Balthazar Patiachvili <balthazar.patiachvili@crans.org>

Composition 2022

Partie 1. Système

Exercice 1-1. Droits et permissions d'accès aux fichiers

Remarque 1

Le format de retour de la commande `ls -l` est le suivant (dans l'ordre) :

1. Type de fichier/permissions,
2. Nombre de liens durs,
3. Propriétaire,
4. Groupe,
5. Taille (en octets),
6. Date de dernière modification,
7. Nom,
8. Fichier pointé (seulement si lien symbolique),

a. L'utilisateur utilisant le terminal est `untel` comme on peut le voir dans l'invite de commande (`[untel:~/Ag]`).

b. Le propriétaire de tous les fichiers est `untel` : on peut le voir dans la troisième colonne.

c. Comme `guntel` et `otel` apparaissent dans la quatrième colonne de la sortie de `ls -l`, on peut en déduire qu'il s'agit de groupes.

d. Le répertoire courant est `Ag` comme on peut le voir dans l'invite de commande.

Remarque 2

Pour déterminer le nombre de fichiers (normaux) et de dossiers, il faut déterminer le type de chaque entrée que l'on peut voir dans la sortie `ls -l`. On regarde pour cela le premier caractère de la première colonne : un `-` indique qu'il s'agit d'un fichier normal, `d` d'un répertoire/dossier et `l` d'un lien symbolique.

e. On peut dénombrer 2 fichiers normaux, `fi.txt` et `oui.sh`, et 1 dossier `sAg` (`.` et `..` sont respectivement le dossier courant et son parent, on ne les compte pas ici comme des enfants du dossier courant). On compte par ailleurs 4 fichiers enfants du dossier courant (car les dossiers et les liens symboliques sont également des fichiers).

f. La première colonne désigne le type de fichier et les permissions associées. Par exemple, pour `oui.sh`, `-` indique qu'il s'agit d'un fichier normal, et `rwxrw-r--` que le propriétaire a les droits de lecture (`r`), d'écriture (`w`) et d'exécution (`x`), le groupe a les droits de lecture et d'écriture (mais pas d'exécution), et les autres utilisateurs uniquement de lecture. Les colonnes 3 et 4 désignent respectivement le propriétaire et le groupe du fichier. Par exemple, pour `oui.sh`, il s'agit de `untel` et `otel`. Finalement, la dernière colonne indique le nom du fichier. De plus, dans le cas de `fc.txt`, comme il s'agit d'un lien symbolique (comme le montre le caractère `l` en début de ligne), le nom à droite de la flèche indique le fichier pointé par le contenu du lien symbolique, c'est-à-dire que `fc.txt` est un lien symbolique vers `fi.txt`.

g. Les droits dont `tritel` dispose sur `sAg` dépendent de si `tritel` appartient au groupe de `sAg`, ici `guntel`. Si `tritel` appartement à `guntel`, alors il aura les droits de lecture et d'écriture, et uniquement le droit de lecture dans le cas contraire. Pour `sAg` qui est un dossier, le droit de lecture signifie pouvoir lister les enfants de `sAg` (par exemple avec `ls sAg`), et le droit d'écriture de créer/modifier des enfants de `sAg`.

> **Remarque 3**
>
> Dans la pratique, il est impossible d'utiliser le droit d'écriture seul sur un dossier : il est nécessaire pour toutes les opérations d'écriture d'avoir également le droit d'exécution.

h. Aujourd'hui, la quasi-totalité des systèmes d'exploitation grand public ont une gestion des droits et permissions d'accès aux fichiers principalement des raisons de sécurité et de vie privée : on ne veut pas que sur une machine multi-utilisateurs qu'un utilisateur puisse accéder sans consentement aux données d'un autre utilisateur. Par ailleurs, cela permet également de s'assurer qu'une application s'exécutant sur une machine ne puisse pas accéder à des données qu'elle n'est pas censé observer/modifier (par exemple, on ne souhaite pas qu'un serveur web puisse accéder aux clefs privées SSH des utilisateurs car cela ne lui est d'aucune utilité). Cette gestion permet également de partager efficacement des fichiers entre plusieurs utilisateurs : si l'on souhaite que plusieurs utilisateurs aient accès à un même fichier partagé, il suffit alors de créer un groupe, d'y ajouter tous les utilisateurs souhaités, et de lui donner les bons droits.

Exercice 1-2. Système et gestion de fichiers

> **Remarque 4**
>
> Un rappel sur l'ambiguïté des préfixes pour les mesures de taille. Si on utilise dans le reste des disciplines scientifiques des préfixes pour les grandeurs qui sont associées au puissance de 1000 (kV, MV, GV, etc.), l'informatique constitue une exception en cela qu'il est souvent plus aisé de parler de puissance de 2. Mais pour pouvoir continuer à écrire de manière lisible de grands nombre, on utilise souvent les puissances de $2^{10} = 1024$, mais on ne le précise pas toujours ! Il existe une notation spécifique, sous la forme: kio, Mio, Gio; mais on utilise parfois (surtout en anglais, même si on trouve aussi parfois écrit kiB, MiB, GiB) la forme ambiguë ko, Mo, Go pour parler en puissance de 1024.
>
> N'hésitez pas à lever l'ambiguïté lorsque vous manipulez ces quantités en précisant leur signification (si vous utilisez la notation ambiguë).

a. Un fichier peut contenir au plus : 12 blocs à adressage direct, 1 bloc à adressage indirect simple, 1 bloc à adressage indirect double et 1 bloc à adressage indirect triple. La taille maximale théorique d'un fichier est : $\underbrace{(12 + 256 + 256^2 + 256^3)}_{\text{nombre de blocs}} \times \underbrace{2048}_{\text{taille d'un bloc}} = 34494504960$ o (on peut aussi écrire $\approx 2^{8\times3+11} = 32$ Gio). Or, la taille d'un fichier est indiquée dans son champ longueur (dont le premier bit n'est pas utilisé), c'est-à-dire que cette information est contenue sur 31 bits. La taille maximale d'un fichier est donc $2^{31} - 1 = 2147483647$ o (≈ 2 Gio), ce qui est plus petit que la première valeur obtenue. La seconde valeur est donc limitante : c'est la vraie taille maximale d'un fichier sur ce système de fichiers.

b. Le nombre maximal de blocs par fichier est de $12 + 256 + 256^2 + 256^3 = 16843020$.

c. Un fichier de 4567 o sera contenu sur 3 blocs : comme $4567 = 2 \times 2048 + 471$, il y aura deux blocs complets et un bloc partiellement rempli.

d. Par un calcul similaire, on trouve qu'un fichier de 573448 o occupera 281 blocs de données. Or, seuls 12 blocs sont directement adressables, il faut donc compter en plus les blocs d'indirection : il y a $281 - 12 = 269$ blocs de données à placer, ce qui peut se faire avec 2 blocs d'indirection. Or, une inode ne peut contenir qu'un seul bloc de simple indirection, il faut donc utiliser un nouveau bloc de double indirection. Au total, 284 blocs sont utilisés, parmi lesquels 281 blocs directs, 2 blocs d'indirection simple et 1 bloc d'indirection double.

