à la version 2.2. En effet lors de l'installation avec le `.deb`, ce dernier ne vous met pas les sources les plus récentes. Avant toute intervention sur votre serveur il est fortement conseillé d'effectuer des sauvegardes de votre solution, pour cela il suffit de faire un `cp -r` de tout le dossier `/etc/zabbix` et une sauvegarde de la base de données (avec la commande `mysqldump` pour les base sous MySQL).

Une fois les sauvegardes effectuées, il suffit de modifier le fichier des sources Zabbix et de mettre à jour les paquets. Dans notre cas nous nous contenterons de tous les mettre à jour, et pas uniquement ceux de la solution. Lors de l'opération, il faudra regarder le log `zabbix_server.log` avec un `tail -f` pour voir si elle s'effectue correctement. Il se peut qu'un `apt-get update && apt-get upgrade` ne suffise pas. Dans ce cas là il faudra effectuer directement un `apt-get dist-upgrade`.

```
root@services $ #Modification du fichier sources
root@services $ vim /etc/apt/sources.list.d/zabbix.list
1 deb http://repo.zabbix.com/zabbix/2.2/debian wheezy main
2 deb-src http://repo.zabbix.com/zabbix/2.2/debian wheezy main

root@services $ #Mise à jour des sources et des paquets
root@services $ apt-get update
root@services $ apt-get upgrade
root@services $ #Vérification de l'évolution de la mise à jour dans les logs
root@services $ tail -f /var/log/zabbix/zabbix_server.log
root@services $ #Si la mise à jour ne s'effectue pas correctement tentez à nouveau avec un dist-upgrade
root@services $ apt-get dist-upgrade
root@services $ tail -f /var/log/zabbix/zabbix_server.log
```

Lorsque les logs affichent correctement la progression de la mise à jour, vous pourrez vérifier en retournant sur l'interface web de Zabbix que tout est correctement opérationnel.

4.5 Documentation pour LDN

Pour alimenter le wiki de LDN et établir une documentation sur l'installation de Zabbix et de l'agent, les tuteurs nous ont demandé une documentation technique simplifiée. Vous pourrez le retrouver en annexe au point 13.7.

5 95^{ème} centile

5.1 Présentation

La mesure du 95^{ème} centile est utilisée par les opérateurs sur internet pour facturer la consommation de bande passante à leurs clients. L'unité de mesure du Mbps consommé est la plus souvent utilisée par les hébergeurs envers leurs clients.

Cette mesure correspond à l'usage de bande passante utile pendant 95% du temps sur la période choisie pour le calcul.

Dans cette perspective, l'association LDN souhaiterait évaluer la consommation de chaque abonné et ainsi leur offrir la possibilité de choisir leur tarification en fonction de leur consommation.

5.2 Mise en place des graphiques automatiques pour le 95^{ème} centile

La solution open source Zabbix offre énormément d'options de supervision, parmi elles la création automatique de graphiques. Chaque graphique devra être créé lors de la découverte d'un nouvelle interface. Les services de Zabbix étant répartis en différents modèles (Template), nous avons utilisé celui lié par défaut à chaque nouvel hôte. Celui-ci regroupe toutes les informations sur l'OS de la machine (Template OS Linux). La génération de ces graphiques ou Graph prototype s'effectue à chaque détection d'un nouvel hôte. Ces graphiques sont disponibles dans le menu Supervision/Graphique de la page d'accueil. Selon notre configuration, les graphiques générés affichent l'évolution du débit entrant et/ou sortant de chaque interface réseau active de l'hôte. La configuration de ces graphiques est plutôt simple (voir image suivante).

The screenshot shows the Zabbix web interface for configuring a new graph. The breadcrumb trail is: Historique: Graphiques personnalisés » Tableau de bord » Graphiques personnalisés » Configuration des hôtes » Configuration des graphiques. The page title is CONFIGURATION DES GRAPHIQUES. The navigation bar includes: Liste d'hôtes, Hôte: Zabbix server, Surveillance, Applications (11), Éléments (71), Déclencheurs (41), Graphiques (14), Règles de découverte (2). The main content area has tabs for Graphique and Aperçu. The Graphique tab is active, showing configuration options for a graph named '95 centile'. The options include: Nom (95 centile), Largeur (900), Hauteur (200), Type de graphique (Normal), Afficher la légende (checked), Afficher les heures ouvrées (checked), Afficher les déclencheurs (checked), Ligne de centile (gauche) (95.00), Ligne de centile (droite) (95.00), Valeur minimale axe Y (Calculé), and Valeur MAX axe Y (Calculé). Below these options is a table of elements with columns: Nom, Fonction, Style de dessin, Coté de l'axe Y, Couleur, and Action. The table contains one entry: 1: Zabbix server: Incoming network traffic on eth0, moy, Ligne, Gauche, C80000, and Supprimer. At the bottom are buttons: Sauver, Clone, Supprimer, and Annuler.

Pour la déclaration d'un nouveau graphique, il y a à disposition au minimum douze variables configurables.

Propriété	Type de variable	Description
Id Graphique	string	(Lecture seule) Id unique du graphique prototype.
Nom	string	Nom du graphique
Largeur	entier	Largeur en pixel du graphique prototype.
Hauteur	entier	Hauteur de pixel du graphique prototype.
Type de graphique	entier	Quatre types de graphiques sont disponibles. Les valeurs possibles : 0- (Normal par défaut), 1- (empilé): Permet de placer les données bout à bout pour comparer la contribution de chaque série à la somme des valeurs, 2- (barre de secteur): Secteur d'un graphique éclaté dans un graphique en colonne empilé, 3- (éclaté) Secteur séparé des autres.
Vue 3D	entier	Il est possible de créer des graphiques en 3D mais cette option n'est disponible qu'avec des graphiques de type secteur ou éclaté. La valeur par défaut est 0 (2D) pour passer en 3D la valeur 1 doit être passée.
Afficher la légende	entier	Activer ou désactiver la légende du graphique : valeur 0 pour cacher et 1 pour activer.
Temps de travail	entier	Activer ou désactiver l'affichage du temps de travail sur les graphiques.
Afficher les déclencheurs	entier	Activer (1) ou désactiver (0) les déclencheurs liés aux graphiques.
Centile gauche	float	Valeur du centile sur l'axe gauche par défaut la valeur est à 0. Nous utilisons cette valeur pour nos calculs avec la valeur 95.
Centile droit	float	Valeur du centile sur l'axe de droite par défaut la valeur est à 0. Nous utilisons cette valeur pour nos calcul avec la valeur 95.
Valeur minimal axe Y	float	Définir une valeur minimale d'affichage de l'axe Y sur le graphique. Nous utilisons la valeur calculée automatiquement pour obtenir un graphique dynamique qui change d'échelle pour une meilleure visualisation du débit entrant et sortant.
Valeur maximal axe Y	float	Définir une valeur maximale d'affichage de l'axe Y sur le graphique. Nous utilisons la valeur calculée automatiquement pour obtenir un graphique dynamique qui change d'échelle pour une meilleure visualisation du débit entrant et sortant.
Éléments	-	Choix des différentes interfaces à utiliser pour la réception des données et la création du graphique. Nous utilisons les interfaces réseau et leur débit entrant et sortant pour créer nos graphiques.

Un aperçu est disponible pour évaluer le résultat de la configuration de notre graphique et ainsi modifier les valeurs de mise en forme pour obtenir une visualisation parfaite. Il est également possible de cloner un graphique déjà créé au préalable et ainsi récupérer une configuration identique.

5.3 Récupération de la valeur

Bien que nous affichons maintenant la ligne du 95^{ème} centile, il nous faudrait maintenant trouver le moyen de l'exploiter. Malheureusement après de nombreuses recherches, il semblerait que cela ne soit pas possible depuis Zabbix. Nous avons néanmoins trouvé une manière de l'obtenir, mais pour cela il aurait fallu appliquer un patch, puis compiler l'installation du service Zabbix. Les tuteurs ne voulaient pas partir sur une solution compilée. Le patch est disponible à l'adresse suivante : <https://support.zabbix.com/browse/ZBXNEXT-756>.

6 API Zabbix

6.1 Qu'est-ce que c'est ?

Zabbix met à disposition un outil intéressant pour obtenir des informations sur les différents hôtes supervisés. Cet outil est une API (Application Programming Interface). L'API peut être utilisable sous différents langages de programmation, le PHP (avec la technologie Jsonrc), le Perl, le Ruby, le Python... Son fonctionnement en PHP est simple il suffit d'envoyer un script PHP par une méthode POST et le navigateur affiche la réponse dans une page web. Son fonctionnement en Perl est aussi très similaire, nous devons créer un script et de le lancer pour obtenir une réponse dans le shell. Utiliser l'API dans un langage de scripting peut être une bonne alternative si il n'y a pas de navigateur web à disposition.

Cet outil, puissant, permet d'étendre les fonctionnalités de Zabbix pour le développement d'applications ou pour l'intégration avec des logiciels tiers.

6.2 Utilisation de l'API

L'API de Zabbix permet d'obtenir une multitude d'informations concernant un hôte ou un composant, mais il est également possible de récupérer des informations sur les utilisateurs, leurs différentes vues sur l'application mais aussi les scripts qui leurs sont liés.

Nous utilisons l'API dans l'optique d'arriver à créer un déclencheur qui avertirait l'utilisateur qu'il dépasse sa limite du 95^{ème} centile défini. Pour cela nous avons recherché un moyen de récupérer la valeur du centile grâce aux méthodes de l'API.

Nous nous sommes vite rendu compte que la valeur du centile était recalculée par Zabbix à chaque rechargement du graphique, mais n'était pas stockée dans la base de données, donc inutilisable. A moins de récupérer les valeurs du débit sur les graphiques grâce à la méthode de l'API et de recalculer le 95^{ème} centile à la main, il était impossible de le récupérer via l'API.

Voici un tableau résumant les différents points contrôlables avec l'API :

Catégories	Propriété	Description
Surveillance	Historique	Permet de récupérer les valeurs recueillies par les processus de surveillance de Zabbix comme la présentation de la machine surveillée et les traitements effectués. Pour obtenir ces valeurs il faut utiliser la méthode suivante : "history.get".
	Événements	Récupère les événements générés par les déclencheurs, la découverte réseau et d'autres systèmes de Zabbix. Pour la récupération des valeurs des événements, il faut utiliser les méthodes suivantes : "event.get" et "event.acknowledge".
	Services	Récupérer les données concernant les informations de disponibilité sur une machine. Pour cela il faut utiliser la méthode : "service.getsla".
Configuration	Hôtes et groupe d'hôtes	Gérer les hôtes, les groupes d'hôtes et tout ce qui touche à leur disposition y compris les interfaces, macros et les périodes de maintenance. Des méthodes à retenir : "host.get", "host.update" et "host.massupdate".
	Déclencheurs	Gérer et configurer les déclencheurs pour informer des problèmes dans le système. Permet aussi de gérer les dépendances de déclenchement. Quelques méthodes importantes : "trigger.adddependencies", "trigger.get" et "trigger.update".
	Graphiques	Modifier les graphiques ou récupérer les valeurs travaillées par celui-ci. Deux méthodes importantes pour la récupération de valeur et de mise à jour des graphes : "graph.get" et "graph.update".
	Templates	Gérer les templates et les relier à des hôtes ou d'autres templates. Les méthodes permettant ces actions : "template.update" et "template.massupdate".
	Découverte de bas niveau	Configurer les règles de découverte de bas niveau ainsi que les déclencheurs et graphiques prototypes. Méthode pour la gestion des graphiques prototypes : "graphprototype.get" et "graphprototype.update".
	Screen (vue utilisateur)	Modification global des screens ou d'un élément particulier de celui-ci. Méthode de modification globale : "screen.update".
Administration	Utilisateurs	Permet l'ajout, la modification, la suppression et la mise à jour des utilisateurs de Zabbix. Méthodes importantes : "user.update", "user.login/logout", "user.addmedia" et "user.get".
	Scripts	Configurer et exécuter des scripts pour aider dans les tâches de supervision. Méthode : "script.get", "script.update" et "script.execute".

