TP 3, 4 et 5 : Etude des algortihmes de routages et calcul de métriques, simulateur OpenSM

Objectifs des 3 séances de TPs :

Durant les trois séances de TPs, vous allez pouvoir étudier quelques algorithmes de routage utilisés pour les topologies de réseaux avancés. Nous allons donc implémenter un programme permattant d'évaluer la performance d'un algorithme de routage selon les topologies étudiées.

Deux topologies (fichiers) ont été générées compatibles avec Ibsim : Topo1.topo et Topo2.topo. Vous pouvez les télécharger depuis Célène (cf TP3&4/supports TP 3, 4 et 5). Egalement, une solution des TP 1 et 2 est aussi disponible sous Célène (cf TP1&2/Solutions).

Travail à rendre :

A l'issue de la séance de TP, un compte rendu (CR) avec en annexe les codes sources de votre application devraint être déposés sous CELENE. Pensez donc à préparer votre CR dès maintenant, à chaque fois que vous répondez à une question.

1. Installation et prise en main d'OpenSM

OpenSM (Subnet Manager) est l'agent qui permet de générer des tables de routage dans votre réseau. Ce programme permet d'avoir des tables de routage pour chaque commutateur du réseau et affectera des identifiants uniques pour chaque noeud.

Une fois que le réseau est virtualisé avec Ibsim, OpenSM l'utilise pour pouvoir générer donc ces dites tables de routages.

Remarque : **pour que OpenSM tourne, il faut lancer Ibsim sur un autre terminal en exécutant le fichier définissant la topologie (par exemple, reprenez votre fichier exemple.topo)**.

Travail à réaliser :

1- Installer OpenSM avec la commande suivante :

```
\# sudo apt-get update
\# sudo apt-get install opensm
```

2. Vérifiez que l'installation a bien été faite en analysant les options offertes par OpenSM.

Maintenant qu'OpenSM est installé, voyons comment fonctionne t'il pour créer les tables de routage.

3- Reprenez l'exemple simple vu lors des TPs précedents "exemple.topo" : fichier décrivant le réseau simple de la Figure 1. Lancer la commande qui permet de charger exemple.topo par Ibsim puis donner la commande permettant de l'exécuter.

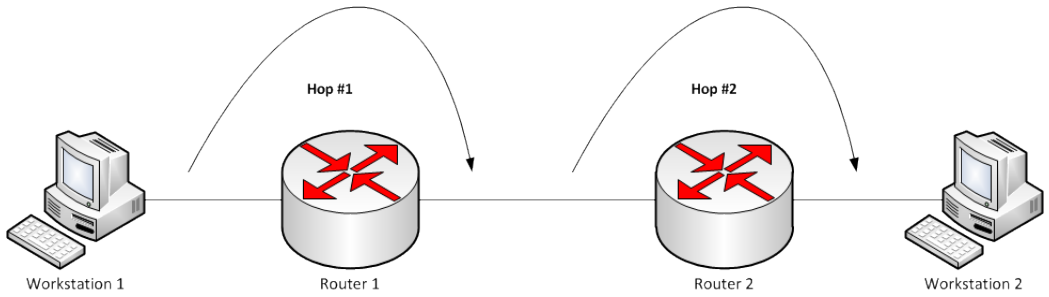

FIGURE 1 –

```
#'Type' 'nombre de port' 'nom'
Hca        1        "workstation1"
#'numérot port' 'nom du device à connecter' 'son port'
[1] "router1"[1]

Hca        1        "workstation2"
[1] "router2"[2]

Switch        2        "router1"
[1] "workstation1"[1]
[2] "router2"[1]

Switch        2        "router2"
[1] "router1"[2]
[2] "workstation2"[1]
```

Question : Remarquez que le champ <**lid**> de chaque équipement du réseau est égal à 0. A quoi correspond chaque LID ? A ce stade, que **lid** sont tous identiques n'est pas contraignant. C'est au moment du calcul des tables de routage, OpenSM leur attribue un **lid** unique.

Commande permettant de charger la bibliothèque libumad2sim : Exécutez la commande suivante dans un nouveau terminale :

```
\# sudo SIM_HOST="H-0013970201000978" OSM_TMP_DIR=./
OSM_CACHE_DIR=./LD_PRELOAD=/usr/lib/umad2sim/libumad2sim.so
```

Attention : Il s'agit d'une seule ligne de commande (pas d'espace, ni de retour à la ligne).

Lancement d'OpenSM : Pour lancer OpenSM, exécutez la commande suivante :

```
\# sudo LD_PRELOAD=/usr/lib/umad2sim/libumad2sim.so opensm -e -v -f ./osm.log
```

Pour visualiser le réseau, vous devez exécuter à nouveau la ligne de commande suivante (bien entendu depuis Ibsim) :

```
\# dump
```

Question : Que constatez-vous ?