Exercice 1-3. Performance

Cet extrait est une allégorie d'un système de fichier, et pose la problématique de la fragmentation des données. On peut en effet voir l'*énorme armoire du ministère* comme un disque dur (ou plus précisément comme une partition d'un disque dur) sur lequel on souhaiterait stocker des données, ici les *dossiers à ranger dans cette armoire*. Chaque dossier représente alors un fichier du système de fichier. On peut souhaiter alors *séparer un dossier en petits groupes de documents*, c'est-à-dire fragmenter un fichier en plusieurs blocs de données non contigus pour ne pas perdre d'espace disponible ni de temps à chercher un espace contigu suffisamment grand pour accueillir le fichier dans son intégralité. Cependant, et cette idée était encore plus d'actualité à l'époque de l'écriture du texte (1998), des données séparées spatialement seront plus lentes à récupérer que des données contiguës. C'est pour cela qu'*une dizaine de garçons remettent tout en ordre* : pour défragmenter les données. On parle d'une défragmentation du disque dur. Dans ce cadre, la *liste des tiroirs vides mise à jour par la deuxième secrétaire* peut correspondre à plusieurs idées comme une liste tableau de booléens indiquant pour chaque bloc de données s'il est libre ou occupé. On retrouve les deux méthodes dans des systèmes de fichiers encore utilisés aujourd'hui : FAT32, exFAT ou encore NTFS utilisent de la fragmentation, alors que ext4 privilégie une bitmap (tableau de booléens). La méthode la plus performante dépend des vitesses de lecture et d'écriture du périphérique de stockage, mais aujourd'hui la majorité des périphériques de stockage utilisent des systèmes de fichiers sans fragmentation pour assurer la localité spatiale des données.

Exercice 1-4. Processus

a.

1. Le programme C donné va entrer dans la boucle `for`, engendrer un enfant par `fork`, puis les deux processus vont à leur tour engendrer de nouveaux enfants, etc.
2. Cela peut effectivement créer un problème : le nombre de processus croît indéfiniment, jusqu'à ce que la machine ne puisse plus exécuter tous les processus et finisse par planter. On parle usuellement de *fork bomb*. On peut régler ce problème en limitant par exemple le nombre total de processus :

```c
int main(int argc, char **argv) {
  int n = atoi(argv[1]);
  for (int i = 0; i < n; i++) {
    int pid = fork();
    if (pid != 0) {
      waitpid(pid, NULL, 0);
    }
  }
  return 0;
}
```

Exécuté avec pour argument n, ce programme générera (au plus) 2^n processus, permettant bien d'éviter une croissance infinie du nombre de processus.

L'erreur est peut-être d'une autre nature: on peut remarquer que l'instruction `return` du programme étudié est indentée comme si elle faisait partie de la boucle ; le concepteur de cette fonction voulait peut-être écrire :

```c
int main(void) {
  for (;;) {
    fork();
    return 0;
  }
}
```

ce qui n'est pas non plus un programme très intéressant mais qui a au moins le mérite de terminer après avoir créé un seul processus fils. En bref, attention à la syntaxe permissive mais parfois trompeuse du langage.

b.

1. On peut modifier la fonction de la façon suivante :

```c
#include <stdio.h>
#include <stdlib.h>
#include <sys/wait.h>
#include <unistd.h>

int calcul(int x, int n) {
  int a, b, pid;
  int p = n / 2;
  int fds1[2], fds2[2];

  if (n == 0)
    return 1;
  if (n == 1)
    return x;

```

```
16    if (pipe(fds1) == -1) {
17      perror("Erreur creation tube");
18      exit(1);
19    }
20
21    pid = fork();
22    switch (pid) {
23    case -1:
24      perror("Erreur creation processus fils");
25      exit(1);
26    case 0: // fils
27      close(fds1[0]);
28      a = calcul(x, p);
29      write(fds1[1], &a, sizeof(int));
30      exit(0);
31    default: // pere
32      close(fds1[1]);
33      if (pipe(fds2) == -1) {
34        perror("Erreur creation tube");
35        exit(1);
36      }
37      pid = fork();
38      switch (pid) {
39      case -1:
40        perror("Erreur creation processus fils");
41        exit(1);
42      case 0:
43        close(fds2[0]);
44        b = calcul(x, n - p);
45        write(fds2[1], &b, sizeof(int));
46        exit(0);
47      default:
48        close(fds2[1]);
49        read(fds1[0], &a, sizeof(int));
50        read(fds2[0], &b, sizeof(int));
51        wait(NULL);
52      }
53
54      wait(NULL);
55    }
56
57    return (a * b);
58  }
```

2. Parmi ces deux fonctions calculant x^n, la première méthode est plus efficace car, même si elles se basent toutes les deux sur la même décomposition du calcul, la seconde méthode créera un processus supplémentaire par rapport à la première méthode. Cependant, même la première fonction est très peu efficace pour calculer x^n : en inversant dans le premier programme les lignes `read(fds[0], &a, sizeof(int));` et `b = calcul(x, n - p);`, on permet d'effectuer les deux calculs en parallèle sans que le père n'attende le fils.

   On remarquera également que, pour une opération aussi simple (qui peut se faire *via* une exponentiation rapide par exemple), le surcoût de cette approche est démesuré : on construit autant de processus (et de tubes) que l'exposant n, qui sont des objets complexes et coûteux du point de vue du noyau.

Partie 2. Nombres flottants

Question 2-1

a. On a $s = 0$, $E = 0$ et $f = 0$ donc le nombre réel correspondant au flottant $0 \mid 0\cdots0 \mid 0\cdots0$ est 0.

b. On a $s = 0$, $E = 0$ et $f = 1$ donc le nombre réel correspondant au flottant $0 \mid 0\cdots0 \mid 0\cdots01$ est $(-1)^0 \times 2^{1-1023} \times \overline{0.\underbrace{0\cdots0}_{51\text{ zéros}}1}^2 = 2^{-1074}$

c. On a $s = 0$, $E = 1$ et $f = 0$ donc le nombre réel correspondant au flottant $0 \mid 0\cdots01 \mid 0\cdots0$ est $(-1)^0 \times 2^{1-1023} \times 1.0 = 2^{-1022}$.

d. On a $s = 0$, $E = 1$ et $f = 1$ donc le nombre réel correspondant au flottant $0 \mid 0\cdots01 \mid 0\cdots01$ est $(-1)^0 \times 2^{1-1023} \times 1.\underbrace{0\cdots0}_{51\text{ zéros}}1 = 2^{-1022} + 2^{-1074}$.

