

# Rapport de stage

Romain BOUJILA

2 avril 2014

## Révision des solutions de supervision

Maître de stage : Jean-Charles BRIATTE



# Table des matières

<b>I</b>	<b>Présentation du projet</b>	<b>4</b>
<b>1</b>	<b>Introduction</b>	<b>5</b>
<b>2</b>	<b>Lorraine Data Network</b>	<b>6</b>
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
<b>II</b>	<b>Partie technique</b>	<b>8</b>
<b>3</b>	<b>Reproduction réseau</b>	<b>9</b>
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
<b>4</b>	<b>Zabbix</b>	<b>13</b>
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
<b>5</b>	<b>95<sup>ème</sup> centile</b>	<b>16</b>
5.1	Présentation	16
5.2	Mise en place des graphiques automatiques pour le 95 <sup>ème</sup> centile	16
5.3	Récupération de la valeur	17
<b>6</b>	<b>API Zabbix</b>	<b>18</b>
6.1	Qu'est-ce que c'est ?	18
6.2	Utilisation de l'API	18
<b>7</b>	<b>RRDTool</b>	<b>20</b>
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
<b>8</b>	<b>Limitation du débit</b>	<b>23</b>
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

<b>9</b>	<b>Supervision du débit</b>	<b>28</b>
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
<b>10</b>	<b>Icinga</b>	<b>31</b>
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
<b>11</b>	<b>Conclusion</b>	<b>35</b>
11.1	Conclusion générale . . . . .	35
11.2	Expérience acquise . . . . .	35
11.3	Remerciements . . . . .	35
<b>III</b>	<b>Annexes</b>	<b>36</b>
<b>12</b>	<b>Annexes</b>	<b>37</b>
12.1	Webographie . . . . .	37
<b>13</b>	<b>Lexique</b>	<b>38</b>
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

# Partie II

## Partie technique

# Partie III

## Annexes

# 1 Annexes

## 1.1 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](http://homeserver-diy.net/wiki/index.php?title=Virtualisation_avec_KVM_via_libvirt)
- Article sur TC : <http://linuxfr.org/wiki/une-introduction-au-contrôle-du-traffic-réseau-avec-linux>

## 2 Lexique

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

95<sup>ème</sup> centile : La mesure du 95<sup>ème</sup> centile est celle utilisée par les opérateurs sur Internet pour facturer la consommation de bande passante de leurs clients. Le 95<sup>ème</sup> 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.

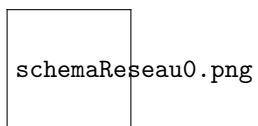
FSI : Fournisseur de Services Internet, organisme qui offre des solutions disponible depuis l'internet (ex : hébergement).

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

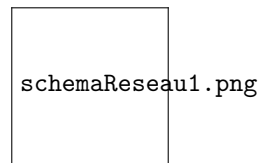
Peering : Le peering est la pratique d'échanger du trafic Internet avec des pairs.

## 2.1 Schéma réseau simplifié

### 2.1.1 Premier schéma



### 2.1.2 Second schéma





## 2.2 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>
```

## 2.3 Les différentes tables de routage

### 2.3.1 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

### 2.3.2 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

### 2.3.3 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

### 2.3.4 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

## 2.4 Les nouvelles routes

### 2.4.1 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

## 2.4.2 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



## 2.5 API Zabbix

### 2.5.1 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 modèle 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";
```

## 2.5.2 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();

}

?>
```

## 2.6 RRDTool

### 2.6.1 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";
```

