Équilibrage de charge par utilisateur

Étude et implémentation par Pierre RIVAULT

Fonctionnement du gestionnaire de charge

Avant d'expliquer les changements apportés à cette partie, il est nécessaire de comprendre comment fonctionnait la répartition de charge avant cette mise à jour.

ALCASAR utilisait directement le programme `ip` du noyau Linux afin de gérer sa répartition de charge. Ceci se faisait en déclarant plusieurs routes par défaut au moyen de la commande :

```
ip route replace default scope global nexthop via ${GW} dev ${IFACE} weight ${WT}
```

On ajoute à cette commande autant de `nexthop` qu'il y a de routes disponibles. Dans chaque `nexthop`, GW désigne l'adresse IP du routeur de sortie, IFACE désigne l'interface par laquelle celui-ci est joignable, et WT désigne le poids que l'on alloue à ce routeur. Ainsi, « ip route » va se trouver à la toute fin de la pile IP, et envoyer les paquets devant passer par la route par défaut sur l'un ou l'autre de ces routeurs en fonction de la charge qu'il estime avoir donné à chaque routeur, et du poids qui leur est associé (un routeur avec un poids de 2 recevra deux fois plus de trafic qu'un routeur avec un poids de 1). Le problème de ce répartiteur de charge est que sa décision de charger tel ou tel routeur se base sur deux critères. Ce fonctionnement, bien qu'il soit toujours correct dans certaines situations, est aujourd'hui relativement obsolète au vu des usages. En effet, la grande majorité de tous les flux réseau d'un utilisateur normal passe par le protocole HTTPS. Les mails, services FTP, sont aujourd'hui souvent passés au travers du protocole HTTPS. Dès lors, le routage se retrouve à rediriger la majorité des flux sur un unique routeur, et laisse les autres routeurs à une charge quasiment nulle, ce qui n'est pas vraiment le but d'un répartiteur de charge.

Au vu de ce constat, la Team ALCASAR a décidé de réaliser son propre répartiteur de charge. Pour pouvoir réaliser ceci, il fallait déterminer un nouveau critère de répartition de charge. Cette tache pouvait aisément être réalisée au sein d'ALCASAR, puisque le logiciel peut en permanence connaitre le nombre de clients connectés au réseau, et donc répartir le trafic entre les routeurs selon ce critère. La répartition de charge allait donc se faire au vu du nombre de clients déjà connectés via tel ou tel routeur, pour connecter tout nouveau client via le routeur le moins chargé au moment de sa connexion.

L'idée est donc la suivante : lors d'une nouvelle connexion, ALCASAR, va regarder combien de clients sont connectés via chaque routeur configuré.

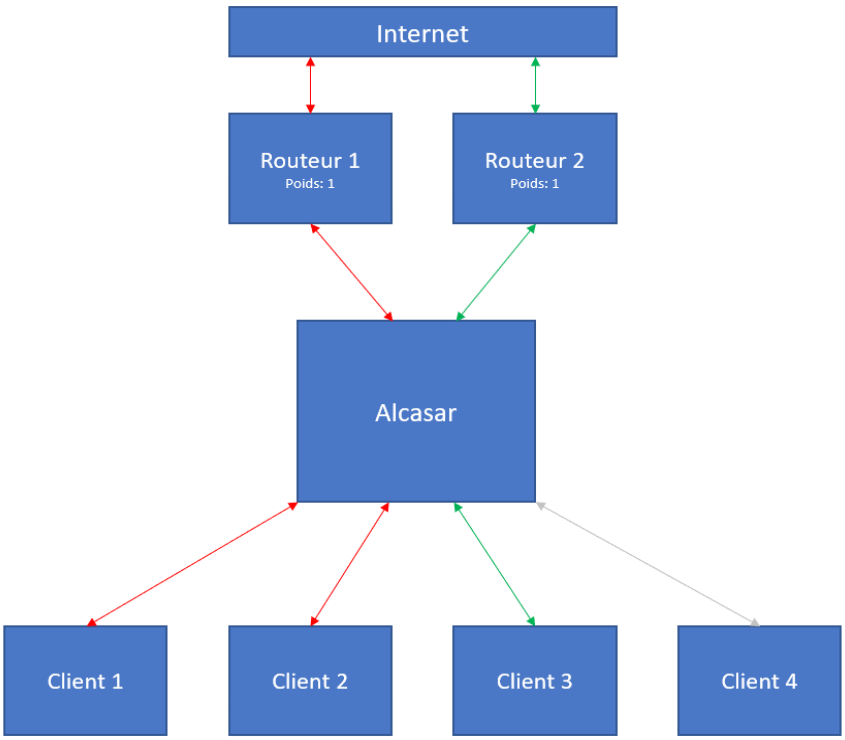

Figure : arrivée d'un nouveau client sur ALCASAR

Ici, on peut voir que 2 clients sont connectés via le routeur 1 et un seul par le routeur 2. Au moment où Client 4 va se connecter au réseau, ALCASAR va donc le connecter via le routeur le moins chargé, à savoir le routeur 2 ici.

