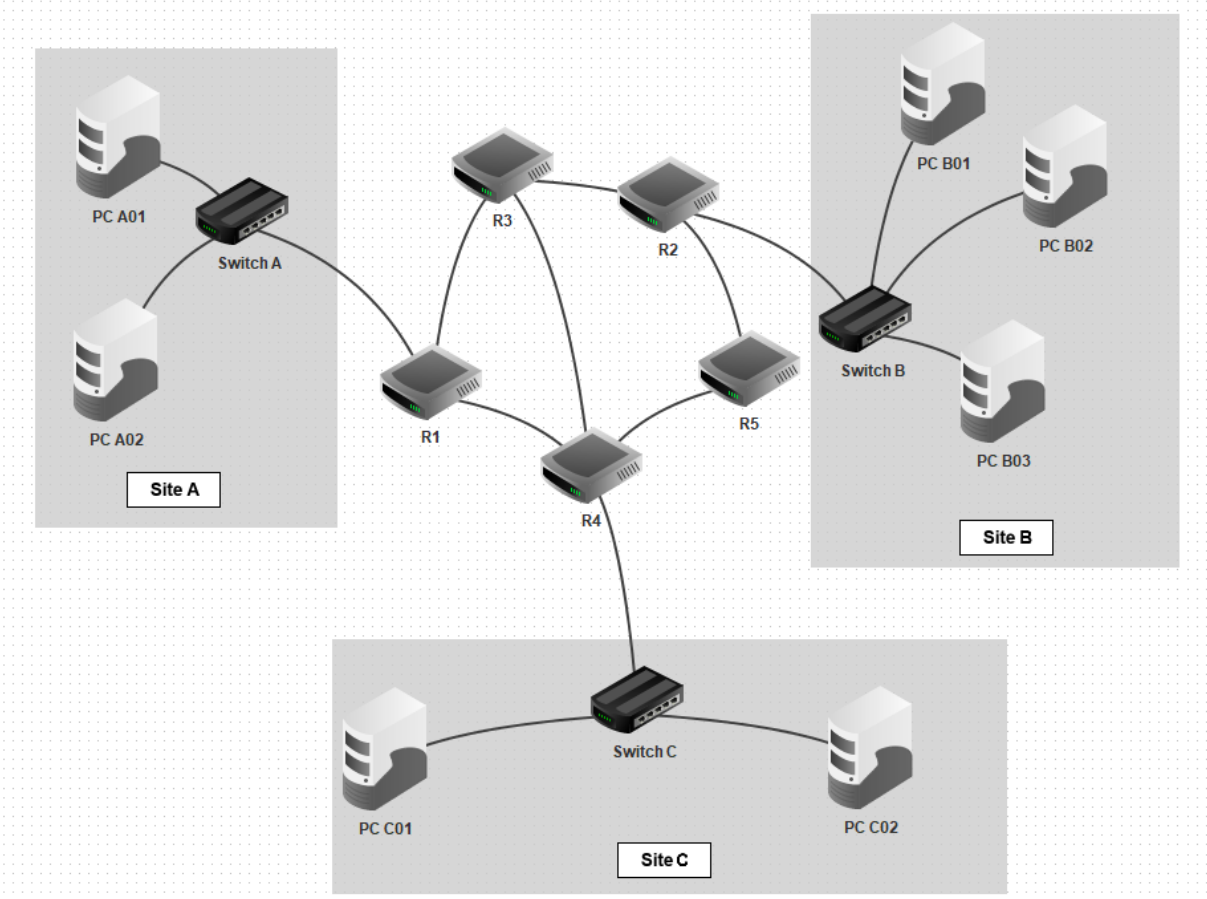
## Exercice 1 (6 points)

*Cet exercice porte sur les réseaux et la programmation orientée objet.*

Une entreprise dispose d'une infrastructure réseau répartie sur plusieurs sites interconnectés à l'aide de routeurs. La figure ci-dessous représente le schéma de ce réseau, où les routeurs sont notés de R1 à R5.




Afin d'assurer la communication entre les différents postes et sous-réseaux, des protocoles de routage tels que RIP ou OSPF sont utilisés.

- Le protocole RIP minimise le nombre de sauts entre deux routeurs ; un "saut" correspond au transfert des données d'un routeur à un autre.
- Le protocole OSPF (Open Shortest Path First) minimise la somme des coûts des liaisons entre les routeurs empruntées par le paquet ; le coût entre deux routeurs voisins dépend du débit de la liaison entre ces routeurs selon la formule $\text{Coût} = \frac{10^8}{\text{Débit}}$ où le débit est exprimé en bit/s et le coût est sans unité.

Le bon fonctionnement de ce réseau repose sur une configuration correcte des adresses IP, des masques de sous-réseau, des routes, ainsi qu'une gestion rigoureuse des ressources (comme la mémoire et les tables de routage).