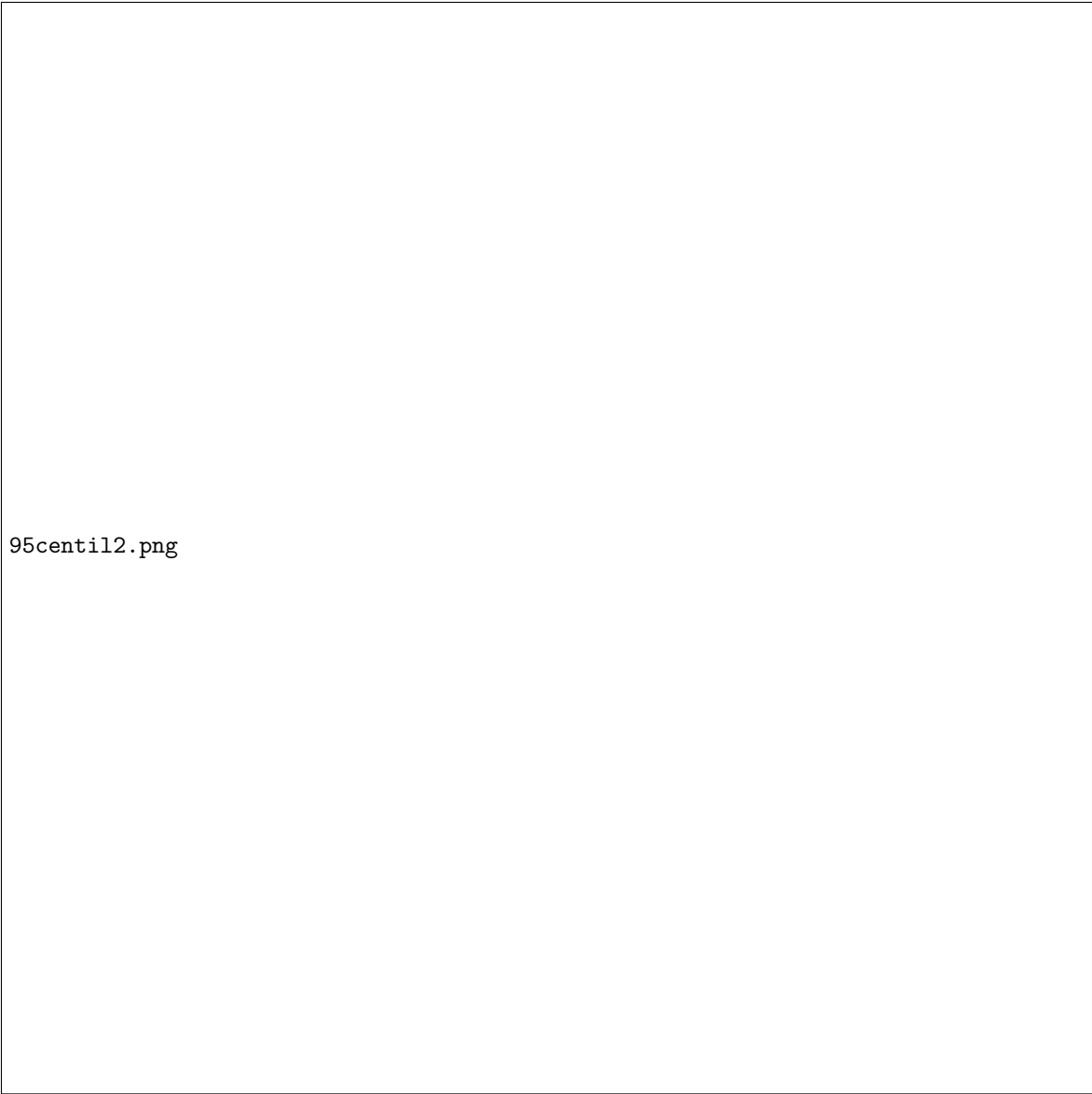
## 2.7 Documentation pour LDN

### 2.7.1 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
```

### 2.7.2 Configuration/création des graphiques prototypes



95centil2.png

```
# 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.
```

### 2.7.3 Configuration/création des déclencheurs prototypes

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



# Configuration des éléments en cliquant sur sélectionner prototype

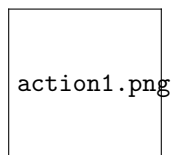


# Sélection de l'élément souhaité : Incomming network traffic on {#IFNAME}.  
# Configuration de l'expression : la dernière (plus récente) valeur T est  $i$ , N  
# Choix de Dernier (T) : 1  
# Choix de N = valeur du centile

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



## 2.8 Image de configuration de l'action



## 2.9 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
}
}
```