2. Edition de tables de routage par OpenSM

Pour récupérer les tables de routages de chaque commutateur, vous devez exécuter la commande suivante dans un nouveau terminal :

```
\# LD_PRELOAD=/usr/lib/umad2sim/libumad2sim.so dump_lfts > exemple.route
```

Question : En vous basant sur le réseaux de l'exemple simple, expliquez le résultat du fichier obtenu (exemple.route) ?

3. Etudes de routage dans deux topologies avancées

3.1 Etudes de topologies avancées

Question : Dites à quelle topologie correspond chacun des deux fichiers Topo1.topo et Topo2.topo par rapport aux deux figures ci-dessous. Justifiez votre réponse en précisant leur nom.

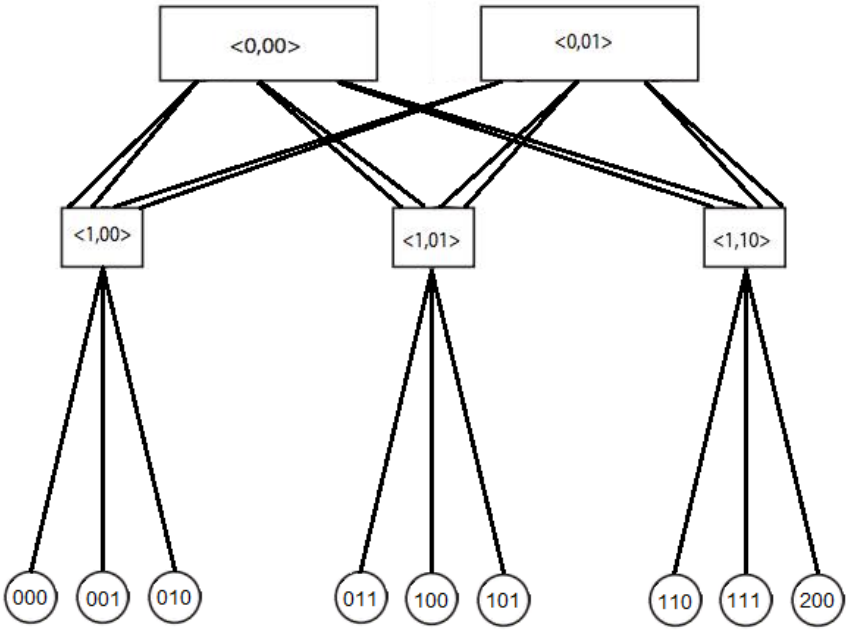

FIGURE 2 –

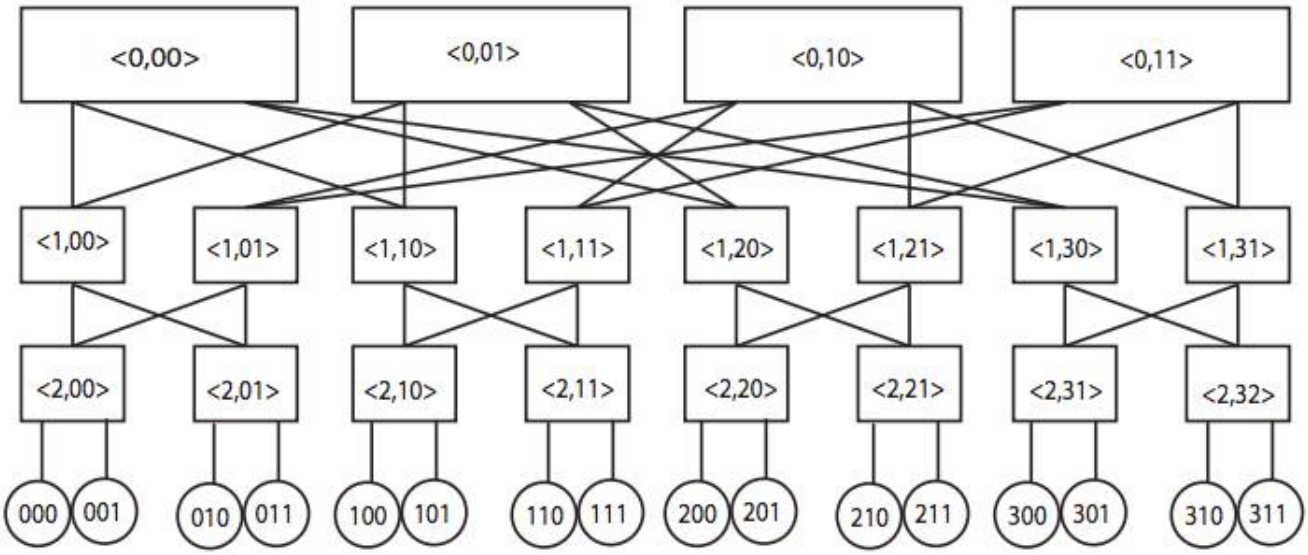

FIGURE 3 –

Question : Citez quelques avantages et inconvénients de chacunne ?

Question : Générez les **tables de routages** de chaque topologie selon la commande vue précédement.