Un aperçu d'un script de l'API est disponible en annexe, voir point 13.5.

7 RRDTool

7.1 Qu'est-ce que c'est ?

RRDTool (Round-Robin Database Tool) est un outil de gestion de base de données RRD créé par Tobias Oetiker. Il est utilisé par de nombreux outils, tels que Nagios ou encore Cacti pour la sauvegarde de données et le tracé des graphiques. Cet outil est très utile pour superviser la bande passante ou la température d'un processeur. Le principal avantage d'une base RRD est sa taille fixe.



Nous avons utilisé cet outil dans l'optique de recréer les graphiques de Zabbix pour par la suite mettre en place des vues personnalisées pour chaque utilisateur.

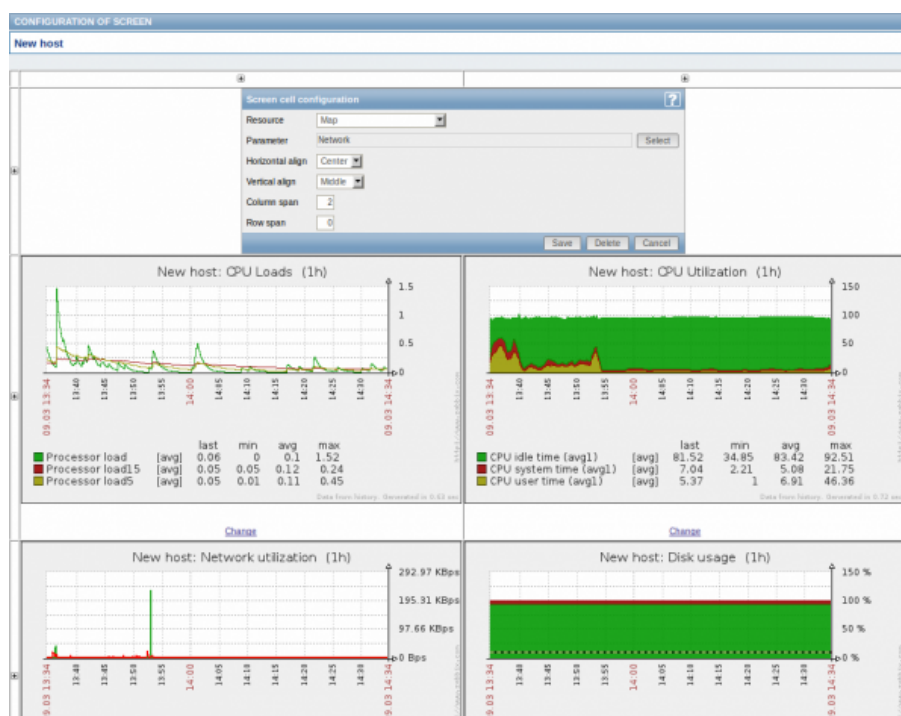
7.2 Les vues utilisateur (Screen)

Zabbix met à disposition un système de vue pour chacun de ses utilisateurs. Ce système, Screen, permet de créer un tableau de bord composé d'éléments définis par l'administrateur. Ces éléments peuvent être très variés, insérer un graphique indiquant la courbe d'évolution de l'activité du CPU de la machine de l'utilisateur par exemple.

Un des objectifs du projet tuteuré étant de trouver un moyen de restreindre la vision de l'utilisateur sur Zabbix à sa seule machine, cet outil semblait remplir les différents critères.

Malheureusement, le plan de notre infrastructure ne convenait pas au bon fonctionnement de cet outil. En effet, n'ayant qu'une seule machine physique, regroupant un grand nombre de machines virtuelles, nous ne pouvions installer l'agent Zabbix uniquement sur cette dernière. Il est impossible de donner accès uniquement à un seul composant (la machine virtuelle) à l'utilisateur. Ce problème s'explique par le fait que le serveur Zabbix est enregistré dans un groupe d'hôtes. Les permissions de chaque utilisateur ne peuvent être reliées qu'à des groupes d'hôtes, nous donnerons accès aux vues des autres machines virtuelles à chaque utilisateur.

Voici un aperçu d'un Screen de Zabbix :



7.3 RRDTool

7.3.1 Requête MySQL

Pour pouvoir intégrer RRDTool dans l'outil de supervision Zabbix nous avons choisi de créer un script en Perl et de l'appeler par la suite lors d'une alerte d'un déclencheur par un script lié à celui-ci.

Au préalable nous avons fait quelques recherches sur la base de données de Zabbix pour localiser l'endroit où sont stockées les valeurs du débit des graphiques.

Voici les quelques requêtes SQL que nous avons effectuées pour la localisation :

```
SHOW COLUMNS FROM graphs;
```

Cette requête permet de retourner toutes les colonnes qui composent la table des graphiques pour nous permettre ainsi d'identifier plus facilement le contenu.

```
SELECT graphid FROM graphs WHERE name='95 centile ...';
```

Cette requête permet de filtrer l'identifiant du graphique du 95 centile.

```
SELECT itemid FROM graphs_items WHERE graphid = 549;
```

Nous sélectionnons l'identifiant de l'item du graphique du 95 centile pour ensuite récupérer les valeurs stockées dans la table item.

```
SELECT * FROM history_uint WHERE itemid=23331; SELECT value FROM history_uint WHERE  
itemid = 23331 AND clock=(SELECT MAX(clock) FROM history_uint WHERE itemid=23331);
```

La première requête permet de voir toutes les valeurs contenues dans la table history_uint. Cette table représente la source de toutes les données de type numérique.

La seconde requête permet de récupérer la valeur du débit pour le graphique avec l'identifiant d'item numéro 23331 et on applique un filtre sur cette valeur pour obtenir la dernière en date. On peut noter que la date est en timestamp unix, c'est un format de date qui représente le nombre de secondes écoulées depuis le 1^{er} janvier 1970 à 0h00:00. Il est possible de récupérer cette valeur de date grâce à la ligne de commande suivante :

```
user@localhost# date +%s
```

7.3.2 RRDTool fonctionnement

Le fonctionnement de RRDTool se tient en trois étapes. Tout d'abord la création d'une base de données vide ainsi que la définition des configurations du graphique. Dans un second temps, une mise à jour des données des informations du graphique. Puis enfin la génération du graphique en précisant la taille de celui-ci afin que les différentes opérations s'effectuent sur les valeurs de la base de données.

```
rrdtool create test.rrd --start 1393426052 DS:test:GAUGE:600:U:U RRA:AVERAGE:0.5:1:24
```

Cette première ligne de commande permet de créer la base de données appelée ici "test.rrd". Nous définissons aussi une date de début pour la génération du graphique avec le paramètre "--start" ici sa valeur est de 1393426052 équivalent au 26 février 2014 à 14h47:32. Le paramètre DS signifie Data Source on lui passe en paramètre le nom de la base de données que l'on souhaite ici "test" ainsi que le type de valeur qu'elle va contenir, ici GAUGE spécifie une valeur entrée par l'utilisateur tel qu'un nombre. Le paramètre RRA tant qu'à lui définit une archive qui se compose d'un certain nombre de valeurs de données ou des statistiques pour chacune des sources de données définies par le DS.

```
rrdtool update test.rrd 1297810801:0:0 1297811101:5:1 ...
```

Cette seconde ligne de commande permet la mise à jour des données de la base. Nous passons en paramètre le nom de la base à cibler par la modification ainsi que les valeurs les unes après les autres. Les valeurs sont définies comme ceci : date-en-timestamp:valeur-x:valeur-y.

```
rrdtool graph test.png -s 1393426052 -e 1393426947 -h 300 -w 600 -t "Graphe de test"  
DEF:test=test.rrd:test:AVERAGE LINE3:test#FF0000:"TEST"
```

Cette dernière ligne de commande permet la génération du graphique RRD. On définit un nom pour le graphique, la base de données à utiliser dans DEF ainsi que sa hauteur et sa largeur avec les paramètres "-h" et "-w". Le paramètre "-t" permet de définir une légende pour le graphique.

Vous trouverez en annexe les différents scripts que nous avons effectués pour nos tests.

8 Limitation du débit

Pour que les utilisateurs puissent contrôler leurs débits, il était prévu initialement de pouvoir mettre en place un système d'alerte, suivant la valeur du 95^{ème} centile, puis de permettre à l'utilisateur de réduire sa consommation de sa propre initiative ou automatiquement. Bien entendu, en tant que serveur de monitoring, Zabbix comprend initialement un système d'alerte automatique. Malheureusement la limite du débit n'est pas incluse à la solution, il a fallu donc s'orienter vers un outil système, `tc`, puis utiliser cet outil grâce à un script depuis l'agent Zabbix.

8.1 TC

`Tc` est un programme en ligne de commande sous Linux qui permet de gérer son trafic réseau. Il est régulièrement utilisé pour l'une de ses capacités très appréciée qui est la prise en charge du QoS (Quality of Service) : c'est la gestion prioritaire de flux, qui permet également de limiter le débit réseau sur une interface. C'est cette fonctionnalité-là que nous allons utiliser.

Le programme permet de créer des conteneurs, aussi appelés gestionnaires ou `qdisc`, qui traiteront les paquets avant leur arrivée sur la carte réseau et décideront à quel moment ces paquets seront transmis sur celle-ci. Dans l'exemple qui va suivre nous utiliserons les conteneurs HTB, Hierarchical Token Bucket.

C'est donc de cette façon que vont être traités les paquets pour brider notre carte réseau, suivant nos conteneurs et nos configurations, `tc` décidera si le paquet est légitime ou non d'être transmis à la carte réseau. La grande faiblesse de cet outil est qu'il contrôle uniquement ce qui sort des interfaces. Il faut donc passer par un routeur, ou bien par des interfaces virtuelles si l'on veut effectuer le contrôle entrant et sortant.

Pour contrôler les flux entrants sur une interface (`eth1` dans notre exemple), il faut procéder de la manière suivante :

```
root@services $ #Création d'un conteneur (ou qdisc) htb sur eth1
root@services $ tc qdisc add dev eth1 root handle 1:0 htb default 1
root@services $ #Création d'une classe fille au conteneur parent avec la limitation de débit, 800000 dans
l'exemple qui suit, attention valeur en bit
root@services $ tc class add dev eth1 parent 1:0 classid 1:1 htb rate 800000
```

On peut ensuite vérifier que la classe est bien créée grâce à la commande `tc -s class show dev eth1`

```
root@services $ tc -s class show dev eth1
class htb 1:1 root prio 0 rate 800000bit ceil 800000bit burst 1600b cburst 1600b
Sent 9718 bytes 135 pkt (dropped 0, overlimits 0 requeues 0)
rate 1920bit 3pps backlog 0b 0p requeues 0
lended: 135 borrowed: 0 giants: 0
tokens: 197141 ctokens: 197141
```

Une fois que tous les tests sont ok, on peut supprimer la classe avec la commande `tc class del dev eth1 parent 1:0 classid 1:1 htb rate 800000`, puis le conteneur avec `tc qdisc del dev eth1 root handle 1:0 htb default 1`.

