Protocoles de routage - Activité d'introduction

Dernière mise à jour le : 30/10/2023

Cette activité propose à travers un travail de programmation de revoir les notions d'**adresse IP**, de **masque de sous-réseau** et d'**adresse réseau**. Ces notions, bien qu'a priori pas vraiment au programme, ont été vues en classe de Première (voir Thème 5 / Chapitre 3 : Architecture d'un réseau) et leur connaissance facilitera grandement la compréhension du chapitre sur les protocoles de routage.

Rappels : appartenance à un même réseau

Deux machines appartiennent au même réseau si elles possèdent la même *adresse réseau*. Dans ce cas, elles peuvent communiquer directement.

On rappelle que pour déterminer l'adresse réseau d'une adresse IP, il suffit d'appliquer l'opérateur logique ET (AND en anglais) bit à bit entre l'adresse IP et le masque de sous-réseau. La table de vérité de cette opérateur est la suivante :

x	y	x ET y
0	0	0
1	0	0
0	1	0
1	1	1

Exemple : Déterminons l'adresse réseau de la machine d'adresse IP `193.55.221.62/24`. Le masque est `/24` donc vaut `255.255.255.0` en notation décimale. On peut alors appliquer le ET logique bit à bit :

	Notation binaire	Notation décimale
Adresse IP (d'une machine)	`11000001.00110111.11011101.00111110`	`193.55.221.62`
Masque	`11111111.11111111.11111111.00000000`	`255.255.255.0`
Adresse réseau (ET bit à bit)	`11000001.00110111.11011101.00000000`	`193.55.221.0`

L'adresse réseau de la machine `193.55.221.62/24` est donc 193.55.221.0.

✍ **Question** : Un réseau local contient une machine d'adresse IP `192.168.0.1` avec le masque `/20`. La machine d'adresse IP `192.168.1.3` appartient-elle au même réseau ?

💻 Partie programmation

L'objectif de cette partie est d'écrire une fonction `meme_sous_reseau(ip_a, ip_b, masque)` qui renvoie un booléen indiquant si les machines A et B font partie du même sous-réseau.

On rappelle quelques instructions qui pourront s'avérer utiles :

```
>>> bin(45)
'0b101101'
>>> chaine = "bonjour à tous !"
>>> chaine.split(' ')  # on sépare la chaîne selon le caractère espace
['bonjour', 'à', 'tous', '!']
```

Vous utiliserez la méthode Test Driven Development (ou *développement piloté par des tests*) à partir du squelette de code ci-dessous, en testant chaque fonction après sa réalisation, jusqu'à la fonction finale.

```python
def et_logique(bit1, bit2):
    """
    Renvoie le résultat de l'opération logique bit1 ET bit2.

    Entrées : deux string d'1 caractère ('0' ou '1')
    Sortie : string
    """
    # à vous de jouer !
    pass

def test_et_logique():
    assert et_logique('0', '0') == '0'
    assert et_logique('0', '1') == '0'
    assert et_logique('1', '0') == '0'
    assert et_logique('1', '1') == '1'

def entier_en_mot_binaire(entier, longueur_mot_binaire):
    """
    Renvoie un mot binaire correspondant à l'entier.

    Entrées :
        entier : int
        longueur_mot_binaire : int de longueur supérieure ou égale au nombre
        de bits nécessaires
    pour convertir entier en base 2

    Sortie : string de longueur longueur_mot_binaire
    """
    # à vous de jouer !
    pass

def test_entier_en_mot_binaire():
    assert entier_en_mot_binaire(1, 8) == '00000001'
    assert entier_en_mot_binaire(192, 8) == '11000000'
    assert entier_en_mot_binaire(168, 8) == '10101000'

def masque_en_mot_binaire(masque):
    """
    Renvoie la conversion du masque en un mot binaire.

    Entrée : masque (int)
    Sortie : string de 32 caractères
    """
    # à vous de jouer !
    pass

def test_masque_en_mot_binaire():
    assert masque_en_mot_binaire(24) == '11111111111111111111111100000000'

def ip_en_liste(ip):
    """
    Renvoie une liste dont les éléments sont les 4 octets de l'adresse ip.

    Entrée : ip (string)
    Sortie : liste d'entiers
    """
    # à vous de jouer !
    pass

def test_ip_en_liste():
    assert ip_en_liste('192.168.0.1') == [192, 168, 0, 1]

def ip_en_mot_binaire(ip):
    """
    Renvoie la conversion d'une ip en un mot binaire.

    Entrée : string contenant une IP (ex "192.168.0.1")
```

```
    Sortie : string de 32 caractères
    """
    # à vous de jouer !
    pass

def test_ip_en_mot_binaire():
    assert ip_en_mot_binaire("192.168.0.1") == '11000000101010000000000000000001'

def adresse_reseau_ip(ip, masque):
    """
    Renvoie l'adresse réseau de l'adresse ip avec le masque `masque`.

    Entrées :
        ip : string contenant une IP (ex "192.168.0.1")
        masque : entier du masque en notation CIDR (ex : 24)
    Sortie : string de 32 caractères
    """
    # à vous de jouer !
    pass

def test_adresse_reseau_ip():
    assert adresse_reseau_ip("192.168.0.1", 24) == '11000000101010000000000000000000'
    assert adresse_reseau_ip("192.168.1.3", 24) == '11000000101010000000000100000000'
    assert adresse_reseau_ip("192.168.0.1", 20) == '11000000101010000000000000000000'
    assert adresse_reseau_ip("192.168.1.3", 20) == '11000000101010000000000000000000'

def meme_sous_reseau(ip_a, ip_b, masque):
    """
    Renvoie un booléen indiquant si ip_a et ip_b sont dans un même réseau
    de masque `masque`.

    Entrées :
    ip_a:  string contenant une IP (ex "192.168.0.1")
    ip_b : string contenant une IP
    masque : entier du masque en notation CIDR (ex : 24)
    """
    # à vous de jouer !
    pass

def test_meme_sous_reseau():
    assert meme_sous_reseau("192.168.0.1", "192.168.1.3", 24) == False
    assert meme_sous_reseau("192.168.0.1", "192.168.1.3", 20) == True
    assert meme_sous_reseau("192.168.0.1", "192.168.0.3", 30) == True
```

Références

- Equipe éducative DIU EIL (cours d'Introduction aux réseaux de P. PASSARD et S. HAMMA), Université de Nantes.
- L'idée de départ de la partie "programmation" vient d'un exercice proposé par Gilles Lassus

---

Germain BECKER, Lycée Mounier, ANGERS

Voir en ligne : info-mounier.fr/terminale_nsi/archi_se_reseaux/protocoles-routage-activite-intro