Chapitre 2

Internet

Objectifs du chapitre

Thème : Internet.
Comprendre comment une information circule sur Internet : découpage en paquets, rôle des protocoles IP et TCP, routage, annuaire DNS, réseaux physiques et réseaux pair-à-pair.
Organisation : programme officiel, cours, méthodes, ateliers Python, exercices corrigés, QCM et fiche de synthèse.

Programme officiel (BO)

Thème « Internet » – Contenus et capacités attendues

— **Protocole TCP/IP : paquets, routage des paquets** : Distinguer le rôle des protocoles IP et TCP. Caractériser les principes du routage et ses limites. Distinguer la fiabilité de transmission et l'absence de garantie temporelle.

— **Adresses symboliques et serveurs DNS** : Sur des exemples réels, retrouver une adresse IP à partir d'une adresse symbolique et inversement.

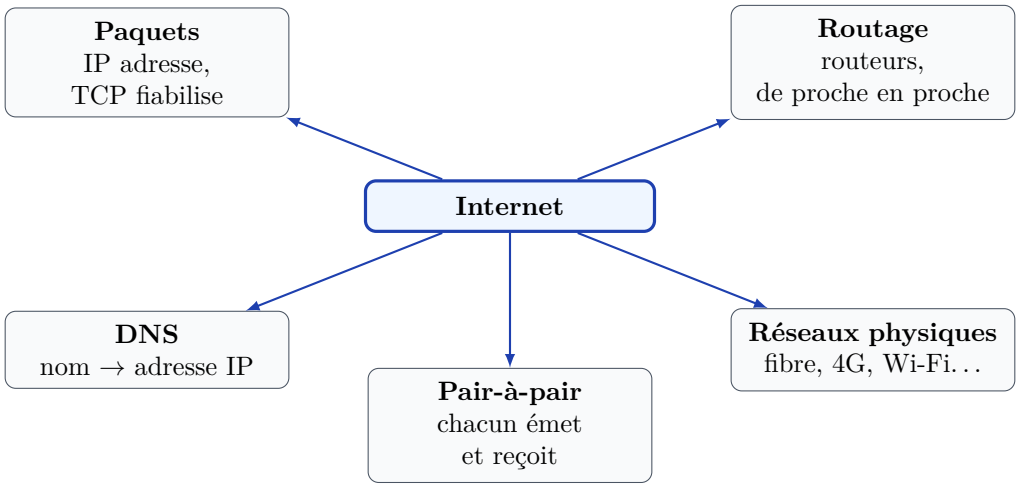
— **Réseaux pair-à-pair** : Décrire l'intérêt des réseaux pair-à-pair ainsi que les usages illicites qu'on peut en faire.

— **Indépendance d'internet par rapport au réseau physique** : Caractériser quelques types de réseaux physiques : obsolètes ou actuels, rapides ou lents, filaires ou non. Caractériser l'ordre de grandeur du trafic de données sur internet et son évolution.

Source : BO spécial n°1 du 22 janvier 2019, programme de Sciences numériques et technologie de Seconde générale et technologique.

2.1 Carte mentale du chapitre

Carte mentale



2.2 Introduction et repères historiques

Grâce à sa souplesse et à son universalité, **Internet** est devenu le principal moyen de communication entre les humains et avec les machines : messagerie, vidéo, jeux en ligne, objets connectés... Internet est un **réseau de réseaux** : il relie entre eux des millions de réseaux locaux (un lycée, une box à la maison, un opérateur mobile).

Date	Repère
Années 1950	Premiers ordinateurs reliés en réseau, très dépendants des constructeurs ou des opérateurs téléphoniques.
1970	**ArpaNet** aux États-Unis : premier grand réseau indépendant des constructeurs.
1971	**Cyclades** en France (Louis Pouzin) : l'idée d'envoyer les données par paquets indépendants.
1974	Vinton Cerf et Robert Kahn décrivent le protocole **TCP/IP**.
1983	ArpaNet adopte TCP/IP : c'est la naissance d'**Internet**. Le système DNS est créé la même année.
1989-1991	Invention du **Web** au CERN : une application qui utilise Internet (chapitre suivant).

Attention

Internet et le Web ne sont pas la même chose. **Internet** est le réseau qui transporte les données ; le **Web** n'est qu'un des services qui l'utilisent, comme la messagerie, la visioconférence ou les jeux en ligne.

