Chapitre 03: La couche Transport

Table des matières

Objectifs

- Connaître la notion de port logiciel.
- Comprendre les particularités respectives des protocoles TCP et UDP.
- Comprendre dans quels cas il est préférable d'utiliser respectivement TCP et UDP.
- Comprendre la notion de redirection de port.

Rappel: couche Transport des modèles OSI et TCP/IP

Dans le premier chapitre, nous avons vu la correspondance suivante entre les couches des modèles OSI et TCP/IP:

OSI	TCP/IP
7 – Application	4 - Application
6 – Présentation	
5 – Session	
4 – Transport	3 - Transport
3 – Réseau	2 - Internet
2 – Liaison de données	1 - Liaison
1 – Physique	

La couche Transport du modèle TCP/IP correspond donc à cette même couche sur le modèle OSI.

La couche Transport s'assure que les données parviennent de leur source à leur destination, dans le bon ordre et sans erreur. Elle identifie aussi les programmes auxquels s'adressent les données. Une unité de données sur la couche Transport se nomme un **segment**.

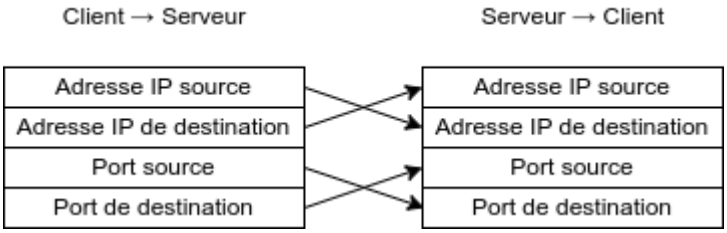
À la différence de la couche Internet, dont les fonctions s'exécutent en grande partie sur les routeurs, les fonctions de la couche Transport ne s'exécutent qu'aux deux bouts d'une communication sur le réseau, c'est-à-dire sur les hôtes.

Les ports

Au chapitre 2, nous avons vu que chaque hôte sur le réseau possède une adresse IP permettant d'y acheminer des paquets. Cette adresse n'est cependant pas suffisante pour acheminer des segments à leur destination sur la couche Transport. En effet, bien qu'une adresse IP permette d'identifier l'hôte de destination, ce dernier peut exécuter plusieurs programmes simultanément, qui peuvent tous communiquer sur le réseau. La destination finale d'un segment n'est donc pas un hôte, mais plutôt un programme s'exécutant sur un hôte. Il faut donc une façon d'identifier ce programme. C'est là qu'entre en jeu la notion de **port**. On parle ici de port logiciel, et non de port matériel (comme un port Ethernet).

Dans le modèle TCP/IP, un port est un nombre codé sur 16 bits. En d'autres termes, il s'agit d'un numéro compris entre 0 et 65 535. Un programme de type serveur « écoute » sur un port précis en indiquant au système d'exploitation qu'il souhaite utiliser ce port. Un programme de type client peut communiquer avec un serveur à l'aide de son adresse IP et son port – il doit donc savoir sur quel port le serveur écoute pour pouvoir communiquer avec lui.

Un segment contient un **port source** (le plus souvent attribué au hasard par le système d'exploitation) et un **port de destination**. Lorsqu'un serveur reçoit un segment provenant d'un client, il sait qu'il doit utiliser le port source de ce segment pour répondre au client. Autrement dit, dans les échanges réseau entre un client et un serveur, le port source des segments envoyés par le client est utilisé comme port de destination des segments envoyés par le serveur. Le même principe est utilisé sur la couche Internet avec les adresses IP source et de destination.



Les ports réservés

Les ports 0 à 1023 sont des **ports réservés** (*well-known ports* en anglais, parfois traduits de façon littérale en « ports bien connus », mais ce terme est déconseillé par l'OQLF[1]). Ce sont des ports utilisés par des protocoles de la couche Application qui sont largement répandus, auxquels ils ont été attribués par l'*Internet Assigned Numbers Authority* (IANA). Sur certains systèmes d'exploitation, un programme doit posséder des droits de superutilisateur pour utiliser un de ces ports.

Voici quelques exemples de ces ports:

Port	Protocole de la couche Application	Description
21	FTP	Protocole de transfert de fichiers.
22	SSH	Protocole sécurisé de contrôle à distance, principalement utilisé pour administrer des serveurs Linux ou Unix.
25	SMTP	Protocole de transfert de courriel.
53	DNS	*Domain Name System*, vu dans le chapitre 2.
80	HTTP	*Hypertext Transfer Protocol*, utilisé principalement pour communiquer avec un serveur Web.
443	HTTPS	Version sécurisée de HTTP.

