MINISTÈRE
DE L'ÉDUCATION
NATIONALE ET
DE LA JEUNESSE

Liberté • Égalité • Fraternité
RÉPUBLIQUE FRANÇAISE

EAE SIN 2

SESSION 2020

AGREGATION
CONCOURS EXTERNE

Section : SCIENCES INDUSTRIELLES DE L'INGÉNIEUR

Option : SCIENCES INDUSTRIELLES DE L'INGÉNIEUR ET INGÉNIERIE INFORMATIQUE

MODÉLISATION D'UN SYSTÈME, D'UN PROCÉDÉ OU D'UNE ORGANISATION

Durée : 6 heures

Calculatrice électronique de poche - y compris calculatrice programmable, alphanumérique ou à écran graphique – à fonctionnement autonome, non imprimante, autorisée conformément à la circulaire n° 99-186 du 16 novembre 1999.

L'usage de tout ouvrage de référence, de tout dictionnaire et de tout autre matériel électronique est rigoureusement interdit.

Si vous repérez ce qui vous semble être une erreur d'énoncé, vous devez le signaler très lisiblement sur votre copie, en proposer la correction et poursuivre l'épreuve en conséquence. De même, si cela vous conduit à formuler une ou plusieurs hypothèses, vous devez la (ou les) mentionner explicitement.

NB : Conformément au principe d'anonymat, votre copie ne doit comporter aucun signe distinctif, tel que nom, signature, origine, etc. Si le travail qui vous est demandé consiste notamment en la rédaction d'un projet ou d'une note, vous devrez impérativement vous abstenir de la signer ou de l'identifier.

Tournez la page S.V.P.

Ce document est composé :

- d'un dossier sujet de la page 2 à la page 24 ;
- d'un dossier technique de la page 25 à la page 27.

Le sujet comporte une introduction au système ainsi que cinq parties de questionnement.

Conseils aux candidats

Les différentes parties du sujet sont indépendantes à l'exception de la partie 5 qui dépend de la partie 4.

Un parcours attentif de l'ensemble du document est conseillé avant de composer.

La présentation des programmes doit respecter les mots clés du langage cible ainsi que l'indentation des structures algorithmiques.

Les réponses doivent être présentées avec clarté, rigueur et concision.

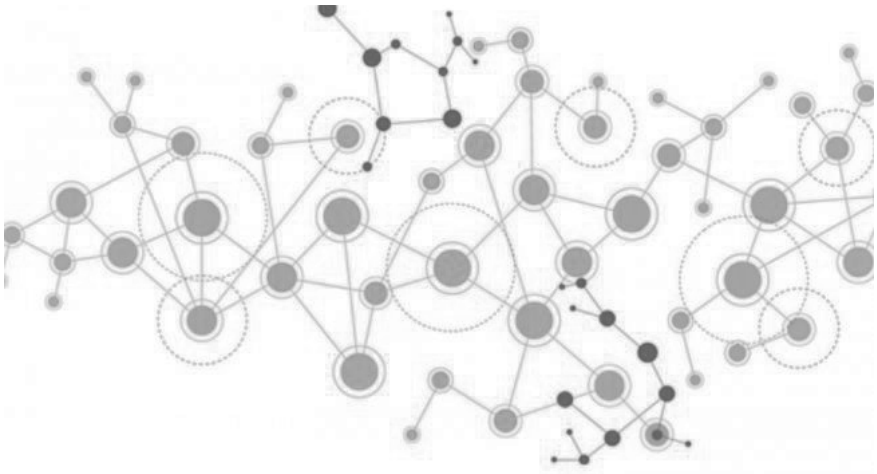
INFORMATION AUX CANDIDATS

Vous trouverez ci-après les codes nécessaires vous permettant de compléter les rubriques figurant en en-tête de votre copie.

Ces codes doivent être reportés sur chacune des copies que vous remettrez.

Concours	Section/option	Epreuve	Matière
EAE	1417A	102	2680

Réseau de diffusion de contenu vidéo



Présentation du système

En 2017, le trafic Internet entrant sur le réseau des quatre principaux fournisseurs d'accès à Internet en France a augmenté de 44 % par rapport à l'année précédente. Cette demande très fortement croissante du trafic conduit à une augmentation proportionnelle des capacités des réseaux pour répondre aux besoins des clients. Le trafic étant variable au cours du temps, les réseaux doivent être dimensionnés pour traiter les pics de charge. La *figure 1* montre un exemple de cette variabilité sur le trafic d'un point d'échange Internet (IXP) où différents fournisseurs d'accès Internet échangent du trafic entre leurs réseaux. Sur cet exemple, il y a un facteur 2,5 entre le trafic le plus bas et celui le plus élevé sur une journée.

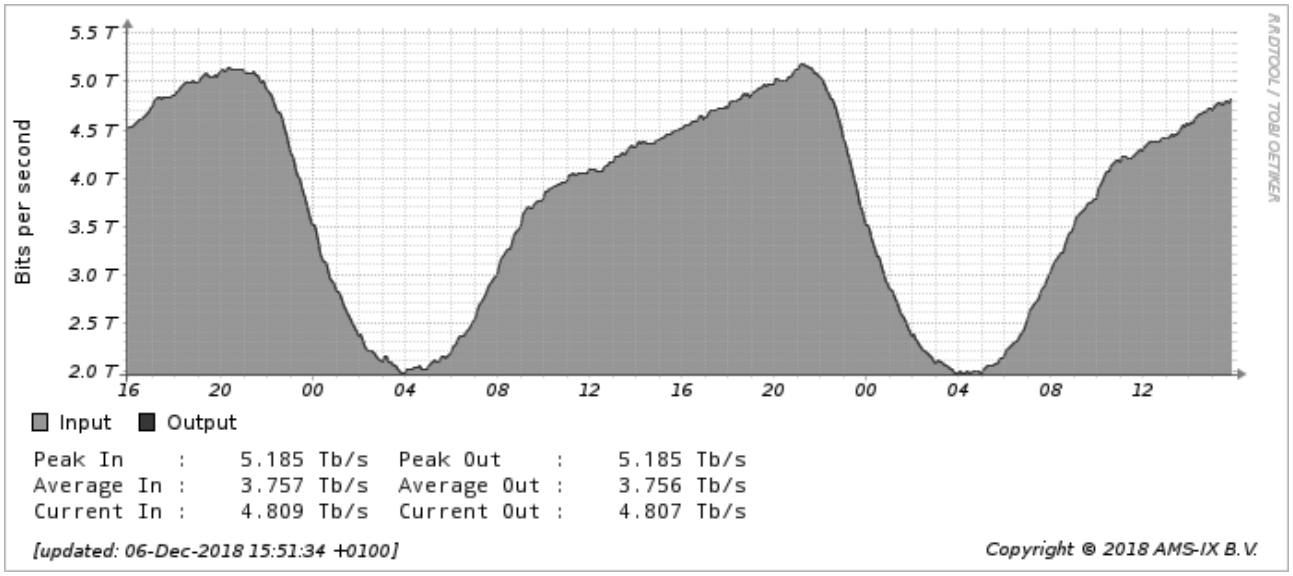


Figure 1 : Trafic agrégé sur l'ensemble des ports réseaux du point d'échange Internet d'Amsterdam (AMS-IX)

Fin 2017, l'Arcep (Autorité de régulation des communications électroniques et des postes) estimait qu'en France, environ 50 % du trafic provenait des quatre principaux fournisseurs de contenu. Cette répartition du trafic est illustrée par la *figure 2*.

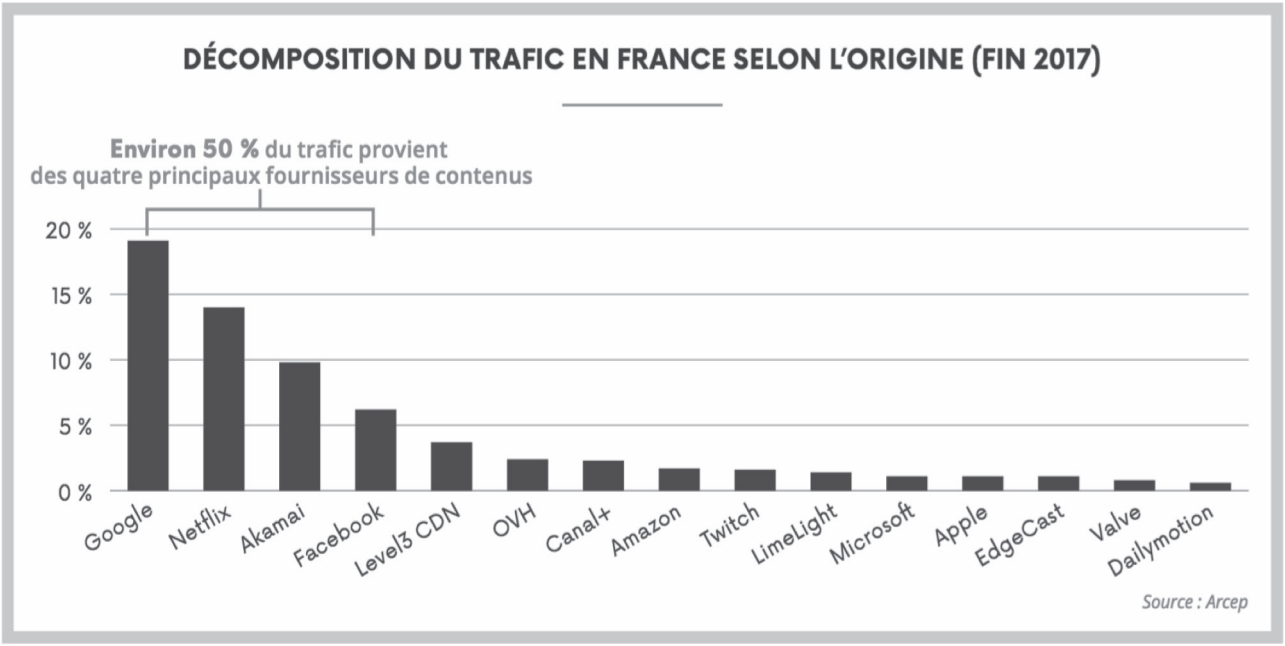


Figure 2 : Trafic Internet en France selon l'Arcep en 2017

L'évolution de l'écosystème de l'Internet a favorisé l'apparition d'une multitude d'acteurs qui interagissent pour permettre aux utilisateurs d'accéder aux contenus numériques et aux applications. L'Arcep définit les principaux acteurs suivants :