2.3 Adresses IP, paquets et protocoles TCP/IP

Définition – Adresse IP

Chaque machine connectée à Internet (ordinateur, téléphone, serveur, objet connecté) est identifiée par une **adresse IP**.

— En **IPv4**, une adresse est formée de 4 nombres entiers compris entre 0 et 255, séparés par des points : `192.168.1.15`. Elle est codée sur 32 bits, soit $2^{32} \approx 4{,}3$ milliards d'adresses, ce qui ne suffit plus.

— En **IPv6**, l'adresse est codée sur 128 bits, soit $2^{128} \approx 3{,}4 \times 10^{38}$ adresses, par exemple

`2001:db8::1`.

Définition – Paquet

Pour être envoyée, une information (texte, image, son, vidéo…) est découpée en **paquets** de taille limitée (environ 1 500 octets). Chaque paquet contient :
— un **en-tête** : adresse IP de l'émetteur, adresse IP du destinataire, numéro du paquet, durée de vie (TTL)… ;
— les **données** : un morceau du message.
Les paquets voyagent indépendamment les uns des autres et peuvent emprunter des chemins différents.

Propriété – Rôles des protocoles IP et TCP

Un **protocole** est un ensemble de règles que les machines respectent pour communiquer.
— **IP** (*Internet Protocol*) donne une adresse à chaque machine et **achemine** chaque paquet vers son destinataire, sans aucune garantie : un paquet peut se perdre, arriver en double ou dans le désordre.
— **TCP** (*Transmission Control Protocol*) **fiabilise** la communication : il découpe et numérote les paquets, le destinataire envoie un **accusé de réception**, les paquets manquants sont redemandés et les paquets sont remis dans l'ordre.
TCP/IP n'est pas installé dans les câbles mais dans **chaque ordinateur connecté** : c'est un protocole logiciel.

Attention

TCP garantit que les paquets **finissent** par arriver (sauf panne matérielle), mais ni Internet ni TCP ne garantissent **quand** ils arrivent : il n'y a **pas de garantie temporelle**. C'est pourquoi une visioconférence ou un jeu en ligne peut saccader : dans une vidéo, on peut perdre une image, mais pas le fil du temps.

Méthode – Calculer le nombre de paquets

Pour envoyer un fichier, on divise sa taille par la quantité de données d'un paquet et on **arrondit à l'entier supérieur** (le dernier paquet est incomplet). Exemple : un fichier de 4 000 octets avec 1 500 octets de données par paquet : $4\,000 \div 1\,500 \approx 2{,}67$, donc il faut **3 paquets**.

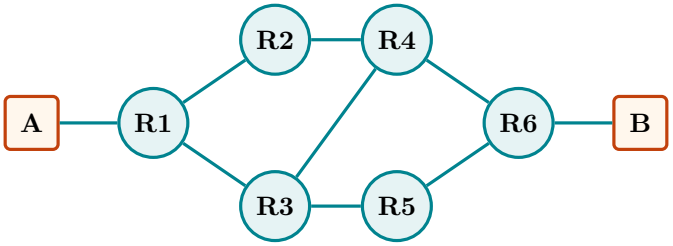
```
import math

taille = 4000            # octets
donnees = 1500           # octets de donnees par paquet
print(math.ceil(taille / donnees))   # arrondi superieur : 3
```

2.4 Le routage des paquets

Définition – Routeur et routage

Un **routeur** est une machine reliée à plusieurs réseaux. Il reçoit des paquets et choisit, pour chacun, le routeur voisin auquel le transmettre. Le **routage** est l'acheminement des paquets de proche en proche, de l'émetteur jusqu'au destinataire.



Dans ce réseau, un paquet envoyé par l'ordinateur A au serveur B peut passer par R1 → R2 → R4 → R6 ou par R1 → R3 → R5 → R6 : il existe plusieurs chemins.

Propriété – Principes du routage

— Chaque routeur possède une **table de routage** : une carte *locale* qui indique à quel voisin envoyer un paquet selon sa destination. Aucun routeur ne connaît la carte complète d'Internet.
— Les routeurs échangent en permanence avec leurs voisins et **mettent à jour** leurs tables quand un routeur est ajouté ou tombe en panne : le réseau se réorganise tout seul.
— Chaque paquet contient un nombre maximal de routeurs à traverser (**TTL**, *Time To Live*). Chaque routeur diminue ce nombre de 1 ; arrivé à 0, le paquet est **détruit** pour ne pas encombrer le réseau.