8.2 Mise en place de script dans Zabbix

Maintenant que l'on a correctement utilisé `tc` pour limiter le débit, nous pouvons tenter de le mettre en place dans un script Zabbix. Pour que l'utilisateur final interagisse le moins possible sur le serveur, on va créer un script bash qui prendra en argument l'interface réseau de la machine virtuelle ainsi que le débit désiré, en kilobit, pour la restriction de débit.

8.2.1 Prérequis

Pour pouvoir lancer un script depuis Zabbix il faut impérativement disposer de l'agent Zabbix, au cas où il n'est pas installé se référer au point 4.3, ainsi que le paquet `sudo` installé (`apt-get install -y sudo`). Ensuite vous pouvez ajouter un shell à l'utilisateur Zabbix. En effet lors de l'installation de l'agent, un utilisateur Zabbix se crée,

mais avec un shell du type `/bin/false`. On peut donc éditer le fichier `passwd` (`vim /etc/passwd`), pour y ajouter un `bash` restreint (`rbash`). Ensuite nous allons donner des droits `sudo` à l'utilisateur pour qu'il puisse lancer la commande `tc`. Enfin on va devoir autoriser l'agent à lancer les scripts dans le fichier de configuration de l'agent, `/etc/zabbix/zabbix_agentd.conf`. Pour effectuer ces modifications :

```
root@services $ #Modification du passwd
root@services $ vim /etc/passwd
25 zabbix:x:107:112::/var/lib/zabbix:/bin/rbash

root@services $ #On donne les droits sudo pour tc
root@services $ visudo
21 zabbix ALL=NOPASSWD: /sbin/tc

root@services $ #Ajout du droit d'utilisation de scripts par l'agent et on le redémarre
root@services $ vim /etc/zabbix/zabbix_agentd.conf
62 # Default:
63 EnableRemoteCommands=1

root@services $ /etc/init.d/zabbix-agent restart
```

8.2.2 Création du script et lancement via Zabbix

Comme dit précédemment, nous allons maintenant pouvoir créer le script en `bash` et le lancer sur le serveur Zabbix. L'objectif du script était de pouvoir laisser le choix à l'utilisateur du débit désiré, en kilobit, et d'auto-compléter la partie sur le choix de l'interface réseau sur laquelle cette restriction va s'effectuer. Malheureusement, malgré de nombreuses recherches sur la documentation Zabbix, il semblerait qu'il n'était pas possible de pouvoir récupérer la valeur de l'interface réseau courante sur laquelle le déclencheur se met en marche. Nous avons donc pris la liberté de mettre en argument l'interface réseau désirée. Voici le script final qui reste donc très simple :

```
root@services $ mkdir /etc/zabbix/scripts && vim /etc/zabbix/scripts/monscript.sh
1 #/bin/bash
2 debit=$((2 * 1024))
3
4 sudo /sbin/tc qdisc add dev $1 root handle 1:0 htb default 1
5 sudo /sbin/tc class add dev $1 parent 1:0 classid 1:1 htb rate $debit

root@services $ chmod +x /etc/zabbix/scripts/monscript.sh
```

Une fois le script créé, nous allons définir le déclencheur automatique pour l'action pour lancer le script à partir de Zabbix.

8.3 Mise en place des déclencheurs automatiques

Il est possible de configurer un déclencheur qui va agir sur plusieurs unités. Par exemple une valeur de consommation limite d'un hôte, une activité/inactivité, etc.. À l'image des graphiques prototypes, il existe la possibilité de créer des déclencheurs qui se lieront automatiquement à un équipement dès sa découverte. L'image suivante est la fenêtre de création et de configuration des déclencheurs prototypes.

Pour la déclaration d'un nouveau graphique, il y a à disposition six variables configurables.

Propriété	Type de variable	Description
Nom	string	Nom du déclencheur prototype, il est possible d'utiliser la variable {#IFNAME} dans le nom du déclencheur. Cette variable permet l'affichage dynamique de l'interface réseau que l'on visualise.
Expression	-	L'expression est l'aspect qui va déclencher l'alerte. Il est possible de définir des alertes pour n'importe quelle valeur retournée par l'agent Zabbix de l'hôte et les autres protocoles. Définition des expressions (voir image suivante)
Constructeur d'expression	lien	Le constructeur est un outil pour créer des liens entre les différentes expressions sélectionnées dans la zone précédente.
Génération d'évènements problème multiples	entier	Activer ou désactiver permet l'affichage de l'ensemble des problèmes relevés par le déclencheur lors d'une alerte.
Description	string	Utilisation de ce champ pour définir l'utilisation du déclencheur ainsi que l'action apportée et la solution/explication de l'alerte.
URL	lien	Ajouter un lien vers une solution pour le problème de l'alerte.
Sévérité	entier	Six valeurs pour classer le déclencheur suivant une hiérarchie de danger : 0- Non classé (sévérité neutre, un évènement qui n'a pas de grade de danger), 1- Information (Grade informatif, le déclencheur donne une information supplémentaire sur l'action en cours), 2-Avertissement (Grade bas niveau, le danger n'est pas important mais doit être consulté et résolu), 3-Moyen (Danger modéré, le système n'est pas en danger mais pourrait le devenir), 4-Haut (Danger important, le système est peut être en danger, problème à résoudre vite), 5-Désastre (Système détérioré).
Activé	-	Active/désactive le déclencheur.

Les déclencheurs actifs sont indiqués dans le menu Supervision/Tableau de bord ou dans Supervision/Déclencheurs. Pour clôturer un déclencheur, il faut l'acquitter. Pour cela il est nécessaire de fournir un descriptif de la solution pour pouvoir le valider dans Zabbix.

Propriété	Type de variable	Description
Éléments	-	Ayant choisi de créer un déclencheur prototype il faut donc sélectionner un autre objet prototype pour effectuer un lien propre qui permettra la création automatique de l'ensemble. Voir image suivante pour les différents éléments disponibles.
Fonction	-	Différentes fonctions sont disponibles, parmi elles nous utilisons la suivante : La dernière (plus récente) valeur T est \geq N.
Dernier (T)	entier	Définition en seconde du temps d'intervalle entre chaque T pour la comparaison des deux valeurs. Il y a deux types d'intervalles disponibles, le type par défaut en seconde ou le type compte qui permet de définir le nombre de valeurs récupérées avant l'opération. Par exemple, si il est renseigné 1 dans Compte, l'opération s'effectuera à toutes les valeurs retournées.
Décalage temporel	entier	Définition du temps de décalage en seconde entre chaque T.
N	entier	Valeur de comparaison à définir.

Nom	Clé	Type	Type d'information	État
Incoming network traffic on {#IFNAME}	net.if.in[{#IFNAME}]	agent Zabbix	Numérique (non signé)	Activé
Outgoing network traffic on {#IFNAME}	net.if.out[{#IFNAME}]	agent Zabbix	Numérique (non signé)	Activé
Total network traffic on {#IFNAME}	net.if.total[{#IFNAME}]	agent Zabbix	Numérique (non signé)	Activé

Dans la fenêtre de choix de l'élément, on peut choisir tous les éléments disponibles sur l'hôte et sur le modèle lié au déclencheur. Ici pour notre configuration nous avons choisi Incoming network traffic on {#IFNAME} et Outgoing network traffic on {#IFNAME} correspondant au débit entrant et sortant de l'interface.

8.4 Les actions

Une fois votre déclencheur mis en place, il va falloir créer une action pour que la restriction du débit via `tc` s'effectue. Cela peut se faire très simplement depuis l'interface web de Zabbix.

Pour mettre en place l'action, nous allons aller dans "configuration", puis dans "actions". Une fois dans cette page, sélectionner le bouton en haut à gauche "Créer une action". Une nouvelle page s'ouvre composée de trois onglets pour configurer votre action. Pour commencer, sélectionner l'onglet "Action" où nous allons tout simplement donner un nom, "Limite débit eth1" dans notre cas. Penser bien évidemment à cocher "Activé" pour que l'action soit mise en place.

Ensuite il y a "Condition". Cet onglet représente la condition que le serveur doit avoir pour pouvoir lancer l'action. Nous allons donc choisir la condition reliée au déclencheur créé précédemment qui va permettre de lancer l'action dès qu'il sera activé. Vous pouvez voir un exemple dans l'image qui suit.

Conditions	Étiquette	Nom	Action
(A)		Déclencheur = router: TEST detect 95e centile eth1	Supprimer

Nouvelle condition

Nom du déclencheur ▼ comme ▼ Ajouter

Sauver Clone Supprimer Annuler

Enfin il faut mettre l'action qui va être effectuée lorsque la condition sera remplie. Pour cela nous devons aller dans l'onglet "Opération". Dans cette partie nous pouvons configurer l'action avec les configurations qui suivent, dans cet exemple le bridage se fera sur la carte réseau eth1 pour une valeur maximum de 800Kb/s (voir l'image de l'exemple sur le point 13.8) :

- Type d'opération : Commande à distance
- Liste des cibles : router
- Type : Script personnalisé
- Exécuté sur : agent Zabbix
- Commandes : `/etc/zabbix/scripts/tc.sh eth1 800`

Vous pouvez désormais sauvegarder et votre bridage s'effectuera au lancement de votre déclencheur.

9 Supervision du débit

9.1 Solution vnStat

9.1.1 Présentation de vnStat

VnStat est un utilitaire console, qui fonctionne sous les noyaux Linux et BSD, permettant de surveiller et de journaliser la consommation de bande passante sur les interfaces réseau de l'hôte courant. L'un de ses avantages, que nous souhaitons utiliser ici, est de pouvoir sortir les résultats de la bande passante en fonction de leur interface et de leur destination (transit ou peering).

9.1.2 Utilisation

Pour utiliser **vnStat** il faut tout d'abord lui lister toutes les interfaces qu'il devra surveiller. Ces interfaces seront ajoutées sous la forme d'une base de données dans le répertoire `/var/lib/vnstat`. Une fois ajoutées, les interfaces sont immédiatement surveillées par **vnStat**. On peut ensuite afficher les statistiques générées en fonction d'une échelle de temps : heure, jour, semaine ou mois. Cette dernière nous semblait intéressante car les abonnés reçoivent leur facturation mensuellement. Nous pouvons voir l'utilisation de cet outil dans l'exemple qui suit, avec l'interface `eth0.800`.

```
root@router $ # Création d'un base sur eth0.800
root@router $ vnstat -u -i eth0.800
root@router $ # Afficher les statistiques de consommation sur eth0.800 sur un mois
root@router $ vnstat -i eth0.800 -m
eth0.800 / monthly
month rx | tx | total | avg. rate
-----+-----+-----+-----
Feb '14 1 KiB | 387 KiB | 388 KiB | 0.00 kbit/s
Mar '14 1.05 MiB | 2.15 MiB | 3.20 MiB | 0.02 kbit/s
-----+-----+-----+-----
estimated 1 MiB | 3 MiB | 4 MiB |
```

9.1.3 Mise en place dans Zabbix

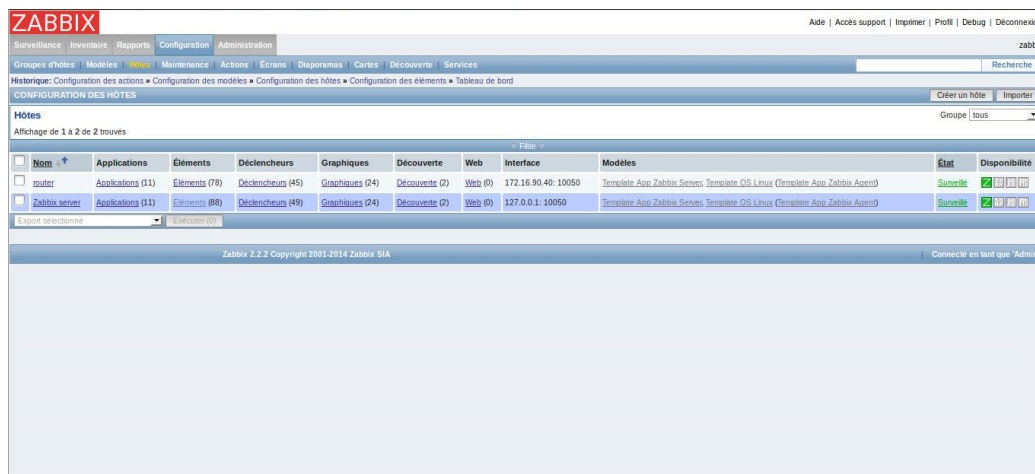
Avant de mettre en place le script pour qu'il soit exécuté par l'agent, il faut donner les droits **sudo** pour la commande **vnstat** à l'utilisateur Zabbix et autoriser l'agent à exécuter les scripts. Vous pouvez retrouver la procédure sur le point 8.2.1, elle est bien entendu à adapter.

