Adobe Flash player doit être installé pour profiter des animations liées à ce document.

La couche TRANSPORT

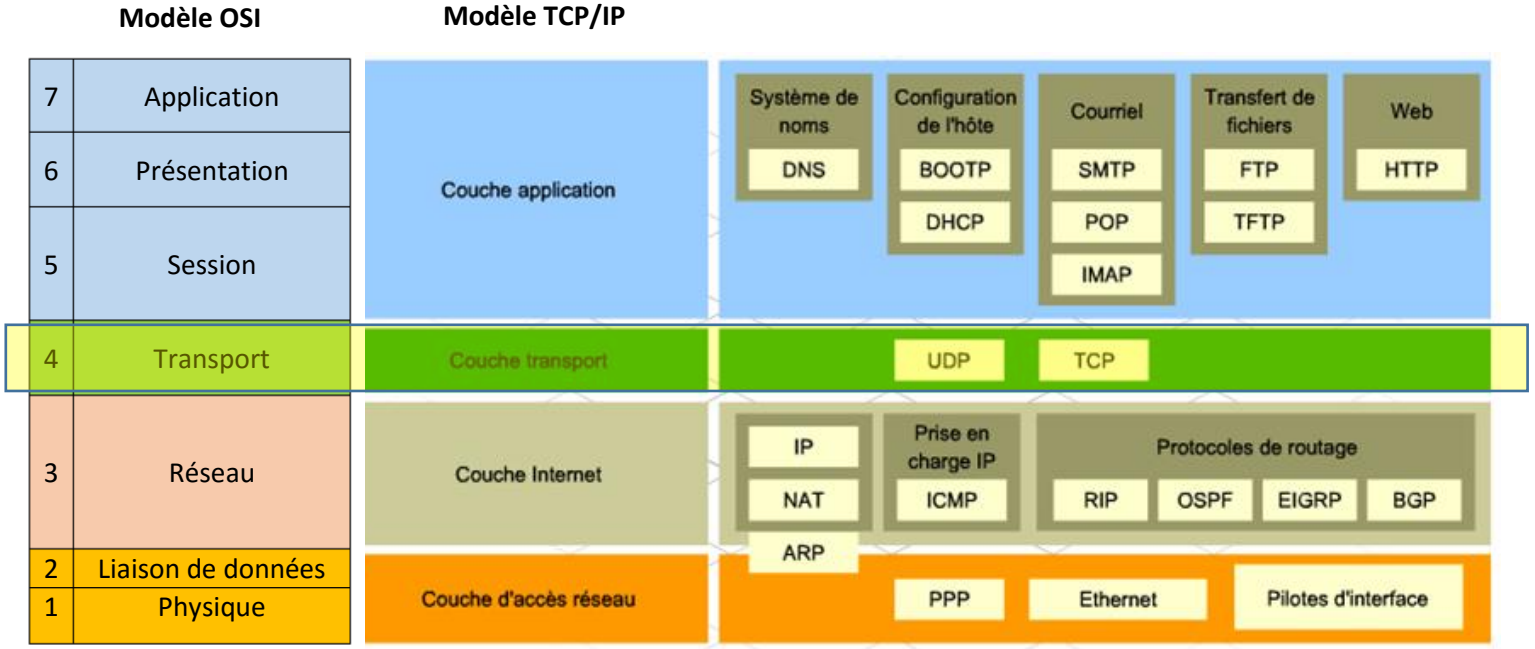


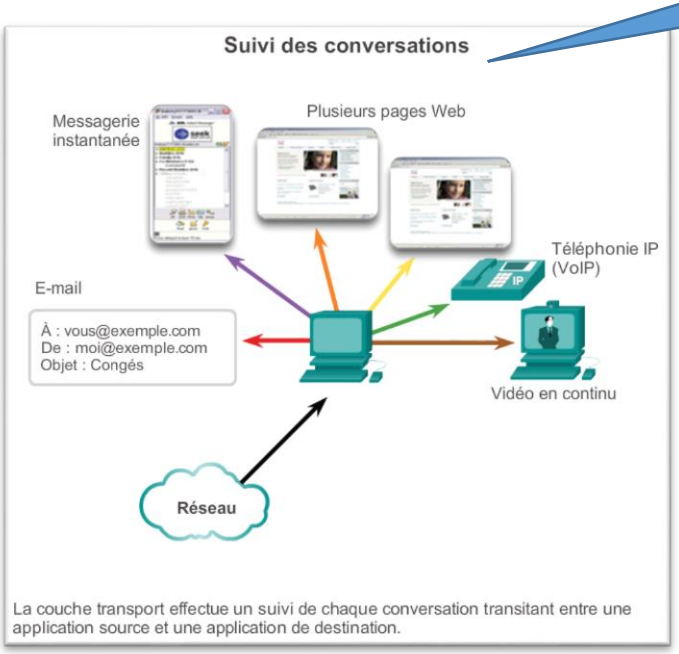
Table des matières

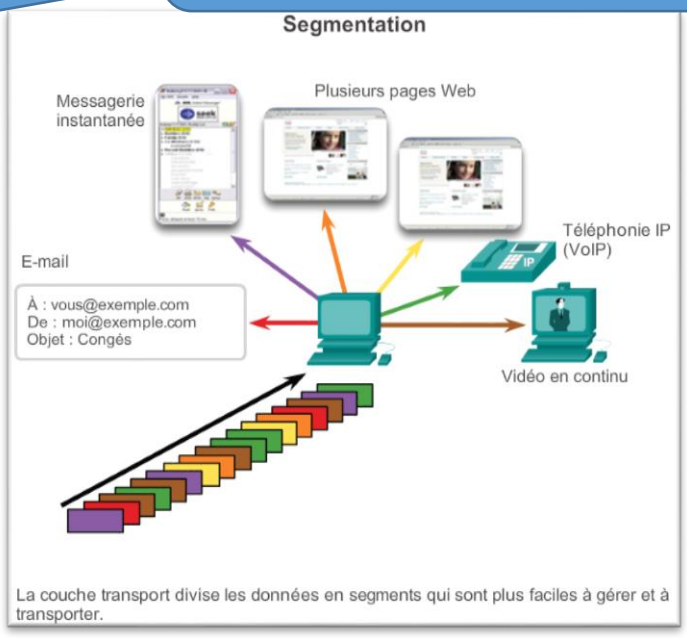
1 Fonctionnalités de la couche Transport

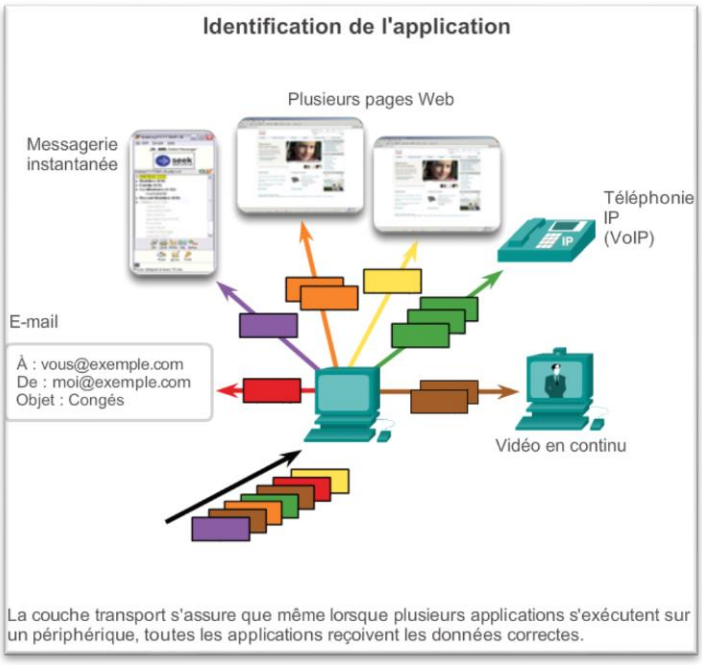
La couche Transport offre 3 services principaux :

- Le suivi des flux réseau.
- La segmentation et le réassemblage des segments.
- L'identification des applications.

Au niveau de la couche transport, chaque ensemble de données transitant entre une application source et une application de destination est appelé une conversation