Propriété – Limites du routage

Un paquet peut ne pas arriver : panne d'une ligne ou d'un routeur, TTL arrivé à 0, ou routeur saturé (**congestion**). Le routage seul ne garantit donc rien : c'est TCP qui redemande les paquets perdus.

Méthode – Observer le réseau avec des commandes

Dans un terminal (sous Windows : touche Windows, puis taper `cmd`) :
— `ipconfig` (Windows) ou `ifconfig` (macOS, Linux) : afficher l'adresse IP de son ordinateur ;
— `ping utspe.com` : vérifier qu'une machine répond et mesurer le temps d'aller-retour d'un paquet ;
— `tracert utspe.com` (Windows) ou `traceroute utspe.com` (macOS, Linux) : afficher la liste des routeurs traversés ;
— `nslookup utspe.com` : interroger le DNS (voir partie suivante).

2.5 Adresses symboliques et serveurs DNS

Définition – Adresse symbolique et DNS

Une **adresse symbolique** (ou nom de domaine) est un nom facile à retenir, comme `utspe.com` ou `wikipedia.fr`. Le **DNS** (*Domain Name System*) est l'annuaire d'Internet : il traduit une adresse symbolique en adresse IP.

Les noms sont **hiérarchiques** et se lisent de droite à gauche : dans `fr.wikipedia.org`, `org` est le domaine de premier niveau, `wikipedia` le domaine, `fr` un sous-domaine. Le DNS est réparti sur un très grand nombre de serveurs dans le monde, constamment mis à jour ; chacun connaît une partie de l'annuaire.

Méthode – Retrouver une adresse IP, et inversement

La commande `nslookup` interroge un serveur DNS. Exemples relevés le 1er novembre 2026 :

```
> nslookup wikipedia.fr
```

```
Name:    wikipedia.fr
Address: 141.94.212.184

> nslookup 8.8.8.8
8.8.8.8.in-addr.arpa   name = dns.google.
```

Dans le premier cas, on obtient l'adresse IP à partir du nom ; dans le second, le nom à partir de l'adresse IP (recherche **inverse**). Une même adresse symbolique peut correspondre à plusieurs adresses IP, et ces adresses peuvent changer : il faut refaire la recherche soi-même.

Propriété – Ce qui se passe quand on tape une adresse

1. L'ordinateur demande au serveur DNS l'adresse IP de `utspe.com`.
2. Le serveur DNS répond (en interrogeant d'autres serveurs DNS si besoin).
3. L'ordinateur envoie ses paquets à cette adresse IP ; ils sont routés jusqu'au serveur.
4. Le serveur répond par d'autres paquets, que TCP remet dans l'ordre à l'arrivée.

2.6 Réseaux physiques et trafic d'Internet

Propriété – Indépendance d'Internet par rapport au réseau physique

Les protocoles d'Internet étant **logiciels**, Internet peut utiliser n'importe quel réseau physique qui les met en œuvre. Un même message peut passer par le Wi-Fi, un câble Ethernet, une fibre optique sous-marine puis la 4G, sans que l'utilisateur s'en aperçoive.

Réseau	**Support**	**Débit (ordre de grandeur)**	**Statut**
Modem téléphonique (RTC)	filaire (cuivre)	56 kbit/s	obsolète
ADSL	filaire (cuivre)	1 à 20 Mbit/s	en voie de disparition
Fibre optique	filaire (lumière)	1 à 8 Gbit/s	actuel
Ethernet	filaire (câble réseau)	100 Mbit/s à 10 Gbit/s	actuel
Wi-Fi	sans fil (ondes radio)	100 Mbit/s à quelques Gbit/s	actuel
4G / 5G	sans fil (antennes relais)	50 Mbit/s / jusqu'à 1 Gbit/s	actuel
Bluetooth	sans fil (courte portée)	1 à 3 Mbit/s	actuel

Attention

Les débits s'expriment en **bits** par seconde, alors que la taille des fichiers s'exprime en **octets**, avec 1 octet = 8 bits. Un débit de 100 Mbit/s transfère donc 12,5 Mo par seconde.

Définition – Unités de quantité de données

1 ko = 10^3 octets ; 1 Mo = 10^6 ; 1 Go = 10^9 ; 1 To = 10^{12} ; 1 Po = 10^{15} ; 1 Eo = 10^{18} ; 1 Zo (zettaoctet) = 10^{21} octets.