Question 2-2

a. La représentation binaire de 1 est $0 \mid 01\cdots1 \mid 0\cdots0$ car $(-1)^0 \times 2^{1023-1023} \times 1.0 = 1$.

b. La représentation binaire de -1 est $1 \mid 01\cdots1 \mid 0\cdots0$ car $(-1)^1 \times 2^{1023-1023} \times 1.0 = -1$.

c. La représentation binaire de 2^{-53} est $0 \mid 01111001010 \mid 0\cdots0$ car $(-1)^0 \times 2^{970-1023} \times 1.0 = 2^{-53}$ et $\overline{01111001010}^2 = 970$.

d. La représentation binaire de $1 + 2^{-50}$ est $1 \mid 01\cdots1 \mid 0\cdots0100$ car $(-1)^0 \times 2^{1023-1023} \times 1.\underbrace{0\cdots0}_{49\text{ zéros}}100 = 1 + 2^{-50}$.

Question 2-3

> **Remarque 5**
>
> Vous pouvez utiliser cet utilitaire en ligne pour voir le fonctionnement de la représentation des flottants encodés sur 32 bits.

a. Si $s = 0$, $E = 0$ et $f \neq 1\cdots1$, alors $x = 2^{-1022} \times 0.f$ donc

$S(x) = 2^{-1022} \times 0.(f + 1) = 2^{-1022} \times (0.f + 2^{-52}) = x + 2^{-1074}$.

b. Si $s = 0$, $1 \leqslant E \leqslant 2^{11} - 2$ et $f \neq 1\cdots1$ alors $x = 2^{E-1023} \times 1.f$ donc

$S(x) = 2^{E-1023} \times 1.(f + 1) = 2^{E-1023} \times (1.f + 2^{52}) = x + 2^{E-1075}$

c. Si $s = 0$, $E = 0$ et $f = 1 \cdots 1$ alors $x = 2^{-1022} \times 0.f = 2^{-1022} \times (1 - 2^{-53}) = 2^{-1022} - 2^{-1075}$.

Or, la représentation de x est $0 \mid 0 \cdots 0 \mid 1 \cdots 1$ donc celle de $S(x)$ est $0 \mid 0 \cdots 01 \mid 0 \cdots 0$.

On a alors $S(x) = (-1)^0 \times 2^{1-1023} \times 1.0 = 2^{-1022} = x + 2^{-1075}$.

d. Si $s = 0, 0 \leqslant E \leqslant 2^{11} - 2$ et $f \neq 1 \cdots 1$, alors d'après les **Questions 2-3-a** et **2-3-b**, on a directement : $S(x) = x + 2^{\max(1,E)-1075}$.

e. Si $s = 0, 0 \leqslant E \leqslant 2^{11} - 3$ et $f = 1 \cdots 1$, alors d'après la question c, si $E = 0$ alors $S(x) = x + 2^{-1075}$, sinon :

$x = 2^{E-1023} \times (2 - 2^{-53}) = 2^{E-1023} - 2^{E-1076}$ et $S(x) = (-1)^0 \times 2^{E-1023+1} \times 1.0 = 2^{E-1022}$, donc $S(x) = x + 2^{E-1076}$.

Donc : $S(x) = x + 2^{\max(1,E)-1076}$ pour $s = 0$, $0 \leqslant E \leqslant 2^{11} - 3$ et $f = 1 \cdots 1$.

De plus, si $E = 2^{11} - 2$ et $f = 1 \cdots 1$, alors $S(x)$ n'est pas défini car la représentation de $S(x)$ est $0 \mid 1 \cdots 1 \mid 0 \cdots 0$.

Remarque 6

Dans la représentation usuelle des flottants, le cas où $E = 2^{11} - 1$ et $f = 0$ correspond aux valeurs $+\infty$ et $-\infty$ (selon la valeur de s). Le cas où $E = 2^{11} - 1$ et $f \neq 0$ correspond à `NaN` (*Not a Number*).

f. À partir des **Questions 2-3-d** et **2-3-e**, et par symétrie entre les cas $s = 0$ et $s = 1$, on a :

- Si $0 \leqslant E \leqslant 2^{11} - 2$ et $f \neq 1 \cdots 1$, alors $S(x) = x + (-1)^s \times 2^{\max(1,E)-1075}$.
- Si $0 \leqslant E \leqslant 2^{11} - 3$ et $f = 1 \cdots 1$, alors $S(x) = x + (-1)^s \times 2^{\max(1,E)-1076}$.

Question 2-4

a. Pour 0, on prend $n = 0$ et $e = 0 : 0 = 0 \times 2^0$.

Pour 1, on prend $n = 1$ et $e = 0 : 1 = 1 \times 2^0$.

Pour $\frac{1}{8}$, on prend $n = 1$ et $e = -3 : \frac{1}{8} = 1 \times 2^{-3}$.

Pour 2^{52}, on prend $n = 2^{52}$ et $e = 0 : 2^{52} = 2^{52} \times 2^0$.

Pour 2^{-52}, on prend $n = 2^{-52}$ et $e = 0 : 2^{-52} = 2^{-52} \times 2^0$.

b. Le plus grand nombre flottant est celui tel que $n = 2^{53} - 1$ et $e = 970$, c'est-à-dire $2^{1023} - 2^{970}$.

Le plus petit nombre flottant strictement positif est celui tel que $n = 1$ et $e = -1074$, c'est-à-dire 2^{-1074}.

c. On raisonne par disjonction de cas sur la valeur de e.

- Si $-52 < e \leqslant 0$, alors $n \cdot 2^e = 1$ avec $n = 2^{-e} \in [\![1, 2^{52}]\!]$.

  Sinon, $\forall n \neq 2^{-e}$, on a directement $n \cdot 2^e \notin [1 - 2^{-52}, 1 + 2^{-52}]$.
- Si $e > 0$, alors $\{n \cdot 2^e \mid |n| < 2^{53}\} \subseteq 2\mathbb{N}$ donc $\{n \cdot 2^e \mid |n| < 2^{53}\} \cap [1 - 2^{-52}, 1 + 2^{-52}] = \emptyset$.
- Si $e < -53$, alors le plus grand flottant représentable est $(2^{53} - 1) \times 2^e \leqslant (2^{53} - 1) \times 2^{-54} = \frac{1}{2} - 2^{-54} < 1 - 1^{-52}$. Donc il n'y a aucun nombre flottant dans $[1 - 2^{-52}, 1 + 2^{-52}]$ tel que $e < -53$.

- Si $e = -53$, alors on a :
  - avec $n = 2^{53} - 1$: $(2^{53} - 1) \times 2^{-53} = 1 - 2^{-53}$
  - avec $n = 2^{53} - 2$: $(2^{53} - 2) \times 2^{-53} = 1 - 2^{-52}$
  - pour toutes les autres valeurs de n, on sort de l'intervalle demandé.
- Si $e = -52$, alors on a :
  - avec $n = 2^{52} + 1$: $(2^{52} + 1) \times 2^{-52} = 1 + 2^{-52}$
  - pour toutes les autres valeurs de n, on sort de l'intervalle demandé.

Donc les flottants de $[1 - 2^{-52}, 1 + 2^{-52}]$ sont : 1, $1 - 2^{-53}$, $1 + 2^{-53}$ et $1 + 2^{-52}$.