1 https://vitrinelinguistique.oqlf.gouv.qc.ca/fiche-gdt/fiche/8871773/port-reserve

Parmi ces numéros de port, vous devriez retenir particulièrement ceux des protocoles HTTP (80) et HTTPS (443).

Un port réservé est, en d'autres mots, le port par défaut utilisé par un protocole de la couche Application. Sans ports réservés, on ne pourrait notamment pas accéder à un site Web sans connaître le port utilisé par le serveur qui l'héberge. Ainsi, lorsqu'on tente d'accéder à l'adresse « https://www.cegepsherbrooke.qc.ca », notre navigateur sait qu'il doit utiliser le port de destination 443 pour communiquer avec le serveur, puisqu'il s'agit du port réservé pour le protocole HTTPS. Si le serveur utilisait un autre port, il faudrait le préciser. Par exemple, si le site Web du Cégep était plutôt accessible sur le port 3000, il faudrait taper « https://www.cegepsherbrooke.qc.ca:3000 » dans la barre d'adresse du navigateur pour accéder à ce site Web.

Les protocoles de la couche Transport

La couche Transport du modèle TCP/IP utilise principalement deux protocoles, soit **TCP** et **UDP**. La vaste majorité des protocoles de la couche Application sont construits par-dessus l'un ou l'autre de ces protocoles de la couche Transport. Dans le cadre du cours, il est essentiel de comprendre les particularités de chacun. Nous n'entrerons cependant pas dans les détails de bas niveau de leur fonctionnement.

Il existe également un troisième protocole de couche Transport appelé **QUIC**, qui a fait son apparition récemment, et dont nous ne parlerons que brièvement dans ce chapitre.

Le protocole TCP

TCP (pour *Transmission Control Protocol*) est un protocole de la couche Transport dont l'objectif est de fournir à la couche Application l'illusion d'un canal de communication **fiable** entre deux hôtes. En d'autres mots, lorsqu'un programme source envoie des données par TCP à un programme de destination, il a une excellente garantie que:

- Les données parviendront au programme de destination;
- Elles lui parviendront une seule fois;
- Elles lui parviendront sans erreur (sans contenu altéré);
- Elles lui parviendront dans le bon ordre.

Or, puisque les réseaux sont imparfaits et que la couche Internet n'offre pas cette garantie directement, le service de la couche Transport correspondant au protocole TCP doit prendre en charge plusieurs opérations qu'il « cache » à la couche Application. Voici quelques-unes de ces opérations:

- **Acquittement** des données: l'hôte source exige de recevoir des accusés de réception pour les segments envoyés. Si aucun accusé de réception n'a été reçu au bout d'un certain temps, les données sont envoyées à nouveau. C'est ce qu'on appelle la **garantie de livraison**.

La durée pour laquelle l'hôte source attend la réception d'un acquittement avant de renvoyer un segment est adaptée aux conditions réelles de communication entre les hôtes. Des algorithmes sont utilisés pour estimer le temps qui devrait s'écouler entre l'envoi d'un segment et la réception de son acquittement. Ce temps varie le plus souvent entre quelques dizaines et quelques centaines de milisecondes, et dépend grandement de la proximité géographique entre les hôtes ainsi que des types de liens de communication utilisés.

- **Ordonnancement**: puisque la couche Internet ne maintient pas nécessairement l'ordre des paquets qu'elle achemine à destination, la couche Transport s'assure de transmettre les segments à la couche Application de la destination dans le même ordre qu'ils ont été envoyés par la source. Cela signifie qu'un segment TCP reçu par la couche Transport n'est pas toujours transféré immédiatement à la couche Application, puisqu'il faut parfois attendre la réception d'un segment précédent qui n'a pas encore été reçu.
- **Détection des erreurs**: la couche Transport utilise une opération mathématique au niveau des bits du segment pour calculer une somme de contrôle (*checksum*) de part et d'autre de la transmission. Cela lui permet de détecter lorsque des bits ont été altérés entre la source et la destination, auquel cas le segment fautif n'est pas transmis à la couche Application et n'est pas acquitté. Il est à noter qu'un mécanisme similaire de détection des erreurs est aussi employé sur la couche Liaison. Dans les faits, cette technique n'est pas infaillible et il y a donc une faible probabilité qu'un segment fautif parvienne tout de même à la couche Application (cela pourrait survenir pour jusqu'à 1 segment sur 16 millions selon une étude[2]). Il est donc judicieux de mettre en place un mécanisme de détection d'erreurs sur cette couche également (ce qu'on devrait faire de toute façon puisque la couche Transport ne valide pas le format des données).
- **Contrôle de congestion:** Lorsqu'on utilise le protocole TCP, la couche Transport peut détecter lorsqu'une grande congestion a lieu sur le réseau (par exemple, en remarquant qu'un grand nombre de paquets est perdu), et adapter son comportement en conséquence, par exemple en ralentissant sa transmission.