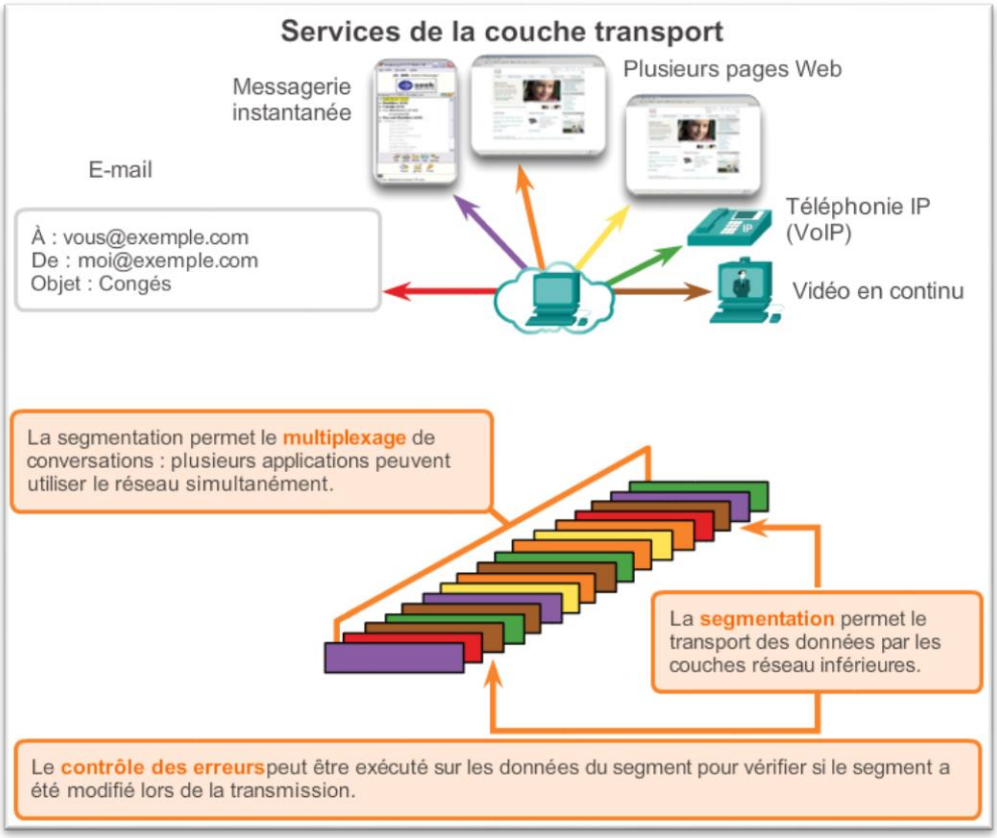






2 Multiplexage des conversations

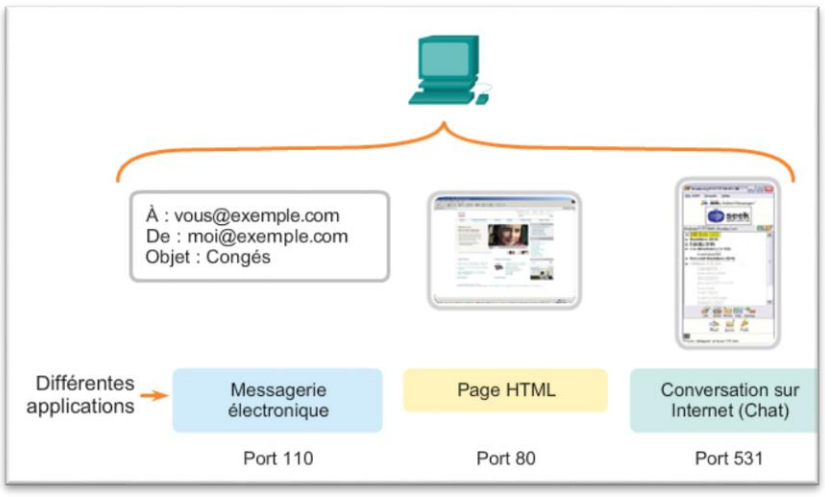
L'envoi de certains types de données (par exemple, un flux vidéo en continu) sur le réseau en tant que flux de communication complet peut utiliser toute la bande passante disponible et empêcher d'autres communications d'avoir lieu en même temps. Ceci rend également difficiles la reprise sur erreur et la retransmission des données endommagées.



La figure montre que la segmentation des données en parties plus petites permet à plusieurs communications différentes, provenant de nombreux utilisateurs, d'être imbriquées (multiplexées) sur le même réseau. La segmentation des données par les protocoles de couche transport permet d'envoyer et de recevoir des données tout en exécutant plusieurs applications simultanément sur un ordinateur.

Pour identifier chaque segment de données, la couche transport ajoute un en-tête contenant des données binaires au segment. Cet en-tête contient des champs de bits. Ce sont les valeurs contenues dans ces champs qui permettent aux différents protocoles de couche transport d'exécuter des fonctions diverses de gestion des communications de données.

