TP - 1,2 : Etude de topologies Fat-tree Simulateur Ibsim

Objectif des TPS :

L'objectif principal des travaux pratiques est de déterminer le meilleur algorithme de routage pour les réseaux avancés.
Pour ce faire, vous installerez un simulateur de réseau et un Subnet Manager qui nous permettrons de simuler la topologie des réseaux.
Il faudra donc prendre en main ces outils afin de comprendre leurs fonctionnements et de pouvoir les utiliser correctement.
Ensuite, certaines métriques seront présentées pour évaluer les algorithmes de routage, vous coderez alors un programme pour évaluer ces algorithmes de routage en définissant vos propres métriques à prendre en compte.

Objectif du TP1 et 2 :

Dans ces deux premiers TPs, vous allez pouvoir installer sur une machine Linux de l'école "UBUNTU 14 64 bits développement", prendre en main les différents outils qui seront utilisés dans la suite des TPs et développer un générateur pour les prochains TPs.

1. Installer Ibsim

Commencer par mettre à jour le cache des paquets en Admin :

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

Téléchargement et installation des packages, librairies d'IBSIM :

```
\# sudo apt-get install libibmad-dev
\# sudo apt-get install libibumad-dev
\# sudo apt-get install ibsim-utils
```

Création d'un dossier pour Ibsim :

```
\# mkdir ibsim
```

Télécharger le projet github dans le dossier ibsim

```
\# git clone https://github.com/jgunthorpe/ibsim.git
```

Pour compiler le projet Ibsim, accéder au répertoire **ibsim/umad2sim** et exécuter la commande suivante :

```
\# make
```

2. Exemple d'utilisation

Pour fonctionner Ibsim à besoin d'un fichier qui décrit l'architecture du réseau, pour qu'il puisse faire la simulation. le fichier doit décrire chaque équipement et ses connexions avec les autres. ce fichier est ensuite chargé par Ibsim qui permet de créer un réseau virtuel.

Voici un exemple de fichier basique pour ce réseau :

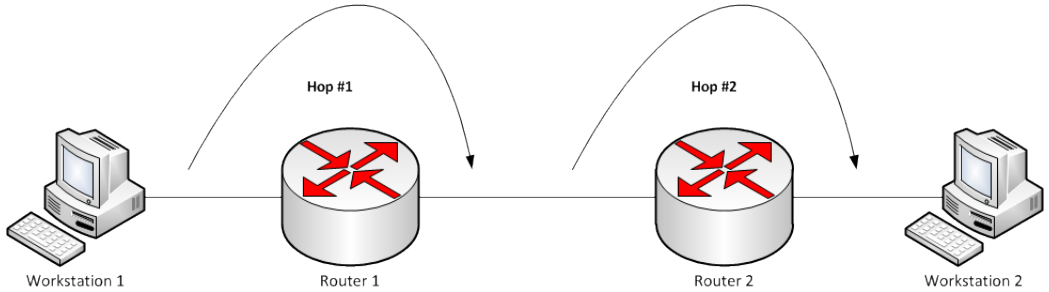

FIGURE 1 –

```
#'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]
```

Hca : Host Channel Adapter, carte d'extension permettant de connecter un équipement informatique à un réseau Infiniband.

La deuxième ligne permet donc de décrire l'équipement 'Workstation1' :
Il est de type Hca possède un port et son nom est 'Workstation1'

La troisième ligne permet de spécifier que le port 1 du Hca 'Workstation1' sera connecté au port 1 d'un équipement dont son nom est 'router1'

Remarque : ici les noms ont été simplifier mais dans la suite on utlisera des identifiants pour chaque équipement qui pourra faciliter la compréhension et avoir une convention pour tout le monde.

3. Chargement dans Ibsim

Créer manuellement un fichier 'exemple.topo' avec la description expliquée précédemment.

Exécuter la ligne de commande suivante dans le répertoire du fichier exemple.topo :

```
\# ibsim -s exemple.topo
```

Ibsim vous confirme que le parçage s'est bien effectué et que la simulation à commencer. Vous pouvez voir que maintenant vous êtes sous un shell de Ibsim.