TCP est aussi un protocole dit « **avec connexion** », c'est-à-dire qu'une session de communication doit être établie entre deux hôtes avant que ceux-ci puissent échanger des données. Cette session de communication a un début et une fin (on peut donc faire une analogie avec une conversation téléphonique). Ce mécanisme est nécessaire pour pouvoir mettre en place les opérations décrites plus haut qui permettent la fiabilité du protocole. L'établissement d'une connexion nécessite trois échanges entre les hôtes :

1. Le client envoie un premier segment au serveur pour lui indiquer qu'il souhaite établir une connexion.
2. Le serveur répond avec un segment indiquant qu'il accepte l'établissement de cette connexion.
3. Le client envoie un autre segment indiquant qu'il a bien reçu la réponse du serveur. La connexion est maintenant établie et l'échange de données peut commencer.

2 https://dl.acm.org/doi/10.1145/347059.347561

La combinaison de ces trois échanges est appelée ***handshake*** (« poignée de main »).

Une fois la connexion établie, la couche Application a accès à un **flux de données** sur lequel elle peut envoyer et recevoir des octets l'un à la suite de l'autre. Le protocole TCP ne s'intéresse pas à la signification des octets transportés, celle-ci concerne seulement la couche Application. De la même façon, la couche Application ne s'intéresse pas aux segments créés par la couche Transport pour transporter les octets.

La fermeture de la connexion peut être initiée par n'importe lequel des deux hôtes. Le processus est semblable à celui permettant d'établir la connexion :

1. Le premier hôte envoie un segment au deuxième hôte indiquant qu'il souhaite mettre fin à la connexion.
2. Le deuxième hôte répond avec un segment confirmant qu'il est prêt à mettre fin à la connexion.
    - Alternativement, si le deuxième hôte a encore des données à transmettre, il peut d'abord acquitter le segment envoyé par le premier hôte, puis lui envoyer les données restantes, puis ensuite seulement lui indiquer qu'il est prêt à mettre fin à la connexion.
3. Le premier hôte confirme la fermeture de la connexion.

Le protocole UDP

Alors que TCP est un protocole fiable et avec connexion, **UDP** (pour *User Datagram Protocol*) est pour sa part un protocole **non fiable** et **sans connexion**. Le seul mécanisme de fiabilité de TCP qui est aussi présent en UDP est la détection des erreurs.

Ainsi, lorsqu'un programme utilise UDP pour envoyer des segments (appelés plus spécifiquement **datagrammes** en UDP), il est possible:

- Que les datagrammes ne parviennent pas au programme de destination;
- Qu'ils lui parviennent plusieurs fois;
- Qu'ils lui parviennent dans le mauvais ordre.

Pourquoi alors utiliserait-on UDP alors qu'il semble n'avoir que des inconvénients en comparaison à TCP? C'est qu'il possède tout de même un avantage majeur, soit une **latence** beaucoup plus faible. On entend par « latence » le temps d'attente entre l'envoi de données par une source et sa réception par la couche Application du destinataire.

En effet, la fiabilité de TCP est offerte au coût d'une latence plus élevée, inhérente aux mécanismes qui garantissent cette fiabilité. Cette latence s'explique principalement par:

- la nécessité d'établir une connexion avant d'échanger des données;
- celle d'acquitter chaque segment;
- celle de transmettre les segments à la couche Application dans le bon ordre.

