Projet Réseaux RE203: "Simulation d'un Réseau de Routeurs"

Mises à jour :

Consulter régulièrement cette page, des modifications seront publiées si nécessaire :

- [03/04/2013] Publication du sujet.

Remarques

- Les élèves doivent s'organiser en équipes de 4-5 élèves.
- La programmation se fera obligatoirement en C et un langage orienté objet (C++, Java, C#). Par exemple le nœud « Contrôleur » en C et les nœuds « Routeur » en Java.
- Un rapport intermédiaire d'avancement est à remettre à la fin de la 5ème séance.
- La 10ème séance est réservée aux soutenances du projet. 15 minutes par équipe (présentation, démo et réponses aux questions). Le rapport final doit rendu avant la soutenance.
- Les fichiers sources doivent être fournis dans une archive à la fin de la 9ème séance

Le but du projet est de réaliser un simulateur distribué de routeurs (en pair-à-pair). Chaque équipe devra programmer un nœud routeur et un nœud contrôleur. Chaque routeur a pour mission de router des paquets qui lui sont envoyés vers la bonne destination. Le nœud contrôleur s'occupera d'informer les routeurs de la topologie actuelle du réseau.

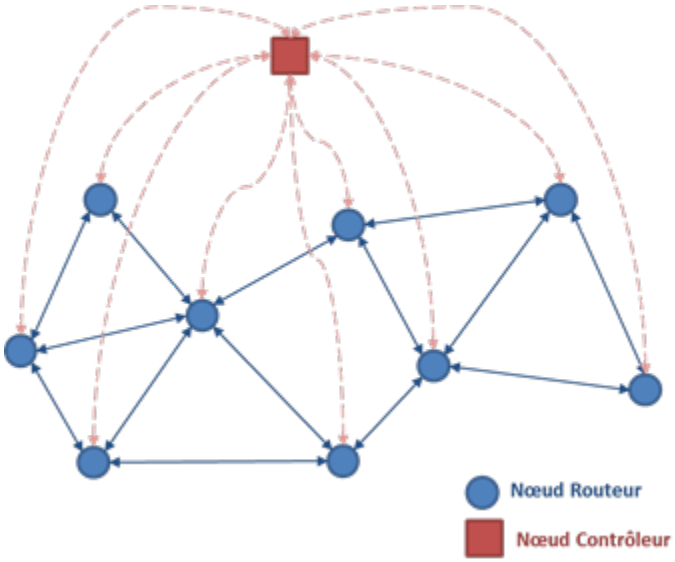


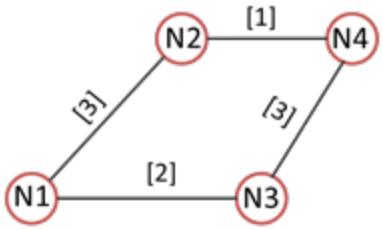
Figure 1: Topologie Réseau

1. Programme "contrôleur"

Le contrôleur s'appuie sur un fichier d'entrée qui décrit la topologie initiale du réseau. Il contient la liste des successeurs et coût associés de chaque routeur composant le réseau. L'exemple suivant illustre la structure du fichier. Il s'agit d'un graphe sous le format "dot", manipulable par le logiciel GraphViz. On peut l'utiliser pour afficher le graphe sous divers format png, pdf, ps, svg, etc. à l'aide de la commande suivante:

```
$ dot –Tpng –o topology.png topology.dot
$ dot –Tpdf –o topology.pdf topology.dot
$ dot –Tsvg –o topology.svg topology.dot
```

```
graph G {
   n1 [label="N1"];
   n2 [label="N2"];
   n3 [label="N3"];
   n4 [label="N4"];
   n1 -- n2 [label="3"];
   n1 -- n3 [label="2"];
   n2 -- n4 [label="1"];
   n3 -- n4 [label="4"];
}
```



Le programme contrôleur devra permettre des commandes simples pour charger une topologie, la modifier en cours du temps, et la sauvegarder si besoin. Les commandes suivantes doivent être implémentées:

($ Signifie l'invite de commande du Contrôleur)

§ Chargement d'une topologie

```
$ load Topology1.txt
    -> Topology loaded (4 nodes in the network) !
```

§ Afficher la topologie actuelle

```
$ show topology
   n1 [label="N1"];
   n2 [label="N2"];
   n3 [label="N3"];
   n4 [label="N4"];
   n1 -- n2 [label="3"];
   n1 -- n3 [label="2"];
   n2 -- n4 [label="1"];
   n3 -- n4 [label="4"];
```

§ Ajouter un lien entre deux nœuds en spécifiant le coût du lien.

```
$ add link N1 N4 5
    -> Link created between N1 and N4. Link cost = 5.
```

§ Supprimer un lien entre deux routeurs.

```
$ del link N1 N4
    -> Link deleted between N1 and N4.
```

§ Supprimer tous les liens à partir d'un routeur. Dans ce cas, le routeur est toujours actif mais il se retrouve déconnecté du réseau.

```
$ disconnect N3
    -> Node N3 is disconnected !
```

§ Modifier le coût d'un lien dans le réseau.

```
$ update link N1 N2 1
```

```
    -> Link updated between N1 and N2. Link cost = 1.
```

§ Enregistrer / exporter la topologie actuelle dans un fichier.

```
$ save Topology2.txt
    -> Topology saved !(4 nodes in the network)
```

2. Programme "routeur"

1ère commande: Message

Au niveau de chaque routeur, l'utilisateur peut envoyer un message texte à n'importe quel autre routeur s'il connait son identifiant. La syntaxe de cette commande est la suivante :

```
$ message N3 "Hello there"
    -> message sent at hh:mm:ss.ms !
    -> message delivered ! RTT = 123.45 ms

$ message N3 "Hello there"
    -> message sent at hh:mm:ss.ms !
      -> error : message not delivered ! (Delta > X
seconds)

$ message N3 "Hello there"
    -> error: unknown destination !
```

À la validation de cette commande, le router envoie le paquet au routeur destinataire ou au routeur susceptible de transmettre le message au routeur destinataire. Si la destination n'est pas connue du routeur, un message d'erreur sera affiché et aucun paquet n'est transmis sur le réseau.

2ème commande: Ping

Grâce à cette commande, l'utilisateur pourrait tester la connectivité avec un autre routeur. La syntaxe de cette commande ainsi que les différents résultats sont les suivants:

```
$ ping N3
    -> from N1 to N3 time=hh:mm:ss.ms
    -> from N1 to N3 time=hh:mm:ss.ms
    -> from N1 to N3 time=hh:mm:ss.ms

    -> Result: x% success, y% failure
    -> RTT   : min=123.45ms avg=123.45 max=123.45

$ ping N3
    -> error: host unreachable !

$ ping N3
    -> error: unknown destination !
```

Cette commande permettrait de mesurer le temps d'aller-retour (RTT: Round Trip Time) entre les deux routeurs.

3ème commande: TraceRoute

Cette commande permettrait de connaitre la route empruntée par les paquets jusqu'à arriver à une certaine destination, ainsi que la route inverse.

La syntaxe de la commande ainsi que les résultats escomptés sont les suivants:

```
$ route N3
    -> error: unknown destination !

$ route N3
    -> from N1 to N3
         Hop 01 = N2
         Hop 02 = N4
         Hop 03 = N3
         Hop 04 = N5
         Hop 05 = N2
         Hop 06 = N1

         Path = 6 hops
         RTT = 123.ms
         Symmetric path = TRUE / FALSE
```

4ème commande: RouteTable

Cette commande permet à l'utilisateur de consulter la table de routage du routeur.

```
$ routetable
    *--------+--------+------*
    |  dest  |  next  | cost |
    *--------+--------+------*
    |   N1   |   -    |   0  |
    |   N2   |   N2   |   1  |
    |   N3   |   N3   |   2  |
    |   N4   |   N3   |   4  |
    |   N5   |   N2   |   3  |
    *--------+--------+------*
```

3. Protocole des communications

Chaque routeur communique périodiquement durant toute sa durée de vie avec le nœud contrôleur. À l'initialisation, il reçoit du nœud contrôleur la liste des voisins directs et établira avec ces derniers des connexions TCP permanentes jusqu'à ce qu'il y ait un changement de la topologie, notifié par le nœud contrôleur. Ainsi, il est indispensable de charger une topologie au niveau du contrôleur avant le lancement des routeurs.