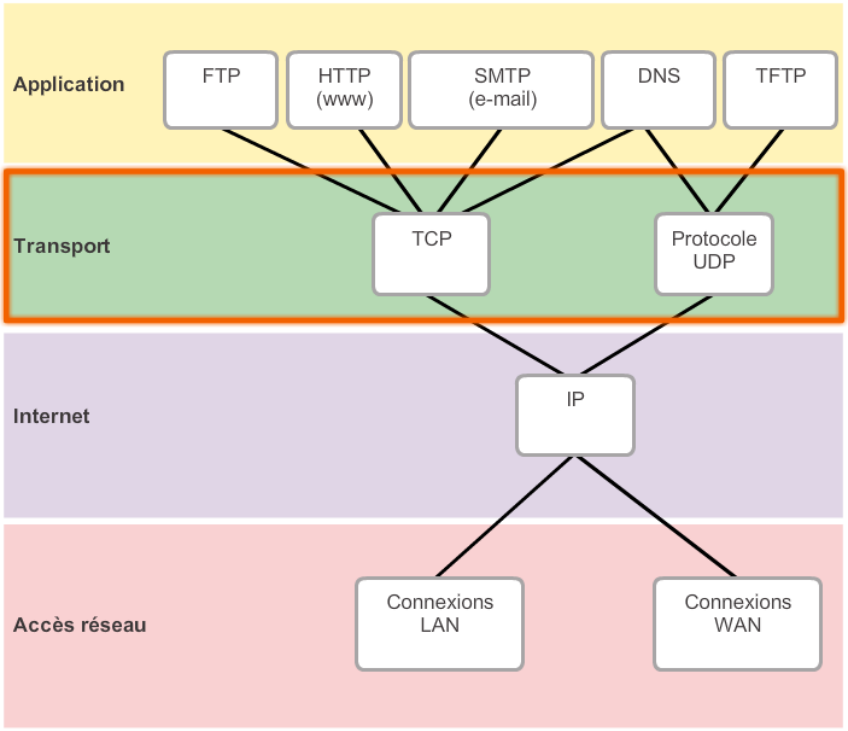
Si deux hôtes veulent communiquer ensemble en utilisant plusieurs applications différentes, il faut trouver un moyen de différencier ces applications sur les hôtes. Pour rendre ce service, **les protocoles de la couche Transport s'appuient sur la notion de port**. Le principe est simple : chaque application se voit attribuer un port virtuel qui l'identifie de manière unique sur l'hôte. Le numéro du port se trouve dans l'entête créé par la couche Transport.



3 Fiabilité de la couche TRANSPORT

La couche transport est également responsable de la gestion des exigences de fiabilité d'une conversation. Des applications différentes ont des exigences différentes en matière de fiabilité du transport.

Le protocole IP ne s'occupe que de la structure, de l'adressage et du routage des paquets. Il ne fixe pas le mode d'acheminement ou de transport des paquets. Les protocoles de transport définissent comment transmettre les messages entre les hôtes. La suite de protocoles TCP/IP propose deux protocoles de couche transport, TCP et UDP, comme illustré dans la figure ci-contre. Le protocole IP utilise ces protocoles de transport pour permettre aux hôtes de communiquer et de transmettre des données.



Le protocole **TCP** est un protocole de couche transport **fiable et complet**, qui garantit que toutes les données arrivent à destination.

En revanche, le protocole **UDP** est un protocole de couche transport **très simple qui ne permet pas de garantir la fiabilité.**

4 Le protocole TCP

Le protocole TCP est un protocole de transport fiable, ce qui signifie qu'il comprend des processus permettant d'assurer un acheminement fiable des données entre les applications par l'utilisation d'accusés de réception. **Le transport TCP revient à envoyer des paquets qui sont suivis de la source à la destination.**

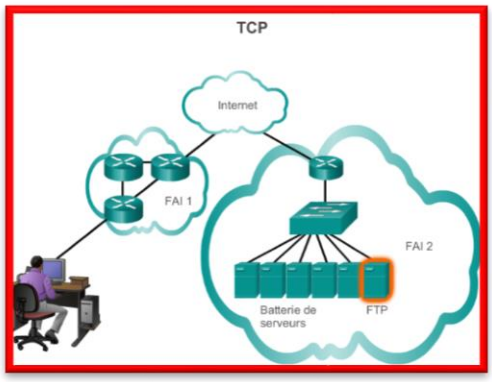
Avec le protocole TCP, les trois fonctions de fiabilité de base sont :

- le suivi des segments de données transmis ;
- les accusés de réception des données ;
- la retransmission des données n'ayant pas fait l'objet d'un accusé de réception.