Question 2-5

2^{52} est représentable par les flottants avec $n = 1$ et $e = 52$, c'est-à-dire $0 \mid 10000110011 \mid 0 \cdots 0$. Le plus petit flottant strictement supérieur est $S(2^{52}) = 2^{52} + 2^{\max(\overline{10000110011}^2, 1) - 1075} = 2^{52} + 2^0 = 2^{52} + 1$. Il n'est donc pas possible de représenter $2^{52} + 2^{-3}$ de manière exacte, et 2^{52} est le flottant le plus proche, d'où $2^{-52} \oplus 2^{-3} = 2^{52}$.

On a donc : $(-2^{52} \oplus 2^{52}) \oplus \frac{1}{8} = \frac{1}{8}$ et $-2^{52} \oplus \left(2^{52} + \frac{1}{8}\right) = 0$. On en conclut donc que $\oplus$ n'est pas associatif.

Question 2-6

On prend $x = 2^{-1074}$, qui est représentable par les flottants d'après la **Question 2-2**.

On a alors : $x \times x = 2^{-2148}$, qui n'est pas représentable pour les flottants. Le flottant le plus proche est alors 0, donc $x \otimes x = 0$.

Cela peut alors causer un problème pour le code python donné car, comme on vient de le voir, il est possible de faire une division par zéro avec `1 / (x * x)` sans que `x` ne vaille `0`. Ce type d'erreur sera par ailleurs particulièrement difficile à repérer, il vaudrait mieux tester à la première ligne `x * x != 0`.

Question 2-7

a. En prenant $x = 0$, on a immédiatement $1 \oplus x = 1$.

b. La représentation de 1 en flottant est $0 \mid 01 \cdots 1 \mid 0 \cdots 0$. De plus, d'après la **Question 2-3**, le plus petit flottant strictement plus grand que 1 est $S(1) = 1 + 2^{\overline{01111111111}^2 - 1075} = 1 + 2^{-52}$.

On cherche donc $1 + x$ à mi-chemin entre 1 et $1 + 2^{-52}$, car alors, puisque 1 a la fraction pair, on aura $\mathcal{N}(1 + x) = 1$ et, pour tout $x < x' \leq 2^{-52}$, $\mathcal{N}(1 + x') = 1 + 2^{-52}$ par définition de $\mathcal{N}$, ce qui garantit qu'un tel x est maximal. Ainsi $1 + x = \frac{1 + (1 + 2^{-52})}{2}$, donc $x = 2^{-53}$ est le plus grand flottant tel que $1 \oplus x = 1$.

Question 2-8

a. En prenant $x = 0$, on a immédiatement $\mathcal{D}(1 + x) = 1$.

b. On cherche donc le plus grand flottant x tel que $1 + x < 1 + 2^{-52}$. Pour x maximal, il faut prendre x tel que $S(x) = 2^{-52}$. Sachant que 2^{-52} est représenté par $0 \mid 01111001011 \mid 0 \cdots 0$, son antécédent (c'est-à-dire x) est $0 \mid 01111001010 \mid 1 \cdots 1$, et donc $x = 2^{-53} \times 1.1 \cdots 1 = 2^{-53}(2 - 2^{-52}) = 2^{-52}(1 - 2^{-53})$.

Question 2-9

a. En prenant $x = 0$, on a immédiatement $\mathcal{U}(1 + x) = 1$.

b. Par définition, $1 + x \leq \mathcal{U}(1 + x)$ d'une part, et d'autre part, on cherche x qui satisfait $\mathcal{U}(1 + x) = 1$, si bien que $x \leq 0$. En particulier $x = 0$ convient, il est donc maximal.

Si on cherche x maximal en valeur absolue, il faut réaliser une analyse essentiellement similaire aux questions précédentes, mais pour la soustraction.

Question 2-10

Soient x et y deux flottants $|x + y| \leqslant 2^{-1022}$. Sans perte de généralité, on suppose que $|x| \geqslant |y|$.

On souhaite montrer que $x + y$ est un flottant, c'est-à-dire qu'il existe n et e des entiers tels que $|n| < 2^{53}$, $-1074 \leqslant e \leqslant 970$ et $x + y = n \cdot 2^e$.

Comme x et y sont des flottants, alors il existe n_x, n_y,e_x et e_y quatre entiers tels que $|n_x|, |n_y| < 2^{53}$, $-1074 \leqslant e_x, e_y \leqslant 970$, $x = n_x \cdot 2^{e_x}$ et $y = n_y \cdot 2^{e_y}$.

On a alors : $x + y = n_x \cdot 2^{e_x} + n_y \cdot 2^{e_y} = \left(n_x + n_y \cdot 2^{e_y - e_x}\right) \cdot 2^{e_x}$. On pose alors : $n = n_x + n_y \cdot 2^{e_y - e_x}$ et $e = e_x$. On a directement que $x + y = n \cdot 2^e$, il reste à montrer que $|n| < 2^{53}$ et $-1074 \leqslant e \leqslant 970$.

Comme $-1074 \leqslant e_x, e_y \leqslant 970$, alors on a directement que $-1074 \leqslant e \leqslant 970$ car $e = e_x$ ou $e = e_y$.

Par ailleurs, on sait que $|x + y| = |n| \cdot 2^e \leqslant 2^{-1022}$, donc

$$\begin{aligned} |n| &\leqslant 2^{-1022-e} \\ &\leqslant 2^{-1022+1074} \text{ par croissance de } x \mapsto 2^x \\ |n| &\leqslant 2^{52} < 2^{53} \end{aligned} \tag{1}$$

Donc $n \cdot 2^e$ est une décomposition valide de $x + y$ en flottant, donc $x + y$ est un flottant, d'où $x + y = x \oplus y$.

Donc : $|x + y| \leqslant 2^{-1022} \implies x + y = x \oplus y$.

Question 2-11

On a exactement : $\left(2^{52} + \frac{1}{8}\right) - \left(2^{52} - \frac{1}{8}\right) = \frac{2}{8} = \frac{1}{4}$.

D'autre part, on a $2^{52} \oplus \frac{1}{8} = 2^{52}$ en reprenant le résultat de la **Question 2-5**. De la même façon, on peut démontrer que $2^{52} \ominus \frac{1}{8} = 2^{52}$ par le même raisonnement (2^{52} est le flottant le plus proche de $2^{52} - \frac{1}{8}$). On en conclut donc que $\left(2^{52} \oplus \frac{1}{8}\right) \ominus \left(2^{52} \ominus \frac{1}{8}\right) = 0$.

Comme $\left(2^{52} + \frac{1}{8}\right) - \left(2^{52} - \frac{1}{8}\right) \neq \left(2^{52} \oplus \frac{1}{8}\right) \ominus \left(2^{52} \ominus \frac{1}{8}\right)$, on peut donc conclure que les arrondis successifs conduisent à un résultat complètement faux (qui peut même mener à des erreurs telle que celle décrite en **Question 2-6**).