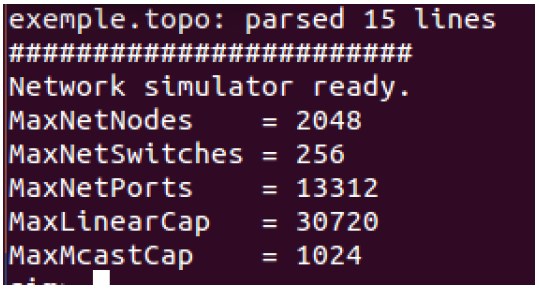

FIGURE 2 –

Pour visualiser la topologie qu'Ibsim a parcée et enregistrée en mémoire il suffit de taper la commande suivante :

```
\# dump
```

Il existe de nombreuses commandes avec Ibsim et plusieurs outils permettant de tracer par exemple le chemin d'un packet à un autre (Ibtracert...). La commande help vous donne une idée sur les commandes qu'on peut utiliser dans ce simulateur pour mieux visualiser chaque composant du réseau.

Dans cette partie, on vous demande donc d'analyser et identifier les différentes commandes offertes par le simulateur et de les tester.

```
\# help
```

4. Générateur de topologie

L'objectif donc est d'utiliser Ibsim pour simuler des réseaux avancés, c'est son réel potentiel. Dans le cadre des TPs vous allez vous concentrer sur la simulation des architectures infiniband s'appuyant sur une topologie Fat-tree.

Pour cela Ibsim a besoin du fichier de description de la topologie Fat-tree.

Vous allez programmer un générateur en **C++** qui génère un fichier **fattree.topo** décrivant la topologie fat-tree.

Explication :

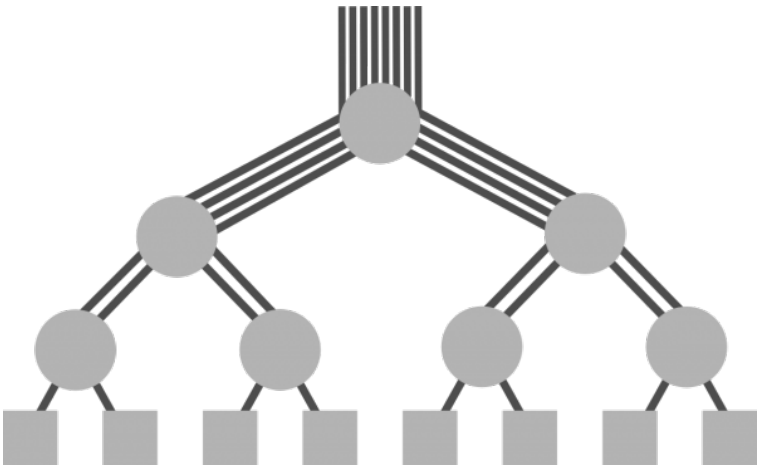

FIGURE 3 – Topologie Fat-Tree

La topologie Fat-Tree a été proposée par Charles E. Leiserson en 1985. La particularité de cette dernière est que pour tout commutateur, le nombre de liens montants est égal au nombre de liens descendants. Le but étant de proposer un réseau non-bloquant de commutateurs, tout en ayant un nombre minimal de commutateurs.