De base, Zabbix permet d'ajouter des scripts personnalisés afin de palier au manque de fonctionnalités de l'agent. Zabbix accepte donc plusieurs langages de programmation comme Ruby, Perl, PHP, ou encore Python.

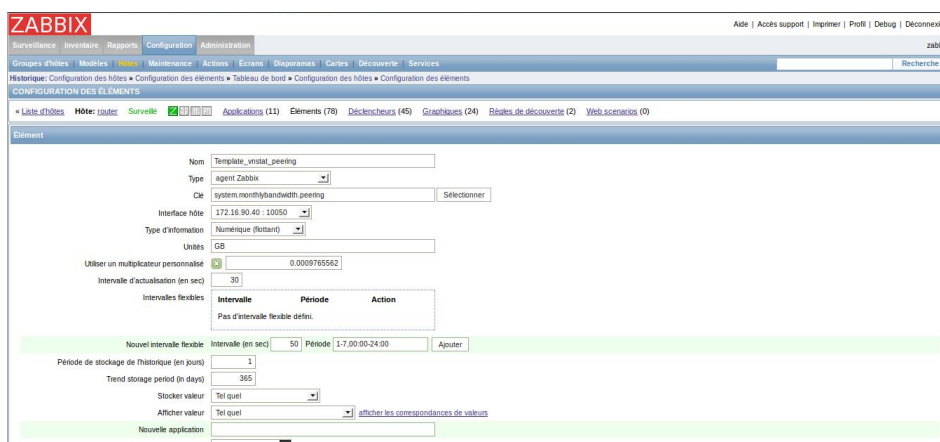
Nous avons commencé par placer le script, fait en bash, dans le répertoire `/etc/zabbix/scripts` que nous avons nous-mêmes créé. Puis nous avons précisé son emplacement dans le fichier `/etc/zabbix/zabbix_agentd.conf` dans l'espace réservé aux paramètres de supervision définis par les utilisateurs : `UserParameter=system.monthlybandwidth,/etc/zab`

Nous avons ajouté un élément sur l'hôte router dans le menu Configuration de Zabbix. Ensuite nous devons préciser une clé qui correspond à celle indiquée précédemment dans le fichier `zabbix_agentd.conf`.

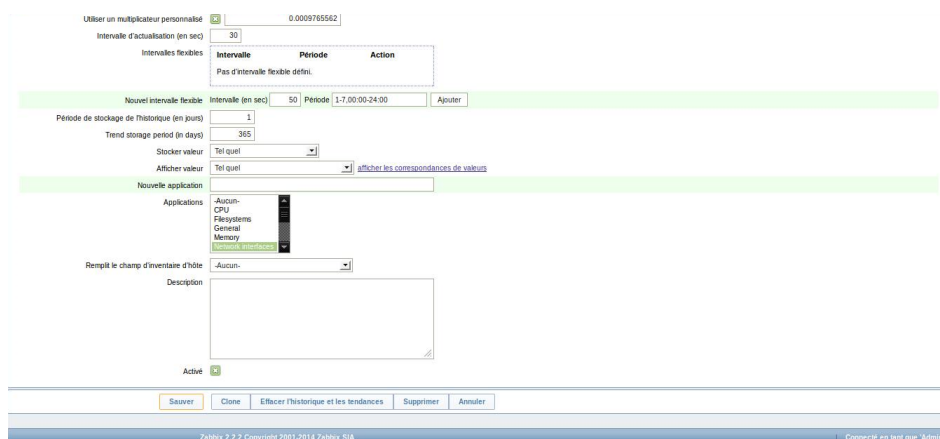
Nous avons ensuite créé un nouveau graphique que nous avons lié à l'élément ajouté précédemment. Pour intégrer le script bash à Zabbix, il faut se rendre dans l'onglet Hôtes du menu Configuration. On clique ensuite sur la partie Éléments de la ligne correspondant à la machine Router. On peut ensuite créer un nouvel élément.



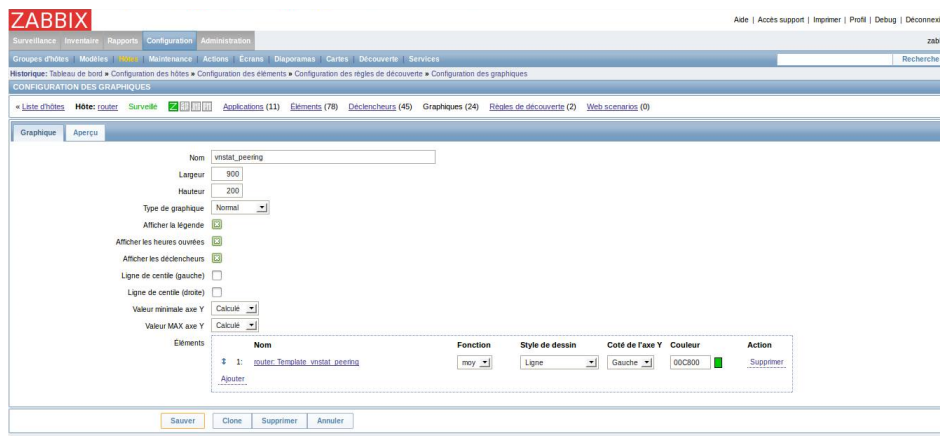
Une fois dans le menu de création d'un nouvel élément, plusieurs champs sont à renseigner. Le premier est le champ "Nom" et permet d'attribuer au nouvel élément un nom pour le distinguer des autres. Dans le champ Clé, on précise le paramètre personnalisé que l'on a ajouté au fichier `/etc/zabbix/zabbix_agentd.conf` précédemment. On choisit l'interface que l'on veut surveiller, ici l'interface eth1. On choisit ensuite le type de valeur retournée par Zabbix puis l'unité de mesure désirée.



On peut ensuite sélectionner Network Interfaces comme application liée au nouvel élément.



Il ne reste plus qu'à lier le nouvel élément créé à un nouveau graphique grâce à la partie Éléments de la page Configuration des graphiques.



9.1.4 Les limites de vnStat

Malgré beaucoup de fonctionnalités intéressantes, la solution vnStat couplée à TC n'était malheureusement pas possible. En effet, L'outil vnStat ne permet qu'une visualisation de statistiques sur la quantité de données ayant transité sur l'interface sélectionnée. Il n'était donc pas possible de surveiller le débit sur une interface réseau.

L'outil **vnStat** ne pouvant pas faire la distinction sur l'adresse IP source des paquets, il fallait contourner le problème, ce que nous pouvions faire en les marquant grâce à **iptables**. Seulement il n'est pas possible non plus d'effectuer une surveillance en fonction de ces marques. Nous avons donc dû nous tourner vers une autre solution plus adaptée à nos besoins.

9.2 Solution de développement d'un sniffer

Nous avons utilisé la librairie Net::Pcap pour développer un sniffer qui nous permettrait d'obtenir les informations sur les paquets transitant et les paquets en peering sur le routeur.

Pcap ou Packet capture est une interface de programmation permettant de capturer toutes les informations d'un trafic réseau. Elle est implémentée sous les systèmes GNU/Linux, BSD et Mac OS X par la bibliothèque libpcap.

Le sniffer récupère les informations des paquets sur les interfaces virtuelles d'eth0, eth0.800 et eth0.267. Sur ces paquets, nous relevons les adresses sources et de destinations ainsi que la taille des paquets en octets. Nous récupérons ces informations dans l'optique de faire un tri des paquets par machine virtuelle.

Pour effectuer ce tri, une comparaison entre les adresses IP sources reçues et la plage d'IP utilisée par les machines virtuelles est effectuée. Grâce à cela il est désormais possible d'identifier les paquets entrants et les paquets sortants de l'infrastructure.

La fonction de la librairie Pcap de Perl permettant la capture des paquets s'effectuant dans une boucle infinie et ne pouvant capturer qu'un seul trafic d'interface, nous avons créé deux processus qui supervisent chacun de leur côté une des interfaces virtuelles de eth0 sur le routeur. Nous profitons de ces deux processus pour indiquer également par quelle interface est passé le paquet (eth0.800 ou eth0.267) pour différencier transit et peering. Nous faisons aussi afficher les informations du paquet par ces deux processus, adresse IP source et taille du paquet en octets.

Vous trouverez le script du sniffer en annexe à ce point 13.9.

10 Icinga

10.1 Présentation de Icinga

Icinga est un fork de Nagios rétro-compatible. Ainsi, les configurations, les plugins et add-ons de Nagios peuvent tous être utilisés avec Icinga. Bien qu'Icinga conserve toutes les caractéristiques existantes de son prédécesseur, il s'appuie sur celles-ci pour ajouter de nombreux correctifs et fonctionnalités longtemps attendues et demandées par la communauté des utilisateurs.

10.2 Les différents composants d'Icinga

Icinga-Core :

Le noyau Icinga permet le suivi des tâches et la réception des résultats de vérification depuis de nombreux plugins. Il communique ensuite les résultats à IDODB via l'interface IDOMOD, puis le service IDO2DB chiffre ces données au travers du SSL. Bien que l'interface IDOMOD et le service IDO2DB (précédemment connu sous le nom IDOUtils) soient ensemble avec le noyau, se sont des composants indépendants pouvant être séparés afin de distribuer l'information à un ensemble de machines.

Icinga-Web :

Icinga-web est une interface web permettant de consulter les alertes de supervision ainsi que d'envoyer des commandes au noyau Icinga. Vous pouvez consulter, en temps réels, l'état des hôtes et services, l'historique des alertes, les notifications et la carte des disponibilités afin de garder un contrôle sur la santé de votre infrastructure. Il est possible d'intégrer la génération de graphiques de performances dans les rapports grâce à PNP ou GrapherV2. L'interface web, développée en Ajax, est flexible et personnalisable grâce au "glisser-déposer" de composants appelés cronks. Pour plus d'informations, vous pouvez consulter l'interface.

Icinga-Mobile :

Si vous avez besoin consulter l'état de votre système d'information lors de vos déplacements, Icinga Mobile est une application fonctionnant sur iPhone, Android et (à venir) Blackberry. Cette interface pour mobile permet de visualiser en détails l'état des hôtes et des services, de trier les données et d'envoyer des commandes comme dans Icinga-Web.

Icinga-Reporting :

La génération de rapports est primordiale dans la supervision. Icinga est donc un outil indispensable que ce soit pour consulter l'état des SLA, de l'utilisation des capacités ou tout simplement pour faciliter la compréhension des graphiques pour les managers. Icinga intègre un composant qui utilise JasperServer et iReport afin de générer des rapports visuels exportables en différents formats.