TCP découpe un message en petits morceaux appelés segments. Les segments, numérotés en séquence, sont ensuite passés au processus IP pour être assemblés en paquets. TCP conserve une trace du nombre de segments qui ont été envoyés à un hôte donné à partir d'une application spécifique. Si l'expéditeur ne reçoit pas d'accusé de réception au bout d'un certain temps, il suppose que les segments ont été perdus, et il les retransmet.

Seule la partie du message qui a été perdue est renvoyée, pas l'intégralité. Sur l'hôte récepteur, TCP est responsable de la reconstitution des segments de message et de leur transmission à l'application. Les protocoles FTP (File Transfer Protocol et HTTP (Hypertext Transfer Protocol) sont des exemples d'applications qui utilisent le protocole TCP pour assurer l'acheminement des données.

Observez le fonctionnement de TCP sur l'animation ci-contre.



Le protocole TCP a été initialement décrit dans le document RFC 793. Outre la prise en charge des fonctions de base de segmentation et de réorganisation des données, le protocole TCP, comme l'illustre la figure, fournit également :

- des conversations orientées connexion en établissant des sessions ;
- un acheminement fiable ;
- une reconstitution ordonnée des données ;
- le contrôle de flux.

L'**établissement d'une session** permet de s'assurer que l'application est prête à recevoir les données.

La **livraison dans un ordre défini** permet de s'assurer que les segments sont remis dans le bon ordre.

L'**acheminement fiable** signifie que les segments perdus sont renvoyés afin que les données soient reçues dans leur intégralité.

Le **contrôle de flux** gère l'acheminement des données en cas d'encombrement au niveau de l'hôte.

Segment TCP

Bit (0)	Bit (15)	Bit (16)	Bit (31)
Port source (16)		Port de destination (16)	
Numéro d'ordre (32)			
Numéro d'accusé de réception (32)			
Longueur d'en-tête (4)	Réservé (6)	Bits de contrôle (6)	Fenêtre (16)
Somme de contrôle (16)		Urgent (16)	
Options (0 ou 32 le cas échéant)			
Données de la couche application (taille variable)			

20 octets

Utilisé pour réorganiser les données.

Indique les données qui ont été reçues.

Utilisée pour le contrôle des erreurs sur l'en-tête et les données de segment.

Indique le nombre de segments qui peuvent être acceptés en même temps.

Indique si les données sont urgentes.

Port source : Ce champ de 2 octets est utilisé pour indiquer quel est le port utilisé par l'émetteur du segment. Une station qui héberge plusieurs applications doit répondre aux sollicitations qui lui parviennent. Il faut être en mesure de distinguer quelle application a émis le paquet et ce champ sert à cela.

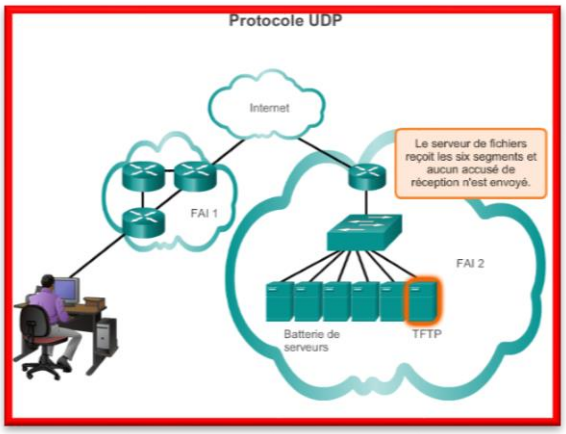
Port destination : Comme pour le port source, ce champ est codé sur 2 octets. Il est utilisé pour indiquer sur quel port virtuel la communication est souhaitée. Lorsque la communication est initiée depuis le client, ce dernier doit connaître le port sur lequel est hébergée l'application qu'il souhaite joindre. Il suffit ensuite à la station de destination d'inverser les données de ports source et destination pour répondre.

Les numéros de port jusqu'à 1024 sont réservés pour des applications bien connues, par exemple le port 80 pour WWW ou le port 21 pour FTP. La liste des numéros de ports réservés par l'IANA peut être consultée sur http://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml
Malgré cela il reste possible d'affecter un port à une application différente notamment si plusieurs instances d'un même service doivent être hébergées sur un même hôte.