- les fournisseurs de contenu et d'application (FCA) : les propriétaires du contenu, qui font appel à plusieurs intermédiaires pour acheminer leur contenu aux utilisateurs finals ;
- les hébergeurs : les propriétaires des serveurs hébergeant un contenu géré par des tiers (FCA ou individus) ;
- les transitaires : les gestionnaires des réseaux internationaux qui font office d'intermédiaires entre les FCA et les FAI pour acheminer le trafic ;
- les points d'échange Internet (IXP - *Internet Exchange Point*) : les infrastructures qui permettent aux différents acteurs de s'interconnecter directement, via un point d'échange, plutôt que par le biais d'un ou de plusieurs transitaires ;
- les réseaux de diffusion de contenu (CDN - *Content Delivery Network*) : les réseaux se spécialisent dans la livraison de volumes de trafic importants vers plusieurs FAI, dans des zones géographiques variées et grâce à des serveurs de cache au plus proche des clients finals ;
- les fournisseurs d'accès Internet (FAI) : les opérateurs de réseaux qui sont chargés de livrer le trafic au client final ;
- les clients finals : les individus qui utilisent leurs propres équipements et contractent un abonnement auprès d'un FAI pour accéder à du contenu sur Internet.

Les réseaux de diffusion de contenu (CDN) permettent de réduire l'utilisation des réseaux en rapprochant les contenus des utilisateurs. Traditionnellement, la distribution de contenu sur Internet s'effectuait depuis un ou plusieurs serveurs sources localisés dans un centre de calcul. Dans un CDN, les serveurs sont distribués géographiquement à travers Internet et contiennent des contenus répliqués pour permettre aux clients d'y avoir accès en répartissant mieux la charge sur le réseau. Ce changement d'architecture est illustré par la *figure 3*.

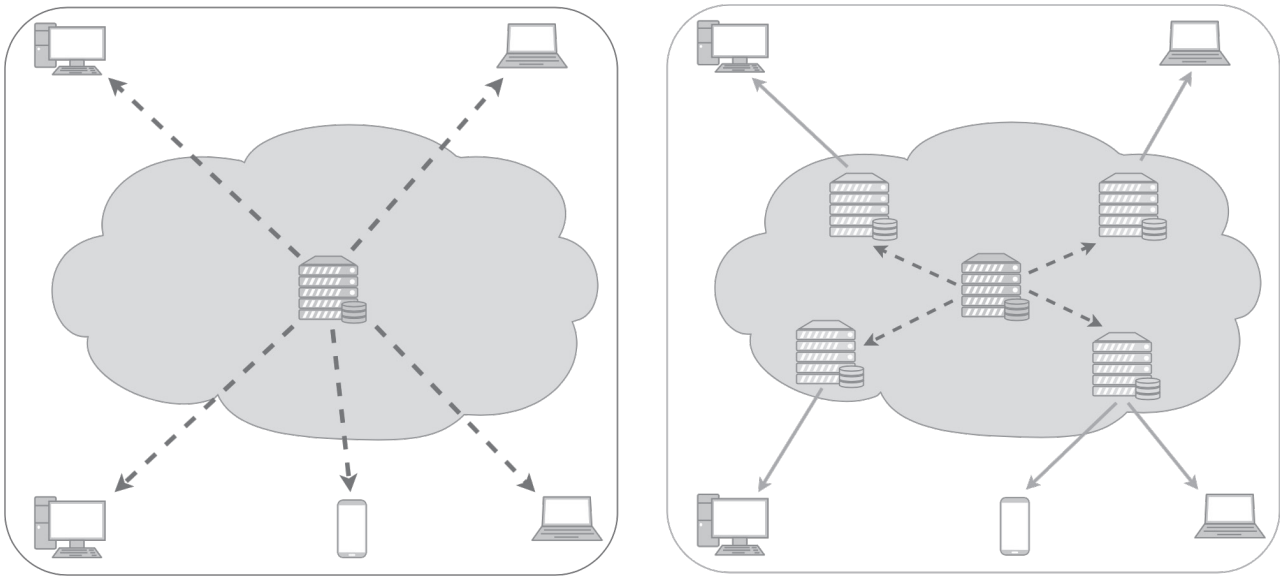


Figure 3 : Architecture traditionnelle de distribution de contenu avec un serveur origine (à gauche) et architecture avec un CDN (à droite)

Un CDN est donc constitué d'un ensemble de serveurs connectés à Internet qui se répartissent les demandes des clients pour accéder à des contenus numériques. Généralement, le serveur qui répond à une requête d'un client est celui situé au plus près du client et possédant le contenu demandé. Cette architecture permet ainsi aux clients d'avoir des latences plus faibles qu'avec une architecture classique impliquant un centre de calcul centralisé regroupant les contenus. De plus, cette architecture permet une réduction du trafic dans le cœur du réseau, réduisant ainsi la demande pour des tailles de réseau plus conséquentes. En effet, le trafic depuis le serveur origine est réduit avec un CDN par rapport à une architecture traditionnelle comme illustré dans la *figure 3*, puisque des serveurs localisés dans des endroits différents répondent aux requêtes des clients. Enfin, cette architecture permet une meilleure répartition de la charge.

Trois types d'acteurs doivent être considérés dans l'architecture d'un CDN : les fournisseurs de contenu et d'applications (FCA) qui sont les clients effectifs du CDN et payent pour pouvoir stocker des contenus dans le CDN, les clients finals qui souhaitent avoir accès aux contenus numériques et le CDN qui stocke les contenus confiés par les fournisseurs et les rend accessibles aux consommateurs. Les fournisseurs de contenus assurent la durabilité du contenu, tandis que les serveurs CDN offrent la disponibilité du contenu en se focalisant sur les performances de diffusion. Il est à noter que certains fournisseurs de contenu opèrent leur propre CDN et assurent ainsi les deux fonctions simultanément (durabilité et disponibilité).

Trois types de contenus peuvent être diffusés par un CDN :

- contenus statiques : des données qui sont stockées une fois chez les fournisseurs de contenu en lecture seule, puis qui sont distribuées. Cette catégorie comprend par exemple les photos, les livres électroniques, ainsi que certains services de partage de fichiers ;
- services interactifs : cette catégorie est principalement liée aux services Web qui permettent aux utilisateurs de conserver leurs propres données ou de gérer des transactions en ligne sur Internet. Par exemple, cela inclut les services de sauvegarde et de synchronisation, les services d'annuaire et le commerce électronique ;
- streaming : les services de streaming utilisent de plus en plus de CDN. Il existe une demande croissante pour ce type de contenu avec un niveau d'attentes en matière de qualité de service en augmentation constante, notamment en termes de temps de réponse et de débit. C'est principalement le cas de la diffusion vidéo, notamment de la vidéo à la demande et de la diffusion en direct.

Le dernier type de contenu, le streaming vidéo, représente une part du trafic en forte croissance et nécessite une gestion fine de la bande passante. Des variations de bande passante disponible induisent des problèmes de qualité à la lecture de la vidéo (résolution vidéo, coupures, temps de chargement avant lecture, etc.).

Les serveurs d'un CDN constituent un stockage distribué de contenu. Suivant l'architecture du CDN, ces serveurs sont répartis sur environ une dizaine de sites (comme c'est le cas par exemple pour Google et Akamai) ou encore de l'ordre de la centaine de sites (pour Netflix par exemple). Un CDN doit relever plusieurs défis :

- il doit faire face à des variations brusques et inattendues de la popularité du contenu qu'il rend accessible aux clients ;
- il doit éviter de gaspiller des ressources matérielles et réduire autant que possible le stockage et l'utilisation du réseau ;
- il doit offrir un faible temps de réponse et une disponibilité de bande passante élevée en optimisant son architecture et en employant des techniques auto-adaptatives.

L'objectif de ce sujet est la modélisation d'un CDN pour du streaming vidéo. Il s'agit d'évaluer sa consommation électrique en phase usage.

Partie 1. Modélisation de la connexion des utilisateurs

Objectifs : Analyser le fonctionnement d'un CDN, déterminer ses caractéristiques.

La *figure 4* représente une séquence simplifiée des paquets échangés par un client souhaitant accéder aux contenus numériques A et B. Il envoie d'abord une requête à un serveur DNS (*Domain Name Service*) pour obtenir l'adresse du serveur possédant les contenus désirés. Il envoie ensuite une requête au serveur CDN pour obtenir le contenu A, qui lui est renvoyé par le serveur CDN. Enfin, il envoie une requête pour obtenir le contenu B. Le serveur CDN ne possédant pas ce contenu, il contacte le serveur FCA pour l'obtenir puis lorsqu'il l'a obtenu, il l'envoie au client. Le RTT (*Round Trip Time*) mesuré entre le client et le serveur DNS est de 35 ms. Le RTT entre le client et le serveur CDN est de 45 ms et le RTT entre le serveur CDN et le serveur FCA est de 100 ms.