10.3 Installation d'Icinga-Core

10.3.1 Installation des pré-requis

Packages des prérequis : Apache, GCC, GD

```
root@services $ #Installation des prérequis pour Icinga-Core
root@services $ apt-get install -y apache2 gcc glibc glibc-common gd gd-devel libjpeg libjpeg-devel libpng
libpng-devel php-soap
```

On installe ensuite les prérequis pour la base de données, nous choisissons MySQL. Icinga fonctionne aussi avec PostgreSQL et Oracle.

```
root@services $ #Installation de la base de données
root@services $ apt-get install -y mysql mysql-server libdbi libdbi-devel libdbi-drivers libdbi-dbd-mysql
```

10.3.2 Installation Icinga avec IDOUtils

Sur Debian, l'utilisation du dépôt Backports permet d'installer les paquets disponibles dans les versions testing, voire unstable de Debian sur une version stable. Ce dépôt a pour avantage d'avoir des versions de paquets plus à jour

que ceux disponibles sur le canal stable.

Autoriser les paquets de Backports :

```
root@services $ #Ajout du dépôt backports dans le fichier sources.list
root@services $ echo "deb http://backports.debian.org/debian-backports squeeze-backports main" >> /etc/apt/sources.list.d/squeeze-backports.list
root@services $ #Mise à jour des fichiers disponibles dans les dépôts apt
root@services $ apt-get update
root@services $ #Installation du client MySQL
root@services $ apt-get install libdbd-mysql mysql-client
```

Installation des paquets d'Icinga dans les sources backport :

```
root@pc-asrall $ apt-get -t squeeze-backports install icinga icinga-cgi icinga-core icinga-doc icinga-idoutils
```

Étant donné qu'Icinga est un fork de Nagios, il est possible d'utiliser directement les plugins de Nagios. Nous installons donc le paquet `nagios-plugin`.

```
root@services $ #Installation du paquet nagios-plugin root@services $ apt-get install nagios-plugins
```

10.3.3 Activation du module idomod

Afin de proposer une solution évolutive permettant l'exploitation de données directement issues d'Icinga, nous installons le module `idomod` qui fera la liaison entre Icinga et MySQL. De ce fait Icinga utilisera la base de données à la place de fichiers texte. Pour cela il suffit de copier le fichier `/usr/share/doc/icinga-idoutils/examples/idoutils.cfg-sample` vers `/etc/icinga/modules/idoutils.cfg`.

10.3.4 Autoriser les commandes externes (CGI)

Activer les commandes externes dans le fichier `/etc/icinga/icinga.cfg` :

```
check_external_commands=1
```

Mise à jour des propriétaires des fichiers :

```
root@services $ #Arrêt du service icinga pour éviter les conflits
root@services $ service icinga stop
root@services $ #Mise à jour des propriétaires de /var/lib/icinga avec droits en lecture et écriture pour les utilisateurs nagios et www-data
root@services $ dpkg-statoverride --update --add nagios www-data 2710 /var/lib/icinga/rw
root@services $ #Mise à jour des propriétaires de /var/lib/icinga
root@services $ dpkg-statoverride --update --add nagios nagios 751 /var/lib/icinga
root@services $ service icinga start
```

10.3.5 Installation de Icinga Web

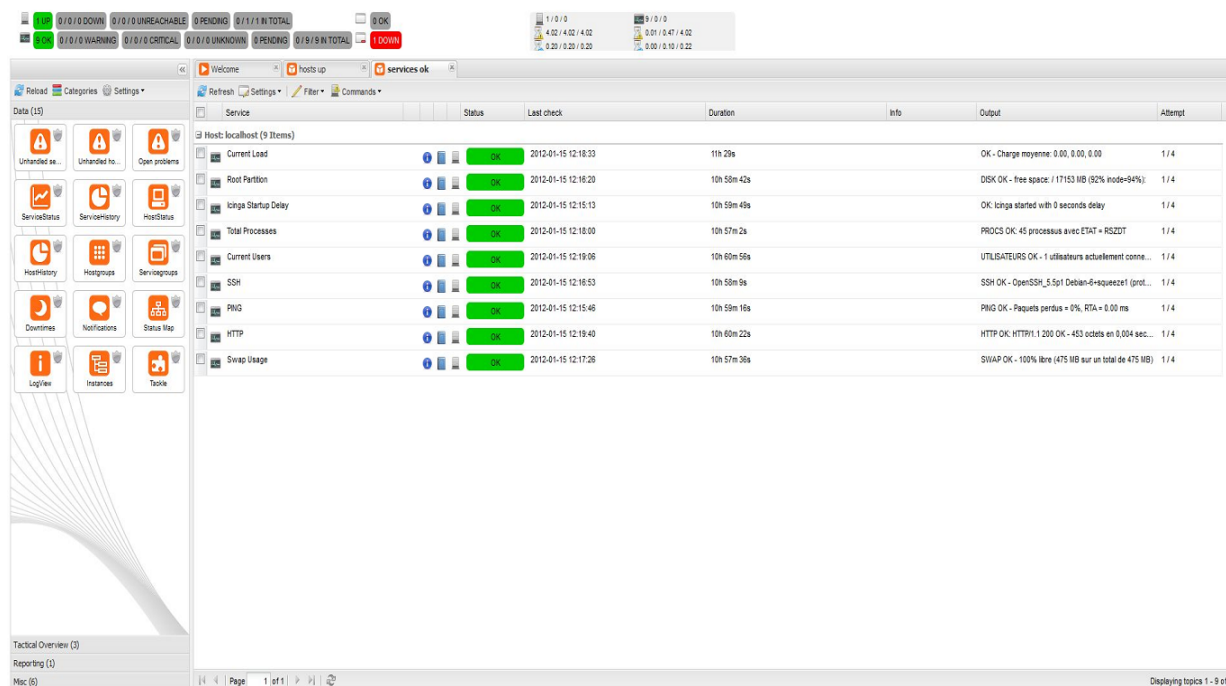
On installe ensuite les paquets prérequis pour le fonctionnement de l'application web d'Icinga.

```
root@services $ #Installation de l'application web d'Icinga
root@services $ apt-get install php5 php5-cli php-pear php5-xmllrpc php5-xsl php5-gd php5-ldap php5-mysql
```

On peut ensuite installer l'application web qui va nous permettre de configurer Icinga et de consulter les données recueillies.

```
root@services $ #Installation de l'application web Icinga
root@services $ apt-get -t squeeze-backports install icinga-web
```

Icinga Web est accessible depuis votre navigateur Web :

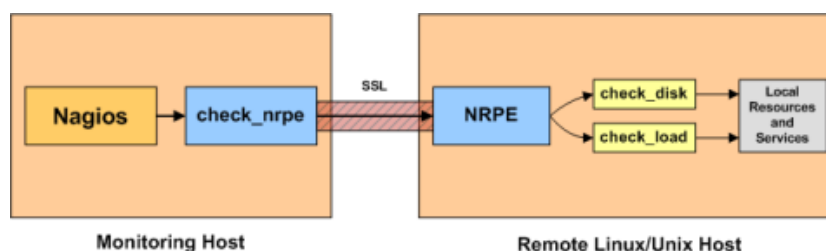


10.4 Addons officiels

Les addons officiels sont des programmes fournis par nagios.org pour améliorer et étendre les fonctionnalités de Nagios et de ses forks.

10.4.1 NRPE :

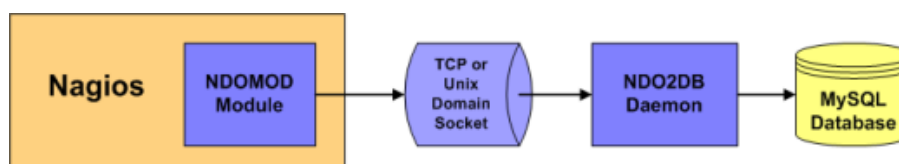
NRPE (Nagios Remote Plugin Executor) est un agent de supervision qui vous permet de récupérer les informations à distance. Son principe de fonctionnement est simple : il suffit d'installer le démon sur la machine distante et de l'interroger à partir du serveur Icinga.



Il est défini comme l'agent d'interrogation de type actif car c'est le serveur Icinga qui va interroger la machine distante.

10.4.2 NDOUtils :

NDOUtils est un addon servant à injecter les informations de Nagios dans la base de données MySQL. Ceci permet de ne plus avoir l'ancienne gestion des archives via fichiers logs et offre donc une plus grande souplesse dans la manipulation des données. Cet addon a permis d'avoir une plus grande ouverture sur l'exploitation des résultats de Icinga et de transformer l'information de la manière que l'on souhaite.



10.5 Comparatif Icinga / Zabbix :

	Icinga	Zabbix
Fonctionnalités	<ul style="list-style-type: none"> - Offre une interface web basée sur les CGI avec gestion des droits pour la consultation. - Génère des rapports de surveillance (SLA) et des documents qui définissent la qualité de service . - Permet de surveiller à distance à travers un pare-feu. - Permet de définir des serveurs esclaves qui prennent le relais si le serveur maître tombe en panne. - Surveille les ressources des serveurs (CPU, mémoire...) - Surveille les services réseaux (http ,ssh , DNS ..) . - Permet l'arrêt temporaire de la supervision locale ou globale. - Génère des graphes par l'interface avec RRD-Tools. 	<ul style="list-style-type: none"> - Monitoring : La partie affichage des statistiques, graphiques, alertes, cartographie..etc.. - Inventory : l'inventaire des machines et équipements . - Report : Statistiques sur le serveur Zabbix et rapport de disponibilité des services sur les machines supervisées . - Configuration : Comme son nom l'indique, permet de configurer entièrement Zabbix - Administration : Permet de gérer les moyens d'alertes (SMS, Jabber, Email, ...) et les utilisateurs .
Architecture	Architecture généralement basée sur le moteur de l'application (écrit en C) qui sert à ordonner les tâches de supervision et une interface web réalisée à l'aide des CGI, décrivant la vue d'ensemble sur système et les anomalies possibles; ainsi que plusieurs plugins qui peuvent être complétés en fonction des besoins.	Architecture de Zabbix est basée sur le cœur du moteur de l'application qui est programmé en C Le « serveur ZABBIX » peut être décomposé en 3 parties séparées : <ul style="list-style-type: none"> - Le serveur de données utilise MySQL, PostgreSQL ou Oracle pour stocker les données - L'interface de gestion est une interface web écrite en PHP. Elle agit directement sur les informations stockées dans la base de données. - L'utilisation de Zabbix Agent permet une meilleure surveillance des hôtes, et donc une supervision plus accrue.
Avantages	<ul style="list-style-type: none"> - Reconnu auprès des entreprises, grande communauté - Énormément de plugins qui permettent d'étendre les possibilités (agents comme zabbix, reporting amélioré, etc...) - Une solution complète permettant le reporting, la gestion de panne et d'alarmes, gestion utilisateurs, ainsi que la cartographie du réseaux. - Beaucoup de documentations sur le web . - Performances du moteur . 	<ul style="list-style-type: none"> - Une solution très complète : cartographie de réseaux, gestion poussée d'alarmes via SMS, Jabber ou Email, gestion des utilisateurs, gestion de pannes, statistiques et reporting . - Une entreprise qui pousse le développement, et une communauté croissante . - Une interface vaste mais claire . - Une gestion des templates poussée, avec import/export xml, modifications via l'interface . - Compatible avec MySQL, PostgreSQL, Oracle, SQLite

11 Conclusion

11.1 Conclusion générale

Le projet fût très intéressant. Il nous a permis de nous pencher sur des cas réseau que nous abordons peu en cours. L'éventualité d'une mise en production des résultats de nos travaux nous a incités à plus d'exigences sur la qualité de ce projet. L'utilisation de l'outil Zabbix nous a beaucoup ralenti du fait d'un manque de documentation et de sa précision. Nous avons donc créé un sujet sur le forum pour les difficultés rencontrées quant à l'obtention de la valeur du 95^{ème} centile dans une variable ou une macro. Seulement, peu de réactivité de la part de la communauté. En effet, après plusieurs semaines, nous avons observé peu de vues (100-200) et personne n'a pu nous aider.