Remarque 7

On peut d'ailleurs observer ce phénomène en Python en tapant la ligne suivante dans un interpréteur (ici Python 3.11.9) :

```python
>>> (2**52 + 1/8) - (2**52 - 1/8)
0.0
```

Question 2-12

Remarque 8

Il s'agit ici d'un cas particulier du lemme de Sterbenz. Vous pouvez consulter la page Wikipédia associée (disponible uniquement en anglais) pour plus de détails.

Soient x et y deux nombres flottants tels que $\frac{y}{2} \leqslant x \leqslant 2y$. De la même façon qu'à la **Question 2-10**, on cherche à montrer que $x - y$ est un flottant, c'est-à-dire qu'il existe n et e deux entiers tels que $|n| < 2^{53}$, $-1074 \leqslant e \leqslant 970$ et $x - y = n \cdot 2^e$.

Comme x et y sont flottants, alors il existe n_x, n_y,e_x et e_y quatre entiers tels que $|n_x|, |n_y| < 2^{53}$, $-1074 \leqslant e_x, e_y \leqslant 970$, $x = n_x \cdot 2^{e_x}$ et $y = n_y \cdot 2^{e_y}$. On considère ici e_x et e_y minimaux parmi tous ceux vérifiant ces propriétés.

- Si $x = 0$, alors $x - y = -y$, qui est toujours une opération exacte car la négation d'un flottant consiste simplement en l'inversion du premier bit.
- Si $x = y$, alors le résultat est 0 donc également exact.
- Si $x < 0$, alors $y < 0$ et comme $\frac{y}{2} \leqslant x \leqslant 2y$ on a : $x - y = -(-x - (-y))$, on peut donc se ramener au cas $x, y \geqslant 0$ (toujours car l'opération de négation d'un flottant est toujours exacte).
- Si $x \leqslant y$, alors on peut écrire $x - y = -(y - x)$ avec $\frac{x}{2} \leqslant y \leqslant 2x$: on peut donc se ramener au cas $x \geqslant y$.
- Si $y = 0$, alors $x - y = x$ est toujours exact.

On peut donc se ramener au dernier cas : $0 < y < x \leqslant 2y$ sans perte de généralité.

On a alors : $x - y = n_x \cdot 2^{e_x} - n_y \cdot 2^{e_y}$. De plus, on a supposé $x \geqslant y$ et e_x et e_y minimaux donc $e_x \geqslant e_y$, d'où : $x - y = \left(n_x \cdot 2^{e_x - e_y} - n_y\right) \cdot 2^{e_y}$. On pose alors : $n = n_x \cdot 2^{e_x - e_y} - n_y$, et $e = e_y$.

Comme $e_x - e_y \geqslant 0$, alors $2^{e_x - e_y}$ est un entier, donc n aussi. De plus, on a :

$$\begin{aligned} & x \leqslant 2y \\ \iff & x - y \leqslant y \\ \iff & n \cdot 2^e \leqslant n_y \cdot 2^{e_y} \\ \iff & 0 < n \leqslant n_y < 2^{53} \end{aligned} \tag{2}$$

Donc : $|n| < 2^{53}$. De plus, $e = e_y$ donc $-1074 \leqslant e \leqslant 970$.

Donc $n \cdot 2^e$ est une représentation en flottant valide de $x - y$, donc $x - y$ est un flottant.

Donc : pour tous flottants x et y, si $\frac{y}{2} \leqslant x \leqslant 2y$, alors $x - y = x \ominus y$.

Partie 3. Logique, déduction naturelle

Exercice 3-1

a. On pose les propositions suivantes :

- R : « Informatix réussit sa preuve. »
- P : « Il pleut. »
- M : « Informatix reste chez lui. »
- E : « Informatix s'entraîne. »

- C : « Informatix est content. »

On peut alors formaliser ces formules de la manière suivante :

1. $R \Rightarrow C$
2. $P \Rightarrow M$
3. $\neg M \Rightarrow C$
4. $M \Rightarrow E$
5. $E \Rightarrow R$

b. On peut déduire de ces axiomes qu'Informatix sera content, i.e $\vdash C$, avec l'arbre de dérivation suivant. On pose : $\Gamma = \{R \Rightarrow C, P \Rightarrow M, \neg M \Rightarrow C, M \Rightarrow E, E \Rightarrow R\}$.

On démontre dans un premier temps que s'il est chez lui, alors il est content. On appelle π_1 cet arbre de preuve :

$$\dfrac{\dfrac{}{\Gamma, M \vdash R \Rightarrow C}\,\text{Ax} \qquad \dfrac{\dfrac{}{\Gamma, M \vdash E \Rightarrow R}\,\text{Ax} \qquad \dfrac{\dfrac{}{\Gamma, M \vdash M \Rightarrow E}\,\text{Ax} \qquad \dfrac{}{\Gamma, M \vdash M}\,\text{Ax}}{\Gamma, M \vdash E}\Rightarrow_{\text{e}}}{\Gamma, M \vdash R}\Rightarrow_{\text{e}}}{\Gamma, M \vdash C}\Rightarrow_{\text{e}} \tag{3}$$

On montre maintenant que, s'il n'est pas chez lui, alors il est content, et on appelle π_2 cet arbre :

$$\dfrac{\dfrac{}{\Gamma, \neg M \vdash \neg M \Rightarrow C}\,\text{Ax} \qquad \dfrac{}{\Gamma, \neg M \vdash \neg M}\,\text{Ax}}{\Gamma, \neg M \vdash C}\Rightarrow_{\text{e}} \tag{4}$$

Finalement, on démontre avec les axiomes Γ qu'Informatix est content, car d'après la règle EM, il est soit chez lui, soit dehors :

$$\dfrac{\dfrac{}{\Gamma \vdash M \vee \neg M}\,\text{EM} \qquad \pi_1 \qquad \pi_2}{\Gamma \vdash C}\vee_e \tag{5}$$

Exercice 3-2

a. Voici des arbres de preuves des formules demandées :

1.

$$\dfrac{\dfrac{}{\Gamma, A \Rightarrow B \vdash A \vee \neg A}\,\text{EM} \qquad \dfrac{\dfrac{\dfrac{}{\Gamma, A \Rightarrow B, A \vdash A \Rightarrow B}\,\text{Ax} \qquad \dfrac{}{\Gamma, A \Rightarrow B, A \vdash A}\,\text{Ax}}{\Gamma, A \Rightarrow B, A \vdash B}\Rightarrow_{\text{e}}}{\Gamma, A \Rightarrow B, A \vdash \neg A \vee B}\vee_{id} \qquad \dfrac{\dfrac{}{\Gamma, A \Rightarrow B, \neg A \vdash \neg A}\,\text{Ax}}{\Gamma, A \Rightarrow B, \neg A \vdash \neg A \vee B}\vee_{ig}}{\Gamma, A \Rightarrow B \vdash \neg A \vee B}\vee_e}{\Gamma \vdash (A \Rightarrow B) \Rightarrow (\neg A \vee B)}\Rightarrow_{\text{i}} \tag{6}$$

2.