UDP est donc un choix pertinent pour les applications où la fiabilité est moins importante que la réception la plus rapide possible des données. Cela est particulièrement le cas dans le cadre des applications temps-réel où une certaine perte de données est tolérable. Pensons à la visioconférence ou la voix sur IP (*VoIP*): on veut que les images et/ou la voix de l'autre personne nous parviennent de la façon la plus immédiate possible, quitte à en perdre quelques morceaux — auquel cas on peut toujours demander à l'autre personne de se répéter! Il n'est par ailleurs pas utile, pour une telle application, d'attendre après des données qui arrivent en retard.

Une autre situation où le protocole UDP est parfois privilégié est celui où un client et un serveur doivent s'échanger une petite quantité de données[3], une seule fois. C'est le cas du protocole DNS: le client veut envoyer une seule requête au serveur, et recevoir une seule réponse. La requête et la réponse contiennent toutes les deux une petite quantité de données. En utilisant UDP plutôt que TCP, le protocole DNS s'évite la lourdeur de devoir établir une connexion et acquitter les segments. Si jamais le client ne reçoit pas de réponse au bout d'un certain temps, il envoie sa requête à nouveau, tout simplement.

Finalement, UDP offre une possibilité que TCP n'a pas: envoyer un segment sur une adresse de diffusion (*broadcast*) ou de *multicast*[4] (que nous n'avons pas vu). TCP, pour sa part, peut seulement transmettre des segments entre deux hôtes spécifiques. Cette limitation de TCP est due à la nécessité d'établir une connexion et d'acquitter chaque segment : cela ne serait tout simplement pas viable pour l'envoi d'un segment à plusieurs hôtes à la fois.

Le protocole QUIC

QUIC est un nouveau protocole de couche Transport développé par Google. Il a été introduit en 2012 et standardisé par l'*Internet Engineering Task Force* (*IETF*) en 2021. Il s'agit d'un protocole fiable et avec connexion, comme TCP. Il vise à apporter des améliorations de performance par rapport à celui-ci pour certains cas d'utilisation.

Bien qu'il soit considéré comme un protocole de la couche Transport, QUIC est en réalité implémenté par-dessus UDP. Cela lui permet d'être d'ores et déjà utilisé sur les réseaux actuels sans que des mises à jour d'équipements aient été nécessaires.

Le présent chapitre se concentre sur TCP et UDP. Nous élaborerons davantage sur QUIC dans le chapitre suivant.

Passage entre les couches

Jusqu'à maintenant, nous avons exploré trois des quatre couches du modèle TCP/IP:

- La couche Liaison, qui transmet des trames;

3 À titre d'ordre de grandeur pour ce qui constitue une petite quantité de données : la taille maximale d'un paquet IP sur la plupart des réseaux Ethernet est de 1500 octets.

4 https://fr.wikipedia.org/wiki/Multicast

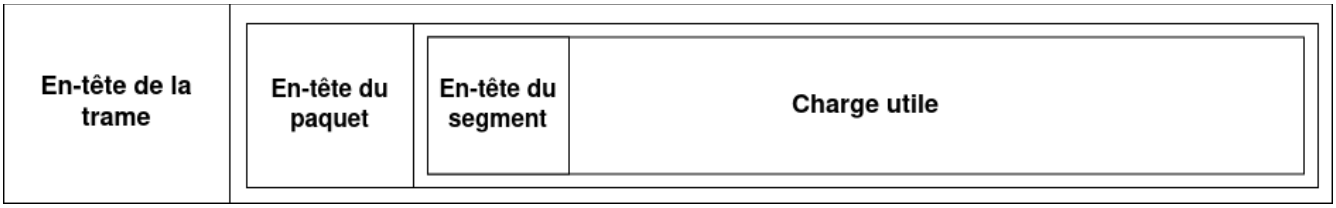
- La couche Internet, qui transmet des paquets;
- La couche Transport, qui transmet des segments.

Nous avons cependant peu exploré la façon dont les données passent d'une couche à l'autre. Il faut en fait savoir que chaque couche **encapsule** les données de la couche supérieure. Ainsi, voici ce qui se passe lorsqu'un programme sur la couche Application envoie des données à un autre programme:

1. La couche Application transfère des données à la couche Transport;
2. La couche Transport crée un segment qui contient un en-tête TCP ou UDP, et une **charge utile** (les données de la couche Application), puis transfère ce segment à la couche Internet;
3. La couche Internet crée un paquet qui contient un en-tête IP et un segment, puis transfère le paquet à la couche Liaison;
4. La couche Liaison crée une trame qui contient un en-tête Ethernet (par exemple) et un paquet, puis le transmet sur un canal de communication.