Le trafic mondial d'Internet prévu pour 2021 était d'environ $\mathbf{3{,}3 \times 10^{21}}$ **octets** par an (3,3 Zo), et il continue d'augmenter chaque année, porté surtout par la vidéo en streaming.

2.7 Réseaux pair-à-pair et enjeux

Définition – Client-serveur et pair-à-pair

— Dans le modèle **client-serveur**, un serveur central fournit les données à tous les clients (un site web, une plateforme de vidéo).
— Dans un réseau **pair-à-pair** (*peer-to-peer*, P2P), chaque ordinateur est à la fois **client et serveur** : il reçoit des morceaux de fichier et en envoie aux autres.

Propriété – Intérêts et usages illicites du pair-à-pair

— **Intérêts** : pas de serveur central à financer ni à saturer, plus on est nombreux, plus la diffusion est rapide, le réseau résiste à la panne d'un participant. Usages légaux : distribution de logiciels libres (Linux), mises à jour de jeux, diffusion de vidéos.
— **Usages illicites** : partage d'œuvres protégées par le droit d'auteur (films, musiques, jeux) sans autorisation. En France, l'Arcom (ex-Hadopi) lutte contre ce piratage ; télécharger ou partager ces fichiers est illégal.

Impacts sur les pratiques humaines

— Internet a fait disparaître le télégramme et le télex, remplacé une grande partie du courrier postal et remplace le téléphone fixe par la **voix sur IP** (VoIP).
— **Déni de service** : une attaque peut saturer un site en lui envoyant un très grand nombre de messages.
— **Neutralité du Net** : les routeurs doivent transmettre tous les paquets de la même façon, quel que soit leur contenu (texte, vidéo…). Ce principe, présent dès l'origine, est régulièrement remis en cause.

2.8 Ateliers Python

Les ateliers se font sur ordinateur, dans un éditeur Python ou directement sur https://utspe.com/snt/chapitre-2. Pour chacun : recopier le programme de départ, suivre les étapes dans l'ordre et exécuter le programme après chaque modification.

Atelier 1 – Découper un message en paquets chaînes, boucle for, math.ceil

Objectif : Simuler le découpage d'un message en paquets numérotés, comme le fait TCP.
1. Exécuter le programme : il calcule le nombre de paquets nécessaires pour un message.
2. `message[debut:debut + 10]` donne les 10 caractères à partir de la position `debut`. Compléter la boucle pour afficher chaque paquet sous la forme `Paquet 0 : Internet e`.
3. Changer `taille_paquet` (5, puis 20) et vérifier que le nombre de paquets affichés correspond au calcul.

Programme de départ :

```
import math

message = "Internet est un reseau de reseaux."
taille_paquet = 10

nb = math.ceil(len(message) / taille_paquet)
print(f"{len(message)} caracteres -> {nb} paquets")

numero = 0
for debut in range(0, len(message), taille_paquet):
    morceau = ""          # a completer
```

```
    # a completer : afficher le paquet avec une f-string
    numero = numero + 1
```

Coup de pouce : `range(0, len(message), taille_paquet)` donne 0, 10, 20, 30 : la position de début de chaque paquet. On écrit `morceau = message[debut:debut + taille_paquet]`.

Corrigé rédigé

```
import math

message = "Internet est un reseau de reseaux."
taille_paquet = 10

nb = math.ceil(len(message) / taille_paquet)
print(f"{len(message)} caracteres -> {nb} paquets")

numero = 0
for debut in range(0, len(message), taille_paquet):
    morceau = message[debut:debut + taille_paquet]
    print(f"Paquet {numero} : {morceau}")
    numero = numero + 1
```

Atelier 2 – Vérifier une adresse IPv4

fonctions, conditions, split()

Objectif : Écrire une fonction qui dit si une adresse IPv4 est valide : 4 nombres, chacun entre 0 et 255.

1. `adresse.split(".")` découpe la chaîne à chaque point et donne la liste des morceaux. Exécuter le programme pour l'observer.
2. Compléter `octet_valide(n)` : elle renvoie `True` si `0 <= n <= 255`.
3. Compléter `ip_valide(adresse)` : il faut exactement 4 morceaux, et chacun doit être un octet valide.
4. Tester avec `192.168.1.15`, `10.0.300.2`, `172.16.5` et `255.255.255.255`.