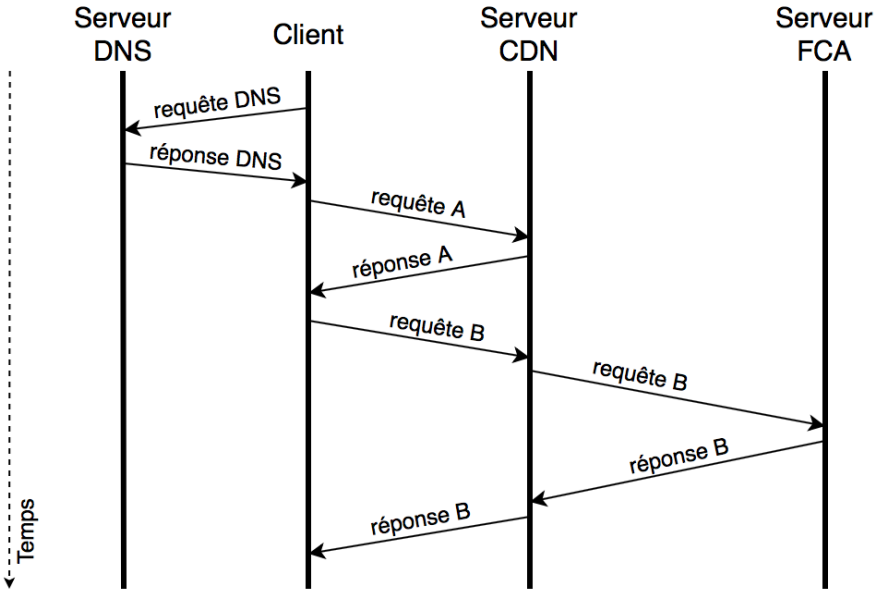


Figure 4 : Séquence simplifiée d'échange de paquets pour 2 requêtes client de demande de contenu

Q1. En supposant que chaque requête et chaque réponse ne requiert qu'un paquet et que le temps de traitement de chaque serveur est négligeable, quelle est la durée totale des échanges représentés dans la *figure 4* ?

Q2. Après la première requête DNS, est-il nécessaire de faire d'autres requêtes DNS pour accéder au contenu fourni par le même FCA ? Justifier.

Q3. Expliquer pourquoi la requête B nécessite une communication avec le serveur FCA.

Q4. Quand le CDN communique-t-il avec le DNS ?

La transmission d'un contenu d'un serveur CDN à un client emprunte les réseaux déployés par des FAI. Supposons que ces réseaux utilisent le protocole Ethernet et suivent donc le principe de commutation de paquets. Une fois la communication établie entre le serveur CDN et le client, le serveur doit envoyer un fichier (une vidéo par exemple) de *N* octets au client. Ce transfert de fichier s'effectue selon les conditions suivantes :

- une communication entre le serveur CDN et le client implique de traverser *k* commutateurs ;
- toutes les liaisons de données utilisées emploient le même protocole de liaison qui ajoute un en-tête de *H* bits pour *P* bits de données utiles dans un paquet ;
- toutes les liaisons ont un débit identique de *B* bps (bit par seconde) ;
- toutes les liaisons ont une latence identique de L secondes (temps que met un bit pour aller d'un équipement à l'autre en empruntant la liaison) ;
- les temps de traitement dans les commutateurs et le temps de traitement des accusés de réception sont négligés ;
- *d* désigne le délai inter-trame (temps d'attente entre la fin de l'envoi de la trame précédente et le début de la trame suivante) ;
- il n'y a pas de congestion sur les liens.

La transmission entre le serveur CDN et le client est illustrée par la *figure 5*.

Figure 5 : Transmission de paquets entre le serveur CDN et le client

Q5. Calculer la durée d'émission T_p d'un paquet comprenant P données utiles, c'est à dire le temps entre l'émission du premier bit du paquet et l'émission du dernier bit du paquet.

Q6. En supposant que le dernier paquet a la même taille que les autres, montrer que la durée totale d'émission T_e du fichier pour le serveur CDN (temps entre l'envoi du premier bit par le serveur et l'envoi du dernier bit par le serveur) est égale à :

$$T_e = T_p.\left\lceil\frac{N}{P}\right\rceil + d.\left(\left\lceil\frac{N}{P}\right\rceil - 1\right)$$

Q7. Calculer la durée totale nécessaire au transfert du fichier (temps entre l'envoi du premier bit par le serveur et la réception du dernier bit par le client).

Q8. Calculer le temps nécessaire pour transmettre un fichier de 4 Gigaoctets avec :

- k = 25 ;
- L = 2 ms ;
- B = 10 Gbps ;
- H = 18 octets ;
- P = 1500 octets ;
- d = 9,6 ns.

Q9. Calculer la durée nécessaire pour transmettre le fichier de 4 Go en utilisant des trames plus grandes (jumbo frames) de P = 9000 octets. Justifier le choix d'utiliser ces trames pour la communication entre le serveur FCA et le serveur CDN.

Q10. Calculer la durée nécessaire pour transmettre le fichier de 4 Go en utilisant k = 10, P = 1500 octets et B = 15 Mbps. Commenter sur l'impact de la bande passante disponible dans le cas du transfert entre le serveur CDN et le client.

Q11. Avec le diagramme de la *figure 4* en considérant les durées de transferts calculées dans les questions précédentes, pour la requête B, quelle communication requiert le plus de temps entre les deux communications nécessaires : celle entre le client et le serveur CDN ou celle entre le serveur CDN et le serveur FCA ?

Les calculs ci-dessus considèrent un cas idéal sans congestion réseau : la communication n'est pas affectée par le trafic des autres utilisateurs. En réalité, la congestion réseau est gérée par des protocoles de la couche transport du modèle OSI. Ces protocoles nécessitent l'établissement d'une connexion entre l'émetteur et le récepteur du fichier. Cette connexion peut s'établir grâce à des sockets réseaux. Pour utiliser cette méthode, il est nécessaire d'avoir un programme informatique sur le client et un programme informatique sur le serveur qui lui répond. Côté client, le programme doit contacter le serveur, établir le socket et recevoir les données envoyées par le serveur. Côté serveur, le programme attend une connexion d'un client, établit le socket et envoie les données au client. Les différentes fonctions en langage Python pour l'utilisation de sockets sont fournies par le document technique DT2.

Q12. Dans le code à trous ci-après qui implémente en Python le côté client du socket, écrire le code qui devrait se trouver en ligne 7 à la place de […].

```
1   import socket                    # Import du module socket
2
3   s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)      # Création du socket
4   hote = 'www.monserveur.fr'       # Adresse du serveur
5   port = 8080                      # Port sur lequel se connecter
6
7   [...]
8
9   with open('monfichier', 'wb') as f:
10      while True:
11          print('Réception des données...')
12          donnees = s.recv(1024)
13          if not donnees:
14              break
15          f.write(donnees)
16
17  f.close()
18  print('Successfully get the file')
19  s.close()
20  print('connection closed')
```


Tournez la page S.V.P.

Q13. Dans le code Python serveur ci-après, écrire le code qui devrait se trouver des lignes 16 à 18

```
1   import socket                       # Import module socket
2
3   port = 8080                        # Port utilisé par le service
4   s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)    # Création du socket
5   hote = socket.gethostname()        # Nom de la machine
6   s.bind((hote, port))               # Bind du socket au port
7   s.listen(5)                        # Attente des clients.
8
9   while True:
10      conn, addr = s.accept()        # Etablissement de la connexion avec le client.
11      data = conn.recv(1024)
12
13      filename='fichier'
14      f = open(filename,'rb')
15      l = f.read(1024)
16      [...]
17         [...]
18         [...]
19      f.close()
20
21      conn.close()
```

Q14. Dans le code d'établissement du socket entre le client et le serveur, à quel moment a lieu la requête DNS de la *figure 4* ?

Q15. Dans le code Python serveur ci-dessus, le serveur ne peut accepter qu'un seul client. Détailler le mécanisme nécessaire au serveur pour accepter plusieurs connexions simultanées.

Q16. En considérant les temps d'établissement des connexions et de transferts de fichiers, pour un fichier volumineux tel que celui d'une vidéo (plusieurs Go) et dans des conditions réalistes pour Internet, justifier si c'est la latence ou la bande passante qui a le plus d'impact sur le temps d'acquisition d'une vidéo par un client depuis un CDN.

Partie 2. Modélisation de l'infrastructure du CDN

Objectifs : modéliser l'infrastructure du CDN et plus particulièrement la façon dont elle est dimensionnée et ses échanges avec les fournisseurs de contenus.

Pour être au plus près des utilisateurs et bénéficier de réseaux ayant une grande bande passante, les CDN placent souvent leurs serveurs à l'intérieur des infrastructures de FAI. Par exemple, en 2018, Akamaï disposait de 240 000 serveurs répartis dans plus de 130 pays et déployés sur plus de 1 700 réseaux à travers le monde. De même, Netflix, qui joue à la fois le rôle de FCA et de CDN, dispose de nombreux serveurs répartis dans

des FAI et connectés à des IXP (points d'échange auxquels sont connectés plusieurs FAI) tout autour du globe. Un CDN doit donc contrôler des serveurs qui sont distribués et interconnectés par des FAI différents.

Pour réduire la latence entre le serveur CDN et le client et réduire l'utilisation globale de la bande passante, il est nécessaire de bien positionner les serveurs CDN. D'une part, il faut limiter leur nombre pour limiter le coût du CDN (coût d'achat, de déploiement et de maintenance notamment). D'autre part, il faut optimiser leur positionnement par rapport aux clients pour optimiser la qualité de service perçue par les utilisateurs.

Les emplacements possibles des serveurs CDN sont modélisés par les nœuds d'un graphe. Chaque nœud représente une interconnexion du réseau des clients à couvrir : ainsi, chaque client est directement connecté à un nœud du graphe. Les nœuds sont reliés par des arrêtes dont le poids représente la latence entre les deux nœuds reliés. Il s'agit d'un graphe non orienté (latence symétrique dans les deux sens).