$$\dfrac{\dfrac{\dfrac{}{\Gamma, \neg A \vee B, A, \neg A \vdash A}\,\text{Ax} \qquad \dfrac{}{\Gamma, \neg A \vee B, A, \neg A \vdash \neg A}\,\text{Ax}}{\Gamma, \neg A \vee B, A, \neg A \vdash B}\neg_e \qquad \dfrac{}{\Gamma, \neg A \vee B, A, B \vdash B}\,\text{Ax}}{\Gamma, \neg A \vee B, A \vdash B}\vee_e}{\Gamma, \neg A \vee B \vdash A \Rightarrow B}\Rightarrow_{\text{i}}}{\Gamma \vdash (\neg A \vee B) \Rightarrow (A \Rightarrow B)}\Rightarrow_{\text{i}} \tag{7}$$

Le séquent $\Gamma \vdash (A \Rightarrow B) \Rightarrow (\neg A \vee B)$ est classique car, comme le montre le premier arbre de preuve, la règle du tiers exclu (EM) qui n'est pas intuitionniste est utilisée.

b. Voici un arbre de preuve de $\Gamma \vdash \exists x\ \neg P(x) \Rightarrow \neg(\forall x\ P(x))$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma, \neg P(t), \forall x\ P(x), P(t) \vdash P(t)}\,\mathrm{Ax}}{\Gamma, \neg P(t), \forall x\ P(x) \vdash P(t)}\,\forall_h}{\Gamma, \neg P(t), \forall x\ P(x) \vdash \bot}\,\neg_h}{\Gamma, \neg P(t) \vdash \neg(\forall x\ P(x))}\,\neg_i}{\Gamma, \exists x\ \neg P(x) \vdash \neg(\forall x\ P(x))}\,\exists_h \quad x \notin \mathrm{vl}(\Gamma, \forall x P(x))}{\Gamma \vdash \exists x\ \neg P(x) \Rightarrow \neg(\forall x\ P(x))}\,\Rightarrow_{\mathrm{i}} \tag{8}$$

c. Il est possible de démontrer que $\vdash \neg\neg A \Rightarrow A$ à partir de la règle $\vdash A \vee \neg A$ car ces deux formules sont logiquement équivalentes. Voici un arbre de preuve de cette formule :

$$\dfrac{\dfrac{\dfrac{}{\neg\neg A \vdash A \vee \neg A}\,\mathrm{EM} \quad \dfrac{}{\neg\neg A, A \vdash A}\,\mathrm{Ax} \quad \dfrac{\dfrac{}{\neg\neg A, \neg A \vdash \neg A}\,\mathrm{Ax} \quad \dfrac{}{\neg\neg A, \neg A \vdash \neg\neg A}\,\mathrm{Ax}}{\neg\neg A, \neg A \vdash A}\,\neg_e}{\neg\neg A \vdash A}\,\vee_e}{\vdash \neg\neg A \Rightarrow A}\,\Rightarrow_{\mathrm{i}}$$

d. On peut prendre la propriété $P(x, y)$: « x est divisible par y » définie pour $x, y \in \mathbb{N}$ et $x, y \geqslant 2$. Ainsi définie, on a bien :

- $\forall x\ \exists y\ P(x, y)$: on peut s'en convaincre en prenant $y = x$: pour tout $x \in \mathbb{N}$, $x \geqslant 2$, x est bien divisible par lui-même.
- il n'existe pas d'entier $y \geqslant 2$ divisant tout entier $x \geqslant 2$ (on s'en convainc en raisonnant par l'absurde et en prenant $x = y + 1$). On a donc : $\neg\exists y\ \forall x\ P(x, y)$.

Donc $\forall x\ \exists y\ P(x, y)$ et $\exists y\ \forall x\ P(x, y)$ ne sont pas équivalents.

Le rapport du sujet met pour cette question l'accent sur la pédagogie en invitant à prendre les prédicats les plus intuitifs possibles, c'est-à-dire ceux pour lesquels la non-équivalence se « prouve » avec le moins de bagage mathématique possible. Par exemple, « y est le père de x ».

Partie 4. Réseaux

Question 4-1

Remarque 9

Le fonctionnement décrit ici est un résumé du véritable protocole DHCP qui est en réalité plus complexe. Pour plus de détails, vous pouvez consulter la page Wikipédia associée ou la RFC 2131 introduisant le protocole DHCP.

Pour que la machine A obtienne son adresse IP, plusieurs trames doivent être échangées dans le réseau pour le bon fonctionnement du protocole DHCP :

- La machine A envoie en broadcast un message `DHCP_DISCOVER` pour vérifier s'il y a un serveur DHCP sur le réseau. Dans la trame échangée de couche 2, l'adresse MAC source est `08:02:3A:52:E8:B1` (celle de la machine A) et celle de destination est `FF:FF:FF:FF:FF:FF` (adresse MAC de broadcast). L'adresse IP de destination est celle de broadcast `255.255.255.255`. L'adresses IP source indiquée dans le paquet de couche 3 n'est pas significative, on met donc `0.0.0.0`.
- Le serveur DHCP répond alors à la machine A en envoyant un message `DHCP_OFFER` pour proposer une adresse IP. Dans la trame envoyée, l'adresse MAC source est `08:02:3A:52:D8:09` (celle du

serveur DHCP), et celle de destination est celle de la machine A, obtenue par le premier message envoyé. Dans le paquet envoyé, l'adresse IP source est `174.24.64.11` (celle du serveur DHCP), et celle de destination est celle proposée par le serveur DHCP à la machine A, par exemple `174.24.64.45`. Le message est donc envoyé en unicast, et contient entre autres le masque de sous-réseau `255.255.192.0`. Le serveur DHCP peut également envoyer un `ping` par le protocole ICMP pour vérifier si l'adresse IP proposée est bien disponible et n'est pas utilisée par un autre appareil sur le réseau.

- La machine A envoie alors un message `DHCP_REQUEST` en broadcast sur le réseau pour accepter l'adresse IP proposée par le serveur DHCP. Dans la trame envoyée, les adresses MAC source et destination sont les mêmes que pour le premier paquet envoyé (diffusion en broadcast), alors que dans le paquet l'IP source est encore `0.0.0.0`, et celle de destination est `174.24.64.11` (celle du serveur DHCP).

  Remarque 10

  Ce message est envoyé en broadcast et non en unicast car cela permet de répondre en un seul message à tous les serveurs DHCP en indiquant quelle adresse IP a été acceptée par la machine A dans le cas où plusieurs serveurs sont présents sur un même réseau et ont envoyé un message `DHCP_OFFER`.

- Le serveur DHCP répond à la machine A par un dernier message `DHCP_ACK` avec les mêmes adresses MAC et IP sources et destinations que pour le message `DHCP_OFFER`. Le message contient par ailleurs des informations supplémentaires comme l'adresse IP du routeur `174.24.64.1`, le masque de sous-réseau `255.255.192.0`, la durée du bail de cette adresse ou encore l'adresse IP d'un serveur DNS `174.24.64.22`.

