TD 10: IP et routage

Module ArcSys

Objectifs pédagogiques:

- Comprendre l'adressage CIDR (E1);
- Savoir configurer manuellement le routage (E2);
- Comprendre le calcul des tables de routage dans un domaine (E3);
- Comprendre le calcul des tables de routage entre domaines (E4);
- Implémenter un algorithme de routage sur un simulateur simplifié (E5).

★ Exercice 1: Calcul d'adresses

Soit l'adresse `192.168.5.8` et le CIDR 22, ou en forme raccourci `192.168.5.8/22`.

▷ **Question 1:** Quel est le masque de sous-réseau ?

▷ **Question 2:** Calculez le nombre maximum de machines et de sous-réseaux.

▷ **Question 3:** Quelle est l'adresse du sous-réseau dont `192.168.5.8` fait partie ?

▷ **Question 4:** Quelle est la première adresse du réseau dont `192.168.5.8` fait partie et la dernière ?

★ Exercice 2: Problème de routage chez votre ami Tchekhov

Votre ami Tchekhov vous invite chez lui afin de profiter de vos connaissances en réseau. Il vous explique très fièrement la configuration de son réseau, qu'il a lui même monté :

Il possède deux ordinateurs (1 fixe et 1 portable), un réfrigérateur connecté, un switch et un routeur qui est connecté à Internet.

Machine	Adresse IP	Masque	Passerelle
Ordinateur fixe	192.168.1.5	255.255.255.224	192.168.1.1
Ordinateur portable	192.168.1.33	255.255.255.0	192.168.1.1
Réfrigérateur	10.28.2.3	255.0.0.0	10.0.0.1 (Internet)
Routeur	192.168.1.1	255.255.255.0	10.0.0.1 (Internet)

Le fixe, le portable et le réfrigérateur sont reliés à un switch qui est lui même relié au routeur. Il vous indique aussi qu'il a configuré manuellement les adresses IP, car il pense que DHCP va lui faire perdre du temps au démarrage de ses ordinateurs pour pouvoir se connecter à Internet.

Cependant, votre ami en appelle à votre expertise car tout ne marche pas comme il avait prévu:

▷ **Question 1:** Pourquoi son réfrigérateur n'a pas accès à Internet ? Cela l'embête, car il aimerait vraiment pouvoir regarder des films le matin en mangeant ses céréales et il s'est assuré d'avoir relié correctement le réfrigérateur au switch.

▷ **Question 2:** Pourquoi est-il plus lent d'envoyer un paquet du fixe au portable que du portable au fixe ? Cela l'intrigue, car selon lui il a relié correctement le fixe et le portable au switch, ils devraient pouvoir communiquer directement.

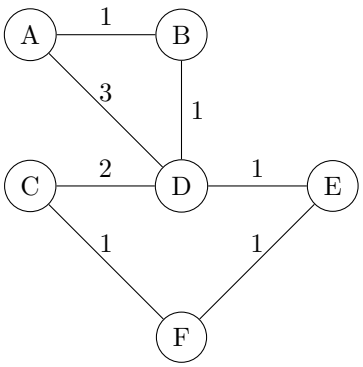
▷ **Question 3:** Comment reconfigurer le réseau correctement ?

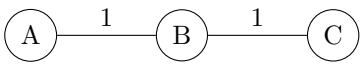
★ Exercice 3: Protocole de routage: Intra

▷ **Question 1:** Rappelez ce qu'est le *Distance Vector Routing*.

▷ **Question 2:** À partir du graphe de droite, établissez le tableau initial et final des informations stockées dans chaque noeud du graphe pour atteindre les autres noeuds.

▷ **Question 3:** Que se passe-t-il si le noeud B est hors-ligne ?





▷ **Question 4:** Dans le graphe à gauche ci-dessus, que se passe-t-il si le noeud A est hors-ligne, B le remarque, mais ne previent pas C assez rapidement?

▷ **Question 5:** Qu'arrive-t-il à un paquet IP qui souhaite atteindre A à ce moment précis depuis C ?

▷ **Question 6:** Avez-vous des idées pour résoudre ce problème ? Attention à faire en sorte que le nouveau protocole puisse accepter l'ancien protocole.

▷ **Question 7:** N'existe-t-il pas un cas ou cela ne fonctionne pas ?