Les graphes valués, non orientés sont représentés en Python par la classe *Lgraphe*. Cette classe utilise un dictionnaire de liste pour représenter les graphes et dispose de deux méthodes pour ajouter respectivement un nœud et une arrête à un graphe.

```
1   from collections import defaultdict
2
3   class Lgraphe:
4       def __init__(self):
5           self.noeuds = set()
6           self.arretes = defaultdict(list)
7           self.latences = {}
8
9       def ajouter_noeud(self, valeur):
10          self.noeuds.add(valeur)
11
12      def ajouter_arrete(self, noeud_src, noeud_dst, latence):
13          self.arretes[noeud_src].append(noeud_dst)
14          self.arretes[noeud_dst].append(noeud_src)
15          self.latences[(noeud_src, noeud_dst)] = latence
16          self.latences[(noeud_dst, noeud_src)] = latence
```

Exemple d'un code pour déclarer un graphe à 3 nœuds et 3 arrêtes :

```
1   h = Lgraphe()
2   h.ajouter_noeud('a')
3   h.ajouter_noeud('b')
4   h.ajouter_noeud('c')
5   h.ajouter_arrete('a', 'b', 10)
6   h.ajouter_arrete('b', 'c', 10)
7   h.ajouter_arrete('a', 'c', 15)
```

Ainsi *h.noeuds* vaut {'b', 'c', 'a'}, *h.arretes* vaut {'a': ['b', 'c'], 'b': ['a', 'c'], 'c': ['b', 'a']} et *h.latences* vaut {('a', 'b'): 10, ('b', 'a'): 10, ('b', 'c'): 10, ('c', 'b'): 10, ('a', 'c'): 15, ('c', 'a'): 15}.

Des éléments du langage Python sont proposés par le Document technique DT1.

Q17. Écrire en Python une fonction ***NbNoeud*** qui prend en entrée un graphe de la classe *Lgraphe* et renvoie le nombre de ses nœuds.

Q18. Écrire en Python une méthode *adjacents*. Cette méthode de la classe *LGraphe* prend en entrée un nœud du graphe et renvoie la liste des nœuds adjacents à ce nœud (c'est à dire directement reliés par une arrête à ce nœud).

Le placement des serveurs CDN doit être optimisé de manière à minimiser la latence perçue par les clients accédant à du contenu vidéo.

La *figure 6* représente un exemple de graphe considéré avec 14 nœuds et 23 arrêtes. Pour simplifier cet exemple, le poids de chaque arrête (latence entre les nœuds ainsi reliés) est supposé égal à 1.

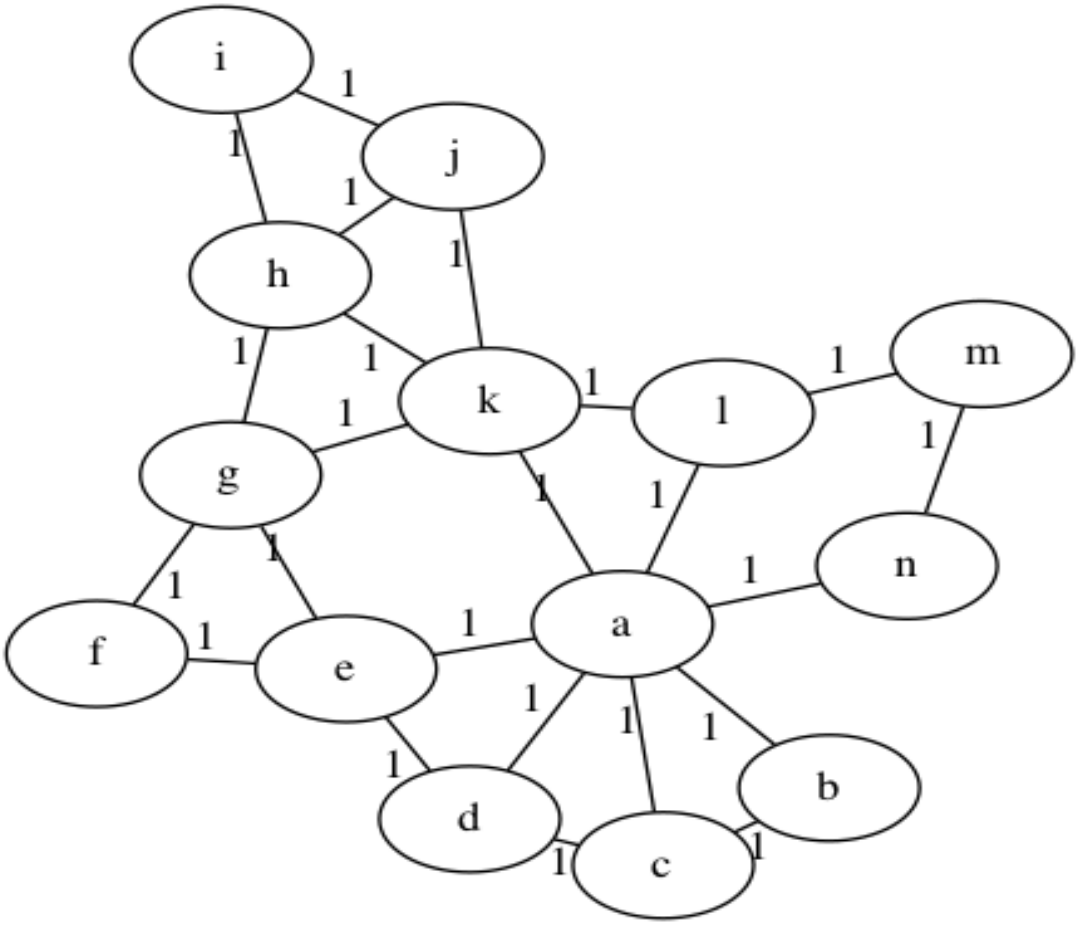


Figure 6 : Exemple de graphe considéré pour la couverture d'un CDN.

Pour un graphe *G(N,A)* avec N l'ensemble des nœuds et A l'ensemble des arrêtes de G, la fonction de distance minimale entre les nœuds i ∈ N et j ∈ N est notée *d(i,j)*. Cette fonction représente donc la somme des latences des arrêtes traversées pour aller de i à j en prenant le plus court chemin.

L'algorithme de Dijsktra permet de déterminer la plus courte distance entre les différents nœuds du graphe. Le pseudocode suivant décrit l'utilisation de l'algorithme de

Dijsktra permettant de déterminer pour un nœud du graphe la distance à chacun des autres nœuds en utilisant le plus court chemin.

```
1    Entrées : un graphe G(N,A) et i un nœud de N
2
3    P = Ø
4    d[a] = +∞ pour chaque nœud a de N
5    d[i] = 0
6
7    while P ≠ (N privé de {i})
8        Choisir un nœud a qui est hors de P ayant la plus petite
     distance d[a]
9        Mettre a dans P
10     foreach nœud b qui est hors de P et est voisin de a
11           d[b] = min(d[b], d[a]+latence(a,b))
12     endfor
13   endwhile
14
15   Sortie : d
```

Algorithme de Dijsktra

Soit ***dijsktra*** la fonction Python, dont le développement n'est pas demandé, qui prend en entrée un *Lgraphe* et un nœud source et renvoie un dictionnaire dont les clés sont les nœuds du graphe et les valeurs sont la distance de chaque nœud au nœud source en utilisant l'algorithme de Dijsktra.

Ainsi pour l'exemple de la *figure 6*, la fonction ***dijsktra****(g, 'a')* renvoie :

{'a': 0, 'b': 1, 'c': 1, 'd': 1, 'e': 1, 'k': 1, 'l': 1, 'n': 1, 'm': 2, 'f': 2, 'g': 2, 'h': 2, 'j': 2, 'i': 3}.

Soit t_{choix} le temps mis pour choisir un nœud à la ligne 7 de l'algorithme et t_{min} le temps mis pour calculer un minimum à la ligne 10.

Q19. Exprimer le temps de calcul total nécessaire pour exécuter l'algorithme de Dijkstra en fonction de |N| le nombre de nœuds, |A| le nombre d'arrêtes, t_{choix} et t_{min}.

Il s'agit de déterminer quels nœuds du graphe sont les meilleurs pour héberger un serveur CDN. Le nombre de connexions simultanées que peut supporter un serveur CDN n'intervient pas ici, mais uniquement la latence perçue lors de la connexion. Chaque utilisateur se connecte au serveur le plus proche. Ainsi, sur l'exemple de la *figure 6*, s'il y a par exemple deux serveurs CDN aux nœuds 'a' et 'i', l'ensemble des nœuds du graphe se trouve à une latence inférieure ou égale à 2 d'un serveur CDN.

Pour déterminer le meilleur emplacement pour un serveur CDN, deux métriques sont considérées : la latence moyenne L_{moy} et la pire latence L_{pc}.

Tournez la page S.V.P.

Soit S l'ensemble des nœuds du graphe où il faut placer un serveur CDN. Ainsi,

$$L_{moy}(S) = \frac{1}{|N|} \sum_{i \in N} min_{j \in S}\, d(i,j)$$

Pour minimiser la latence moyenne pour l'ensemble des clients du CDN, l'objectif est donc de trouver les emplacements des nœuds de S tels que |S| est minimal et $L_{moy}(S)$ est minimal.

Pour minimiser le pire cas, il faut $L_{pc}(S) = max_{i \in N}\, min_{j \in S}\, d(i,j)$. De manière similaire, minimiser le pire cas revient à minimiser $|S|$ et $L_{pc}(S)$.

Q20. Dans l'exemple de la *figure 6*, pour $|S|$ = 1, quel est le nœud qui minimise la latence moyenne ?

Q21. Dans l'exemple de la *figure 6*, pour $|S|$ = 1, quel nœud minimise la pire latence ?

Q22. Écrire en Python la fonction ***NoeudMoyen*** qui prend en entrée un *Lgraphe* et retourne un nœud du graphe qui minimise L_{moy}. Quelle est la complexité de cette fonction en temps de calcul ?

```
1   def MaFonction(lgraphe):
2       lesnoeuds = {}
3
4       for noeud in lgraphe.noeuds:
5           visite = dijsktra(lgraphe, noeud)
6           lmax = 0
7           for dst in visite:
8               if lmax < visite[dst]:
9                   lmax = visite[dst]
10          lesnoeuds[noeud] = lmax
11
12      return min(lesnoeuds, key=lesnoeuds.get)
```

Q23. Que renvoie la fonction Python ***MaFonction*** donnée ci-avant ? A quoi sert-elle ? Quelle est la complexité de cette fonction en temps de calcul ?

Q24. Écrire en Python la fonction ***NoeudsMoyens2*** qui prend en entrée un *Lgraphe* et retourne deux nœuds du graphe ($|S|$ = 2) qui minimisent L_{moy}.