Question 4-2

Voici les trames échangés pour établir une session WEB entre la machine A et le serveur WEB :

- Il faut tout d'abord effectuer une requête DNS pour que la machine A trouve l'adresse IP du serveur WEB. Pour cela, la machine A doit contacter le serveur DNS, dont elle connaît l'adresse IP grâce au protocole DHCP.
  - Il faut en premier lieu que la machine trouve l'adresse MAC du serveur DNS afin de pouvoir lui envoyer des messages. La machine A va donc envoyer une requête ARP en broadcast avec l'adresse IP `174.24.64.22` du serveur DNS comme destination, afin de retrouver son adresse MAC `08:02:3A:52:E8:12` dans la réponse qu'enverra le serveur DNS.
  - Désormais, il est possible de communiquer avec le serveur DNS : la machine A va réaliser une requête DNS de type A (pour retrouver l'adresse IPv4) pour le nom de domaine `www.agreg-info.org` au serveur DNS. Celui-ci possède en cache cette information : le serveur renverra comme réponse à la machine A l'adresse IP `185.26.126.225` du serveur WEB.

    Remarque 11

    Vous pouvez consulter la page Wikipédia sur le DNS pour davantage de détails sur le fonctionnement du protocole DNS, notamment à propos des différents types d'enregistrement.

- La machine A peut maintenant contacter le serveur WEB. Toutes les paquets de couche 3 entre la machine A et le serveur sont échangés de la même manière : la machine A envoie une trame au routeur R1 (dont l'adresse MAC a été récupérée par une requête ARP comme pour le serveur DNS), qui transmettra à son tour une trame au routeur R2, etc. jusqu'au serveur WEB. En effet, la machine A détecte que le serveur WEB ne se trouve pas sur le même réseau qu'elle, graĉe à l'adresse IP et au masque de réseau connu par la machine A : toutes les trames envoyées par la machine A vers le serveur WEB devront alors passer par le routeur R1. Les paquets envoyés par le serveur WEB jusqu'à la machine A fonctionnent exactement de la même façon mais dans le sens inverse. On pourra également noter l'utilisation de la fonction NAT par le routeur R1 pour traduire sa propre adresse IP publique en l'adresse IP privée de la machine A.
  - La machine A commencera par envoyer un paquet au serveur WEB pour demander l'établissement d'une session TCP. Le serveur WEB répondra alors qu'il accepte la mise en place de cette session par un autre paquet, puis la machine renverra un dernier paquet de confirmation d'établissement de connexion. Il s'agit d'un handshake TCP.
  - Après la création de la session TCP, la véritable requête HTTP peut avoir lieu : la machine A enverra une requête de méthode `GET` à l'emplacement `/` au serveur `www.agreg-info.org` pour consulter le site WEB `http://www.agreg-info.org`. Le serveur WEB renverra alors la page demandée sous format HTML. Le client WEB verra alors que d'autres ressources sont nécessaires pour afficher correctement la page WEB (feuilles de style, images, documents, …) et fera de nouvelles requêtes HTTP pour chacune d'entre elles. Le serveur WEB répondra alors à chaque requête avec la ressource demandée associée.

Question 4-3

Dans la question précédente, deux protocoles applicatifs sont utilisés : la protocole DNS et le protocole HTTP. Ces deux protocoles n'utilisent pas les mêmes protocoles de transport (de couche 4) : le protocole DNS utilise UDP tandis que le protocole HTTP utilise TCP, expliquant le besoin de la mise en place d'une session TCP pour la requête HTTP alors que cela n'était pas nécessaire pour la requête DNS.

Question 4-4

Remarque 12

Pour plus de détails sur chaque sous-question, vous pouvez lire la page Wikipédia sur le protocole IPv4 contenant une description de chaque champ de l'en-tête IPv4.

a. Sur le lien R2-R1, l'unité maximale de transmission est 1020 octets, expliquant le découpage de la charge utile en deux paquets, dont le plus grand contient 1020 octets.

On peut vouloir limiter la taille maximale des trames sur les supports de transmission pour éviter la congestion de certains liens sur le réseau, ou encore pour limiter le coût d'une retransmission d'un paquet très gros en cas d'erreur sur la couche physique.

b. Le protocole IP ne réassemble les fragments qu'à la destination finale car il n'y a aucune garantie que tous les fragments du message initial empruntent le même chemin. Il est donc plus simple de tout réassembler au niveau de la destination finale.

c. Comme le champ `TTL` d'un paquet baisse de 1 à chaque routeur rencontré, ces deux datagrammes ont traversé 9 entre la source et le routeur R1. Entre la source et la destination, 10 routeurs auront donc été traversés.

d. Le champ `Total Length` des deux datagrammes correspondent à la taille totale en octet de chaque datagramme, en-têtes compris. Il faut donc soustraire à ce champ la taille des en-têtes pour obtenir `x` et `y`. Comme il s'agit ici d'un datagramme IP de version 4 (champ `Ver.`), alors les adresses IP sont des adresses IPv4, donc encodées sur 32 bits. On en déduit qu'une ligne du schéma fait 32 bits, donc 4 octets, et il y a 5 lignes d'en-têtes (visible sur le schéma ou lisible dans le champ `IHL`). On peut donc en déduire que `x` est 1000 et que `y` est 460.

e. La fonction de contrôle est réalisée au niveau de chaque routeur traversé. Cela permet de s'assurer à chaque routeur qu'aucune mauvaise information n'a été transmise, et que si c'est le cas le routeur ayant reçu un mauvais paquet peut demander à ce qu'il soit retransmis directement au précédent routeur traversé (ou à la machine source). De plus, comme le champ `TTL` décroît de 1 à chaque routeur traversé, chaque routeur recalculera un nouveau checksum et changera l'en-tête IPv4 en conséquence.

Question 4-5

a. Le protocole FTP utilise également le protocole de transport TCP, s'appuyant sur l'utilisation de sockets réseaux. Un socket est un triplet (protocole, adresse IP, port) et est utilisé en couche 4 (transport), cette entité n'est donc utilisée que pour les machines source et destination car les routeurs sont des appareils de couche 3 qui sont transparents à l'établissement d'une session TCP.

b. Il est tout à fait possible d'avoir plusieurs connexions simultanées avec une même machine. Il suffit de créer plusieurs paires de sockets (client, serveur). On distinguerait alors les deux par les ports utilisés. Par exemple, les deux paires de sockets pouvant être utilisées pourraient être les suivantes :

- FTP : `<174.24.64.45, 20, 185.26.126.225, 21>`
- WEB : `<174.24.64.45, x, 185.26.126.225, 80>`

Remarque 13

Il est important de distinguer ici les **sockets**, qui sont des triplets `(protocole, adresse IP, port)`, des **paires de sockets** qui sont des quadruplets `(adresse IP source, port source, adresse IP dest, port dest)`.

c. Il est techniquement possible que des segments de la première connexion interfèrent avec la seconde, mais cela est en réalité assez peu probable pour plusieurs raisons :

- Après fermeture d'une connexion, sur différents systèmes et notamment ceux utilisant Linux, il n'est pas possible de réutiliser directement un port réseau : une petite durée est généralement mise en place pour éviter ce cas de figure. Cela est d'autant plus vrai que du point de vue du client, le port de connexion utilisé est déterminé aléatoirement.