★ Exercice 4: Protocole de routage: Inter

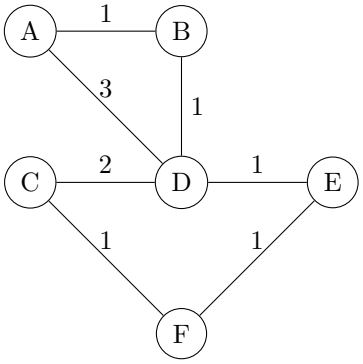
▷ **Question 1:** Rappelez pourquoi on ne peut pas utiliser l'algorithme vu précédemment pour du *Inter-Domaing Routing* ? et quel protocole est utilisé pour cela ?

▷ **Question 2:** Pourquoi permet-il d'empêcher les boucles ?

▷ **Question 3:** Pourquoi est-il nécessaire d'avoir un réseau maillé pour iBGP ?

★ Exercice 5: Programmer Bellman-Ford.

On souhaite coder en C une version très simplifiée du protocole à vecteurs d'état de Bellman-Ford. Notre version ne tiendra pas compte des réseaux et de leurs masques: chaque machine existante sera explicitement listée dans la table de routage. Le nombre de machines est constant et connu à l'initialisation. Les distances sont mesurées en millisecondes. Dans les premières questions, nous utiliserons le graphe ci-contre. Tout le code est à écrire dans le fichier `table.c` fourni dans l'archive disponible sur le site du cours.




```c
struct table {
  int *next_hop; // next_hop[i]: ID of the node to use to reach i
  int *distance; // distance[i]: time needed to reach i
  int size;      // number of nodes
};
```

▷ **Question 1:** Écrivez les fonctions suivantes `table_new()` et `table_free()` (prototypes ci-dessous), qui permettent respectivement d'allouer et libérer la mémoire nécessaire pour stocker une table. `table_new()` reçoit en paramètres le nombre de machines connectées sur le réseau (qui est la taille de la table de routage), ainsi que l'identifiant de la machine sur laquelle cette fonction est appelée. Les distances sont initialisées à `MAXINT` pour les machines distantes et à 0 pour la machine locale.

```c
struct table* table_new(int taille, int ID);
void table_free(struct table* t);
```

▷ **Question 2:** Écrivez la fonction `table_send()`, utilisée pour envoyer les informations de la table de routage à un noeud `destination` du réseau. Là où Bellman-Ford envoie des couples $< ID, distance >$, nos simplifications nous permettent de n'envoyer que le vecteur de distances. Vous pourrez utiliser la fonction `send`, dont le prototype est donné dans le fichier "routing.h".

```c
void table_send(struct table* t, int destination);
```

▷ **Question 3:** Écrivez la fonction `table_recv()`, utilisée pour recevoir les informations de routage d'une machine distante `source`.

La fonction reçoit également en paramètre `millis`, la distance entre la machine émettrice et la machine locale. Il faut modifier la table de routage pour tenir compte des informations reçues. Entre autres, le `next_hop` de certaines destinations peut passer à `source` si cette solution est plus avantageuse.

```c
void table_recv(struct table* t, int source, int millis);
```

▷ **Question 4:** Vous pouvez maintenant tester votre solution sur le premier scénario proposé. Compilez votre code et testez le avec les commandes suivantes :

```
make
./routing 1
```

La table de routage obtenue est-elle correcte ?

▷ **Question 5:** Nous allons maintenant étudier un scénario où la qualité de la connexion évolue au cours du temps. Exécutez le scénario 2 avec la commande suivante :

```
./routing 2
```

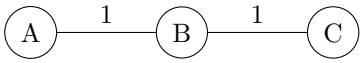
Nous allons maintenant étudier des scénarios où des machines disparraissent du réseau. Pour vous compliquer la tâche, le simulateur que nous avons mis au point retarde volontairement l'information du crash d'une machine : si le noeud B apprend que le noeud A n'est plus disponible, il n'enverra pas de message pendant un cycle.

▷ **Question 6:** Écrivez la fonction `table_failed()`, utilisée pour indiquer qu'un lien vers la machine `neighb` ne fonctionne plus. La distance à cette machine est fixé à `MAXINT` de façon à ce qu'une autre route soit trouvée par les mises à jour futures.

```
void table_failed(struct table* t, int neighb);
```

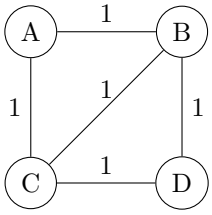
▷ **Question 7:** Testez votre nouveau code sur le scénario 3. Qu'observez vous ?

Dans le scénario 4, le réseau étudié devient :



▷ **Question 8:** Relancez votre code sur ce scénario et commentez ce qu'il se passe. Comment pouvez-vous corriger ce phénomène ? Implémentez votre solution.

Enfin, dans le scénario 5, on utilise le graphe suivant :



▷ **Question 9:** Relancez votre code sur le scénario 5. Qu'observez vous ?