Q25. Quelle est la complexité en temps de calcul dans le cas général de la fonction qui retourne S, un ensemble de nœuds qui minimisent $|S|$ et $L_{moy}(S)$? Qu'implique cette complexité en pratique pour de grands graphes ?

L'algorithme de Gonzalez, défini ci-dessous, permet de placer *k* serveurs CDN sur un graphe.

```
Entrées : un graphe G(N,A) et un entier k

S = ∅
Choisir arbitrairement un nœud et l'ajouter à S

while |S| < k
   ajouter à S le point le plus éloigné des points de S
endwhile

Sortie : S
```

Algorithme de Gonzalez

Q26. Écrire en Python la fonction ***PlusLoin*** qui prend en entrée un *Lgraphe*, un nœud initial et *k* le nombre de serveurs CDN à placer et retourne l'ensemble des nœuds incluant le nœud initial et suivant l'algorithme de Gonzalez.

Q27. Comparer la complexité en temps de calcul de l'algorithme de Gonzalez et la qualité de la solution fournie avec l'approche exacte précédemment implémentée. Quel est le meilleur algorithme ?

Q28. Comment choisir le nœud initial dans l'algorithme de Gonzalez ? Est-ce que ce choix a un impact sur la qualité de la solution calculée ?

Q29. Est-il réaliste de considérer des latences fixes entre les nœuds du graphe ? Expliquer pourquoi. Comment obtenir en pratique ces latences ?

Q30. Donner deux limites du modèle proposé pour décrire l'architecture du réseau CDN. Peut-on adapter le modèle pour prendre en compte ces limites ?

Partie 3. Modélisation du flux vidéo

Objectif : modéliser les flux de données effectivement échangés entre le CDN et les clients.

Une fois déterminé l'emplacement des serveurs CDN, il convient de déterminer le nombre de serveurs nécessaires à chaque emplacement et les volumes de données qui transitent entre les clients et les serveurs CDN.

Une vidéo fournie par un FCA est disponible dans plusieurs formats sur le serveur CDN afin que sa qualité soit adaptée au débit de téléchargement du client.

La *figure 7* représente la courbe idéale de téléchargement d'un fichier vidéo par un client du CDN.

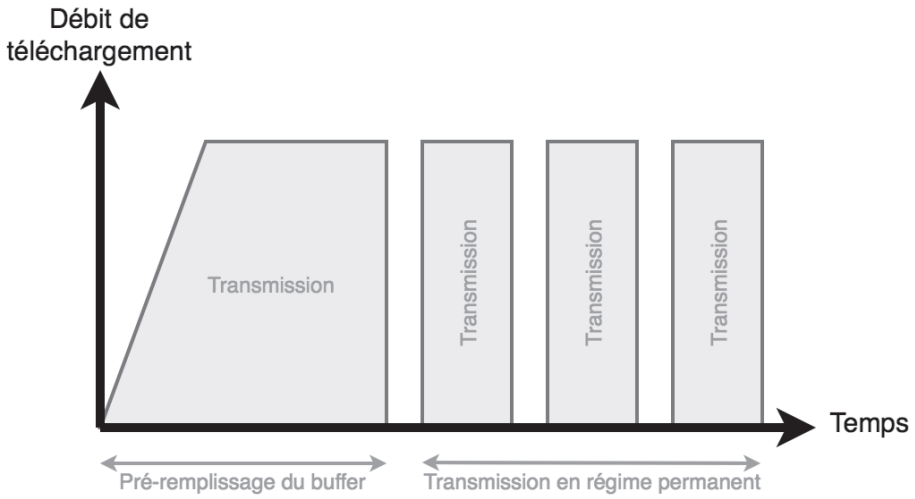


Figure 7 : Courbe de téléchargement d'un fichier vidéo par un client du CDN

Q31. Sur la *figure 7*, quelle est l'utilité de la phase de pré-remplissage du buffer ?

Q32. Sur la *figure 7*, en phase de transmission en régime permanent, pourquoi le téléchargement n'est-il pas continu ? Quel est l'intérêt ?

Le Moving Picture Expert Group (MPEG) a proposé en 2011 le *Dynamic Adaptive Streaming over HTTP* (DASH), un standard de format de diffusion vidéo sur Internet. Le principe consiste à découper un fichier vidéo en segments qui correspondent chacun à quelques secondes. Chaque segment est disponible en plusieurs formats (appelés représentations) qui encodent le fichier à des résolutions ou qualités différentes. Cela permet un streaming adaptatif : la représentation utilisée peut varier au cours du téléchargement d'une même vidéo.

La *figure 8* représente l'architecture client-serveur utilisée par le standard DASH. Le serveur dispose de *n* représentations dans cet exemple. Un manifeste nommé *Media Presentation Description* (MPD) permet de décrire les méta-données du fichier, la durée totale du contenu et les différentes représentations disponibles. Pour chaque représentation, le manifeste indique l'URL HTTP permettant d'accéder à chaque segment. Chaque segment est adressable par une URL unique.

Le client choisit la représentation qu'il souhaite et peut en changer dynamiquement. Ce choix est opéré par l'algorithme *Adaptive Bitrate* (ABR) implémenté par le client.

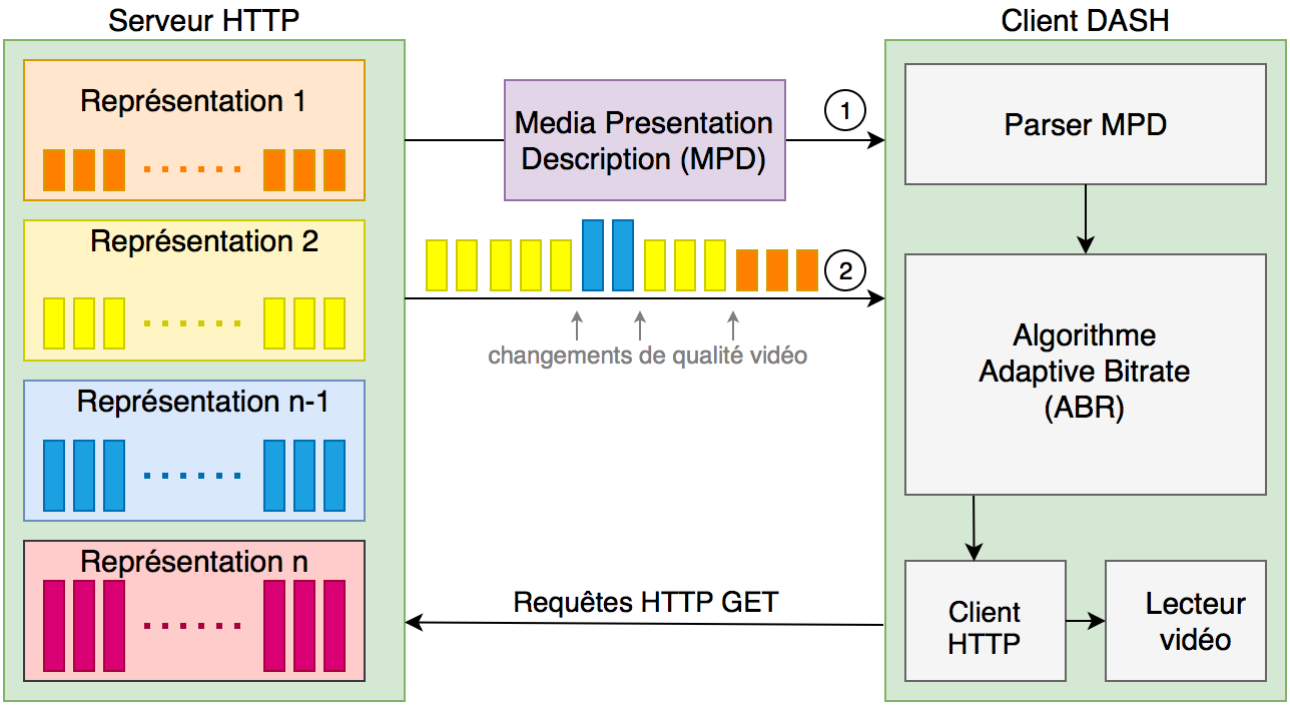


Figure 8 : Architecture client-serveur DASH (Dynamic Adaptive Streaming Over HTTP)

Q33. Quel est l'intérêt de permettre un débit d'émission adaptatif ? Quelle contrainte cela impose sur la taille des segments des différentes résolutions ?

Q34. Pourquoi le contrôle adaptatif du débit d'émission est-il réalisé par le client ?

Q35. Quel est l'intérêt de s'appuyer sur le protocole HTTP pour faire du streaming vidéo ?

Q36. Pourquoi avoir choisi d'utiliser TCP plutôt qu'UDP (*User Datagram Protocol*) dans l'implémentation de DASH ?

Q37. Quelles sont les deux métriques sur lesquelles le client peut s'appuyer pour réaliser le contrôle adaptatif du débit d'émission ?

Q38. Quels sont les deux mécanismes fonctionnant comme des boucles de rétroaction qui permettent de réguler le débit d'émission du serveur en fonction des capacités de téléchargement du client ?

Les flux vidéo transmis par les serveurs CDN sont compressés pour faciliter leur transmission. Pour un débit de téléchargement fixé par le client, il est ainsi possible de faire correspondre une ou plusieurs représentations disponibles. Une représentation spécifie en effet une résolution d'image et un encodage, les deux combinés permettent de calculer le débit nécessaire pour cette représentation.

La *figure 9* représente le PSNR (*Peak Signal to Noise Ratio*) pour trois résolutions d'images différentes et 5 débits d'émission différents par résolution. Le PSNR est un rapport signal sur bruit et permet de mesurer la qualité de reconstruction de l'image décompressée par le client par rapport à l'image initiale. Il est défini par :

$\text{PSNR} = 10\, log_{10}\left(\frac{MAX^2}{MSE}\right)$ avec $MSE = \frac{1}{m.n}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[I(i,j) - K(i,j)]^2$ où MAX est la valeur maximum possible pour un pixel, I est l'image originale de taille $m.n$ et K l'image décompressée par le client.