Programme de départ :

```
def octet_valide(n):
    return True           # a completer

def ip_valide(adresse):
    morceaux = adresse.split(".")
    print(morceaux)
    # a completer
    return True

print(ip_valide("192.168.1.15"))
```

Coup de pouce : Si `len(morceaux) != 4`, on renvoie `False`. Sinon, on parcourt les morceaux avec `for m in morceaux:` et on renvoie `False` dès que `octet_valide(int(m))` est faux ; après la boucle, on renvoie `True`.

Corrigé rédigé

```
def octet_valide(n):
    return 0 <= n <= 255

```

```python
def ip_valide(adresse):
    morceaux = adresse.split(".")
    if len(morceaux) != 4:
        return False
    for m in morceaux:
        if not octet_valide(int(m)):
            return False
    return True

for ip in ["192.168.1.15", "10.0.300.2", "172.16.5", "255.255.255.255"]:
    print(f"{ip} : {ip_valide(ip)}")
```

Atelier 3 – La durée de vie d'un paquet (TTL)

boucle while, conditions

Objectif : Simuler un paquet qui traverse des routeurs : à chaque routeur, son TTL diminue de 1 ; à 0, il est détruit.

1. Exécuter le programme : le paquet doit traverser 8 routeurs avec un TTL de 64.
2. Remplacer le TTL par 5 : que se passe-t-il ? Compléter la fin du programme pour afficher « Paquet arrivé » ou « Paquet détruit ».
3. Quel est le plus petit TTL qui permet au paquet d'arriver ?

Programme de départ :

```python
ttl = 64
routeurs = 8          # routeurs a traverser
traverses = 0

while traverses < routeurs and ttl > 0:
    ttl = ttl - 1
    traverses = traverses + 1
    print(f"Routeur {traverses} : TTL = {ttl}")

# a completer : paquet arrive ou detruit ?
```

Coup de pouce : Après la boucle, le paquet est arrivé si `traverses == routeurs`. Pour la dernière question, essayer des valeurs de `ttl` autour de 8.

Corrigé rédigé

```python
ttl = 5
routeurs = 8          # routeurs a traverser
traverses = 0

while traverses < routeurs and ttl > 0:
    ttl = ttl - 1
    traverses = traverses + 1
    print(f"Routeur {traverses} : TTL = {ttl}")

if traverses == routeurs:
    print("Paquet arrive")
else:
    print(f"Paquet detruit apres {traverses} routeurs")
# Il faut un TTL d'au moins 8 pour traverser 8 routeurs.
```

Atelier 4 – Un mini-annuaire DNS

dictionnaire, fonctions, boucle for

Objectif : Programmer un petit annuaire qui traduit un nom en adresse IP, et inversement.

1. Un **dictionnaire** associe des clés à des valeurs : `annuaire["wikipedia.fr"]` donne l'adresse IP de `wikipedia.fr`. Exécuter le programme.
2. Compléter `resoudre(nom)` : elle renvoie l'adresse IP, ou `"inconnu"` si le nom n'est pas dans l'annuaire (tester avec `nom in annuaire`).
3. Compléter `inverse(ip)` : elle parcourt l'annuaire et renvoie le nom qui correspond à l'adresse IP.
4. Ajouter un nom de votre choix, après avoir trouvé son adresse avec `nslookup`.

Programme de départ :

```
annuaire = {
    "wikipedia.fr": "141.94.212.184",
    "dns.google": "8.8.8.8",
    "utspe.com": "217.160.191.3",
}

def resoudre(nom):
    return annuaire[nom]       # a ameliorer

def inverse(ip):
    return "inconnu"           # a completer

print(resoudre("wikipedia.fr"))
```

Coup de pouce : Dans `resoudre` : `if nom in annuaire:`. Dans `inverse` : `for nom in annuaire:` puis `if annuaire[nom] == ip:` on renvoie `nom`.

