Table des matières

ADSL VPN Bonding 3
Côté serveur 4
Côté client 5
Bonding 6
Côté serveur 7
Côté client 8
Firewall & NAT 8
Test de la configuration 9
Balancing manuel 10
QoS 11
ICMP redirect 11

Dernière mise à jour : 05/12/2017
Cette solution ne fonctionne qu'avec un kernel 3.x côté serveur. Le monitoring ARP utilisé par le bonding ne fonctionne plus de la même façon avec un kernel 4.x et je n'ai pas eu le temps de tester/vérifier/faire fonctionner.

Donc cette doc est destinée à un kernel 3.x côté serveur et 4.x côté client.

ADSL VPN Bonding

Le but de cette doc est de montrer comment on peut agréger 2 ou plusieurs lignes ADSL pour avoir :

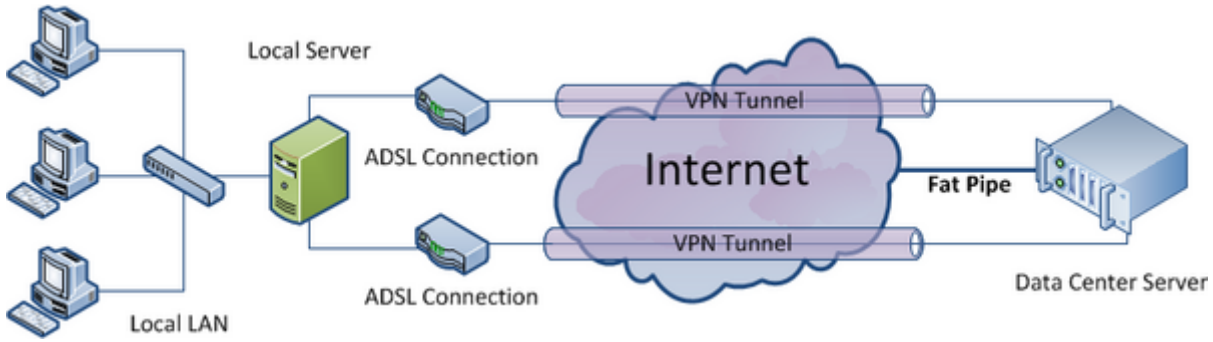
- Agrégation de bande passante **même avec une seule connexion TCP ouverte**
- Failover

On obtient les mêmes résultats avec le nouvelle box d'OVH : https://www.ovhtelecom.fr/overthebox/. Les technos utilisées ne sont simplement pas les mêmes (mTCP pour overthebox, VPN bonding utilisé ici).

On aura besoin de :

- 2 lignes ADSL ;
- 2 modems routeur (les box des FAI suffisent) ;
- Un serveur externe ;
- Un serveur local sous Linux pour aggréger les lignes (raspberry, dd-wrt recyclé, etc);
- Les packages *ifenslave* et *openvpn* sont indispensables sur le serveur et la machine locale

Voici un schéma qui résume ce qu'on veut faire (merci à Simon Mott pour son schéma). Le principe est de créer un tunnel VPN constitué de 2 liens (1 par box et donc par connexion). Les 2 liens seront agrégés via le bonding dispo en standard sous linux.



- **Réseau local** : 192.168.1.0/24
- **Adresse IP du serveur local** : 192.168.1.252
- **Réseau VPN** : 172.16.16.0/30
- **Adresse IP box #1** : 192.168.1.240
- **Adresse IP box #2** : 192.168.1.254
- **Adresse IP VPN distant** : 172.16.16.1
- **Adresse IP VPN local** : 172.16.16.2
- **Adresse IP distant** : A.B.C.D

Ci-dessous le hardware/software utilisé dans ce guide :

Hardware

	CPU	RAM	Modèle
serveur distant	Atom D425 @ 1.80GHz	4 Gb	Kimsufi
serveur local	ARMv7 @ 1 Ghz	1 Gb	Raspberry Pi 3

Software

	OS	Kernel	OpenVPN	ifenslave	bonding driver
serveur distant	Ubuntu 14.04.5 LTS	3.13.0-135-generic	2.3.2	2.4ubuntu1.2	v3.7.1
serveur local	Ubuntu 16.04.3 LTS	4.9.61-v7+	2.3.10	2.7ubuntu1	v3.7.1

Coût

Environ 50€ / mois :

- Box ADSL #1 : 30/35€ par mois

- Frais installation 2ème ligne : entre 40 et 150€ selon le FAI
- Box ADSL #2 : ~2€ via une vente privée ou autre
- Serveur type Kimsufi ou Dedibox : entre 7 et 10€ / mois
- Raspberry : 50€ (avec boitier, alim et carte SD)

Côté serveur

Le but ici est d'avoir 2 interfaces locales et distantes **tap0** et **tap1** qu'on agrégera ensuite pour n'avoir qu'une interface **bond0**.

