RAPPORT NOUVELLES TECHNOLOGIES DE L'IP

MULTICAST ET IPv6

Auteurs:
Morin Thibaud & de Broux Mathias

ESIREM 5A - SQR

Table des matières

Table des figures

1 Tests/modification d'une application d'envoi/réception en multicast IPv6

ping6

Tout d'abord, on se propose ici de tester deux commandes :

```
$ ping6 -I em1 FF02::1
```

Avec cette commande, nous obtenons un ping normal car c'est une adresse qui est faite pour écouter tous les messages qui arrivent dessus.
En revanche, lorsque nous exécutons la deuxième commande :

```
$ ping6 -I em1 FF02::1
```

Nous observons une perte de paquets de 100% étant donné que cette adresse ne sert qu'à envoyer et ne peut pas recevoir de paquets exceptés des requêtes du type "router discovery".

in2multi6 et multi2out

Le programme multi2out6 sert de récepteur : il s'abonne au groupe qu'on lui donne avec la commande :

```
$ ./multi2out -i em1 FF02::9269
```

Le programme in2multi6 sert quant à lui d'émetteur : il envoie au groupe qu'on lui précise de la même manière que multi2out6 écoute. Lorsque l'on envoie un message sur le terminal dans lequel in2multi6 est entrain d'être exécuté, le terminal avec multi2out6 affiche ce message s'il est abonné au même groupe que in2multi6. De cette manière, il est possible de mettre plusieurs récepteurs au sein du groupe et ils recevront tous le message envoyé par l'émetteur.

MLD et Wireshark

Pour analyser les messages d'adhésion à un groupe multicast, nous avons utiliser l'outil Wireshark. Notre poste possédant l'adresse `fe80::fab1:56ff:fecf:9794`, nous avons décidé avec notre voisin du groupe `FF02::9456`. On obtient bien une trame d'initiation, comme vu à la Figure 1 pour l'accès au groupe multicast, aussi appelé MLD.