Corrigé rédigé

```
annuaire = {
    "wikipedia.fr": "141.94.212.184",
    "dns.google": "8.8.8.8",
    "utspe.com": "217.160.191.3",
}

def resoudre(nom):
    if nom in annuaire:
        return annuaire[nom]
    return "inconnu"

def inverse(ip):
    for nom in annuaire:
        if annuaire[nom] == ip:
            return nom
    return "inconnu"

print(resoudre("wikipedia.fr"))
print(resoudre("exemple.org"))
print(inverse("8.8.8.8"))
```

Atelier 5 – Mini-projet : TCP remet les paquets dans l'ordre listes, boucles imbriquées, conditions

Objectif : Les paquets arrivent dans le désordre et l'un d'eux est perdu. Reconstituer le message et détecter le paquet à redemander.

1. Chaque paquet reçu est un couple `(numero, texte)`. Exécuter le programme pour afficher les paquets reçus.
2. Pour chaque numéro de 0 à `nb_paquets - 1`, chercher le paquet qui porte ce numéro et ajouter son texte au message.
3. Si aucun paquet ne porte ce numéro, afficher « Paquet k manquant : à redemander ».

Programme de départ :

```
recus = [(2, "eau de "), (0, "Inter"), (4, "x."), (1, "net est un res")]
nb_paquets = 5

for paquet in recus:
    print(paquet)

message = ""
# a completer
```

Coup de pouce : Écrire `for k in range(nb_paquets):`, puis une variable `trouve = False` ; dans une deuxième boucle `for numero, texte in recus:`, si `numero == k`, ajouter `texte` au message et passer `trouve` à `True`.

Corrigé rédigé

```
recus = [(2, "eau de "), (0, "Inter"), (4, "x."), (1, "net est un res")]
nb_paquets = 5

message = ""
for k in range(nb_paquets):
    trouve = False
    for numero, texte in recus:
        if numero == k:
            message = message + texte
            trouve = True
    if not trouve:
        print(f"Paquet {k} manquant : a redemander")
        message = message + "[?]"
print(message)
```

2.9 Exercices corrigés

Exercice

Exercice 1 – Adresses IPv4. Parmi les adresses suivantes, lesquelles sont des adresses IPv4 valides ? Justifier.
`192.168.0.1` ; `10.0.300.2` ; `172.16.5` ; `255.255.255.255` ; `8.8.8.8.8`.

Corrigé rédigé

`192.168.0.1`, `255.255.255.255` : valides (4 nombres entre 0 et 255). `10.0.300.2` : non, $300 > 255$. `172.16.5` : non, seulement 3 nombres. `8.8.8.8.8` : non, 5 nombres.

Exercice

Exercice 2 – Combien d'adresses ? 1. Une adresse IPv4 est codée sur 32 bits. Combien d'adresses différentes existe-t-il ? Comparer au nombre d'humains (environ 8 milliards).
2. Même question pour IPv6 (128 bits). Pourquoi a-t-on créé IPv6 ?

Corrigé rédigé

1. $2^{32} = 4\ 294\ 967\ 296 \approx 4{,}3$ milliards d'adresses : moins d'une par habitant, alors que chacun possède plusieurs appareils connectés.
2. $2^{128} \approx 3{,}4 \times 10^{38}$ adresses : IPv6 a été créé parce que les adresses IPv4 sont épuisées.

```
print(2 ** 32)
print(f"{2 ** 128:.2e}")
```

Exercice

Exercice 3 – Nombre de paquets. Une photo de 3 Mo (3 000 000 octets) est envoyée par Internet. Chaque paquet transporte 1 500 octets de données et un en-tête de 40 octets.
1. Combien de paquets faut-il ?
2. Quelle quantité totale d'octets est réellement transmise ?

Corrigé rédigé

1. $3\ 000\ 000 \div 1\ 500 =$ **2 000 paquets** (la division tombe juste).
2. $2\ 000 \times (1\ 500 + 40) =$ **3 080 000 octets**, soit environ 2,7 % de plus que la photo.

```
import math

nb = math.ceil(3000000 / 1500)
print(nb, "paquets")
print(nb * (1500 + 40), "octets transmis")
```

Exercice

Exercice 4 – Routage. On reprend le réseau de la partie 3 (routeurs R1 à R6, ordinateur A relié à R1, serveur B relié à R6).
1. Donner tous les chemins de R1 à R6 qui passent par 3 liaisons.
2. Le routeur R4 tombe en panne. Par quel chemin les paquets peuvent-ils encore passer ?
3. Compléter la table de routage de R3 : pour chaque destination R1, R2, R4, R5, R6, indiquer le voisin auquel R3 envoie le paquet.