Sur le serveur distant on va créer 2 instances OpenVPN :

⇒ une qui écoute sur A.B.C.D:XXXX
⇒ une qui écoute sur A.B.C.D:YYYY

- On génère d'abord la clé secrète :

```
cp -R /usr/share/doc/openvpn/examples/easy-rsa/2.0/ /etc/openvpn/easy-rsa/
cd /etc/openvpn/easy-rsa/
openvpn --genkey --secret keys/ta.key
cp keys/ta.key /etc/openvpn
```

- On créé ensuite les 2 fichiers de conf :

Il vous faudra peut-être bidouiller les paramètres mssfix/tun-mtu/fragment

/etc/openvpn/ovh.conf

```
local A.B.C.D
port 1194
proto udp
txqueuelen 1000
dev tap0
dev-type tap
mode p2p
cipher AES-256-CBC
secret ta.key
keepalive 1 2
user nobody
group nogroup
persist-tun
persist-key
comp-lzo no
verb 4
log-append /var/log/ovpn-ovh.log
replay-window 512 15
status /var/log/ovpn-ovh-status.log
```

/etc/openvpn/orange.conf

```
local A.B.C.D
port 1196
proto udp
dev tap1
dev-type tap
txqueuelen 1000
mode p2p
cipher AES-256-CBC
secret ta.key
keepalive 1 2
user nobody
group nogroup
persist-tun
persist-key
comp-lzo no
verb 4
log-append /var/log/ovpn-orange.log
replay-window 512 15
status /var/log/ovpn-orange-status.log
```

- On peut ensuite démarrer les 2 instances OpenVPN sur le serveur distant et vérifier la présence des 2 interfaces **tapX** :

```
/etc/init.d/openvpn start
```

```
root@external_server:/# ifconfig tap0
tap0      Link encap:Ethernet  HWaddr 8e:e6:1a:ff:11:d3
          inet6 addr: fe80::8ce6:1aff:feff:11d3/64 Scope:Link
          UP BROADCAST RUNNING SLAVE MULTICAST  MTU:1500  Metric:1
          RX packets:56846 errors:0 dropped:0 overruns:0 frame:0
          TX packets:137310 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:3118416 (3.1 MB)  TX bytes:26369758 (26.3 MB)

root@external_server:/# ifconfig tap1
tap1      Link encap:Ethernet  HWaddr 8e:e6:1a:ff:11:d3
          inet6 addr: fe80::8ce6:1aff:feff:11d3/64 Scope:Link
          UP BROADCAST RUNNING SLAVE MULTICAST  MTU:1500  Metric:1
          RX packets:133267 errors:0 dropped:0 overruns:0 frame:0
          TX packets:62763 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:7587496 (7.5 MB)  TX bytes:28598900 (28.5 MB)
```

Côté client

Routage

```
                box1 ----- A.B.C.D:XXXX ---------> serveur dédié
               /
 serveur local =>
               \
                box2 ----- A.B.C.D:YYYY----------> serveur dédié
```

La connexion à *A.B.C.D:XXXX* s'effectue via une route statique configurée sur la machine locale (dans mon cas : *box1 - 192.168.1.240*). On ne spécifie pas de gateway par défaut. Celle-ci sera définie par la suite.

```
auto eth0
allow-hotplug eth0
iface eth0 inet static
    address 192.168.1.252
    netmask 255.255.255.0
    post-up /sbin/ip route add A.B.C.D via 192.168.1.240 dev eth0
    post-up /sbin/modprobe bonding
```

Pour forcer la connexion à *A.B.C.D:YYYY* via la *box2 - 192.168.1.254* on créé une table de routage spécifique et on marque les paquets avec iptables :

```
cat << EOF >> /etc/iproute2/rt_tables
2 box2
EOF
```

```
ip rule add from all fwmark 2 table box2
ip route add default via 192.168.1.254 dev eth0 table box2

iptables -t mangle -A OUTPUT -p udp --dport YYYY -j MARK --set-mark 2
iptables -t mangle -A PREROUTING -p udp --dport YYYY -j MARK --set-mark 2
```

- De la même façon on va créer 2 instances OpenVPN “client” pour obtenir 2 interfaces **tapX**.