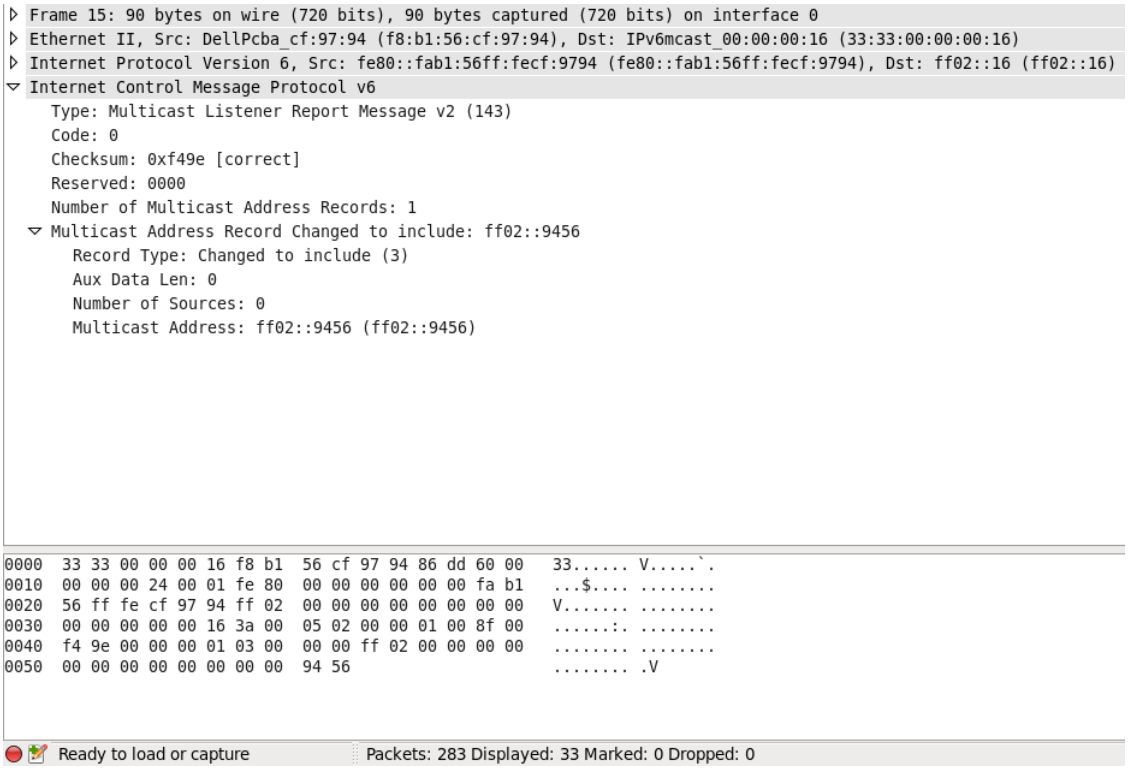

FIGURE 1: Analyse d'un message MLD d'adhésion via Wireshark

L'adresse source de cette demande est celle de notre poste soit `fe80::fab1:56ff:fecf:9794`, et l'adresse de destination est celle du groupe soit `33:33:00:00:00:16`. Une fois ceci terminé, il enverra un nouveau message afin de terminer la communication (Multicast Address Record Changed to exclude : FF02 : :9456) comme montré a la Figure 2

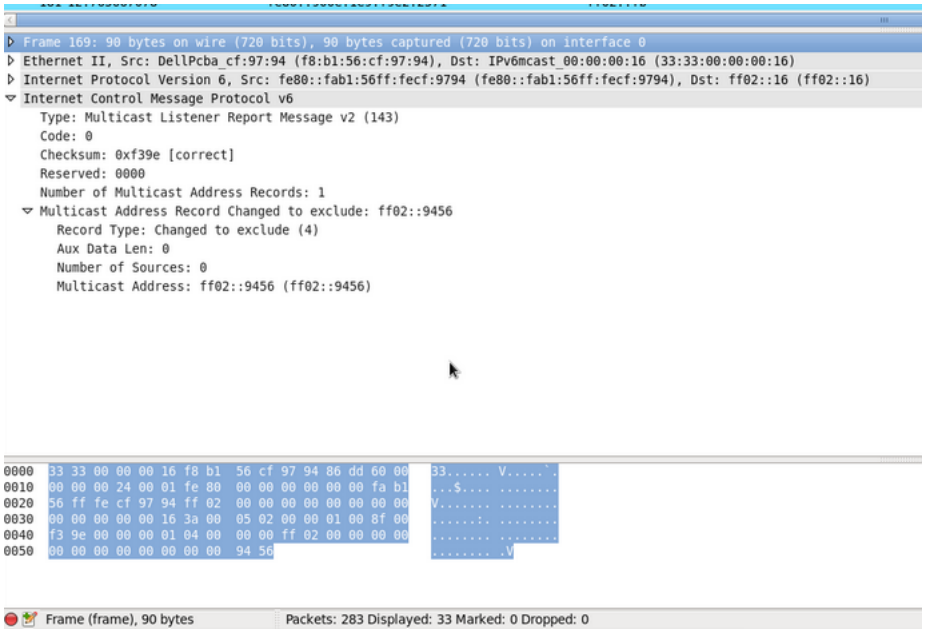

FIGURE 2: Analyse d'un message MLD d'exclusion via Wireshark

Modifications programme

Nous avons eu à afficher le message puis l'adresse IPv6 émettrice, pour ce faire on procède comme suit :

1. Nous rajoutons une structure afin de contenir l'adresse :

```
struct sockaddr_in6 src6
```

2. On rajoute cette ligne pour que b contienne le message reçu :

```
b = (recvfrom(s, buf, 10240, 0,(struct sockaddr *) &src6, &recv6_len))
```

3. Ensuite, on affecte tout le message avec le nombre de caractères (il prendra en compte maintenant b et plus cc) :

```
ccb = write(1, buf, b)
```

4. Pour afficher la source de chaque message, on ajoute l'adresse IPv6 dans l'affichage du message grâce à la fonction `inet_ntop` :

```
inet_ntop(AF_INET6, &src.sin6_addr, str_src, INET6_ADDRSTRLEN);
printf("Message venant de l'adresse %s : \n", str_src);
```

Les modifications majeures peuvent être vues dans la Figure 3 ci-dessous :

```
[tm340623@ad.u-bourgogne.fr@gr13-01l Téléchargements]$ ./multi2out -i em1 FF02::
9456
 Message venant de l'adresse fe80::fab1:56ff:fecf:971f :
salut
<--[6]--
^C
```

FIGURE 3: Aperçu modifications fichier

Le terminal affiche bien l'adresse IPv6 de notre voisin, ainsi que son message et le nombre de caractère du message reçu (en comptant le point final invisible).