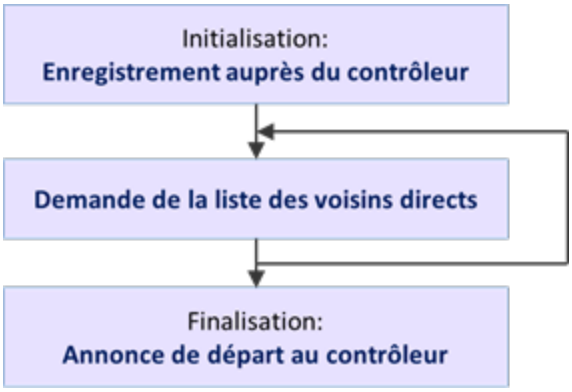


Figure 2: Communications Routeur- Administrateur

Le schéma des communications entre un routeur et le contrôleur est illustré sur la Figure 2. Au début, le routeur contacte le contrôleur dont l'adresse et le numéro de port sont connus de tous les nœuds routeurs.

Après l'enregistrement, le routeur demande périodiquement la liste de ses voisins directs au contrôleur. Au début, le nœud routeur obtient une liste de voisins. Un voisin est décrit par son identifient, le cout du lien associé, l'adresse IP et le port d'écoute.

Lors des demandes futures, s'il n'y a aucun changement du voisinage, le nœud contrôleur n'envoie pas de liste.

Si un nœud routeur reçoit une nouvelle liste de voisins (**NewList**), il doit adapter ses connexions:

- En se déconnectant des voisins qui ne sont plus présents dans la nouvelle liste, i.e. **OldList - NewList**
- En se connectant aux nouveaux voisins, i.e. **NewList - OldList**
- En maintenant les connexions avec les voisins, i.e. **OldList Ç NewList**

Conventions:

- Dans ce qui suit, le signe ">" indique le message qui a été envoyé, et le signe "<" le message reçu en réponse.
- Tous les messages sont en texte, et sont terminé par le caractère "*".
- Un caractère d'échappement devra être utilisé pour le caractère *. Exemples:
  - 2*3=6 Le paquet contiendra la charge utile : 2*3=6
  - 6\3=2 Le paquet contiendra la charge utile : 6\\3=2

A. Communications Routeur <-> Contrôleur

1. Initialisation

```
> log in as ID port p*
< greeting ID*
```

À l'initialisation, le routeur communique au contrôleur son port d'écoute. Il peut aussi spécifier un identifiant à utiliser.

§ Si l'identifiant demandé est libre, le contrôleur autorise son utilisation.

```
> log in as N3 port 3333*
< greeting N3*
```

§ Si le routeur n'a pas demandé d'identifiant, le contrôleur lui attribue un identifiant libre

```
> log in port 2222*
< greeting N2*
```

§ Si l'identifiant demandé n'existe pas dans la topologie, ou est déjà attribué à un autre routeur, le contrôleur renvoi un autre identifiant libre.

```
> log in as N404 port 4040*
< greeting N4*
```

2. Demande périodique de la topologie:

a. *Première demande ou lors d'un changement du voisinage*

```
> poll*
< neighborhood newlist [ID1,Cost1,IP1:Port1; ID2,Cost2,IP2:Port2; ...
; IDN,CostN,IPN:PortN]*
```

La topologie retournée par le contrôleur peut aussi être vide. Dans ce cas, le nœud va couper toutes les connexions déjà établies avec d'autres routeurs.

b. *Aucun changement du voisinage*

```
> poll*
< neighborhood ok*
```

3. Finalisation

```
> log out*
< bye*
```

Si au bout d'un certain temps, le contrôleur, ne reçoit aucun message d'un nœud routeur. Celui-ci est supprimé de la topologie.