/etc/openvpn/ovh.conf

```
remote A.B.C.D 1194  # kimsufi (main)
#remote A.B.C.D 1194 # online (backup)
proto udp
dev tap0
dev-type tap
up-delay 2
down-pre
down /root/del_tap0
up /root/add_tap0
txqueuelen 1000
mode p2p
secret ta.key
cipher AES-256-CBC
keepalive 1 2 # server-side controlled, useless here
nobind
persist-key
comp-lzo no
verb 4
log-append /var/log/ovpn-ovh.log
```

```
replay-window 512 15
```

/etc/openvpn/orange.conf

```
remote A.B.C.D 1196  # kimsufi (main)
#remote A.B.C.D 1196 # online (backup)
proto udp
dev tap1
dev-type tap
up-delay 2
down-pre
down /root/del_tap1
up /root/add_tap1
txqueuelen 1000
mode p2p
secret ta.key
cipher AES-256-CBC
keepalive 1 2 # server-side controlled, useless here
nobind
persist-key
comp-lzo no
verb 4
log-append /var/log/ovpn-orange.log
replay-window 512 15
```

- On copie la clé **ta.key** du serveur dans le répertoire */etc/openvpn* du client.
- Créer les 4 scripts ci-dessous :

/root/add_tap0

```
#!/bin/bash

/sbin/ifenslave bond0 tap0
exit 0
```

/root/add_tap1

```
#!/bin/bash

/sbin/ifenslave bond0 tap1
exit 0
```

/root/del_tap0

```
#!/bin/bash

/sbin/ifenslave -d bond0 tap0
exit 0
```

/root/del_tap1

```
#!/bin/bash

/sbin/ifenslave -d bond0 tap1
exit 0
```

Bonding

Pour le bonding 2 modes sont utilisables :

balance-rr

Doublement effectif de bande passante même avec une connexion TCP/UDP unique. Ce mode, avec les connexions TCP, peut faire arriver les paquets dans le désordre. Par exemple si le serveur envoie les paquets 1,2,3,4 ils peuvent arriver dans l'ordre 1,2,4,3. L'algo TCP va considérer que le paquet 3 est perdu car il a reçu le paquet 4 trop tôt. TCP va donc retransmettre les paquets 3 et 4. Avec pour conséquence de dégrader fortement la bande passante (packet reordering) ;

balance-xor

Doublement effectif de bande passante si l'appli cliente est capable d'initier plusieurs connexions en parallèle. Une interface est affectée à l'envoi vers une même adresse MAC. Ainsi les transferts sont parallélisés et le choix de l'interface suit la règle : (Adresse MAC de la source XOR Adresse MAC de la destination) modulo nombre

d'interfaces.

xmit_hash_policy : définit la règle à utiliser pour déterminer l'interface pour les modes balance-xor et 802.3ad. Cette option peut prendre 2 valeurs :

- layer2 : utilise XOR de l'adresse MAC dont la formule est : (source MAC XOR destination MAC) modulo le nombre d'interfaces ;
- layer3+4 : La répartition du trafic se fait par un hash XOR (eXclusive OR ou OU exclusif) en fonction des arguments sélectionnables suivants : les adresses MAC (source et ou destination), les adresses IP (source et ou destination) et le port applicatif (destination);
- layer2 est la valeur par défaut

Dans cet exemple j'ai utilisé le mode *balance-rr* qui a donné de bons résultats. A chacun de tester.

Côté serveur

- Créer le fichier **/etc/modprobe.d/aliases-bond.conf**. Le monitoring MII ne fonctionne pas avec les interfaces **tapX**. On utilisera l'ARP monitoring en spéficiant l'adresse IP distante pour tester la connectivité.

```
alias bond0 bonding
```

- Rajouter l'interface **bond0** au fichier **/etc/network/interfaces** :

/etc/network/interfaces

```
auto bond0
iface bond0 inet static
    address 172.16.16.1
    netmask 255.255.255.252
    bond-slaves none
    bond-mode 0
    bond-arp_interval 300
    bond-arp_ip_target 172.16.16.2

pre-up /etc/init.d/openvpn start
pre-up /sbin/modprobe bonding
post-up /sbin/ip route add 192.168.1.0/24 via 172.16.16.2 dev bond0
post-down /etc/init.d/openvpn stop
post-down /sbin/rmmod bonding

auto tap0
iface tap0 inet manual
        bond-master bond0

auto tap1
iface tap1 inet manual
        bond-master bond0
```

La route créée ici permet au serveur dédié de router les paquets vers notre LAN.