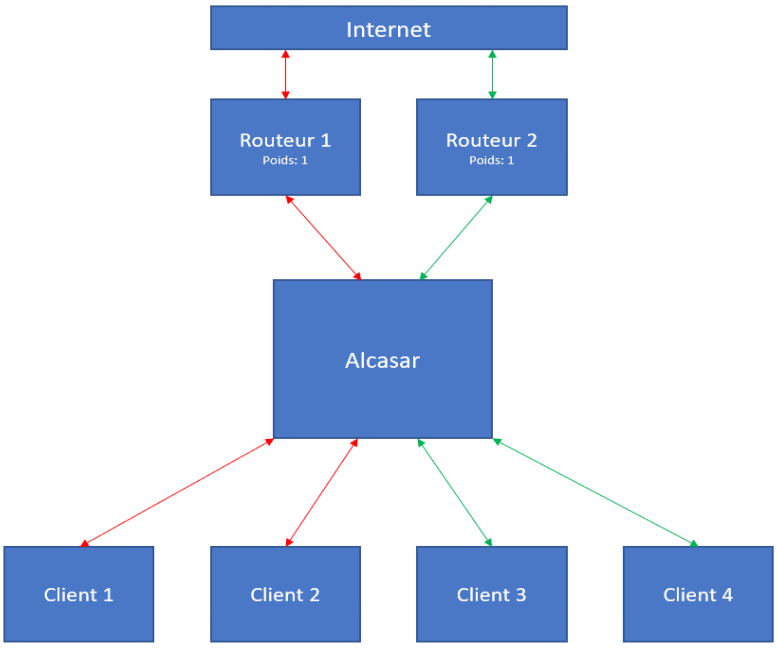

Figure : le client est associé au routeur 2

Cependant, la notion de poids de chaque routeur va être conservée dans cette implémentation de la répartition de charge. Si ici, le routeur 1 avait eu un poids associé de 3 et le routeur 2 un poids associé de 1, Client 4 aurait été connecté via le routeur 1.

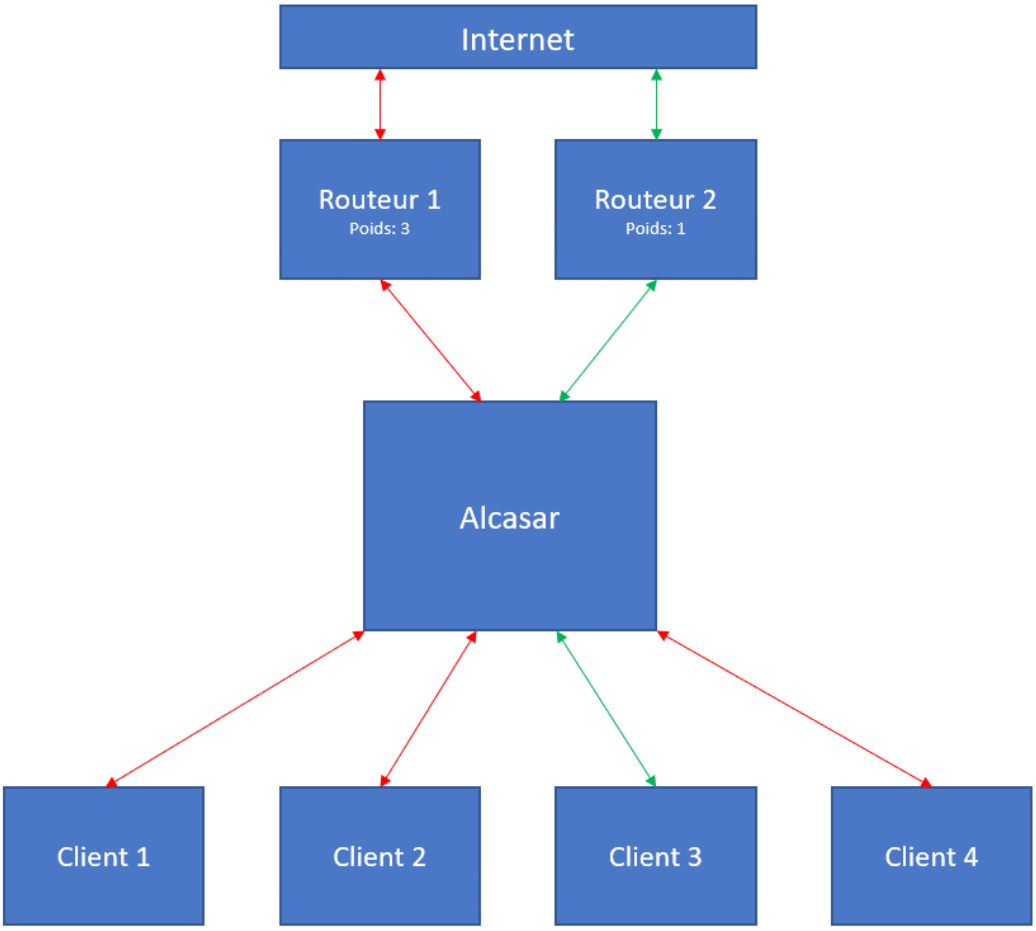

Figure : le client est associé au routeur 1 car son poids est plus important

Pour réaliser cette répartition de charge, de nombreuses briques ont été mises en place :