À l'inverse, lorsque la couche Liaison sur un hôte reçoit une trame:

1. Elle en extrait le paquet, qu'elle transfère à la couche Internet;
2. La couche Internet extrait le segment du paquet, puis le transfère à la couche Transport;
3. La couche Transport extrait la charge utile du segment, puis la transfère à la couche Application;
4. La couche Application traite les données contenues dans la charge utile.

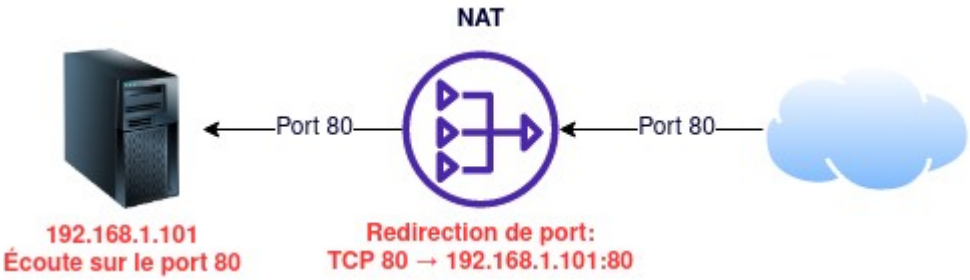


La couche Transport et le NAT

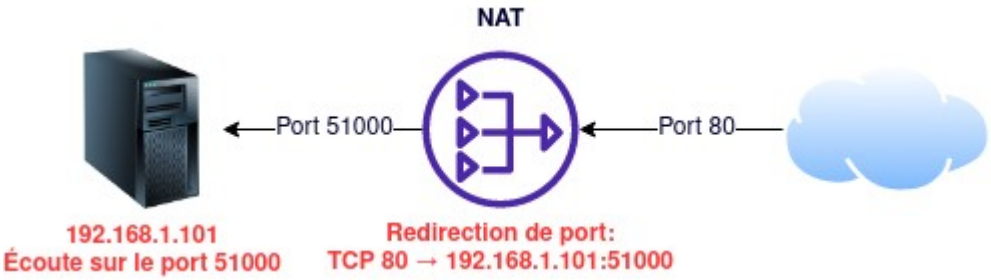
Dans le chapitre 2, nous avons vu que le NAT fait la traduction entre les adresses IP publiques et privées, et modifie les en-têtes des paquets en conséquence. Or, pour ce faire, il doit avoir un moyen de savoir à quelles adresses IP privées les paquets qui proviennent du WAN sont destinés. Le NAT utilise le plus souvent les ports de la couche Transport à cette fin.

Pour qu'un serveur situé derrière un NAT soit accessible depuis le WAN, il faut mettre en place une **redirection de port** dans la configuration du NAT. Une redirection de port consiste à dire au NAT: « Si tu reçois un segment à destination du port X, tu dois le transmettre à telle adresse IP privée, sur le port Y. »

Par exemple, si un serveur Web sur notre LAN écoute sur le port 80, et qu'on veut qu'il soit accessible depuis Internet, également sur le port 80, alors il faut créer une redirection du **port public** 80, vers le **port privé** 80 sur l'adresse IP du serveur.



Le port public et le port privé peuvent être différents. Le serveur pourrait par exemple écouter sur le port 51000 et être tout de même accessible sur le port 80 depuis Internet. Il faudrait cependant utiliser le port 51000 pour accéder au serveur à partir du réseau local.