- On peut ensuite démarrer l'interface et checker le bonding :

```
ifup bond0
```

```
root@serveur_distant:~# cat /proc/net/bonding/bond0
Ethernet Channel Bonding Driver: v3.7.1 (April 27, 2011)

Bonding Mode: load balancing (round-robin)
MII Status: up
MII Polling Interval (ms): 0
Up Delay (ms): 0
Down Delay (ms): 0
ARP Polling Interval (ms): 300
ARP IP target/s (n.n.n.n form): 172.16.16.2

Slave Interface: tap0
MII Status: down
Speed: 10 Mbps
Duplex: full
Link Failure Count: 3
Permanent HW addr: ee:b5:b6:90:1a:b4
Slave queue ID: 0

Slave Interface: tap1
MII Status: down
Speed: 10 Mbps
Duplex: full
Link Failure Count: 5
```

```
Permanent HW addr: 52:e9:c3:59:16:8e
Slave queue ID: 0
```

Les 2 liens passeront UP quand la config sera terminée côté client.

- Modifier le fichier /etc/sysctl.conf :

```
net.ipv4.ip_forward = 1
```

Côté client

On fait la même chose côté client :

- Créer le fichier **/etc/modprobe.d/aliases-bond.conf**.

```
alias bond0 bonding
```

- Rajouter l'interface **bond0** au fichier **/etc/network/interfaces** :

/etc/network/interfaces

```
auto bond0
iface bond0 inet static
    address 172.16.16.2
    netmask 255.255.255.252
    bond-mode 0
    pre-up sleep 2
    pre-up /etc/init.d/openvpn start
    post-up /sbin/ip route add 0.0.0.0/0 via 172.16.16.1 dev bond0
    post-down /usr/sbin/service openvpn stop
    post-down /sbin/rmmod bonding
```

- La route par défaut a été ajouté pour faire transiter le trafic par l'IP distante du serveur dédié.
- **/etc/sysctl.conf**

```
net.ipv4.ip_forward = 1
```

- On peut maintenant activer l'interface bond0 :

```
ifup bond0
```

- Et vérifier que tout est OK :

```
root@serveur_local:~#  cat /proc/net/bonding/bond0
Ethernet Channel Bonding Driver: v3.7.1 (April 27, 2011)

Bonding Mode: load balancing (round-robin)
MII Status: up
MII Polling Interval (ms): 0
Up Delay (ms): 0
Down Delay (ms): 0
ARP Polling Interval (ms): 300
ARP IP target/s (n.n.n.n form): 172.16.16.1

Slave Interface: tap1
MII Status: up
Speed: 10 Mbps
Duplex: full
Link Failure Count: 0
Permanent HW addr: 06:eb:b1:94:c5:25
Slave queue ID: 0

Slave Interface: tap0
MII Status: up
Speed: 10 Mbps
Duplex: full
Link Failure Count: 0
Permanent HW addr: 0e:12:c7:d2:e7:b3
Slave queue ID: 0
```

Firewall & NAT

- Ajouter les règles iptables suivantes sur le serveur distant (On peut modifier les règles ci-dessous pour n'autoriser que certaines IPs à se connecter au VPN ⇒ option **-s** de iptables)

```
for i in XXXX YYYY
do
$IPTABLES -A INPUT -i eth0 -p tcp  --sport 1024: --dport $i -m state --state NEW,ESTABLISHED -j ACCEPT
$IPTABLES -A OUTPUT -o eth0 -p tcp --dport 1024: --sport $i -m state --state ESTABLISHED -j ACCEPT
done
```

```
$IPTABLES -A FORWARD -o eth0 -i bond0 -s 192.168.1.0/24 -m conntrack --ctstate NEW -j ACCEPT
$IPTABLES -A FORWARD -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
$IPTABLES -A POSTROUTING -t nat -o eth0 -j MASQUERADE
```