- ALCASAR doit en permanence avoir une liste associant chaque client (un client est en réalité représenté par une adresse IP) à un routeur ;
- ALCASAR, doit être capable de rediriger correctement ce trafic vers le bon routeur ;
- Il faut qu'ALCASAR puisse associer un nouveau client au routeur le moins chargé ;
- Il faut qu'ALCASAR répartisse à nouveau les charges correctement lorsqu'un client se déconnecte ;
- ALCASAR doit présenter une interface d'administration pour l'ensemble de ces routeurs.

Liste d'association Client-Routeur

Le point très important à prendre en compte lors de la création de cette liste est qu'elle devra être lisible par plusieurs scripts. Il existe de nombreuses méthodes pour partager des données entre différents programmes, que cela soit l'envoi de signaux, la mise en place d'une zone de mémoire partagée ou d'un sémaphore. Toutes ces solutions sont complexes à mettre en place. ALCASAR utilise pour sa part un programme appartenant directement au noyau Linux, qui s'appelle `ipset`. Ce dernier va permettre de stocker des listes de variables liées à la configuration réseau, que cela soit des adresses IP, des adresses MAC ou des numéros de port. Ces variables seront alors disponibles en lecture pour tous les scripts qui en auront besoin, et ils pourront les modifier au moyen d'un appel système à `ipset`. D'autre part, ipset possède l'énorme avantage d'être nativement utilisable dans la création de règles de filtrage iptables. Cela permet ainsi de modifier certains filtres à la volée sans avoir à supprimer et recréer la règle associée.

Pour réaliser la répartition de charge, j'ai choisi d'utiliser une liste ipset par routeur en place. Chaque liste contiendra ainsi la liste des IP clientes associées à un routeur. Par exemple, la liste *gw0* contiendra la liste de tous les clients à faire passer par le premier routeur.

Ces listes seront ainsi utilisées dans plusieurs scripts. Premièrement, le script qui met en place les différentes règles iptables au lancement d'ALCASAR devra également initialiser autant de listes ipset vides qu'il y a de routeurs, et ce afin de pouvoir créer les règles iptables associées, sur lesquelles je reviendrai plus loin dans ce rapport. Ensuite, le script de connexion d'un utilisateur doit pouvoir ajouter un élément au sein des différentes listes ipset, et le script de déconnexion d'un utilisateur doit pouvoir retirer un élément de ces mêmes listes. D'autre part, ces listes seront également lues par Netfilter lorsqu'il réalisera le filtrage des paquets qui lui arrivent.

Redirection du trafic client vers le bon routeur.

Dans l'entièreté de ce chapitre, les clients seront tous supposés authentifiés

Chaque client ayant maintenant son adresse IP contenue dans une liste spécifique à un routeur, il faut être capable de rediriger le flux venant de chaque client sur le bon routeur. Ceci se passe en plusieurs étapes.

Premièrement, lorsqu'un client veut ouvrir une connexion vers une machine externe, ALCASAR va recevoir un paquet en ce sens. Ce paquet va tout d'abord être présenté à Netfilter, dans la partie prerouting, c'est-à-dire avant que le routage ait eu lieu. Netfilter va marquer le paquet avec un nombre en fonction de la liste ipset dans laquelle se trouve l'ip source de ce paquet.

```
#Marquage pour le load balancing
if [ "$MULTIWAN" == "on" ] || [ "$MULTIWAN" == "On" ]; then
    temp_index=200
    for i in $gw_list; do
        $IPTABLES -A PREROUTING -t mangle -i $TUNIF -m set --match-set $i src -j MARK --set-mark $temp_index
        temp_index=$(($temp_index+1))
    done
fi
```

Figure 4: marquage des paquets en fonction de leur appartenance à un ipset

Le paquet reçu va ainsi se retrouver marqué de la manière suivante : 200 s'il doit être routé au travers du premier routeur, 201 s'il doit être routé au travers du second, et ainsi de suite.

Une fois ce marquage fait, le paquet va suivre la pile IP de manière tout à fait normale, en étant supprimé s'il correspond à un filtre de blocage, et en ayant son adresse IP source modifiée puisque ALCASAR réalise du SNAT afin que le routeur puisse renvoyer le paquet de retour à ALCASAR.

À la toute fin de la pile IP, au moment du routage du paquet, c'est le programme `ip route` qui va s'occuper de déterminer vers quel routeur envoyer le paquet, et c'est là que le marquage intervient.