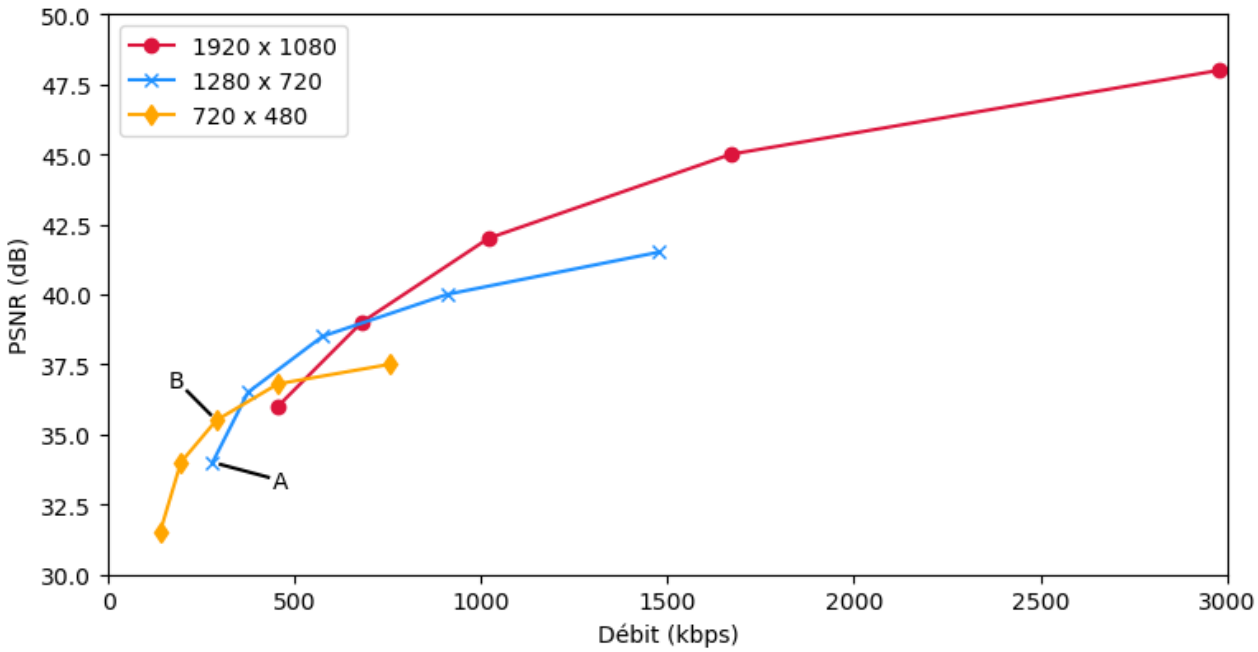


Figure 9 : Exemple pour Netflix de PSNR pour différentes résolutions et débits d'émission

Q39. Pour l'exemple de la *figure 9*, expliquer pourquoi le point A qui a une résolution de 1280x720 obtient un moins bon PSNR que le point B qui a une résolution de 720x480 pour le même débit d'émission. Qu'est-ce que cela implique sur le choix de la meilleure résolution possible ?

Q40. Pour l'exemple de la *figure 9*, citer une méthode permettant de déterminer la courbe de meilleur débit-distorsion.

Q41. En plus du débit, quel autre paramètre permet de déterminer la résolution la plus adaptée pour le streaming d'un flux vidéo pour un client donné ?

Q42. Pour la *figure 9*, expliquer pourquoi cette courbe de meilleur débit-distorsion est différente pour chaque vidéo. Est-il envisageable de calculer cette courbe pour toutes les vidéos hébergées par le CDN ?

Q43. Dans ce système, quelle méthode permettrait de construire rapidement la courbe de meilleur débit-distorsion sans calculer la distorsion induite par chaque résolution à chaque débit possible ?

Q44. Pour l'exemple de la *figure 9*, considérant que tous les formats disponibles fournis par le CDN pour une vidéo donnée sont représentés sur la figure, combien d'exemplaires différents de la vidéo sont stockés par le CDN ? Donnez deux raisons pour lesquelles chaque exemplaire n'est pas calculé à la volée (ou à la demande du client) à partir de la vidéo originale.

Q45. Considérant une vidéo avec une résolution de 1920 x 1080 pixels avec 24 images par seconde (format 1080p) et chaque pixel encodé sur 24 bits, quel débit théorique faudrait-il pour la diffuser telle quelle ? Comparer avec les débits offerts sur l'exemple de la *figure 9* pour la même résolution de vidéo et donner deux raisons pour expliquer la différence observée entre le débit théorique calculé et les débits représentés sur la *figure 9*.

Les hypothèses de modélisation sont les suivantes :

- les vidéos proposées par les fournisseurs de contenu (FCA) ont toutes une durée identique ;
- une fois encodées dans les représentations souhaitées, elles occupent la même taille d'espace disque ;
- le FCA propose 100 000 vidéos qui nécessitent chacune 8 Go de disque pour être stockées (ces 8 Go incluent toutes les représentations souhaitées pour une même vidéo) ;
- le CDN dispose de 10 emplacements géographiquement distants qui contiennent les serveurs utilisés pour diffuser ces 100 000 vidéos ;
- chaque emplacement a la même capacité en terme de stockage et de serveurs ;
- chaque emplacement stocke au maximum un exemplaire de toutes les représentations d'une vidéo.

Q46. Expliquer pourquoi le CDN doit placer plus de 80 To de stockage dans chaque emplacement pour diffuser ces vidéos à l'ensemble des clients. Dans le même contexte, expliquer pourquoi le CDN place probablement moins de 800 To de stockage dans chaque emplacement.

Partie 4. Modélisation de la consommation énergétique

Objectifs : modéliser l'énergie consommée par un CDN, ses clients et les infrastructures réseau permettant aux clients d'accéder aux services du CDN.

Pour modéliser la consommation énergétique d'un équipement informatique, il faut considérer deux parties :

- une partie fixe, qui correspond à l'énergie consommée par l'équipement lorsqu'il est sous tension, mais ne fait rien (pas de calcul ni de transfert de donnée par exemple) ;
- une partie dynamique, qui correspond à l'énergie supplémentaire (c'est-à-dire en plus de la partie fixe) qui dépend des tâches effectuées par l'équipement.

Pour la suite, il est admis que l'énergie consommée par un équipement (notée $E_{équipement}$) est modélisée par le produit de sa puissance moyenne (notée $P_{équipement}$) et de la durée d'utilisation considérée.

La *figure 10* représente la puissance instantanée moyenne lors du visionnage d'une vidéo sur un ordinateur fixe ; la consommation de l'écran n'est pas incluse. Il s'agit ici uniquement de la partie dynamique et 3 résolutions de vidéo sont considérées : basse, moyenne et élevée. Les données représentées sur la *figure 10* correspondent à des mesures effectuées sur un même ordinateur fixe utilisé pour visionner des extraits vidéos de 2 minutes d'un corpus représentatif de 202 vidéos distribuées par Netflix, chaque visionnage étant répété 5 fois. Dans la publication dont est issue cette figure, les auteurs indiquent que l'impact de la carte réseau sur la consommation énergétique totale de l'ordinateur fixe est négligeable. Ils observent en effet moins de 1 W de variation pour des débits de tests allant de 1 Mbps à 60 Mbps.

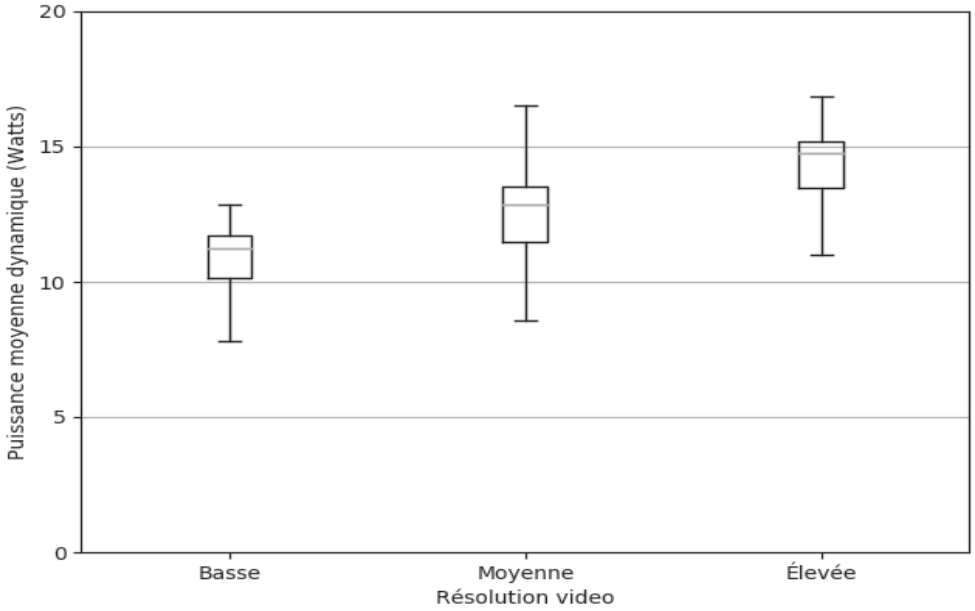


Figure 10 : Consommation énergétique utile pour le visionnage sur poste client d'une vidéo

Q47. Pour la *figure 10*, expliquer pourquoi la puissance consommée varie avec la résolution de la vidéo. Pour une résolution donnée, donner un facteur qui explique la variation de la puissance.

Prendre pour hypothèse que la partie dynamique de l'ordinateur pendant le visionnage d'une vidéo en haute qualité représente 14,5 W pour un débit de 8 Mbps.

Q48. Dans ces conditions, calculer la partie dynamique de la consommation de cet ordinateur pour le visionnage d'une vidéo d'1 Go. Noter cette valeur $E_{ordinateur\text{-}fixe}$.

Le tableau ci-après indique la puissance moyenne consommée par une « box Internet » (Thomson Triple Play VDSL2+) en fonction du débit de la vidéo téléchargée. Ici, la mesure correspond à la totalité de la consommation de la box (puissance fixe et dynamique).