Question : Selon chaque topologie, déduisez les chemins pour un packet partant (justifiez vos réponses et illustrez les sur les schémas) :

Du **HCA 000** vers **HCA 010**

Du **HCA 000** vers **HCA 200**

3.2 Quelques algorithmes de routage

OpenSM implémente plusieurs algorithmes de routage, comme :

- **MINHOP**
- **UPDN**
- **FTREE**
- **DOR (Dimension Order Routing)**
- **LASH (Layered Shortest Path)**

Question : Expliquez le principe de chacun de ces algorithmes (que cherchent-ils à optimiser) ?

3.3 Quelques métriques pour l'étude des performances des algorithmes de routage

Par défaut OpenSM utilise l'algorithme **MINHOP**. C'est l'algorithme de routage qu'on va étudier ses performances dans la suite. L'objectif ici est de pouvoir évaluer les performances de MINHOP appliqué sur les deux réseaux Topo1 et Topo2. Nous vous proposons donc de développer et d'implémenter deux métriques suivantes :

- **Le nombre de sauts minimal**

Cette métrique permet d'évaluer si un algorithme utilise le plus court chemin pour arriver à une destination ou non. Il calcule pour chaque liaison/chemin, le nombre de sauts nécessaires pour arriver à la bonne destination (i.e : la longueur du chemin). La fonction objectif de cette métrique est la suivante :

$$max\left(min_{\forall(s,t)}\,|\pi_{s,t}|\right)$$

Algorithme : calcul du nombre de sauts maximum à partir des tables de routages obtenues

Data: Nodes : Liste des noeuds
Result: Le nombre de sauts max dans la topologie étudiée
foreach *src in Nodes* **do**
  cpt ← 0 ; maxVal ← 0 ;
  foreach *dest in Nodes* **do**
    if (src == dest) continue ;
    hopCount ← getHopCount(src, dest) ;
    cpt ← max(cpt, hopCount) ;
  end
  maxVal ← max(maxVal, cpt) ;
end

Algorithm 1: Calcul du nombre de sauts

- **Route balancing**

Cette métrique, calcule le nombre de chemin passant par chaque arête. Cela signifie que l'algorithme essaie d'équilibrer la charge sur le réseau (maximiser le nombre de chemins disjoints). La fonction objectif selon cette métrique est donc la suivante :

$$min(max_{\forall(e\in E)}\,\textstyle\sum_{\forall\pi} x_e^{\pi})$$
$$x_e^{\pi} = \{1\ si\ e \in E, \quad 0\ sinon$$

Algorithme de calcul de chemins disjoints

Le but de cet algorithme est de calculer le nombre de chemins maximal empruntant un même port.

Data: Nodes : Liste des noeuds
Result: Le nombre de chemins pour chaque liaison
foreach *node in Nodes* **do**
  foreach *port of the node* **do**
    port.nbRoutes ← nombre de chemins sortants par ce port ;
    if *port.nbRoutes > cpt* **then**
      cpt ← port.nbRoutes ; ;
    end
  end
  Return cpt
end

Algorithm 2: Calcul du nombre de chemins max empruntant un même port

Travail à réaliser

Avant propos : Vous devez lire attentivement la suite avant de commencer le travail demandé !

1- Télécharger : le projet sous **CELENE/Réseaux Avancés et Télécom / chapitre 6/ supports TP 3, 4 et 5**. Les codes fournis (à compléter) permet à partir d'une table de routage, de calculer le nombre de saut minimum et la charge minimale. Analyser les codes et classes fournies.

2- Tester : Compilez et exécutez ce projet.

3- Complétez la classe Calculation : Implémentez les fonctions selon les deux algorithmes précédents (calcul du nombre de sauts et calcul du nombre de chemins) pour permettre d'avoir le nombre de sauts minimum et la charge minimale pour chaque topomlogie étudiée.

En effet, **Calculation** est la classe sur laquelle vous allez travailler (toutes les fonctions sont créées, seulement leur implémentation n'est pas complète).

Quatre fonctions sont donc à compléter :

- **getHopCount :** Méthode qui doit calculer le nombre de sauts d'un noeud source jusqu'à un noeud destination.
- **calculate :** Méthode qui doit implémenter la métrique Minhop.
- **getRoute :** Méthode qui calcule la charge d'un noeud source jusqu'à un noeud destination.
- **balance :** Méthode qui doit implémenter la métrique Route balancing.

En considérant les deux métriques précédentes, implémentez donc ces fonctions et concluez pour chaque topologies le meilleur algorithme ? Des tests/résultats expérimentaux sont donc à réaliser.

Indications :

Le projet téléchargé, vous avez :

— **Main :** Permet de prendre en paramètre le fichier décrivant la topologie et le fichier de la table de routage.
— **topology :** Permet de charger le fichier décrivant la topologie et offre des fonctionnalités et des informations sur ce dernier.
— **routing :** Permet de charger le fichier fichier de la table de routage et offre des fonctionnalités et des informations sur ce dernier.
— **calculation :** Doit permettre l'implémentation des métriques décrites ci-dessus.