Pour réaliser cette association, le programme `ip rule` va créer une table d'association entre le marquage d'un paquet et la table `ip route` à utiliser pour ce type de paquet.

```
alcasar-rasacla:~# ip rule
0:      from all lookup local
32763:  from 192.168.182.0/24 fwmark 0xca lookup 202
32764:  from 192.168.182.0/24 fwmark 0xc9 lookup 201
32765:  from 192.168.182.0/24 fwmark 0xc8 lookup 200
32766:  from all lookup main
32767:  from all lookup default
```

Figure 5: table d'association ip rule dans un exemple à 3 routeurs

Ici, on peut voir les règles `ip rule` créées dans un exemple à 3 routeurs. La partie *fwmark* de chaque ligne désigne le marquage des paquets, et la partie *lookup* désigne la table `ip route` à utiliser pour ce type de marquage. Les lignes 0, 32766 et 32767 sont les règles par défaut nécessaires à la connectivité internet d'ALCASAR en lui-même.

Il est possible de configurer différentes tables dans ip route, qui vont permettre de router les paquets différemment les uns des autres.

```
alcasar-rasacla:~# ip route show table 200
default via 192.168.50.1 dev enp0s3
192.168.50.0/24 dev enp0s3 scope link src 192.168.50.20
alcasar-rasacla:~# ip route show table 201
default via 192.168.50.2 dev enp0s3
192.168.50.0/24 dev enp0s3 scope link src 192.168.50.20
alcasar-rasacla:~# ip route show table 202
default via 192.168.50.3 dev enp0s3
192.168.50.0/24 dev enp0s3 scope link src 192.168.50.20
```

Figure 6: exemple d'une configuration avec 3 routeurs

Ainsi, un paquet marqué 200 ne sera pas envoyé sur la route par défaut, mais sur la route par défaut de la table 200, qui sera ici le premier routeur qui aura été configuré. De même, un paquet marqué 201 sera envoyé vers la route par défaut de la table 201, qui sera le second routeur configuré dans ALCASAR.

La répartition de charge peut ainsi être résumée de la manière suivante :