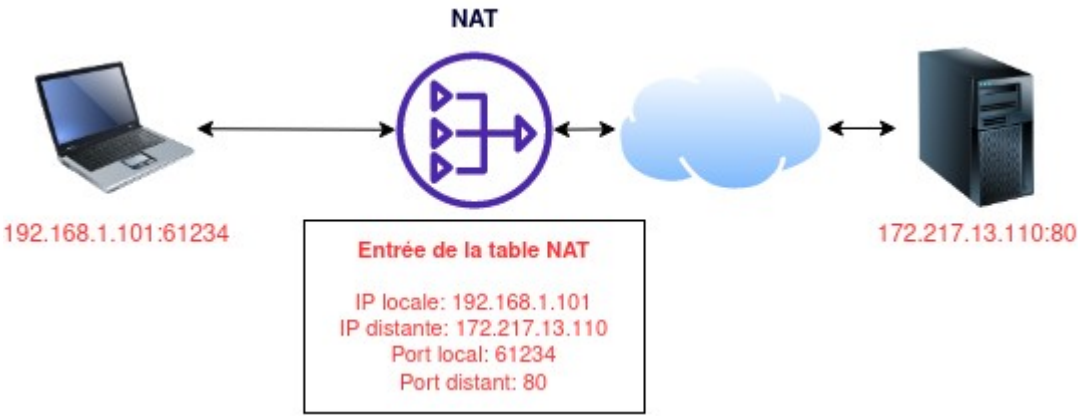


Il est à noter que chaque redirection de port est configurée pour un protocole de transport spécifique (TCP, UDP ou les deux). Il est donc possible de configurer deux redirections de port différentes pour le même port public (une redirection pour les segments UDP, et une autre pour les segments TCP).

Un client, pour sa part, ne nécessite habituellement pas qu'une redirection de port soit explicitement configurée pour pouvoir recevoir des données d'un serveur externe, à condition d'avoir établi une communication avec ce serveur au préalable. Pour rendre cela possible, le NAT peut par exemple sauvegarder les informations suivantes dans sa table chaque fois qu’un hôte envoie un segment à une adresse externe au réseau :

- Adresse locale (l'adresse IP privée du client);
- Adresse distante (l'adresse IP du serveur)
- Port local (le port utilisé par le client)
- Port distant (le port utilisé par le serveur)

Dans cet exemple, si le NAT reçoit par la suite un paquet provenant de l'adresse distante et dont les ports source et de destination correspondent respectivement au port distant et au port local sauvegardés plus tôt, il achemine le paquet à l'adresse locale correspondante.



Dans le cas de TCP, les entrées de la table peuvent être conservées en mémoire jusqu'à la fin de la connexion. Dans le cas d'UDP, puisqu'il n'y a pas de connexion, elles doivent être effacées après un certain temps, ce qui peut poser problème si le client et le serveur ne s'envoient pas des données à intervalles régulières.

Le fonctionnement exact du NAT peut différer d'un type de NAT à l'autre (les différents types de NAT sont hors de la portée du cours). Le fonctionnement décrit ici doit donc être vu comme un exemple de fonctionnement possible.

Synthèse

- La couche Transport utilise un **numéro de port** pour identifier le programme auquel un segment est destiné sur un hôte.
- Les ports de 0 à 1023 sont des **ports réservés** attribués à des protocoles bien connus.
  - Le **port 80** appartient à HTTP.
  - Le **port 443** appartient à HTTPS.
- **TCP** est un protocole de transport **fiable** et **avec connexion**.
- **UDP** est un protocole de transport **non fiable** et **sans connexion**.
- **TCP** est utile lorsqu'il est important de recevoir toutes les données et de les recevoir dans le bon ordre, même au détriment de la vitesse.
- **UDP** est utile lorsque:
  - Une faible latence est plus importante que la fiabilité, par exemple pour les applications temps-réel.
  - Un client doit effectuer une seule requête à un serveur et recevoir une seule réponse, et que les deux contiennent une petite quantité de données.
- Le **NAT** peut utiliser les ports de la couche Transport pour savoir à quelles adresses IP privées il doit acheminer les paquets entrants.

Index

Références

- TANENBAUM, Andrew et WETHERALL, David, *Réseaux, 5e édition*, Paris, Pearson Education France, 2011, p. 529-645.
- https://www.sfml-dev.org/tutorials/2.5/network-socket-fr.php
- https://fr.wikipedia.org/wiki/Port_(logiciel)
- https://fr.wikipedia.org/wiki/Transmission_Control_Protocol
- https://cisco.goffinet.org/ccna/ipv4/couche-transport-tcp-et-udp/
- https://stackoverflow.com/questions/3830206/can-a-tcp-checksum-fail-to-detect-an-error-if-yes-how-is-this-dealt-with
- https://networkengineering.stackexchange.com/questions/52200/if-tcp-is-a-reliable-data-transfer-method-then-how-come-its-checksum-is-not-100

- https://dl.acm.org/doi/10.1145/347059.347561
- https://www.twingate.com/blog/tcp-vs-udp/
- https://networkengineering.stackexchange.com/questions/62764/how-does-a-dns-response-reach-a-client-behind-a-nat#62767
- https://en.wikipedia.org/wiki/QUIC
- https://en.wikipedia.org/wiki/Maximum_transmission_unit