Nous sommes déçus de n'avoir pu obtenir de meilleurs résultats. Néanmoins, ils pourront servir à l'association pour une conclusion finale de l'outil Zabbix et de ses limites. Nous aurions souhaité avoir plus de temps pour pouvoir effectuer des recherches supplémentaires et obtenir des résultats concluants et directement utilisables en production.

Le peu de ressources matérielles en classe nous a aussi beaucoup ralenti. Par exemple, il nous a fallu attendre plusieurs jours pour obtenir l'installation d'une deuxième carte réseau sur nos deux serveurs. Nous étions donc freinés dans l'avancement du projet, n'ayant plus notre environnement de test. L'association nous a donc mis à disposition du matériel : un câble croisé et un commutateur, éléments manquants en salle bloquants pour relier nos serveurs en direct et ne pas être parasités.

11.2 Expérience acquise

Le travail avec l'association LDN nous a permis de comprendre comment fonctionnait un FAI associatif sur un plan technique mais aussi commercial. Nous ignorions jusqu'alors la contrainte du transit, peering et la facturation au 95^{ème} centile.

La reproduction du réseau a été très bénéfique pour l'acquisition de nombreuses compétences, sur les configurations réseau d'un système Debian. Elle nous a permis une meilleure compréhension et nous nous sommes orientés vers un changement de nos méthodes de travail. L'utilisation des outils comme `ip` ou `tc` était jusqu'à présent inconnue bien qu'utilisée en entreprise. Le travail avec un FAI associatif nous a fait comprendre la réelle pénurie d'adresses IPv4 et des contraintes qu'elle génère ; ainsi nous avons pu effectuer notre première expérience avec les adresses IPv6.

Il a été fort agréable de découvrir un nouvel outil de monitoring comme Zabbix, qui reste très puissant malgré le fait qu'il ne peut pas répondre totalement à nos attentes dans le cadre du projet.

11.3 Remerciements

Nous tenons à remercier nos deux tuteurs, Sébastien JEAN et Julien VAUBOURG, pour le temps qu'ils nous ont dédié, pour nous avoir assistés lors des difficultés rencontrées, leur présence hebdomadaire, leur réactivité pour nous répondre, ainsi que pour la proposition du projet. Nous remercions aussi l'association LDN pour le prêt du matériel et l'accès à leur base de connaissances via le wiki.

Partie III

Annexes

12 Annexes

12.1 Webographie

- Lorraine Data Network : <http://ldn-fai.net/>
- Services LDN : <http://ldn-fai.net/services-de-lassociation/>
- Wiki LDN : <https://wiki.ldn-fai.net>
- Documentation libvirt sur les interfaces réseaux : <http://libvirt.org/formatnetwork.html>
- Zabbix : <http://zabbix.com/>
- Document de Zabbix : <https://www.zabbix.com/documentation/doku.php?id=2.2/manual>
- Document réseau libvirt : <http://libvirt.org/formatnetwork.html>
- Tutoriel KVM/libvirt : http://homeserver-diy.net/wiki/index.php?title=Virtualisation_avec_KVM_via_libvirt
- Article sur TC : <http://linuxfr.org/wiki/une-introduction-au-controle-du-traffic-reseau-avec-linux>

13 Lexique

LDN : Lorraine Data Network est une association pour la défense d'un Internet libre, neutre et décentralisé situé en Lorraine.

95^{ème} centile : La mesure du 95^{ème} centile est celle utilisée par les opérateurs sur Internet pour facturer la consommation de bande passante de leurs clients. Le 95^{ème} centile est la valeur telle que 95% des valeurs sont en dessous et 5% sont au dessus.

API Zabbix : Outil de Zabbix, utilisé pour étendre Zabbix à d'autres applications pour une plateforme ou pour l'intégration à un logiciel.

JSON-RPC : Protocole permettant la définition de plusieurs types de données et de commandes. JSON-RPC permet d'effectuer des requêtes à un serveur pour obtenir des informations systèmes ou des éléments de base de données.

FAI : Un Fournisseur d'Accès à Internet est un organisme offrant une connexion à Internet.

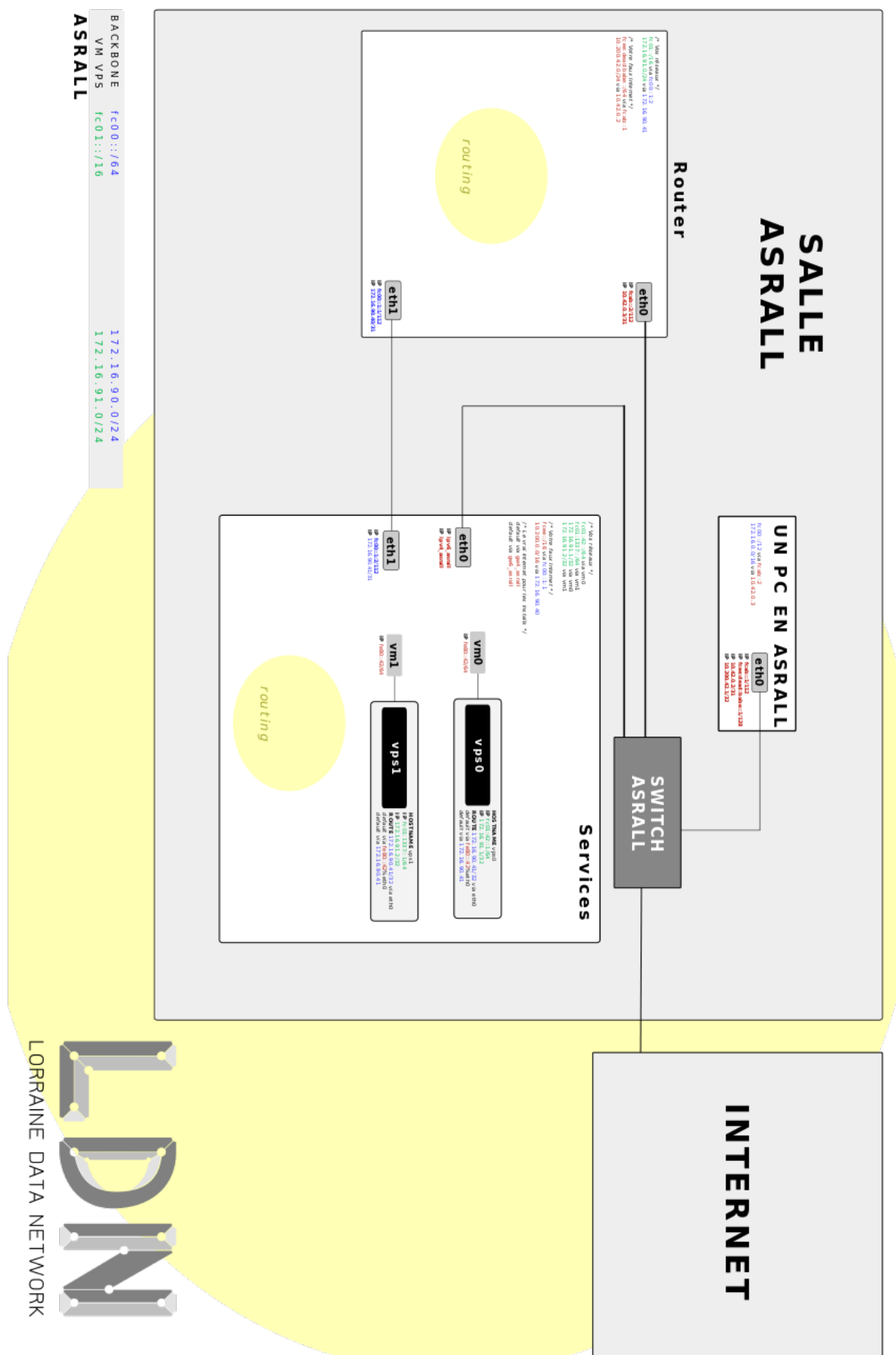
FSI : Fournisseur de Services Internet, organisme qui offre des solutions disponible depuis l'internet (ex : hébergement).

RRD : Round-Robin Database est une base de donnée pour la génération de graphique.

Peering : Le peering est la pratique d'échanger du trafic Internet avec des pairs.

13.1 Schéma réseau simplifié

13.1.1 Premier schéma



[illegible]

13.2 Exemple xml de vm

```

<!--
WARNING: THIS IS AN AUTO-GENERATED FILE. CHANGES TO IT ARE LIKELY TO BE
OVERWRITTEN AND LOST. Changes to this xml configuration should be made using:
    virsh edit vm0
or other application using the libvirt API.
-->

<domain type='kvm'>
  <name>vm0</name>
  <uuid>26e7e9a0-5798-bac6-84e6-2745edc2c24a</uuid>
  <memory unit='KiB'>1048576</memory>
  <currentMemory unit='KiB'>1048576</currentMemory>
  <vcpu placement='static'>1</vcpu>
  <os>
    <type arch='x86_64' machine='pc-1.1'>hvm</type>
    <boot dev='hd' />
  </os>
  <features>
    <acpi/>
    <apic/>
    <pae/>
  </features>
  <clock offset='utc' />
  <on_poweroff>destroy</on_poweroff>
  <on_reboot>restart</on_reboot>
  <on_crash>restart</on_crash>
  <devices>
    <emulator>/usr/bin/kvm</emulator>
    <disk type='file' device='disk'>
      <driver name='qemu' type='raw' />
      <source file='/var/lib/libvirt/images/vm2.img' />
      <target dev='vda' bus='virtio' />
      <address type='pci' domain='0x0000' bus='0x00' slot='0x04' function='0x0' />
    </disk>
    <disk type='block' device='cdrom'>
      <driver name='qemu' type='raw' />
      <target dev='hdc' bus='ide' />
      <readonly />
      <address type='drive' controller='0' bus='1' target='0' unit='0' />
    </disk>
    <controller type='usb' index='0'>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x01' function='0x2' />
    </controller>
    <controller type='ide' index='0'>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x01' function='0x1' />
    </controller>
    <interface type='bridge'>
      <mac address='52:54:00:ef:6e:27' />
      <source bridge='virbr0' />
      <model type='virtio' />
      <address type='pci' domain='0x0000' bus='0x00' slot='0x03' function='0x0' />
    </interface>
    <serial type='pty'>
      <target port='0' />
    </serial>
    <console type='pty'>
      <target type='serial' port='0' />
    </console>
    <input type='tablet' bus='usb' />
    <input type='mouse' bus='ps2' />
  </devices>
</domain>

```

```
<graphics type='vnc' port='-1' autoport='yes' />
<video>
  <model type='cirrus' vram='9216' heads='1' />
  <address type='pci' domain='0x0000' bus='0x00' slot='0x02' function='0x0' />
</video>
<memballoon model='virtio'>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x05' function='0x0' />
</memballoon>
</devices>
</domain>
```

13.3 Les différentes tables de routage

13.3.1 Vm0

Table de routage IP du noyau

Destination	Passerelle	Genmask	Indic	Metric	Ref	Use	Iface
0.0.0.0	172.16.90.41	0.0.0.0	UG	0	0	0	eth0
172.16.90.41	0.0.0.0	255.255.255.255	UH	0	0	0	eth0