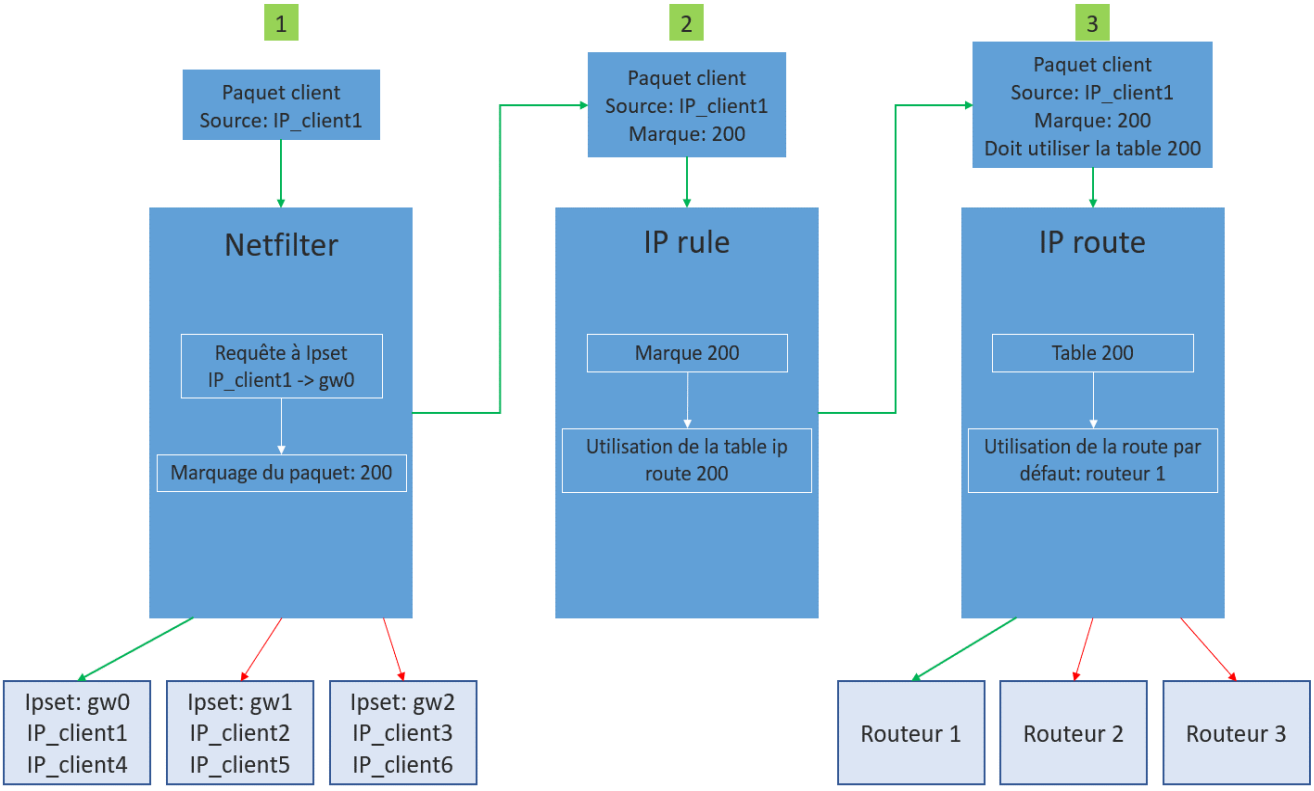

Figure 7: schéma de routage pour la répartition de charge

Connexion d'un nouveau client à ALCASAR

Lorsqu'un nouveau client se connecte depuis le réseau local, et que son authentification a réussi, il faut ajouter son IP dans la bonne liste ipset afin qu'il soit routé vers le bon routeur, à savoir le routeur le moins chargé au moment de sa connexion. Pour réaliser ceci, Coovachilli, qui est le programme qui gère la connexion des utilisateurs, possède un Hook-up lors d'une connexion réussie, c'est-à-dire qu'il va exécuter un script défini par ALCASAR à chaque fois qu'une connexion aura réussi. C'est dans ce script qu'ALCASAR va, entre autres, pouvoir remplir la bonne liste ipset.

Le choix du routeur à utiliser se fait finalement selon une relation mathématique très simple. Pour chaque routeur, on calcule la valeur déterminée par le nombre d'utilisateurs associés à ce routeur, c'est-à-dire le nombre d'entrées dans la liste ipset associée à ce routeur, et on divise ce nombre par le poids que l'on a attribué à ce routeur. On ajoute alors le nouveau client au routeur ayant obtenu le score le plus petit (le premier de la liste en cas d'égalité).

Ce travail est ainsi effectué par le script suivant :

```
#put the user in an ipset for load-balancing
gw_min="gw0"
weight=`grep ^PUBLIC_WEIGHT= $CONF_FILE | cut -d"=" -f2`
already=`ipset list $gw_min | grep Number\ of\ entries: | cut -d":" -f2`
#The *1000 is here to avoid working on floats in bash
gw_min_value=$((1000 * $already / $weight))

nb_gw=`grep ^WAN $CONF_FILE | wc -l`
for (( i = 1 ; i <= $nb_gw ; i++ ));do
    gw="gw${i}"
    weight=`grep ^WAN$i= $CONF_FILE | awk -F'"' '{ print $2 }' | awk -F ',' '{ print $2 }'`
    already=`ipset list $gw | grep Number\ of\ entries: | cut -d":" -f2`
    value=$((1000 * $already / $weight))
    if [ $value -lt $gw_min_value ]
    then
        gw_min_value=$value
        gw_min=$gw
    fi
done
ipset add $gw_min $FRAMED_IP_ADDRESS
```

Figure 8: script de sélection du routeur pour le nouveau client

On peut voir ici que le script fait simplement défiler tous les routeurs, et conserve en permanence le routeur ayant le score le plus petit. On peut également remarquer que ce score est multiplié par 1000. Ceci permet d'obtenir des nombres comparables, étant donné que Bash ne sait pas travailler avec les nombres à virgule (tout du moins pas nativement, il existe des moyens de « tricher » avec awk, mais ils sont lourds et complexes à mettre en œuvre).

Déconnexion d'un client d'ALCASAR

Lors de la déconnexion d'un client d'ALCASAR, deux possibilités ont été envisagées :

- On recalcule la répartition de l'intégralité des utilisateurs après la déconnexion afin que la répartition de charge soit optimale en permanence.
- On supprime simplement l'utilisateur de sa liste ipset, et la place sera fort probablement prise par le prochain client à se connecter à ALCASAR.

La seconde solution a été retenue, d'une part parce qu'elle est beaucoup moins longue et lourde à effectuer pour ALCASAR, et d'autre part parce qu'elle est beaucoup plus simple à implémenter. Ipset comporte en effet une option pour supprimer une entrée d'une liste, aussi le script suivant suffit à la suppression de l'utilisateur du bon ipset.

```
# Remove IP address from ipset of load balancing
nb_gw=`grep ^WAN $CONF_FILE | wc -l`
for (( i = 0 ; i <= $nb_gw ; i++ ));do
    gw="gw$i"
    ipset test $gw $FRAMED_IP_ADDRESS 1>/dev/null 2>&1
    if [ $? -eq 0 ];then
        ipset del $gw $FRAMED_IP_ADDRESS
        break
    fi
done
```

Figure 9: Suppression d'un client d'ipset après sa déconnexion

On a ainsi une manière simple et efficace de supprimer les clients déconnectés.

Afin de rentre l'utilisation de cette répartition de charge plus simple, j'ai également ajouté une interface web pour gérer et configurer ses routeurs.

En effet, dans les précédentes versions d'ALCASAR, la configuration de la répartition de charge ne se faisait que dans un terminal en éditant un fichier de configuration à la main. Avec un répartiteur de charge plus efficace, il était logique d'également simplifier son utilisation.