Corrigé rédigé

1. R1 → R2 → R4 → R6 ; R1 → R3 → R4 → R6 ; R1 → R3 → R5 → R6.
2. Il reste R1 → R3 → R5 → R6 : les routeurs mettent à jour leurs tables et contournent la panne.
3. Table de R3 : R1 → R1 ; R2 → R1 (ou R4) ; R4 → R4 ; R5 → R5 ; R6 → R4 ou R5 (les deux chemins ont 2 liaisons).

Exercice

Exercice 5 – IP ou TCP ? Pour chaque affirmation, dire si elle concerne le protocole IP ou le protocole TCP.

a) Il donne une adresse à chaque machine.
b) Il redemande les paquets perdus.
c) Il achemine chaque paquet vers son destinataire, sans garantie.
d) Il remet les paquets dans l'ordre à l'arrivée.
e) Il garantit que la vidéo arrive sans saccade.

Corrigé rédigé

a) IP ; b) TCP ; c) IP ; d) TCP ; e) **aucun des deux** : ni IP ni TCP n'offrent de garantie temporelle.

Exercice

Exercice 6 – Télécharger un film. Un film fait 4 Go. Rappel : 1 octet = 8 bits.
1. Convertir 4 Go en mégabits (Mbit).
2. Calculer la durée de téléchargement avec l'ADSL (8 Mbit/s), la 4G (50 Mbit/s) et la fibre (1 Gbit/s = 1 000 Mbit/s).
3. Écrire une fonction `duree(taille_go, debit_mbits)` qui renvoie la durée en secondes, et afficher les trois durées en minutes et secondes.

Corrigé rédigé

1. 4 Go = 4 000 Mo = 4 000 × 8 = **32 000 Mbit**.
2. ADSL : 32 000 ÷ 8 = 4 000 s = 66 min 40 s (plus d'une heure) ; 4G : 32 000 ÷ 50 = 640 s = 10 min 40 s ; fibre : 32 000 ÷ 1 000 = 32 s.

```
def duree(taille_go, debit_mbits):
    return taille_go * 1000 * 8 / debit_mbits

for nom, debit in [("ADSL", 8), ("4G", 50), ("Fibre", 1000)]:
    d = int(duree(4, debit))
    print(f"{nom} : {d // 60} min {d % 60} s")
```

Exercice

Exercice 7 – Le trafic d'Internet. Le trafic d'Internet prévu pour 2021 était de $3{,}3 \times 10^{21}$ octets.
1. Combien d'octets cela représente-t-il par seconde ? Exprimer le résultat en To/s.
2. Avec environ 8 milliards d'habitants, combien de gigaoctets par habitant et par an ?

Corrigé rédigé

1. Une année compte $365 \times 24 \times 3\,600 = 31\,536\,000$ s. $3{,}3 \times 10^{21} \div 31\,536\,000 \approx 1{,}05 \times 10^{14}$ octets par seconde, soit environ **105 To/s**.
2. $3{,}3 \times 10^{21} \div (8 \times 10^{9}) \approx 4{,}1 \times 10^{11}$ octets, soit environ **412 Go** par habitant et par an.

```
trafic = 3.3e21
secondes = 365 * 24 * 3600
print(f"{trafic / secondes / 1e12:.0f} To/s")
print(f"{trafic / 8e9 / 1e9:.0f} Go par habitant")
```

Exercice

Exercice 8 – Pair-à-pair. Une distribution Linux (fichier de 2 Go) est téléchargée par 10 000 personnes le jour de sa sortie.
1. Pourquoi un serveur unique risque-t-il d'être saturé ?
2. Expliquer comment un réseau pair-à-pair résout ce problème.
3. Donner un usage légal et un usage illicite des réseaux pair-à-pair.

Corrigé rédigé

1. Le serveur devrait envoyer 10 000 × 2 Go = 20 To à lui seul : sa connexion serait saturée.
2. En pair-à-pair, chaque personne qui a reçu des morceaux du fichier les renvoie aux autres : plus il y a de participants, plus il y a de sources.
3. Légal : distribuer un logiciel libre ou des mises à jour. Illicite : partager un film ou de la musique protégés par le droit d'auteur sans autorisation.

2.10 QCM – Teste tes connaissances

Pour chaque question, une seule réponse est correcte.