5 Le protocole UDP

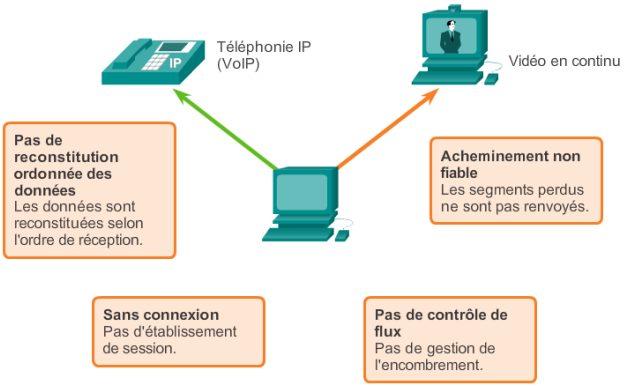
Tandis que les fonctionnalités de fiabilité TCP assurent l'efficacité des communications entre les applications, elles engendrent également une surcharge supplémentaire et des éventuels retards de transmission. Un compromis doit être établi entre la valeur accordée à la fiabilité et la charge qu'elle représente sur le réseau. Imposer une surcharge pour garantir la fiabilité de certaines applications peut réduire l'utilité de l'application et peut même porter préjudice à l'application. Dans ce cas, le protocole UDP représente un meilleur protocole de transport.

Le protocole UDP fournit uniquement des fonctions de base permettant d'acheminer des segments de données entre les applications appropriées avec peu de surcharge et de vérification des données. Le protocole UDP est un protocole d'acheminement au mieux. Dans le contexte des réseaux, l'acheminement au mieux est considéré comme n'étant pas fiable car aucun accusé de réception ne confirme que les données sont arrivées à destination. Avec le protocole UDP, aucun processus de couche transport ne signale à l'expéditeur si la transmission a réussi.



L'UDP revient à poster une lettre normale, sans accusé de réception.

Le protocole UDP est un protocole de transport d'acheminement au mieux, décrit dans le document RFC 768. Le protocole UDP est un protocole de transport léger qui offre les mêmes fonctions de segmentation et de réorganisation des données que le protocole TCP, mais sans la fiabilité et le contrôle de flux du protocole TCP. C'est un protocole simple, qui est généralement décrit en indiquant ce qu'il ne fait pas par rapport au protocole TCP.



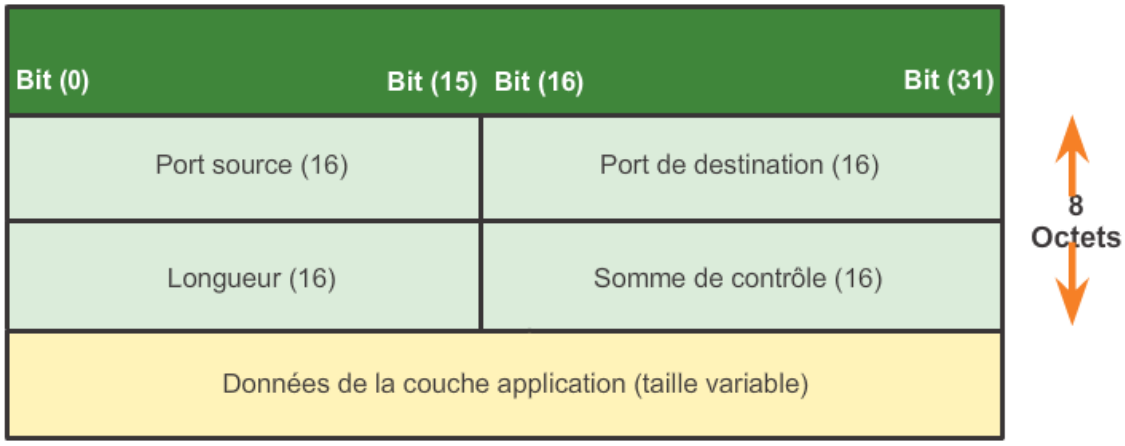
Comme l'illustre la figure, les fonctionnalités suivantes constituent le protocole UDP :

- **Sans connexion** – le protocole UDP n'établit pas de connexion entre les hôtes avant que les données puissent être envoyées et reçues.

- **Acheminement non fiable** – le protocole UDP ne fournit pas de services garantissant que les données sont acheminées de façon fiable. Il n'existe pas de processus dans le protocole UDP permettant de faire retransmettre à l'expéditeur les données perdues ou endommagées.
- **Aucune reconstitution ordonnée des données** – parfois, les données sont reçues dans un ordre différent de celui dans lequel elles ont été envoyées. Le protocole UDP n'offre aucun mécanisme permettant de réorganiser les données dans leur ordre initial. Les données sont simplement remises à l'application dans l'ordre où elles arrivent.
- **Aucun contrôle de flux** – le protocole UDP ne propose aucun service permettant de contrôler la quantité de données envoyées par la source pour éviter de submerger le périphérique de destination. La source envoie les données. Si les ressources sur l'hôte de destination sont surexploitées, l'hôte de destination abandonne généralement les données envoyées jusqu'à ce que des ressources soient disponibles. Contrairement au protocole TCP, le protocole UDP ne fournit aucun mécanisme permettant de retransmettre automatiquement les données abandonnées.