Le réseau utilise des adresses IPv4, c'est-à-dire des adresses de la forme `IP/S`` où :

- IP est une suite de 4 entiers entre 0 et 255 séparés par un point ;
- S est un entier entre 1 et 32.

L'adresse IP est codée en machine sur 4 octets, soit 32 bits. L'entier S indique que les S premiers bits de IP correspondent à la partie fixe de l'adresse et les suivants à la partie propre à la machine.

Deux adresses IP sont réservées :

- l'adresse du réseau local qui est constituée de la partie fixe de l'adresse complétée par des bits égaux à 0 ;
- l'adresse de diffusion (broadcast) qui est constituée de la partie fixe de l'adresse complétée par des bits égaux à 1.

On considère le poste PC C01 du site C ayant pour adresse IPv4 : 172.16.2.1 /16.

1. Déterminer l'adresse IP du réseau local dédié au site C.
2. Déterminer l'adresse IP de diffusion du réseau local dédié au site C.
3. Donner le nombre maximal de machines que l'on peut connecter sur le réseau local dédié au site C, y compris les machines déjà présentes.
4. Recopier et compléter la table de routage du routeur R1 obtenue avec le protocole RIP.

   Table de routage R1 :

| Destination | Passe par | Nombre de sauts |
|---|---|---|
| R2 | | |
| R3 | | |
| R4 | | |
| R5 | | |

5. Déterminer le chemin que suit un paquet envoyé depuis PC A01 (Site A, relié à R1) vers PC B01 (Site B, relié à R2) en supposant qu'on utilise le protocole RIP.
6. Déterminer la nouvelle route qu'un paquet peut suivre de R1 à R2 si le routeur R3 tombe en panne, toujours en supposant qu'on utilise le protocole RIP.
7. Recopier et compléter le tableau suivant donnant le coût pour le protocole OSPF des liaisons réseau selon leur type de connexion.

| Type de connexion | Débit (bit/s) | Coût |
|---|---|---|
| Ethernet | $10^7$ | |
| Fast Ethernet | $10^8$ | |
| Fibre | $10^9$ | |

Le tableau ci-dessous indique le type connexion pour chaque liaison du réseau.

| Liaison | Type de connexion |
|---|---|
| R1-R3 | Fibre |
| R1-R4 | Ethernet |
| R2-R3 | Ethernet |
| R5-R4 | Fast Ethernet |
| R3-R4 | Fast Ethernet |
| R5-R2 | Fast Ethernet |

8. Déterminer, en justifiant, le chemin choisi par le protocole OSPF entre R1 et R4.

On décide pour la suite de représenter en Python chaque routeur par une chaîne de caractères, par exemple 'R1', et de représenter les routes par des listes de routeurs, par exemple `['R1','R3','R2']`.

Un code Python permettant de définir une liste de routes (chaque route étant une liste de routeurs) a été créé. Ce code est composé :

- d'une liste vide appelée `liste_routes` qui permettra de contenir les routes ;
- d'une constante `MAX_ROUTES` indiquant le nombre maximum de routes que peut contenir la liste `liste_routes` ;
- d'une fonction `ajouter_route` qui ajoute la route donnée en paramètre à la liste `liste_routes`.

```
1 liste_routes = []
2 MAX_ROUTES = 5
3
4 def ajouter_route(route):
5     liste_routes.append(route)
```

9. Justifier que le code proposé ci-dessus ne permet pas de s'assurer que la contrainte d'un nombre maximal de routes contenues dans la liste est respectée.

On souhaite créer une classe Routage qui contient :

- le constructeur `__init__` qui permet d'initialiser un objet avec une capacité maximale donnée ;
- une méthode `ajouter` qui ajoute une route uniquement si la capacité maximale n'est pas atteinte ; sinon, elle affiche un message d'erreur.

Un objet de type Routage doit contenir les attributs :

- `capacite` : un nombre entier initialisé à 5 par défaut ;
- `routes` : une liste pour stocker les routes.

```
 1 class Routage:
 2     def __init__(self, capacite = 5):
 3         self.capacite = ...
 4         self.routes = []
 5
 6     def ajouter(self, route):
 7         if ...
 8             ...
 9         else:
10             ...
```

10. Recopier et compléter les lignes 3, 7, 8 et 10 du code ci-dessus pour respecter les contraintes de la classe.

11. Écrire une méthode `afficher` de la classe `Routage` qui affiche toutes les routes présentes dans la liste `routes`. Par exemple si l'attribut `routes` de l'objet contient les routes `['R1','R3','R2']` et `['R1','R4','R5','R2']`, l'affichage sera :

```
R1
R3
R2
---
R1
R4
R5
R2
---
```