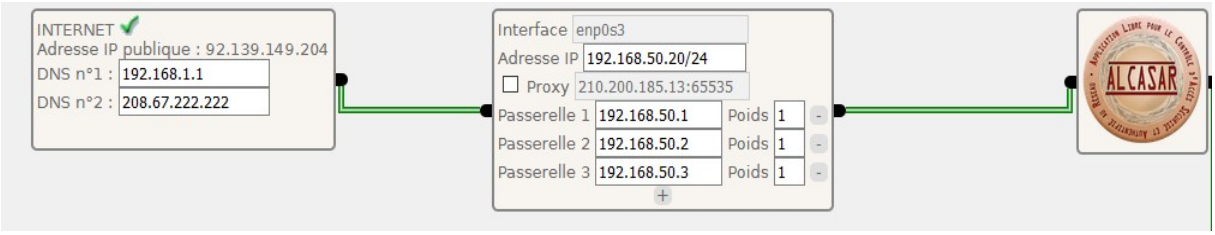

Figure 10: interface de gestion des routeurs

L'interface de configuration a été réduite au strict nécessaire. Il est possible d'ajouter une passerelle (un routeur), de supprimer chaque passerelle indépendamment, et de donner à chaque passerelle un poids. De plus, lorsqu'il n'y a qu'un seul routeur, il n'est pas possible de lui donner un poids.

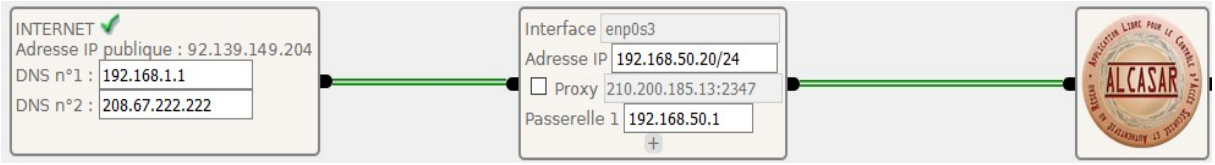

Figure 11: le poids n'est pas disponible avec un seul routeur

D'autre part, la possibilité de configurer un proxy a également été ajoutée à l'interface web d'ALCASAR. Lorsqu'un proxy est configuré, il ne peut pas y avoir de répartition de charge, puisque tout le trafic est envoyé vers une seule et même machine, donc l'interface ne laisse pas la possibilité de configurer les deux simultanément.

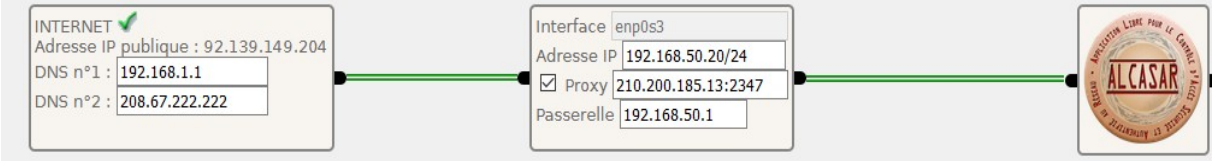

Figure 12: Une seule passerelle est disponible lorsqu'un proxy est présent

Afin d'appliquer ces paramètres, la page web va simplement aller remplir le fichier de configuration général d'ALCASAR, nommé `alcasar.conf`. Cependant, elle ne va pas remplir ce fichier avec des informations qui pourraient être erronées, et qui pourraient créer une instabilité du système. La page va effectuer une vérification du format des données entrées par l'utilisateur avec de simples regex, ce qui va permettre d'éviter la majorité des fautes de frappe. D'autre part, cette vérification permet également de signifier à l'administrateur dans quelle case une faute a été entrée.

```
$reg_ip      = '/^(([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])\.){3}([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])$/';
$reg_ip_cidr = '/^(([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])\.){3}([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])(\/([0-9]|[1-
$reg_ip_port = '/^(([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])\.){3}([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])(\:([1-9]|[1-
$reg_mac     = '/^([0-9A-Fa-f]{2}[:-]){5}([0-9A-Fa-f]{2})$/';
$reg_host    = '/^[a-zA-Z0-9-_]+$/';
$reg_weight  = '/^[0-9]*$/';
```

Figure 13: exemple de regex utilisées

```
if (isset($_POST['int_ip_0']) && (trim($_POST['int_ip_0']) !== $conf['PRIVATE_IP'])) {
    if (!preg_match($reg_ip_cidr, $_POST['int_ip_0'])) {
        $ext_conf_error = true;
        $ext_conf_error_list[] = $l_error.': '.$l_ip_address.' LAN 0: '.$l_error_bad_ip_CIDR;
    }
    file_put_contents( filename: TEMP_FILE, str_replace( search: 'PRIVATE_IP='.$conf['PRIVATE_IP'],
    $modification_internal = true;
}
```

Figure 14: chaque variable va voir son format vérifié

Lorsque le script effectue la vérification du format de chaque variable, si le format n'est pas respecté, il va passer la valeur d'une variable booléenne à 1. Dans ce cas, le script continue de passer en revue toutes les variables entrées par l'administrateur, afin de détecter d'autres erreurs éventuelles, mais il n'écrira pas les changements dans le fichier de configuration.