- En fonction de votre firewall il faudra accepter le traffic sur les interfaces **bond0** et **tapX** :

```
for i in tap0 tap1 bond0
do
$IPTABLES -A OUTPUT -o $i -j ACCEPT
$IPTABLES -A INPUT -i $i -j ACCEPT
$IPTABLES -A FORWARD -i $i -j ACCEPT
done
```

- Si vous hébergez un site web en local ou tout autre service il faudra rajouter les règles de NAT suivantes :

```
$IPTABLES -A FORWARD -p tcp --dport 80 -i eth0 -o bond0 -d 172.16.16.2 -m state --state NEW,ESTABLISHED,RELATED -j ACCEPT
$IPTABLES -A FORWARD -p tcp --sport 80 -i bond0 -o eth0 -s 172.16.16.2 -m state --state ESTABLISHED,RELATED -j ACCEPT
$IPTABLES -A PREROUTING -t nat -i eth0 -p tcp -m tcp --dport 80 -j DNAT --to-destination 172.16.16.2:80
```

- A partir de maintenant vous devriez être en mesure de tester un ping de chaque côté :

```
root@local_server:~# ping -c6 172.16.16.1
PING 172.16.16.1 (172.16.16.1) 56(84) bytes of data.
64 bytes from 172.16.16.1: icmp_req=1 ttl=64 time=50.6 ms
64 bytes from 172.16.16.1: icmp_req=2 ttl=64 time=52.4 ms
64 bytes from 172.16.16.1: icmp_req=3 ttl=64 time=38.9 ms
64 bytes from 172.16.16.1: icmp_req=4 ttl=64 time=38.1 ms
64 bytes from 172.16.16.1: icmp_req=5 ttl=64 time=24.7 ms
64 bytes from 172.16.16.1: icmp_req=6 ttl=64 time=37.0 ms

--- 172.16.16.1 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5007ms
rtt min/avg/max/mdev = 24.722/40.328/52.407/9.245 ms
```

```
root@external_server:~# ping -c4 172.16.16.2
PING 172.16.16.2 (172.16.16.2) 56(84) bytes of data.
64 bytes from 172.16.16.2: icmp_seq=1 ttl=64 time=66.2 ms
64 bytes from 172.16.16.2: icmp_seq=2 ttl=64 time=39.2 ms
64 bytes from 172.16.16.2: icmp_seq=3 ttl=64 time=39.7 ms
64 bytes from 172.16.16.2: icmp_seq=4 ttl=64 time=27.7 ms
64 bytes from 172.16.16.2: icmp_seq=5 ttl=64 time=25.9 ms
64 bytes from 172.16.16.2: icmp_seq=6 ttl=64 time=38.5 ms

--- 172.16.16.2 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5007ms
rtt min/avg/max/mdev = 25.929/39.581/66.282/13.169 ms
```

Test de la configuration

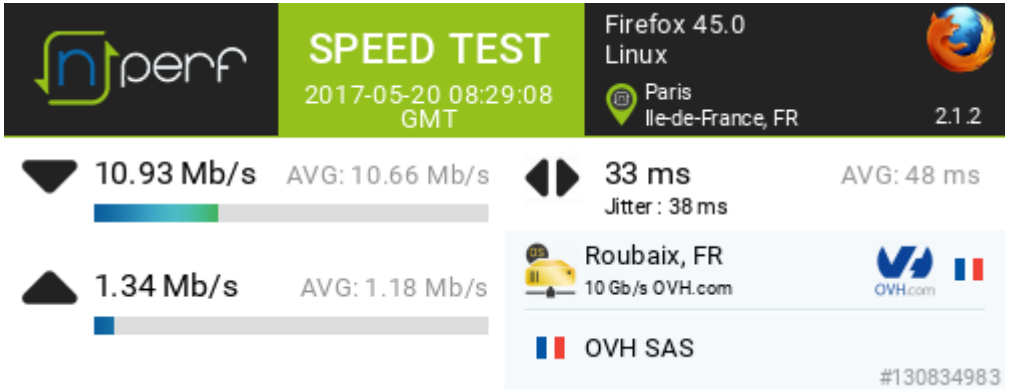
Toutes vos machines du LAN peuvent maintenant utiliser **192.168.1.252** comme gateway et profiter de l'agrégat des deux connexions ADSL.

Si vous surfez sur un site comme http://www.whatismyip.com/ vous devriez voir apparaitre l'IP externe de votre serveur dédié.