Réécriture programme en Java

Le programme écrit en Java à l'aide du package Netbrans est disponible en pièce jointe. Le programme `SendMulticast` est l'équivalent Java de `in2multi6` tandis que `ReceiveMulticast` est l'équivalent Java de `multi2out6`

2 Pour aller plus loin

Question 1

Pour permettre la mise en place de réseaux multicast, le mieux serait une topologie en 3 réseaux distincts. Chaque réseau posséderait un récepteur (donc 3 récepteurs au total) et 2 des réseau auraient un émetteur. Cet émetteur va alterner d'un réseau à l'autre tout en envoyant des informations et le but sera de vérifier si les 3 récepteurs le reçoivent.

Question 2

À priori il n'est pas possible d'utiliser le protocole PIM-SM sur des hôtes linux utilisant le package mcast-tools car ce package n'implémente pas ce protocole. Il faudra donc ajouter d'autres packages supportés.

Question 3

Il est possible, en utilisant le logiciel de simulation GNS3, de simuler le protocole PIM-SM afin de réaliser des tests.Cependant, pour ce faire, il faudra désigner un routeur comme Rendezvous Point (RP).

3 Travail de synthèse

Protocole HIP

Le but d'une adresse IP est en première partie, d'identifier une machine sur un réseau donné, et en seconde partie, elle permet de savoir quelle route utilisée pour rejoindre cette machine, à l'aide des réseaux et sous réseaux. A l'heure actuelle, il existe de fortes contraintes sur les réseaux qui nous pousseraient à vouloir se tourner vers un modèle d'adressage IP fixe. Cependant, en matière de sécurité et d'évolutivité de l'environnement, cette pratique est contradictoire. Le protocole HIP permet de répondre à ces contradictions en introduisant la notion de clé cryptographique et d'identifiant. Une machine posséderait toujours une adresse IP, cependant, cette dernière ne jouerait plus que le rôle d'identifiant sur le réseau. C'est la clé cryptographique qui serait désignée comme identificateur.

Le protocole HIP à été rendu disponible en version stable en 2006 sous le RFC 4423 cependant, l'IETF [1] avait commencé l'étude de ce protocole dès 1999. Cette version resta très expérimentale jusqu'à la sortie de l'a version 2.0 d'HIP, le RFC 7401 publié en 2015. Un des plus gros problèmes de la première version était la sécurité même au sein du protocole. La version de 2015 vient pallier à ce problème en ajouté en plus de la sécurité, la notion de mobilité et de `namespace` ainsi qu'une nouvelle couche appelée Host Identity Layer comme présenté dans la Figure 4

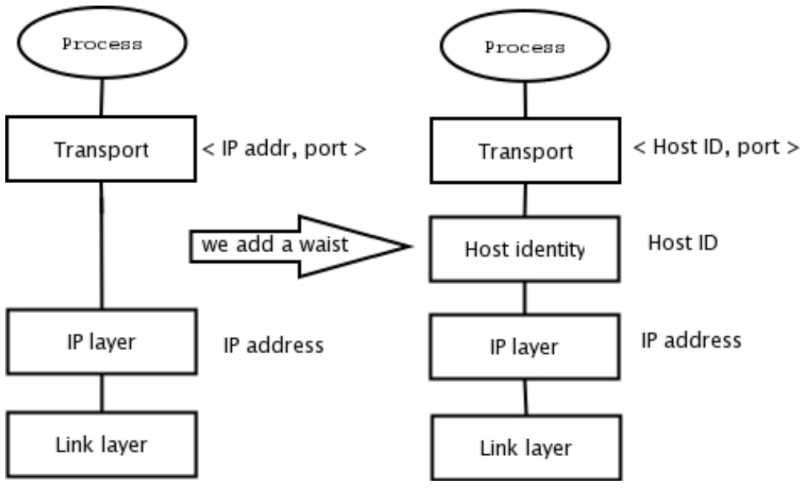

FIGURE 4: Comparaison HIP v1 et HIP v2 - RustRepo©

1. Internet Engineering Task Force

La couche Host Identity est composée de Host Identifiers. Un Host Identifier est une clé publique issue d'une paire asymétrique. Un hôte doit posséder au moins un Host Identifier mais en général il en possédera plus d'un. Quand on utilise ce protocole, on l'encapsule généralement dans IPSec. Pour ce qui est de l'implémentation, il faut ajouter le namespace directement directement dans le noyau IP des machines souhaitant l'utiliser. Un Host Identity Tag est une représentation d'un Host Identity au format 128-bits. Ils sont rangés dans un préfixe IPv6 spécial (`2001:20::/28`) pour éviter les conflits avec les adresses IPv6 normales.