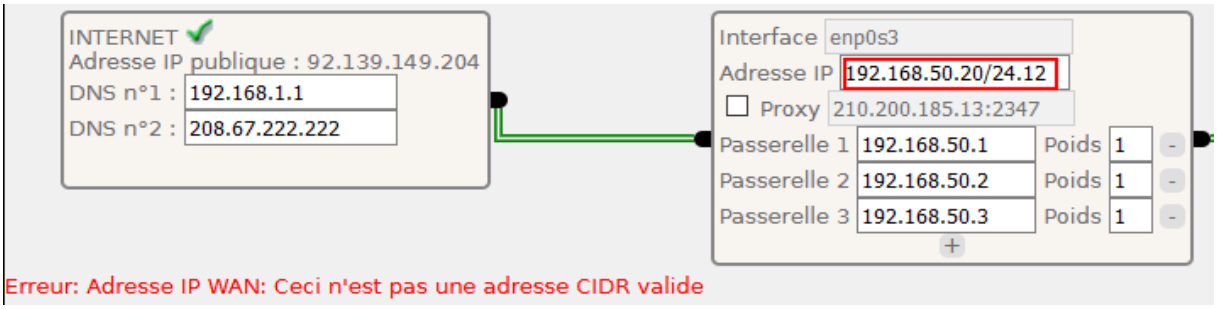

Figure 15: affichage d'une erreur de configuration, l'adresse IP externe d'ALCASAR ne respecte pas le format CIDR

L'administrateur pourra alors modifier la variable non conforme et resoumettre ses modifications. Le script est capable de gérer un très grand nombre de routeurs à la fois (testé jusqu'à 12 routeurs en parallèle, situation hautement improbable en réalité).

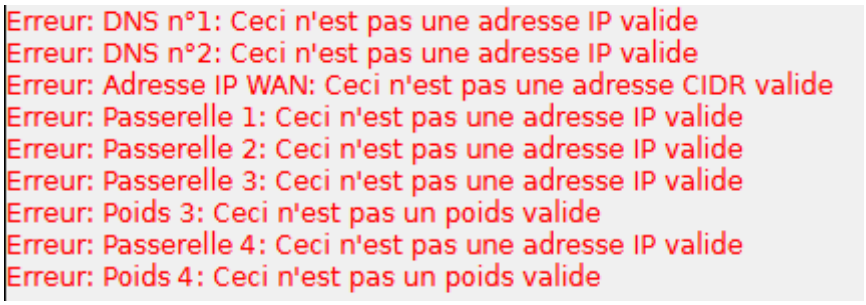

Figure 16: erreurs nominatives pour un grand nombre de routeurs

Le script intègre également une petite correction automatique d'erreur sur les poids des passerelles. Ainsi, lorsqu'on ajoute une passerelle dans l'interface, le poids est défini par défaut à 0, ce qui n'est pas un poids valide étant donné qu'il va servir en tant que diviseur (comme indiqué dans le paragraphe sur le fonctionnement de la répartition de charge). Lorsque le script va rencontrer une valeur de poids égale à 0, il ne va pas la compter comme une erreur, mais simplement modifier cette valeur par 1.

Enfin, lorsqu'aucune erreur n'est rencontrée, le script le signifie à l'utilisateur et va aller remplir le fichier de configuration d'ALCASAR.

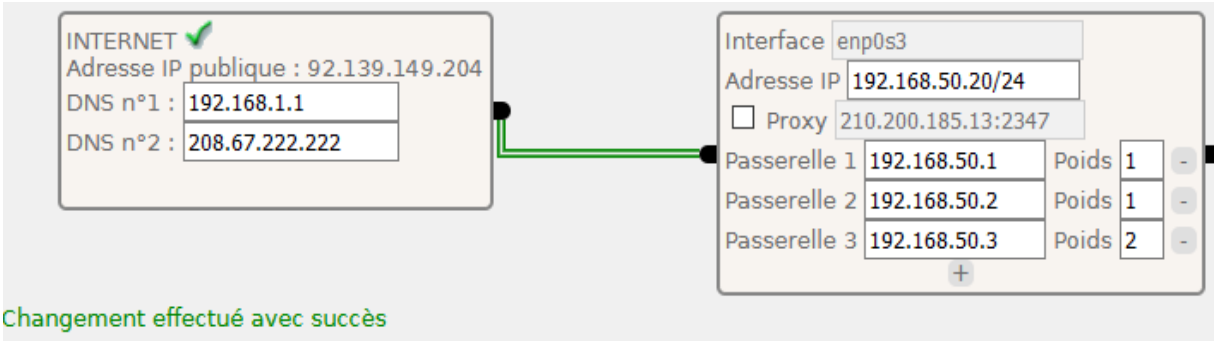

Figure 17: aucune erreur de configuration n'a été détectée, les changements sont appliqués

```
46 ## WANx=GWx,Weight
47 MULTIWAN=On
48 WAN1="192.168.50.2,1"
49 WAN2="192.168.50.3,2"
```

Figure 18: Données sauvegardées dans le fichier de configuration

Application des paramètres

Une fois que les paramètres ont été rentrés par l'administrateur, vérifiés par le script, et sauvegardés dans le fichier de configuration, il reste encore à les appliquer dans ALCASAR.