B. Communications Routeur <-> Routeur

1. Connexion

Après réception de la liste des voisins, un nœud routeur doit établir une connexion TCP avec chaque voisin. La connexion est finalisée par l'envoi d'un message "*Link*" auquel le voisin répond par un acquittement positif.

```
> link*
< link ok*
```

2. Échange périodique des vecteurs de distance

Une fois la connexion établie, les routeurs vont s'échanger périodiquement (période paramétrable, par défaut = 10sec) leurs vecteurs de distance respectifs. Dans le vecteur envoyé, on trouve toutes les destinations connues du routeur ainsi que le cout calculé ou estimé vers cette destination.

```
> vector [Dst1,Cost1; Dst2,Cost2; . . . ; DstN,CostN]*
< vector ok*
```

3. Paquets

Un routeur qui veut envoyer un message à un autre routeur construit et envoie un paquet

contenant son adresse, l'adresse de destination finale, une valeur pour la durée de vie du paquet en termes de nombre de sauts (TTL: Time To Live), et enfin la charge utile du paquet (une chaine de caractères). Ce paquet sera transmis de proche en proche jusqu'à arriver à la destination finale. Le routeur de destination devra notifier la bonne réception du message en envoyant un acquittement à la source initiale du message.

```
> packet src idSrc dst idDst ttl x data …*
< packet src idDst dst idSrc ok*
```

Si en cours d'acheminement, la valeur du TTL atteint la valeur 0, le paquet est détruit et une notification est envoyée au nœud source

```
> packet src idSrc dst idDst ttl x data …*
< packet src idDst dst idSrc toofar*
```

Un routeur peut tester la connectivité avec un autre routeur dans le réseau. À cet effet, il envoie un paquet de contrôle "ping". Ce paquet sera transmis de proche en proche jusqu'à atteindre le nœud destinataire. Ce dernier devra répondre avec un paquet de contrôle "pong" qui fera le chemin inverse jusqu'au nœud source. Le nœud source pourra donc mesure le temps entre l'émission du paquet "ping" et la réception du paquet "pong".

```
> ping src id dst id ttl val*
< pong src id dst id ttl val*
```

Finalement, un routeur pourrait explorer la route le séparant d'un autre routeur en envoyant une série de paquets "ping" en incrémentant à chaque paquet la valeur du TTL et en commençant à 1 (cf. outil traceroute).

4. Configuration des programmes:

Chaque programme devra s'appuyer sur un fichier de configuration disponible dans le même dossier du fichier exécutable avant le lancement du programme.

Fichier de configuration (*router.cfg*):

```
# Numéro de port d'écoute
router-port = 8888

# Adresse IP du nœud contrôleur
controller-address = a.b.c.d

# Numéro de port TCP d'écoute du nœud contrôleur
controller-port = 12345

# Nombre de secondes avant de recontacter le contrôleur.
controller-update-interval = 30

# Intervalle en secondes pour l'échange périodique de vecteurs de
distances.
router-update-interval = 10

# Valeur par défaut du TTL.
default-ttl-value = 8
```

```
# Valeur par défaut du nombre de paquets émis lors d'un ping.
default-ping-packet-count = 3

# Valeur par défaut du TimeOut pour la réponse d'un message
default-packet-timeout-value = 1

# Valeur par défaut du TimeOut pour retirer un routeur du voisinage.
default-dv-timeout-value = 15
```

Fichier de configuration (controller.cfg):

```
# Numéro de port TCP d'écoute
controller-port = 12345
# Temps entre 2 requêtes d'un routeur au-delà duquel le routeur est retiré
du réseau.
poll-timeout-value = 45
```