Le HIT peut être comparé à une empreinte SSH, mais à contrario, il peut être utilisé par toutes les applications. Ce protocole prend également en charge les noms compatibles avec IPv4 appelés LSI [2]. Les Tags dans HIP sont statistiquement uniques et très sûrs car ils sont basés sur des clés publiques, ce qui rend la chose presque impossible à fausser.

A l'inverse du schéma standard, les sockets des protocoles de transport comme TCP, sont liés à ces tags plutôt qu'aux adresses IP directement. D'abord, la pile réseau va traduire les tags en adresses IP avant de les transmettre. L'inverse se produit sur l'hôte qui reçoit les paquets HIP. Lorsque l'hôte change de réseau, la pile réseau change l'adresse IP pour la traduction. L'application ne remarque pas le changement car elle utilise la constante HIT. Grâce à ça, ce protocole règle le problème de mobilité des utilisateurs sans contraintes. La Figure 5 présente un échange classique entre deux machines par le biais du protocole HIP :

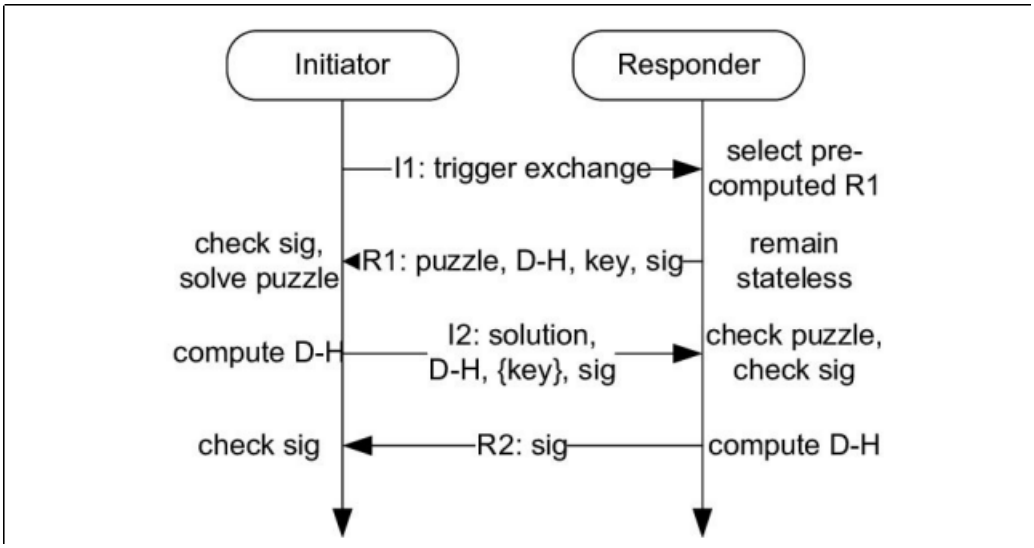

FIGURE 5: Échange entre 2 machines avec le protocole HIP - Researchgate©

2. Local Scope Identifiers

La première étape "trigger exchnage" est initiée par la machine souhaitant une connexion (“Initiator” pour Initiateur dans le schéma). Elle envoie donc son Host Identity Tag et celui que l'autre machine (“Responder” pour Répondeur sur le schéma) devra utiliser. La réponse, appelée R1 contiendra la confirmation de l'utilisation des tags (ainsi que ces derniers) et un puzzle de la machine qui le reçoit doit résoudre. Ce puzzle sert au Responder à contrôler sa charge de traitement et adapte la complexité du puzzle en fonction de celle ci, ce qui peut être très apprécié lors d'attaques, par exemples, par déni de service. Le message I2 contient la solution du puzzle ainsi que les tags. Si le puzzle est valide, cela constitue l'authentification auprès des deux machines avec une confirmation (R2).

Pour conclure, les avantages du protocole HIP sont la résolution des problèmes de multi-connexion et de mobilité ainsi que sa sécurité. Son principal désavantage est qu'il nécessite d'être implémenté dans la pile IP des machines.

Sources

— https ://www.slideserve.com/chiara/host-identity-protocol
— https ://www.bortzmeyer.org/separation-identificateur-localisateur.html
— https ://www.rfc-editor.org/rfc/rfc6079.html
— https ://hal.archives-ouvertes.fr/hal-01373423/document
— https ://tools.ietf.org/html/rfc4423
— https ://www.bortzmeyer.org/hip-resume.html
— https ://www.security7.net/news/what-is-host-identity-protocol-hip