Pour ce faire, le script de la page d'administration va relancer le moins de services possible au vu des modifications effectuées, afin de prendre le moins de ressource possible sur ALCASAR.

```
//DNS values modification, several services needs to be reloading, reloads the full server.
if ($modification_dns) {
    exec( command: 'sudo /usr/local/bin/alcasar-conf.sh -apply');
}
//External network modifications, no service reloading
if ($modification_network) {
    exec( command: 'sudo /usr/local/bin/alcasar-network.sh');
    exec( command: 'sudo /usr/local/bin/alcasar-iptables.sh');
}
//If only the proxy has been modified, only the firewall needs a change
else if ($modification_proxy) {
    exec( command: 'sudo /usr/local/bin/alcasar-iptables.sh');
}
```

Figure 19: choix des scripts à exécuter pour appliquer la configuration

On peut voir ici que la modification la plus lourde est le changement d'un des deux serveurs DNS du serveur, car son application va relancer l'intégralité des services d'ALCASAR. En effet, plusieurs services ont besoin de connaître l'adresse de ces serveurs à leur lancement, et ils ne peuvent pas être lancés dans n'importe quel ordre. Le plus simple consistait donc à tout relancer. Heureusement, cette modification n'est que très rarement effectuée.

Dans le cas d'une modification du proxy, seul iptables a besoin d'être relancé. Une simple règle de DNAT suffit pour utiliser le proxy, mais le script est intégralement exécuté afin de supprimer les éventuelles règles devenues obsolètes.

Dans le cas d'une modification de la configuration IP externe d'ALCASAR, les modifications effectuées sont plus importantes, c'est pourquoi un nouveau script a été ajouté à ALCASAR. Ce script va tout d'abord modifier la configuration IP de l'interface externe d'ALCASAR, en modifiant simplement le fichier `/etc/sysadmin/network-scripts/ifcfg-extif`. Ensuite, ce script va créer les règles de routage associant le marquage des paquets à une table de routage.

```
#remove the routing rules
ip rule flush
#set back the main and default rules
ip rule add from all lookup main pref 32766
ip rule add from all lookup default pref 32767
#add the rule for the first gateway
ip rule add from ${PRIVATE_NETWORK_MASK} fwmark 200 lookup 200
```

Figure 20:suppression des anciennes règles de routage et création de la première route

Une fois cette première règle créée, le script va, pour chaque routeur, créer la règle qui lui est associée. Il va également créer les tables de routage pour chaque routeur

```
for ((i=0 ; $i < $nb_gw_supp ; i++)); do
    #this number is used to mark the paquets in order to route them to the choosen gateway
    table=$(($i + 201))
    GW=`grep ^WAN$(($i + 1))= $CONF_FILE|awk -F'"' '{ print $2 }' | awk -F, '{print $1}'`
    #add the others route in their respective tables
    ip route add ${NET} dev ${EXTIF} src ${IP} table $table
    ip route add default via ${GW} table $table
    #add the rule for each rule depending of the mark set by the firewall
    ip rule add from ${PRIVATE_NETWORK_MASK} fwmark $table lookup $table
    #add the added gateway into the default gateway
    routecmd="${routecmd} nexthop via ${GW} dev ${EXTIF}"
done
```

Figure 21: création des règles et tables de routage pour les autres routeurs

Une fois toutes ces règles et ces tables de routage créées, le script d'application d'iptables est exécuté, et la répartition de charge d'ALCASAR est fonctionnelle.

Problème connu

Pour fonctionner, la répartition de charge d'ALCASAR a besoin de marquer les paquets. Cependant, d'autres briques d'ALCASAR vont également intervenir sur ces paquets, afin de journaliser leur présence et d'analyser leur contenu. C'est notamment le cas d'E²Guardian, un proxy HTTP et HTTPS, qui n'est actuellement actif que sur l'HTTP sur ALCASAR. Le problème est que le passage par le proxy fait perdre le marquage aux paquets, et ceux-ci possèdent également tous la même adresse IP source après le passage dans E²Guardian, à savoir celle d'ALCASAR lui-même. La sélection de routeurs pour ce paquet est alors impossible, et il part simplement par la route par défaut. Le problème est que pour remettre en place une répartition de charge selon le même schéma, il faudrait pouvoir récupérer l'adresse source réelle d'un paquet après son passage dans le proxy, ce qui n'est pas faisable simplement avec un pare-feu. Ceci pourrait être une piste de recherche pour ALCASAR dans le futur.