Débit en Mbps	**0**	**0,8**	**2**	**4**	**10,5**
Puissance moyenne en W	14,05	14,06	14,08	14,12	14,23

Le modèle utilisé considère que la puissance de la box s'exprime par une relation linéaire en fonction du débit q (en bps) : $P_{box} = 14{,}05 + 0{,}017 \times 10^{-6} \times q$.

Q49. En utilisant la relation ci-avant, calculer l'énergie dépensée par la box pour le visionnage d'une vidéo de 1 Go en supposant un débit utile parfait de 10 Mbps, c'est-à-dire sans considérer les bits ajoutés par les protocoles réseau et en supposant qu'il n'y a pas de perte de paquet ni de congestion. Noter cette valeur E_{box}.

Q50. En utilisant la relation précédente, calculer la partie dynamique de la consommation de la box pour le visionnage d'une vidéo de 1 Go avec un débit utile parfait de 10 Mbps. Noter cette valeur $E_{box\text{-}dynamique}$. Quelle hypothèse est faite si uniquement cette partie dynamique de la consommation de la box est considérée pour modéliser l'impact du visionnage d'une vidéo ?

La *figure 11* montre la puissance moyenne d'un serveur de CDN en fonction de son débit de sortie qui représente le débit cumulé pour toutes les vidéos qu'il transmet à des clients à un instant donné. Au maximum, ce serveur peut atteindre un débit cumulé de 1,8 Gbps.

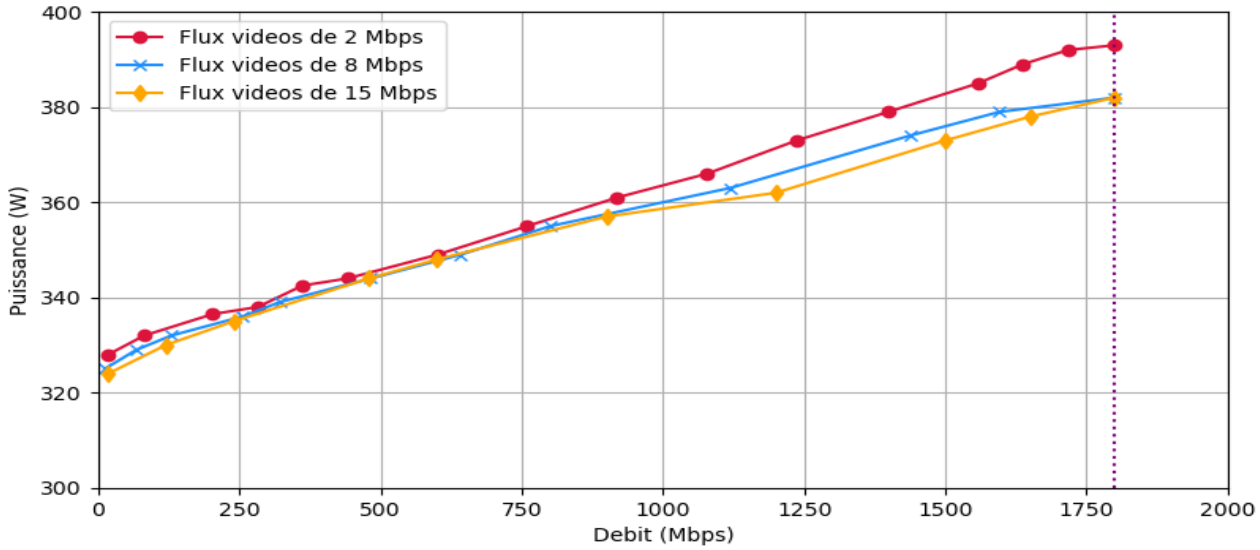


Figure 11 : Consommation énergétique d'un serveur diffusant des vidéos

La consommation énergétique d'un serveur CDN dépend de son débit cumulé q_c grâce à la relation suivante : $P_{serveurCDN} = 325 + 0{,}33 \times 10^{-6} \times q_c$.

Q51. En supposant que le serveur est toujours chargé à son maximum et en utilisant la relation précédente, quelle est sa consommation énergétique pour diffuser 1 Go de vidéo ? Noter cette valeur $E_{serveur}$.

Q52. Comparer $E_{serveur}$ (obtenu à la question Q51) avec $E_{ordinateur\text{-}fixe}$ (obtenu à la question Q48), E_{box} (obtenu à la question Q49) et $E_{box\text{-}dynamique}$ (obtenu à la question Q50). Pour un volume donné de 1 Go de vidéo, quel équipement consomme le plus entre le serveur CDN, la box Internet du client et l'ordinateur fixe du client ?

Le PUE (*Power Usage Effectiveness*) est un indicateur utilisé pour qualifier l'efficacité énergétique d'un centre de calcul. Il se calcule comme suit :

$$PUE = \frac{\text{Energie totale consommée par le centre de calcul}}{\text{Energie consommée par les équipements informatiques}}$$

Les équipements informatiques comprennent les serveurs de calcul et de stockage ainsi que les équipements réseau du centre. L'énergie totale englobe l'énergie liée :

- au système de refroidissement du centre, nécessaire à cause de la dissipation de chaleur des serveurs ;
- aux installations électriques, batteries et systèmes de secours (tels que des groupes électrogènes) utilisés pour éviter les coupures d'électricité en cas de panne d'alimentation ;
- aux équipements informatiques (serveurs et équipements réseau) ;
- aux installations du centre, telles que l'éclairage par exemple.

Pour tenir compte des coûts énergétiques des centres de calcul dans leur ensemble, le coût énergétique lié à l'utilisation d'un serveur est modélisé en multipliant la consommation effective du serveur par le PUE. Ainsi, cette consommation environnée s'exprime comme suit : $\mathrm{E}_{\mathrm{serveurCDNenvironné}} = \mathrm{E}_{\mathrm{serveurCDN}} \times \mathrm{PUE}$

Dans la suite, il est admis que le *PUE* des centres de calcul qui hébergent les serveurs du CDN est de 1,5.

Q53. En s'appuyant sur la relation précédente et $E_{serveur}$ calculé à la question Q51, calculer la consommation environnée d'un serveur CDN pour diffuser 1 Go de vidéo. Noter cette valeur $E_{serveur\text{-}environné}$. Est-ce que cela change l'ordre des valeurs trouvé à la question Q52 ?

Pour modéliser la consommation des réseaux reliant les serveurs CDN et les clients, les mesures sont effectuées sur un routeur Cisco GSR 12000. Ce routeur a une capacité de crête de 5 Gbps. Le modèle suppose que le routeur est chargé en permanence à 50 %, ce qui correspond à un débit moyen de 2,5 Gbps. La puissance moyenne de ce routeur à ce débit est de 375 W.

Q54. Calculer la consommation énergétique de ce routeur pour transmettre 1 Go de données. Noter cette valeur $E_{routeur}$. En supposant que les routeurs entre le client et le serveur CDN sont tous identiques à celui-ci, déterminer à partir de combien de routeurs la consommation du réseau (reliant le CDN au client) surpasse la consommation du serveur CDN pour 1 Go de vidéo.

Comme expliqué dans la *figure 3*, le CDN place généralement ses serveurs au plus près des utilisateurs, allant jusqu'à utiliser les infrastructures des FAI pour bénéficier de leurs réseaux d'accès. Dans ce contexte, le modèle retenu suppose qu'il y a en moyenne

quatre routeurs entre un client et le serveur CDN qui sert sa requête. Cependant, ce rapprochement des utilisateurs nécessite un plus grand facteur de réplication : une vidéo donnée est stockée dans plusieurs sites du CDN.

Les mesures sont effectuées sur un équipement de stockage HP 8100 EVA qui a une capacité de 600 To et une consommation moyenne de 4,9 kW.

Q55. En supposant que ce stockage est utilisé à pleine capacité, en utilisant une modélisation de la consommation énergétique proportionnelle à la taille de stockage, quelle part d'énergie est consommée pour 1 Go de données ? Noter cette valeur $E_{stockage}$. Du point de vue de la consommation énergétique du système global (incluant le CDN et le réseau FAI), citer une condition nécessaire pour que l'utilisation d'un CDN soit plus avantageuse qu'un unique centre de calcul centralisé hébergeant toutes les vidéos et donc avec une distance moyenne en nombre de routeurs plus grande par rapport à un CDN.

Partie 5. Synthèse

Objectifs : conclure cette modélisation et comparer la consommation énergétique d'un client visionnant une vidéo en utilisant un CDN ou un DVD commandé en ligne.

La *figure 12,* issue d'une publication de 2010, donne le résultat de la comparaison de la consommation énergétique d'une vidéo de 2 heures lorsqu'elle est regardée en streaming et depuis un DVD commandé en ligne. Les auteurs considèrent deux scénarios pour le streaming : le premier scénario concerne une architecture classique utilisant un centre de calcul centralisé, le second scénario emploie des optimisations énergétiques pour les centres de calcul et les routeurs.

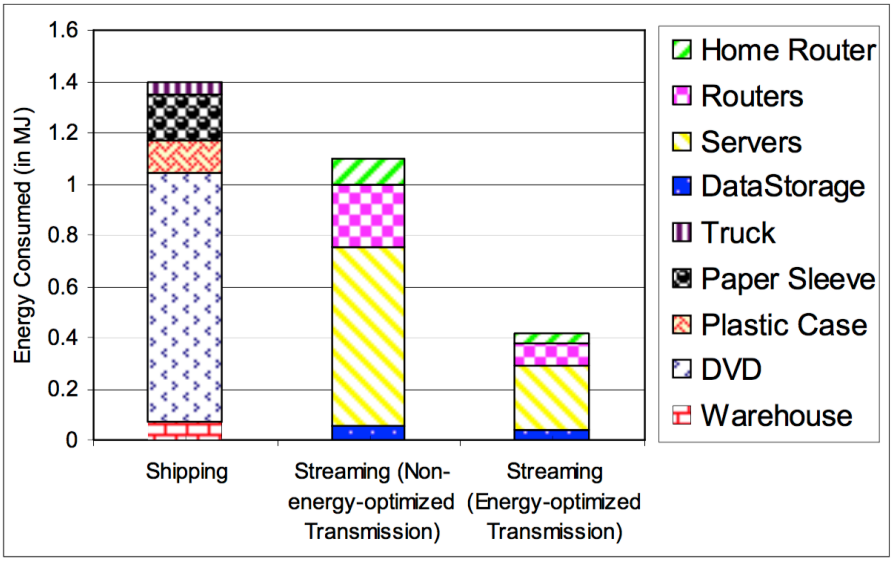


Figure 12 : Consommation énergétique d'une vidéo en streaming ou sur DVD commandé en ligne

Par soucis de performance et de tolérance aux pannes, les infrastructures CDN ont recours à la redondance des équipements et la réplication des données. Pour le stockage, cela se traduit par le fait qu'un format vidéo donné est stocké à l'identique en plusieurs exemplaires. Ainsi, une vidéo d'1 Go est stockée en plusieurs formats et chaque format est stocké en plusieurs exemplaires. Dans cette modélisation, un facteur de réplication moyen de 25 est considéré pour un site du CDN : chaque vidéo d'1 Go représente une occupation effective du stockage de 25 Go en moyenne.