Q1. Internet est…

a) un logiciel de navigation
b) un réseau de réseaux
c) un site web
d) un câble sous-marin

Q2. Laquelle est une adresse IPv4 valide ?

a) 192.168.1.300
b) 10.0.0
c) 172.16.254.1
d) 256.1.1.1

Q3. Le rôle principal du protocole IP est…

a) d'acheminer les paquets vers leur destinataire
b) de redemander les paquets perdus
c) d'afficher les pages web
d) de chiffrer les données

Q4. Que fait TCP quand un paquet est perdu ?

a) rien
b) il le redemande
c) il arrête la communication
d) il envoie un message d'erreur à l'utilisateur

Q5. Pourquoi une visioconférence peut-elle saccader ?

a) TCP perd définitivement les paquets
b) Internet n'a pas de garantie temporelle
c) le DNS est trop lent
d) les paquets sont trop gros

Q6. Un routeur connaît…

a) la carte complète d'Internet
b) une carte locale : ses voisins et vers qui envoyer
c) uniquement l'adresse de l'émetteur
d) le contenu des paquets

Q7. Un paquet a un TTL de 3. Que se passe-t-il s'il doit traverser 5 routeurs ?

a) il arrive en retard
b) il est détruit en route
c) il arrive normalement
d) il repart à l'émetteur

Q8. À quoi sert le DNS ?

a) à traduire un nom de domaine en adresse IP
b) à router les paquets
c) à accélérer la connexion
d) à stocker les pages web

Q9. Quelle commande affiche les routeurs traversés jusqu'à un site ?

a) ping
b) nslookup
c) traceroute (tracert)
d) ipconfig

Q10. Un débit de 80 Mbit/s permet de transférer par seconde…

a) 80 Mo
b) 10 Mo
c) 640 Mo
d) 8 Mo

Q11. Dans un réseau pair-à-pair…

a) un serveur central fournit tous les fichiers
b) chaque ordinateur est à la fois client et serveur
c) les échanges sont toujours illégaux
d) il n'y a pas besoin d'Internet

Q12. Qu'affiche ce programme ?

```
1 import math
2 print(math.ceil(3200 / 1500))
```

a) 2
b) 2.13
c) 3
d) 1500

Corrigé rédigé

Réponses du QCM.
Q1. **b)** Internet relie entre eux des millions de réseaux ; le Web n'est qu'un service qui l'utilise.
Q2. **c)** 4 nombres, chacun entre 0 et 255.
Q3. **a)** IP adresse et achemine les paquets, sans garantie ; c'est TCP qui fiabilise.
Q4. **b)** Sans accusé de réception, TCP redemande le paquet manquant.
Q5. **b)** Les paquets finissent par arriver, mais on ne sait pas quand.

Q6. **b)** Chaque routeur a une table de routage locale, mise à jour avec ses voisins.
Q7. **b)** Le TTL diminue de 1 à chaque routeur ; à 0, le paquet est détruit.
Q8. **a)** Le DNS est l'annuaire d'Internet.
Q9. **c)** `traceroute` (ou `tracert` sous Windows) liste les routeurs traversés.
Q10. **b)** $80 \div 8 = 10$ Mo par seconde (1 octet = 8 bits).
Q11. **b)** Chacun reçoit et renvoie des morceaux de fichiers.
Q12. **c)** $3200 \div 1500 \approx 2{,}13$ et `math.ceil` arrondit à l'entier supérieur : 3 paquets.

2.11 Fiche de synthèse

Fiche de synthèse

— **Adresses et paquets :** Adresse IP : IPv4 = 4 nombres de 0 à 255 (32 bits), IPv6 = 128 bits. Les données sont découpées en paquets (en-tête + données).

— **IP et TCP :** IP achemine les paquets sans garantie. TCP numérote, accuse réception, redemande les paquets perdus et les remet dans l'ordre. Aucune garantie temporelle.

— **Routage :** Les routeurs transmettent les paquets de proche en proche grâce à leur table de routage locale. Le TTL limite le nombre de routeurs traversés.

— **DNS :** Le DNS traduit une adresse symbolique en adresse IP (et inversement) ; commande `nslookup`.

— **Réseaux physiques :** Internet est logiciel : il fonctionne sur la fibre, l'ADSL, le Wi-Fi, la 4G… Débits en bit/s, tailles en octets : 1 octet = 8 bits.

— **Pair-à-pair :** Chaque machine est client et serveur : diffusion rapide et robuste, mais usages illicites possibles (œuvres protégées).