Table de routage IPv6 du noyau

Destination	Next Hop	Flag	Met	Ref	Use	If
fc01:42::/64	::	U	256	0	0	eth0
fe80::/64	::	U	256	0	0	eth0
::/0	fe80::42	UG	1024	0	0	eth0
::/0	::	!n	-1	1	1	lo
::1/128	::	Un	0	1	2	lo
fc01:42::1/128	::	Un	0	1	0	lo
fe80::5054:ff:feef:6e27/128	::	Un	0	1	0	lo
ff00::/8	::	U	256	0	0	eth0
::/0	::	!n	-1	1	1	lo

13.3.2 Services

Table de routage IP du noyau

Destination	Passerelle	Genmask	Indic	Metric	Ref	Use	Iface
0.0.0.0	192.168.1.254	0.0.0.0	UG	0	0	0	eth0
10.200.0.0	172.16.90.40	255.255.0.0	UG	0	0	0	eth1
172.16.90.40	0.0.0.0	255.255.255.254	U	0	0	0	eth1
172.16.91.1	0.0.0.0	255.255.255.255	UH	0	0	0	vnet0
172.16.91.2	0.0.0.0	255.255.255.255	UH	0	0	0	vnet1
192.168.1.0	0.0.0.0	255.255.255.0	U	0	0	0	eth0
192.168.122.0	0.0.0.0	255.255.255.0	U	0	0	0	virbr0

Table de routage IPv6 du noyau

Destination	Next Hop	Flag	Met	Ref	Use	If
fc00::1:0/112	::	U	256	0	3	eth1
fc01:42::/64	::	U	1024	0	0	vnet0
fc01:1337::/64	::	U	1024	0	0	vnet1
fcee::/16	fc00::1:1	UG	1024	0	0	eth1
fe80::/64	::	U	256	0	0	eth0
fe80::/64	::	U	256	0	0	eth1
fe80::/64	::	U	256	0	0	vnet1
fe80::/64	::	U	256	0	0	vnet0
fe80::/64	::	U	256	0	0	vnet2
::/0	::	!n	-1	1	997	lo
::1/128	::	Un	0	1	61359	lo
fc00::1:0/128	::	Un	0	1	0	lo
fc00::1:2/128	::	Un	0	1	290	lo
fe80::/128	::	Un	0	1	0	lo
fe80::/128	::	Un	0	1	0	lo
fe80::/128	::	Un	0	1	0	lo
fe80::/128	::	Un	0	1	0	lo
fe80::/128	::	Un	0	1	0	lo
fe80::42/128	::	Un	0	1	4	lo
fe80::42/128	::	Un	0	1	3	lo
fe80::21e:4fff:fed1:d08f/128	::	Un	0	1	0	lo
fe80::240:5ff:fea6:e841/128	::	Un	0	1	24	lo
fe80::fc54:ff:fe24:2036/128	::	Un	0	1	0	lo
fe80::fc54:ff:fe64:7e1c/128	::	Un	0	1	0	lo
fe80::fc54:ff:feef:6e27/128	::	Un	0	1	0	lo
ff00::/8	::	U	256	0	0	eth0
ff00::/8	::	U	256	0	0	eth1
ff00::/8	::	U	256	0	0	vnet1
ff00::/8	::	U	256	0	0	vnet0
ff00::/8	::	U	256	0	0	vnet2
::/0	::	!n	-1	1	997	lo

13.3.3 Router

Table de routage IP du noyau

Destination	Passerelle	Genmask	Indic	Metric	Ref	Use	Iface
10.42.0.2	0.0.0.0	255.255.255.254	U	0	0	0	eth0
172.16.90.40	0.0.0.0	255.255.255.254	U	0	0	0	eth1
172.16.91.0	172.16.90.41	255.255.255.0	UG	0	0	0	eth1

Table de routage IPv6 du noyau

Destination	Next Hop	Flag	Met	Ref	Use	If
fc00::1:0/112	::	U	256	0	5	eth1
fc01::/16	fc00::1:2	UG	1024	0	0	eth1
fcab::/112	::	U	256	0	0	eth0
fe80::/64	::	U	256	0	0	eth1
fe80::/64	::	U	256	0	0	eth0
::/0	::	!n	-1	1	48	lo
::1/128	::	Un	0	1	31	lo
fc00::1:0/128	::	Un	0	1	0	lo
fc00::1:1/128	::	Un	0	1	15	lo
fcab::/128	::	Un	0	1	0	lo
fcab::2/128	::	Un	0	1	0	lo
fe80::/128	::	Un	0	1	0	lo
fe80::/128	::	Un	0	1	0	lo
fe80::206:29ff:fe4f:259/128	::	Un	0	1	18	lo
fe80::218:8bff:fe20:ecbb/128	::	Un	0	1	0	lo
ff00::/8	::	U	256	0	0	eth1
ff00::/8	::	U	256	0	0	eth0
::/0	::	!n	-1	1	48	lo

13.3.4 PC ASRALL

Table de routage IP du noyau

Destination	Passerelle	Genmask	Indic	Metric	Ref	Use	Iface
0.0.0.0	192.168.1.254	0.0.0.0	UG	0	0	0	eth1
10.42.0.2	0.0.0.0	255.255.255.254	U	0	0	0	eth1
172.16.0.0	10.42.0.3	255.255.0.0	UG	0	0	0	eth1
192.168.1.0	0.0.0.0	255.255.255.0	U	0	0	0	eth1

Table de routage IPv6 du noyau

Destination	Next Hop	Flag	Met	Ref	Use	If
fe80::/64	::	U	256	0	0	peth1
fe80::/64	::	U	256	0	0	eth1
::/0	::	!n	-1	1	7	lo
::1/128	::	Un	0	1	3026	lo
fe80::f24d:a2ff:fe2c:8ddd/128	::	Un	0	1	0	lo
fe80::f24d:a2ff:fe2c:8ddd/128	::	Un	0	1	0	lo
ff00::/8	::	U	256	0	0	peth1
ff00::/8	::	U	256	0	0	eth1
::/0	::	!n	-1	1	7	lo

13.4 Les nouvelles routes

13.4.1 Router

Table de routage IP du noyau

Destination	Passerelle	Genmask	Indic	Metric	Ref	Use	Iface
10.42.0.2	0.0.0.0	255.255.255.254	U	0	0	0	eth0.800
10.43.0.2	0.0.0.0	255.255.255.254	U	0	0	0	eth0.267
10.200.42.0	10.42.0.2	255.255.254.0	UG	0	0	0	eth0.800
172.16.90.40	0.0.0.0	255.255.255.254	U	0	0	0	eth1
172.16.91.0	172.16.90.41	255.255.255.0	UG	0	0	0	eth1

Table de routage IPv6 du noyau

Destination	Next Hop	Flag	Met	Ref	Use	If
fc00::1:0/112	::	U	256	0	1	eth1
fc01::/16	fc00::1:2	UG	1024	0	0	eth1
fcab::/112	::	U	256	0	4	eth0
fcee:dead:babe::/64	fcab::1	UG	1024	0	0	eth0
fe80::/64	::	U	256	0	0	eth0
fe80::/64	::	U	256	0	0	eth0.267
fe80::/64	::	U	256	0	0	eth0.800
fe80::/64	::	U	256	0	0	eth1
::/0	::	!n	-1	1	10	lo
::1/128	::	Un	0	1	68	lo
fc00::1:0/128	::	Un	0	1	0	lo
fc00::1:1/128	::	Un	0	1	0	lo
fcab::/128	::	Un	0	1	0	lo
fcab::2/128	::	Un	0	1	0	lo
fe80::/128	::	Un	0	1	0	lo
fe80::/128	::	Un	0	1	0	lo
fe80::/128	::	Un	0	1	0	lo
fe80::/128	::	Un	0	1	0	lo
fe80::206:29ff:fe4f:259/128	::	Un	0	1	0	lo
fe80::218:8bff:fe20:ecbb/128	::	Un	0	1	0	lo
fe80::218:8bff:fe20:ecbb/128	::	Un	0	1	0	lo
fe80::218:8bff:fe20:ecbb/128	::	Un	0	1	0	lo
ff00::/8	::	U	256	0	0	eth0
ff00::/8	::	U	256	0	0	eth0.267
ff00::/8	::	U	256	0	0	eth0.800
ff00::/8	::	U	256	0	0	eth1
::/0	::	!n	-1	1	10	lo

13.4.2 PC ASRALL

Table de routage IP du noyau

Destination	Passerelle	Genmask	Indic	Metric	Ref	Use	Iface
0.0.0.0	192.168.1.254	0.0.0.0	UG	0	0	0	eth1
10.42.0.2	0.0.0.0	255.255.255.254	U	0	0	0	eth0.800
10.43.0.2	0.0.0.0	255.255.255.254	U	0	0	0	eth0.267
172.16.0.0	10.42.0.3	255.255.0.0	UG	0	0	0	eth0.800
192.168.1.0	0.0.0.0	255.255.255.0	U	0	0	0	eth1

Table de routage IPv6 du noyau

Destination	Next Hop	Flag	Met	Ref	Use	If
fc00::/12	fcab::2	UG	1024	0	0	eth0.800
fcab::/112	::	U	256	0	1	eth0.800
fcac::/112	::	U	256	0	0	eth0.267
fcee:dead:babe::1/128	::	U	256	0	0	eth0.800
fcff:dead:beef::1/128	::	U	256	0	0	eth0.267
fe80::/64	::	U	256	0	0	eth0
fe80::/64	::	U	256	0	0	eth0.800
fe80::/64	::	U	256	0	0	eth0.267
fe80::/64	::	U	256	0	0	eth1
::/0	::	!n	-1	1	866	lo
::1/128	::	Un	0	1	1001	lo
fcab::1/128	::	Un	0	1	0	lo
fcac::1/128	::	Un	0	1	0	lo
fcee:dead:babe::1/128	::	Un	0	1	0	lo
fcff:dead:beef::1/128	::	Un	0	1	0	lo
fe80::221:91ff:fe8c:e863/128	::	Un	0	1	0	lo
fe80::221:91ff:fe8c:e863/128	::	Un	0	1	0	lo
fe80::221:91ff:fe8c:e863/128	::	Un	0	1	0	lo
fe80::f24d:a2ff:fe2c:5e6a/128	::	Un	0	1	0	lo
ff00::/8	::	U	256	0	0	eth0
ff00::/8	::	U	256	0	0	eth0.800
ff00::/8	::	U	256	0	0	eth0.267
ff00::/8	::	U	256	0	0	eth1
::/0	::	!n	-1	1	866	lo

13.5 API Zabbix

13.5.1 Script Perl

```
#!/usr/bin/perl

use 5.010;
use strict;
use warnings;
use JSON::RPC::Client;
use Data::Dumper;

#-----
# Authentification sur le serveur Zabbix
#-----

my $client = new JSON::RPC::Client;
my $url = 'http://192.168.1.35/zabbix/api_jsonrpc.php'; #URL de connection
my $authID;
my $response;

#-----
#Déclaration de la variable $json en identifiant la version de json utilisée, la méthode utilisée le login
#-----

my $json = {
    jsonrpc => "2.0",
    method => "user.login",
    params => {
        user => "admin",
        password => "pwasrall"
    },
    id => 1
};

$response = $client->call($url, $json);

#-----
# Check si la réponse est OK
#-----

die "Fail Authentication\n" unless $response->content->{'result'};

$authID = $response->content->{'result'};
print "Authentication Success. Auth ID: " . $authID . "\n";

#-----
# Création de la requête JSON-RPC ici un appel à la modèl graph.get avec comme nom de graphique CPU sur 1
#-----

$json = {
    jsonrpc=> '2.0',
    "method"=>"graph.get",
    "params"=>{
        "output"=>"extend",
        "search"=>{
            "name"=>"CPU"
        },
        "filter"=>{
            "host"=>[
                "Zabbix-server"
            ]
        }
    }
}
```

```
    },
    "limit"=>2
  },
  id => 2,
  auth => "$authID",
};
$response = $client->call($url, $json);

#-----
# Check si la réponse est OK
#-----

die "graph.get failed\n" unless $response->content->{result};

#-----
#Affichage de la réponse
#-----

print "Liste de graphe\n-----\n";
foreach my $host (@{$response->content->{result}}) {
print  "Graphe: ".$graph->{"name"}."\n";
```