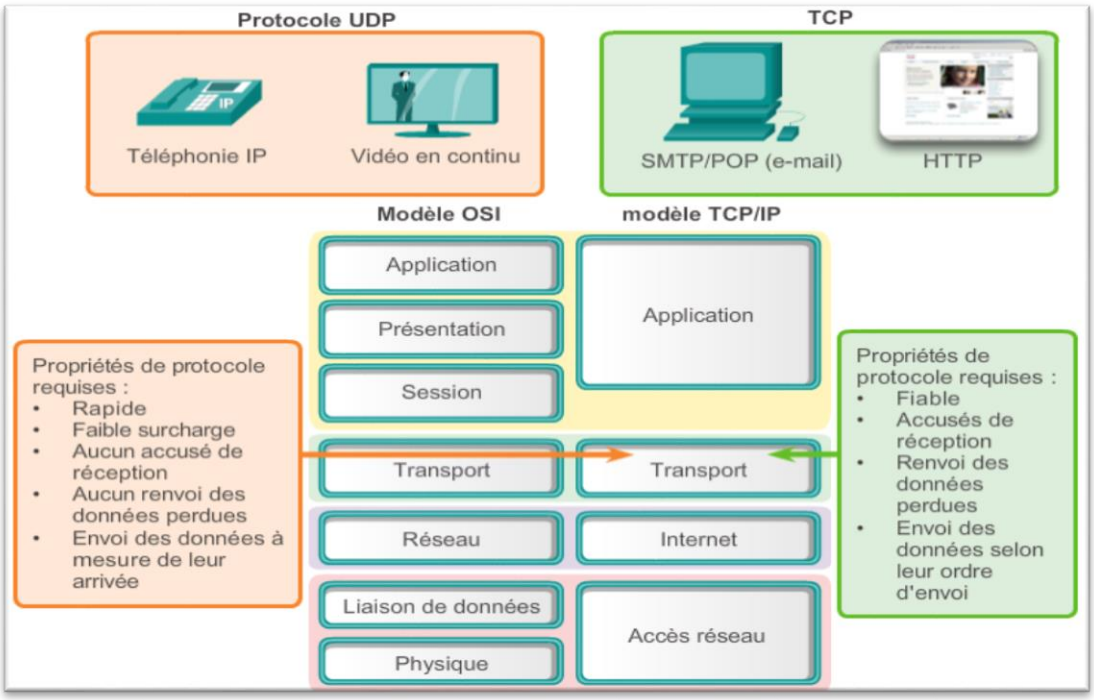
Bien que le protocole UDP n'inclue pas les mécanismes de fiabilité et de contrôle de flux du protocole TCP, comme l'illustre la figure, la faible surcharge pour l'acheminement des données du protocole UDP fait de ce dernier un protocole de transport idéal pour les applications qui peuvent tolérer certaines pertes de données. Les blocs de communications utilisés dans le protocole UDP sont appelés des datagrammes. Ces datagrammes sont envoyés « au mieux » par le protocole de couche transport. Le système de noms de domaine (DNS), la transmission vidéo en continu et la voix sur IP (VoIP) comptent parmi les applications utilisant le protocole UDP.