  Remarque 14

  Sur Linux, il est possible de se lier à un port aléatoire (comme avec la fonction `connect`) en indiquant comme port `0`.

- Dans le cas où le port utilisé est le même qu'avant, les différents mécanismes mis en place lors d'une connexion TCP permettrait de voir immédiatement que le protocole n'a pas été respecté du point de vue du serveur, qui réinitialiserait alors la connexion. Le client devra alors ré-ouvrir une connexion sur le serveur WEB.

Question 4-6

a. Pour déterminer le prochain saut dans le routage d'un paquet, il faut consulter la table de routage et choisir la ligne où la destination correspond le plus précisément à la destination finale du routage. Plus précisément, il faut choisir la ligne où la destination a le plus long préfixe commun avec la destination finale (en bits), et parmi ceux avec la même longueur choisir celui avec le plus petit sous-réseau (en terme d'adresses IP contenues). Dans le cas où aucune ligne n'a une destination avec un préfixe commun avec la destination finale, une destination par défaut sera choisie (celle correspondant à la destination `0.0.0.0/0`).

Destination	Route	Prochain saut	Interface
`174.24.64.45`	`174.24.64.0/18`	R1	if1
`188.114.255.2`	`188.114.255.0/30`	R3	if3
`188.114.255.4`	`0.0.0.0/0`	R5	if2
`67.44.21.97`	`67.44.16.0/20`	R3	if3

Remarque 15

Précisions :

- `188.114.255.4` n'est pas dans le sous-réseau `188.114.255.0/30` qui ne contient que les adresses IP entre `188.114.255.0` et `188.114.255.3`
- `67.44.21.97` n'est pas dans le sous-réseau `67.44.16.0/22` car il ne contient que les adresses IP entre `67.44.16.0` et `67.44.19.255`

b. Pour des paquets de même source, même destination et de même taille, la latence de transmission peut varier fortement. Les causes principales de cette latence sont la congestion du réseau, pouvant causer un ralentissement de la vitesse de retransmission d'un paquet par les équipements intermédiaires, ou encore la correction d'erreurs liées à la transmission physique des informations. De plus, le protocole IP fonctionne en mode non-connecté, c'est-à-dire que tous les paquets ne prennent pas nécessairement la même route, ce qui peut faire fortement varier les délais de transmission.

Question 4-7

a. Voici les tables de routage des routeurs R2 et R3 :

Routeur R2

Destination	Prochain saut	Interface
R1	R1	if1
R3	R1	if1
R4	R4	if2
R5	R5	if4
R6	R6	if3

Routeur R3

Destination	Prochain saut	Interface
R1	R1	if3
R2	R1	if3
R4	R4	if4
R5	R1	if3
R6	R1	if3

L'ordre dans lequel les messages RIP circulent ont une importance car, pour des destinations pouvant être accédées par deux chemins de même longueur débutant par deux prochains sauts différents, le premier routeur ayant transmis ses informations sera enregistré avant dans la table de routage. C'est

pour cela qu'on a pris la convention de faire passer tous les prochains sauts de toutes les destinations excepté R4 par R1 pour la table de routage de R3.

b. Lors de l'exécution du protocole RIP, tous les routeurs testent la présence de tous leurs voisins en envoyant de manière régulière des pings. Ainsi, le routeur R2 a pu découvrir la panne de la liaison R2 —R6 car le ping effectué par R2 pour tester la présence de R6 sur l'interface R3 ne répondait plus. Le routeur R2 marquera alors le routeur R6 comme inatteignable, puis recevra un message RIP du routeur R5 mentionnant R6 comme voisin : R2 mettra alors à jour sa table de routage pour mentionner que pour aller au routeur R6, le prochain saut doit être R5. R3 quant à lui détecte la panne grâce à la propagation de la table de routage de R2 qui indique notamment que la distance entre R2 et R6, et donc entre R1 (le prochain saut pour atteindre R2) et R6 a également augmenté. Cette information se propagera alors aux autres routeurs par les messages RIP indiquant l'état des voisins pour tous les routeurs du réseau : plus précisément : R5 transmettra l'information à R2, R2 à R1 et R4, et R1 et R4 à R3.

Finalement, voici les tables de routage des routeurs R2 et R3 après convergence :

Routeur R2

Destination	Prochain saut	Interface
R1	R1	if1
R3	R1	if1
R4	R4	if2
R5	R5	if4
R6	**R5**	**if4**

Routeur R3

Destination	Prochain saut	Interface
R1	R1	if3
R2	R1	if3
R4	R4	if4
R5	R1	if3
R6	R1	if3

c. Lors d'une rupture de lien, les phénomènes décrits en question précédente surviennent, notamment la mise à jour des voisins et la propagation de l'information de la nouvelle distance pour atteindre un routeur. Une panne de routeur est plus importante car elle équivaut à une rupture de lien entre ce routeur et tous ses voisins. Comme du point de vue de la topologie du réseau le routeur n'existe plus, en plus des phénomènes présents pour la rupture de lien, on peut imaginer qu'à terme, le routeur en question sera supprimé de la table de routage de tous les routeurs du réseau car il n'existe plus pour eux.

Question 4-8

a. Le temps d'émission d'une trame de longueur minimale de $512\text{o} = (512 \times 8)\text{b} = 4096\text{b}$ sur un réseau Ethernet 1Gb/s est de $\frac{4096\text{b}}{10^9\text{b/s}} = 4,096 \times 10^{-6}\text{s}$, c'est-à-dire environ 4 microsecondes.

b. La période de vulnérabilité d'un réseau est définie comme étant la durée minimale qu'il faut attendre après l'émission d'une trame avant d'en envoyer une autre pour être certain qu'aucune collision n'aura lieu. Cette durée est égale par ailleurs au temps de propagation nécessaire entre les deux équipements réseaux les plus éloignés sur le support physique de transmission, ici un câble Ethernet. Cette durée est ici au plus la durée de transmission d'une trame, puisque tout autre transmission faite jusqu'à la fin de la transmission de la première trame aurait pu provoquer une collision, c'est-à-dire ici environ 4 microsecondes.

c. Le débit support est la quantité d'information totale théorique transmise ou reçue par unité de temps : par exemple, pour des câbles Ethernet de 1 Gbit/s, le débit support sera de 1 Gbit/s. Le débit utile est la quantité d'information utile transmise ou reçue par unité de temps, c'est-à-dire l'information d'une

trame restreinte à sa charge utile, à savoir son contenu sans tous les en-tête. Le débit réel est la quantité d'information totale transmise ou reçue par unité de temps.

Par définition, on a toujours que le débit support est le plus élevé des trois, suivis par le débit réel, puis le débit utile qui doit tenir compte des traitements nécessaires à toutes les couches intermédiaires.