0.0.1 Construction d'une topologie Fat-Tree

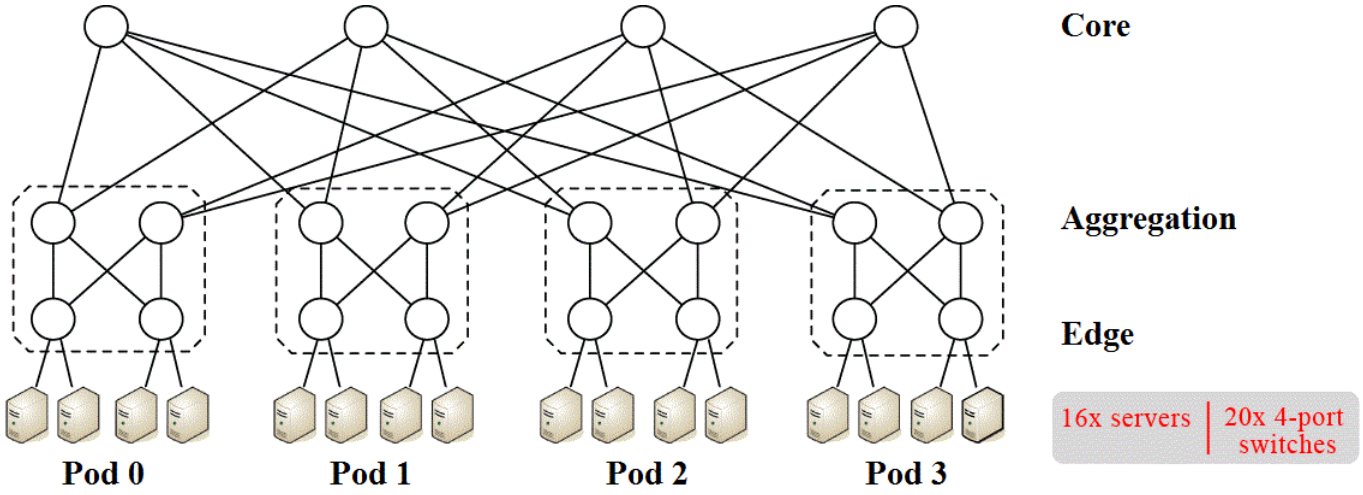

FIGURE 4 – Topologie Fat-Tree K=4

La topologie Fat-Tree se compose de 3 niveaux : Edge, Aggregation et Core.
Pour construire une telle topologie au sein d'un Data center, des règles ont déjà été définies qu'on listera ci-dessous :

1. Définir le paramètre k qui représente le nombre de pod (Point of delivery est un module du réseau ex : machines de calculs, stockages...), dont chaque pod contient $(k/2)^2$ servers et 2 niveaux de k/2 commutateurs à k ports.
2. Chaque commutateur du niveau Edge peut connecter jusqu'à k/2 serveurs et k/2 commutateurs d'agrégation.
3. Chaque commutateur d'agrégation peut connecter jusqu'à k/2 commutateur du niveau Edge et Core.
4. $(k/2)^2$ commutateurs du niveau core dont chacun connecte k pods.

L'image précédente représente une topologie Fat-Tree avec un paramètre k égale à 4.

Convention :