13.5.2 Script PHP

```
<?php

//-----
// Chargement ZabbixApi
//-----

require 'ZabbixApiAbstract.class.php';
require 'ZabbixApi.class.php';

try {

//-----
// connection à l'API Zabbix
//-----

    $api = new ZabbixApi('http://192.168.1.35/api_jsonrpc.php','zabbix','pwasrall');

//-----
// Récupération de tous les graphes
//-----

    $graphs = $api->graphGet();

//-----
// Affichage des valeurs ID des graphiques
//-----

    foreach($graphs as $graph)
    echo $graph->graphid."\n";

} catch(Exception $e) {

// Exception in ZabbixApi catched
    echo $e->getMessage();

}

?>
```

13.6 RRDTool

13.6.1 Script

Script de génération des graphiques RRD depuis les valeurs de la base de donnée de Zabbix.

```
#!/usr/bin/perl

use warnings;
use strict;
use Encode;
use utf8;
use DBI;
use Date::Parse;
```

```
#-----
#Déclaration des variables pour la connection à la base et déclaration des tableaux de stockage des
données.
#-----

my $bd = 'zabbix';
my $serveur = 'localhost';
my $id = 'root';
my $mdp = 'pwasrall';
my $port = '';
my $i=0;
my @result_horloge = ();
my @result_network = ();
my $date;

#-----
#Connection à la base de donnée MYSQL de Zabbix
#-----

print " BASE CONNEXION : $bd\n";
my $dbh = DBI->connect("DBI:mysql:database=$bd;host=$serveur;port=$port",$id,$mdp,{ RaiseError => 1, }
) or die "Connection impossible BASE $bd !\n $! \n $@\n$DBI::errstr";

#-----
#Préparation de la requête SQL pour la récupération des données dans la base de Zabbix
#-----

print "Affichage des valeurs de debit entrant sur eth0 de la machine service \n";

my $requete_network = <<"SQL";

SELECT value,clock
FROM history_uint
WHERE itemid = 23331 AND (clock < (SELECT MAX(clock) FROM history_uint WHERE itemid = 23331) AND clock >
(SELECT (MAX(clock)-1000) FROM history_uint WHERE itemid = 23331));

SQL

#-----
#Lancement de la requête SQL et récupération des valeurs ligne à ligne
#-----

my $prep = $dbh->prepare($requete_network) or die $dbh->errstr;
$prep->execute() or die "Echec de la requete : $requete_network\n";

print "RESULTAT DEBIT NETWORK: \n";
while ( my $refdonnees= $prep->fetchrow_hashref) {
print "\t $refdonnees->{clock}:$refdonnees->{value}\n";
push(@result_horloge,$refdonnees->{clock});
push(@result_network,$refdonnees->{value});
}

$prep->finish();

#-----
#Création de la base RRD, mise à jour des données de la base et génération du graphique
#-----

my $test_clock = $result_horloge[0]-1;
my $create_rrd = "rrdtool create test.rrd --start $test_clock DS:test:GAUGE:600:U:U RRA:AVERAGE
:0.5:1:12";
print "Creation de la base de donnee RRD et config du graphique\n";
print '$create_rrd';
```

```
my $size = $#result_horloge+1;

print "$size\n";
my $update_rrd = "rrdtool update test.rrd ";

for($i=0 ;$i<$size;$i++)
{
$update_rrd = "$update_rrd $result_horloge[$i]:$result_network[$i]";
}
print '$update_rrd';

print "Generation du graphique\n";
my $gen_graph = "rrdtool graph test.png -s $test_clock -e $result_horloge[$size-1] -h 300 -w 600 -t
\"Graphe de test\" DEF:test=test.rrd:test:AVERAGE LINE3:test#FF0000:\"TEST\"";
print '$gen_graph';

$date= str2time('10/03/2014 00:00:00');
print "$date \n";
```

13.7 Documentation pour LDN

13.7.1 Installation de Zabbix serveur et agent

```
## Zabbix-server
# wget http://repo.zabbix.com/zabbix/2.0/debian/pool/main/z/zabbix-release/zabbix-release_2.0-1wheezy_all.deb
# dpkg -i zabbix-release_2.0-1wheezy_all.deb
# apt-get update
# apt-get install zabbix-server-mysql zabbix-frontend-php

## Zabbix-agent (sur la cible)
# wget http://repo.zabbix.com/zabbix/2.0/debian/pool/main/z/zabbix-release/zabbix-release_2.0-1wheezy_all.deb
# dpkg -i zabbix-release_2.0-1wheezy_all.deb
# apt-get update
# apt-get install zabbix-agent
## Ajout de l'IP du serveur dans le fichier conf de l'agent # vim /etc/zabbix/zabbix_agentd.conf +86
```

13.7.2 Configuration/création des graphiques prototypes

```
# Menu Configuration/Modèles/Template OS Linux/Découverte/Network interface/graph prototype
# Création de graphique
# Configuration de la valeur du centile gauche et droit (95.00)
# Sélection des éléments à récupérer (débit entrant et sortant des interfaces réseau)
# Création des graphiques lors de la prochaine mise à jour des hôtes.
```

13.7.3 Configuration/création des déclencheurs prototypes

```
# Menu Configuration/Modèles/Template OS Linux/Découverte/Network interface/déclencheurs prototypes
# Création des déclencheurs
```

Déclencheur

Nom

Expression [Ajouter](#)

[Constructeur d'expression](#)

Génération d'événements PROBLEME ☐

multiples

Description

URL

Sévérité **Non classé** Information Avertissement Moyen Haut Désastre

Activé ☒

[Sauver](#) [Annuler](#)

Configuration des éléments en cliquant sur sélectionner prototype

zabbix: Condition - Google Chrome

192.168.1.35/zabbix/popup_trexp.php?dstfrm=triggersForm&dstfld1=expression&srctbl=exp

Condition de l'expression du déclencheur

Élément [Sélectionner](#)

Sélectionner le prototype

Fonction La dernière (plus récente) valeur T est = N

Dernier (T) 0 Secondes

Décalage temporel Secondes

N 0

[Insertion](#) [Annuler](#)

Sélection de l'élément souhaité : Incoming network traffic on {#IFNAME}.

Configuration de l'expression : la dernière (plus récente) valeur T est = N

Choix de Dernier (T) : 1

Choix de N = valeur du centile

Création du nouveau déclencheur lors de la prochaine mise à jour des hôtes.

13.8 Image de configuration de l'action

Durée de l'étape d'opération par défaut

60 (60 secondes minimum)

Opérations d'action

Étapes

Détails

Démarrer dans

Durée (sec)

Action

1

Exécuter des commandes distantes sur l'hôte courant

Immédiatement

Défaut

Édition

Supprimer

Détails de l'opération

Étape

De

1

A

1 (0 - infini)

Durée de l'étape

0 (60 secondes minimum, 0 - action par défaut)

Type d'opération

Commande à distance

Liste des cibles

Cible

Action

Hôte: router

Supprimer

Nouveau

Type

Script personnalisé

Exécuter sur

agent Zabbix

Serveur Zabbix

Commandes

/etc/zabbix/scripts/icmp.sh eth1 800

Conditions

Étiquette

Nom

Action

Nouveau

Actualiser

Annuler

13.9 Script Sniffer

```
#!/usr/bin/perl

use strict;
use Net::Pcap;
use NetPacket::Ethernet;
use NetPacket::IP;
use NetPacket::TCP;

# Variable Declarations
my $filter_t;
my ($tcp,$ip,$ethernet);
my ($net,$mask,$err);
my $dev = $ARGV[0]; #takes the network card interface as the first parameter
my $dev2 = $ARGV[1]; #takes the network card interface as the second parameter
my $filter;
my $optimize = 1;
my $pid;

# Determine network number and mask for use later on when we're compiling our filter
if (Net::Pcap::lookupnet($dev, \$net, \$mask, \$err) == -1){
die 'Cannot determine network number and subnet mask - ' , $err;
}

if (Net::Pcap::lookupnet($dev2, \$net, \$mask, \$err) == -1){
die 'Cannot determine network number and subnet mask - ' , $err;
}

$pid = fork();
if ($pid != 0) {
#Dans le père

my $pcap_object = Net::Pcap::open_live($dev, 1500, 0, 0, \$err);
if (defined $err){
die 'Failed to create live capture on - ' , $dev , ' - ' , $err;
}

Net::Pcap::compile($pcap_object, \$filter_t, $filter, $optimize, $mask); #compile our filter , $filter
and return it in the $filter_t variable

Net::Pcap::loop($pcap_object, -1, \&capture_packets, '') || die 'Unable to perform packet capture';
#loop or sniff packets on the network infinitely

Net::Pcap::close($pcap_object); #close the pcap object gracefully

} else {
#Dans le fils

my $pcap_object_2 = Net::Pcap::open_live($dev2, 1500, 0, 0, \$err);
if (defined $err){
die 'Failed to create live capture on - ' , $dev2 , ' - ' , $err;
}

Net::Pcap::compile($pcap_object_2, \$filter_t, $filter, $optimize, $mask); #compile le our filter ,
$filter and return it in the $filter_t variable

Net::Pcap::loop($pcap_object_2, -1, \&capture_packets, '') || die 'Unable to perform packet capture';
```

```
#loop or sniff packets on the network infinitely

Net::Pcap::close($pcap_object_2); #close the pcap object gracefully
exit(0);

}

# subroutine to handle each packet that is sniffed
sub capture_packets {
my($user_data, $hdr, $pkt) = @_; #this line should always be present to handle the incoming packets,
You refer to the incoming packets from $pkt as you would see from the next lines of code
my $ethernet = NetPacket::Ethernet->decode($pkt); #decodes the ethernet frame
my $ip = NetPacket::IP->decode($ethernet->{data}); # decodes the IP headers
my $tcp = NetPacket::TCP->decode($ip->{data}); # decodes the TCP data

if($pid!=0){
print "INTERFACE eth0.800 \n";

if($ip->{src_ip} =~ /172\.16\..*\$/){
print "OUT($ip->{src_ip}): $ip->{len} octets\n";
}
else {
print "IN($ip->{dest_ip}): $ip->{len} octets\n";
}
print "IP Source : $ip->{src_ip} -> IP Destination : $ip->{dest_ip} : "; # prints source to destination
IP's
print ": $ip->{len} octets\n"; #prints the data contained in this packet
} else {
print "INTERFACE eth0.267\n";

if($ip->{src_ip} =~ /172\.16\..*\$/){
print "OUT($ip->{src_ip}): $ip->{len} octets\n";
}
else {
print "IN($ip->{dest_ip}): $ip->{len} octets\n";
}
print "IP Source : $ip->{src_ip} -> IP Destination : $ip->{dest_ip} : "; # prints source to destination
IP's
print ": $ip->{len} octets\n"; #prints the data contained in this packet
}
}
```