Le surdimensionnement de l'infrastructure touche également les serveurs CDN qui ne sont pas utilisés à pleine capacité pour ne pas risquer la surcharge et pallier rapidement les pannes éventuelles. De plus, comme illustré dans la figure 1, l'utilisation des serveurs varie au cours du temps. Dans cette modélisation, un facteur de redondance moyenne de 3 est considéré : une vidéo d'1Go nécessite en moyenne l'utilisation de 3 serveurs CDN pour être diffusée à un client.

Q56. En utilisant les valeurs $E_{routeur}$, $E_{stockage}$, $E_{serveur\text{-}environné}$, E_{box} calculées dans la partie précédente, les facteurs de réplication moyen et de redondance moyenne, en supposant qu'il y a 4 routeurs entre la box et le serveur CDN, calculer le pourcentage représenté par chaque partie (box, routeurs, serveurs et stockage) dans la consommation énergétique totale pour diffuser 1 Go de vidéo à un client. Comparer ces pourcentages avec les proportions présentées sur la figure 12 pour les cas de streaming.

Q57. Avec l'hypothèse du second scénario de streaming, conformément à la *figure 12*, en passant du DVD au streaming, la consommation énergétique globale de la vidéo est-elle réduite ? Quel facteur principal fait qu'au cours des dernières années, la consommation énergétique mondiale due à la vidéo a augmenté malgré les progrès liés à l'efficacité énergétique des équipements informatiques ?

Ce sujet s'inspire librement des sources suivantes :

- Amsterdam Internet exchange point : https://www.ams-ix.net/ams
- « *L'état d'Internet en France* », rapport d'activité, tome 3, ARCEP, juin 2018.
- Guthemberg Da Silva Silvestre. « *Designing Adaptive Replication Schemes for Efficient Content Delivery in Edge Networks* », thèse de doctorat de l'Université Pierre et Marie Curie - Paris VI, 2013.
- « *Open Connect Everywhere: A Glimpse at the Internet Ecosystem through the Lens of the Netflix CDN* », T. Böttger, F. Cuadrado, G. Tyson, I. Castro et S. Uhlig, SIGCOMM Comput. Commun. Rev. 48, 1, pages 28-34, avril 2018.
- « *The environmental footprint of a distributed cloud storage* », Lorenzo Posani, rapport, mars 2018.
- « *Client-Side Energy Costs of Video Streaming* », O. Ejembi et S. N. Bhatti, IEEE International Conference on Data Science and Data Intensive Systems, pages 252-259, 2015.
- « *Complexity-based consistent-quality encoding in the cloud* », J. De Cock, Z. Li, M. Manohara and A. Aaron, IEEE International Conference on Image Processing, pages 1484-1488, 2016.
- « *A Survey of Rate Adaptation Techniques for Dynamic Adaptive Streaming Over HTTP* », J. Kua, G. Armitage and P. Branch, IEEE Communications Surveys & Tutorials, vol. 19, no. 3, pages 1842-1866, 2017.
- « *Greening the Internet with Nano Data Centers* », V. Valancius, N. Laoutaris, L. Massoulié, C. Diot, et P. Rodriguez, ACM international conference on emerging networking experiments and technologies, pages 37-48, 2009.
- « *Shipping to streaming: is this shift green?* », A. Seetharam, M. Somasundaram, D. Towsley, J. Kurose et P. Shenoy, ACM SIGCOMM workshop on Green networking, pages 61-68, 2010.
- « *The controller placement problem* », B. Heller, R. Sherwood et N. McKeown, workshop on Hot topics in software defined networks, pages 7-12, 2012.

Document technique DT1 Méthodes et fonctions de manipulation d'objets conteneurs standard de Python

Fonctions natives

`len(s)`

Donne la longueur (nombre d'éléments) d'un objet. L'argument peut être une séquence (telle qu'une chaîne, un objet bytes, tuple, list ou range) ou une collection telle qu'un dict.

Méthodes des objets de type liste

`list.append(x)`

Ajoute un élément à la fin de la liste. Équivalent à `a[len(a):] = [x]`

`list.extend(iterable)`

Étend la liste en y ajoutant tous les éléments de l'itérable. Équivalent à `a[len(a):] = iterable`.

`list.insert(i, x)`

Insère un élément à la position indiquée. Le premier argument est la position de l'élément courant avant lequel l'insertion doit s'effectuer, donc `a.insert(0, x)` insère l'élément en tête de la liste et `a.insert(len(a), x)` est équivalent à `a.append(x)`.

`list.remove(x)`

Supprime de la liste le premier élément dont la valeur est égale à x. Une exception ValueError est levée s'il n'existe aucun élément avec cette valeur.

`list.pop([i])`

Enlève de la liste l'élément situé à la position indiquée et le renvoie en valeur de retour. Si aucune position n'est spécifiée, `a.pop()` enlève et renvoie le dernier élément de la liste (les crochets autour du i dans la signature de la méthode indiquent que ce paramètre est facultatif.

`list.clear()`

Supprime tous les éléments de la liste. Équivalent à `del a[:]`.

`list.index(x[, start[, end]])`

Renvoie la position du premier élément de la liste dont la valeur égale x (en commençant à compter les positions à partir de zéro). Une exception ValueError est levée si aucun élément n'est trouvé.

Les arguments optionnels start et end sont interprétés de la même manière que dans la notation des tranches et sont utilisés pour limiter la recherche à une sous-séquence particulière. L'index renvoyé est calculé relativement au début de la séquence complète et non relativement à start.

`list.count(x)`

Renvoie le nombre d'éléments ayant la valeur x dans la liste.

`list.sort(key=None, reverse=False)`

Ordonne les éléments dans la liste.

`list.reverse()`

Inverse l'ordre des éléments dans la liste.

`list.copy()`

Renvoie une copie superficielle de la liste. Équivalent à **`a[:]`**

Document technique DT1 (Suite)

Méthodes et fonctions des objets de type dictionnaire

Un **dictionnaire** est similaire à une **liste** mais qui utilise des **clés au lieu d'index** , c'est à dire des valeurs autres que numériques.
Pour ajouter des valeurs à un dictionnaire il faut indiquer une clé ainsi qu'une valeur :

```
a = {}
a["nom"] = "engel"
a["prenom"] = "olivier"
## a contient alors {'nom': 'engel', 'prenom': 'olivier'}
```

dict.get()

Permet de récupérer une valeur dans un dictionnaire. Si la clé est introuvable, une valeur à retourner par défaut peut être définie :

```
data = {"name": "Olivier", "age": 30}
a = data.get("name")
## a contient : 'Olivier'
A = data.get("adresse", "Adresse inconnue")
## a contient : 'Adresse inconnue'
```

Il est possible de supprimer une entrée en indiquant sa clé,

```
del a["nom"]
a contient alors : {'prenom': 'olivier'}
```

dict.keys()

Permet de récupérer les clés :

```
dict = {"nom":"engel","prenom":"olivier"}
for cle in dict.keys():
    print (cle)
```

affiche :

nom
prenom

dict.values()

Permet de récupérer les valeurs

```
dict = {"nom":"engel","prenom":"olivier"}
for valeur in dict.values():
    print (valeur)
```

Affiche :

engel
olivier

dict.items()

Renvoie une liste de tuple double :

```
dict = {'Name': 'Zara', 'Age': 7}
print ("Valeurs : %s" %  dict.items())
```

affiche :

Valeurs : [('Age', 7), ('Name', 'Zara')]

Document technique DT2

Module Socket

accept()	accepte une connexion, retourne un nouveau socket et une adresse client
bind(addr)	associe le socket à une adresse locale
close()	ferme le socket
connect(addr)	connecte le socket à une adresse distante
connect_ex(addr)	connecte, retourne un code erreur au lieu d'une exception
dup()	retourne un nouveau objet socket identique à celui en cours
fileno()	retourne une description du fichier
getpeername()	retourne l'adresse distante
getsockname()	retourne l'adresse locale
getsockopt(level, optname[, buflen])	retourne les options du socket
gettimeout()	retourne le timeout ou none
listen(n)	commence à écouter les connexions entrantes
makefile([mode, [bufsize]])	retourne un fichier objet pour le socket
recv(buflen[, flags])	reçoit des données
recv_into(buffer[, nbytes[, flags]])	reçoit des données (dans un buffer)
recvfrom(buflen[, flags])	reçoit des données et l'adresse de l'envoyeur
recvfrom_into(buffer[,nbytes,[,flags])	reçoit des données et l'adresse de l'envoyeur (dans un buffer)
sendall(data[, flags])	envoie toutes les données
send(data[, flags])	envoie des données, si possible, retourne le nombre d'octets effectivement transmis
sendto(data[, flags], addr)	envoie des données à une adresse donnée
setblocking(0 | 1)	active ou désactive le blocage le flag I/O
setsockopt(level, optname, value)	définit les options du socket
settimeout(None | float)	active ou désactive le timeout
shutdown(how)	ferme les connexions dans un ou les deux sens.