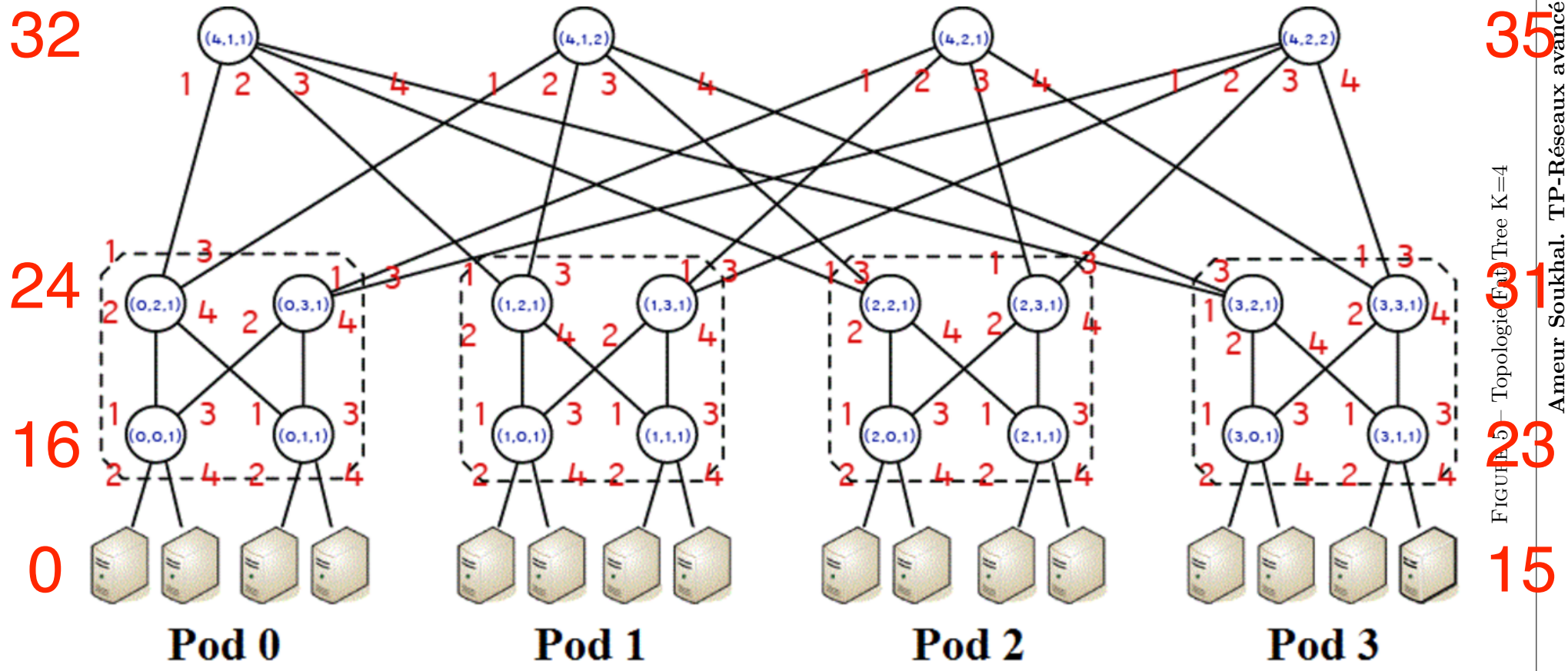

FIGURE 5 – Topologie Fat Tree K=4

POLYTECH TOURS

Votre programme prendra un paramètre k qui variera entre 4,6,8,10,... et qui doit retourner un fichier décrivant la topologie fat-tree selon le k.

Votre programme doit être compilable en commande Unix comme cet exemple :

```
\# g++ topogen.cpp -o generator
\# ./generator 4
```

Voiçi un apperçu du fichier resultat.topo pour k=4

```
#fat tree topology file.
#Value of k = 4
#Total number of hosts = 16
#Number of hosts under each switch = 2
#####################################################

Hca         1         "Node(0)"
[1] "Edge(0 0 1)"[2]

Hca         1         "Node(1)"
[1] "Edge(0 0 1)"[4]
...
Hca         1         "Node(15)"
[1] "Edge(3 1 1)"[4]

Switch         4         "Edge(0 0 1)"
[1] "Aggr(0 2 1)"[2]
[2] "Node(0)"[1]
[3] "Aggr(0 3 1)"[2]
[4] "Node(1)"[1]
...
Switch         4         "Aggr(3 3 1)"
[1] "Core(4 2 1)"[4]
[2] "Edge(3 0 1)"[3]
[3] "Core(4 2 2)"[4]
[4] "Edge(3 1 1)"[3]
...
Switch         4         "Core(4 1 1)"
[1] "Aggr(0 2 1)"[1]
[2] "Aggr(1 2 1)"[1]
[3] "Aggr(2 2 1)"[1]
[4] "Aggr(3 2 1)"[1]

Switch         4         "Core(4 1 2)"
[1] "Aggr(0 2 1)"[3]
[2] "Aggr(1 2 1)"[3]
[3] "Aggr(2 2 1)"[3]
[4] "Aggr(3 2 1)"[3]
```

```
Switch        4        "Core(4 2 1)"
[1] "Aggr(0 3 1)"[1]
[2] "Aggr(1 3 1)"[1]
[3] "Aggr(2 3 1)"[1]
[4] "Aggr(3 3 1)"[1]

Switch        4        "Core(4 2 2)"
[1] "Aggr(0 3 1)"[3]
[2] "Aggr(1 3 1)"[3]
[3] "Aggr(2 3 1)"[3]
[4] "Aggr(3 3 1)"[3]
```

...