Voici quelques tests effectués avec les 2 lignes ADSL ci-dessous :

	Ligne 1 (ovh)	**Ligne 2 (free)**
Down (ATM)	7232 Kbps	7072 Kbps
Up (ATM)	896 Kbps	832 Kbps
Latence (ICMP)	35 ms	15 ms

Soit environ 14 Mbps ATM. Les FAI utilisent la formule : débit IP = 0,83 x débit ATM. Ici on peut espérer environ 11 Mbps en dowload et 1,4 Mbps en upload. On obtient ces débits même avec une seule connexion TCP.



- download

```
bwm-ng v0.6 (probing every 0.500s), press 'h' for help
 input: /proc/net/dev type: rate
        iface                   Rx                   Tx                Total
 ==============================================================================
            tap0:         717.19 KB/s           44.76 KB/s          761.95 KB/s
            tap1:         696.50 KB/s           44.79 KB/s          741.29 KB/s
           bond0:        1413.69 KB/s           89.55 KB/s         1503.24 KB/s
 ------------------------------------------------------------------------------
           total:        2827.38 KB/s          179.09 KB/s         3006.47 KB/s
```

- upload

```
bwm-ng v0.6 (probing every 0.500s), press 'h' for help
 input: /proc/net/dev type: rate
 -         iface                   Rx                   Tx                Total
 ==============================================================================
            tap0:           5.99 KB/s           95.62 KB/s          101.61 KB/s
            tap1:           6.66 KB/s           85.63 KB/s           92.29 KB/s
           bond0:          12.65 KB/s          181.25 KB/s          193.90 KB/s
 ------------------------------------------------------------------------------
           total:          25.30 KB/s          362.50 KB/s          387.80 KB/s
```

```
root@local_server:~# wget http://test-debit.free.fr/65536.rnd -O /dev/null
--2017-05-20 10:30:01--  http://test-debit.free.fr/65536.rnd
Resolving test-debit.free.fr (test-debit.free.fr)... 212.27.42.153, 2a01:e0c:1:1598::3
Connecting to test-debit.free.fr (test-debit.free.fr)|212.27.42.153|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 67108864 (64M) [text/plain]
Saving to: `/dev/null'

100%[======================================================================================>] 67,108,864  1.20M/s   in 54s

2017-05-20 10:30:56 (1.18 MB/s) - `/dev/null' saved [67108864/67108864]
```

Balancing manuel

On peut forcer le trafic à passer par une box en particulier pour bypasser complètement l'agrégat :

- On créé une table de routage spécifique :

```
echo "1 games" >> /etc/iproute2/rt_tables
```

- On créé une règle en indiquant que les paquets marqués '1' feront partie de la table *games* :

```
ip rule add from all fwmark 1 table games
```

- Les paquets de la table *games* transiteront par la box dont l'IP est 192.168.1.254

```
ip route add default via 192.168.1.254 dev eth0 table games
```

- On marque les paquets avec iptables :

```
iptables -t mangle -A OUTPUT -p tcp --dport 6112  -j MARK --set-mark 1              #Guild Wars 2 & Battlenet
iptables -t mangle -A PREROUTING -p tcp --dport 6112  -j MARK --set-mark 1          #Guild Wars 2 & Battlenet

iptables -t mangle -A OUTPUT -p udp --dport 27000:30000 -j MARK --set-mark 1        #Half-Life and co
iptables -t mangle -A PREROUTING -p udp --dport 27000:30000 -j MARK --set-mark 1    #Half-Life and co
```

QoS

J'utilise une version modifiée de wondershaper. Le script est dispo ici.

ICMP redirect

- linux

```
# Disable ICMP Redirect Acceptance
net.ipv4.conf.all.accept_redirects = 0
net.ipv4.conf.lo.accept_redirects = 0
net.ipv4.conf.eth0.accept_redirects = 0
net.ipv4.conf.br0.accept_redirects = 0
net.ipv4.conf.default.accept_redirects = 0
```

- freebsd

```
net.inet.ip.redirect=0
net.inet.icmp.drop_redirect=1
net.inet.icmp.log_redirect=0
```

From:
https://unix.ndlp.info/ - **Where there is a shell, there is a way**

Permanent link:
https://unix.ndlp.info/doku.php/informatique:reseau:xdsl:xdsl_bonding?rev=1512470425

Last update: **2017/12/05 11:40**