DATAGRAMME UDP

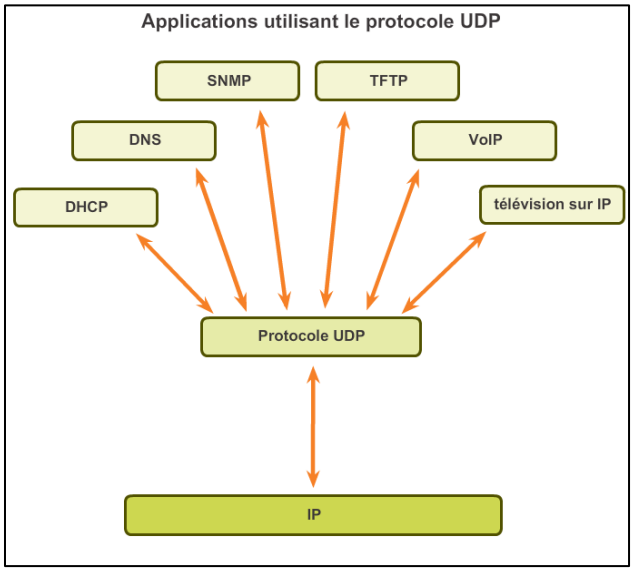


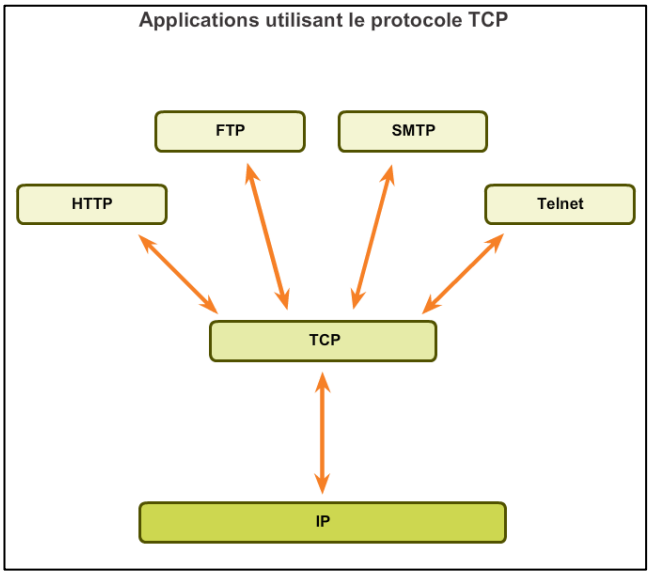
UDP est un protocole sans état, c'est-à-dire que ni le client ni le serveur ne sont tenus de surveiller l'état de la session de communication. Comme l'illustre la figure, le protocole UDP n'est pas concerné par la fiabilité et le contrôle de flux. Les données peuvent être perdues ou reçues dans le désordre sans qu'aucun mécanisme UDP ne puisse récupérer ou réorganiser les données. Si la fiabilité est nécessaire dans le cadre de l'utilisation d'UDP comme protocole de transport, elle doit être prise en charge par l'application.

6 Comparaison des protocoles UDP et TCP



Entrainez-vous avec cet exercice et cet exercice





7 Adressage des ports avec TCP ou UDP

L'en-tête de chaque segment ou datagramme contient un port source et un port de destination. Le numéro de port source est le numéro associé à l'application d'origine sur l'hôte local pour cette communication. Le numéro de port de destination est le numéro de cette communication associé à l'application de destination sur l'hôte distant.

Lorsqu'un message est transmis à l'aide du protocole TCP ou UDP, les protocoles et services demandés sont identifiés par un numéro de port. Un port est un identifiant numérique, présent dans chaque segment, qui est utilisé pour conserver la trace de certaines conversations et de certains services de destination demandés. Chaque message envoyé par un hôte contient un port source et un port de destination.

Port de destination : Le client place un numéro de port de destination dans le segment pour informer le serveur de destination du service demandé. Par exemple, **le port 80 renvoie au service HTTP ou Web**. Lorsque le client spécifie le port 80 comme port de destination, le serveur qui reçoit le message sait que des services Web sont demandés. Un serveur peut proposer plusieurs services simultanément. Par exemple, il peut proposer des services Web sur le port 80 et, en même temps, l'établissement d'une connexion FTP sur le port 21.

Port source : Le numéro du port source est généré de manière aléatoire par le périphérique émetteur pour identifier une conversation entre deux périphériques. Ainsi, plusieurs conversations peuvent s'effectuer simultanément. En d'autres termes, un périphérique peut envoyer plusieurs requêtes de service HTTP à un serveur Web en même temps. Un suivi des différentes conversations est effectué sur la base des ports sources.

Les ports sources et de destination sont placés à l'intérieur du segment. Les segments sont ensuite encapsulés dans un paquet IP. Le paquet IP contient l'adresse IP de la source et de la destination. La combinaison des adresses IP source et de destination ainsi que des numéros de port source et de destination est appelée un socket. L'interface de connexion sert à identifier le serveur et le service demandés par le client. Chaque jour, des milliers d'hôtes communiquent avec des millions de serveurs différents. **Ces communications sont identifiées par les sockets**.

La combinaison du numéro de port de la couche transport et de l'adresse IP de la couche réseau de l'hôte suffit à identifier de manière unique un processus d'application particulier exécuté sur un périphérique hôte individuel. Cette combinaison **est appelée un socket**. Une paire de sockets, composée des adresses IP et numéros de port source et de destination, est également unique et identifie la conversation spécifique entre les deux hôtes.

Un socket client peut se présenter comme suit, 1099 représentant le numéro de port source : `192.168.1.5:1099`

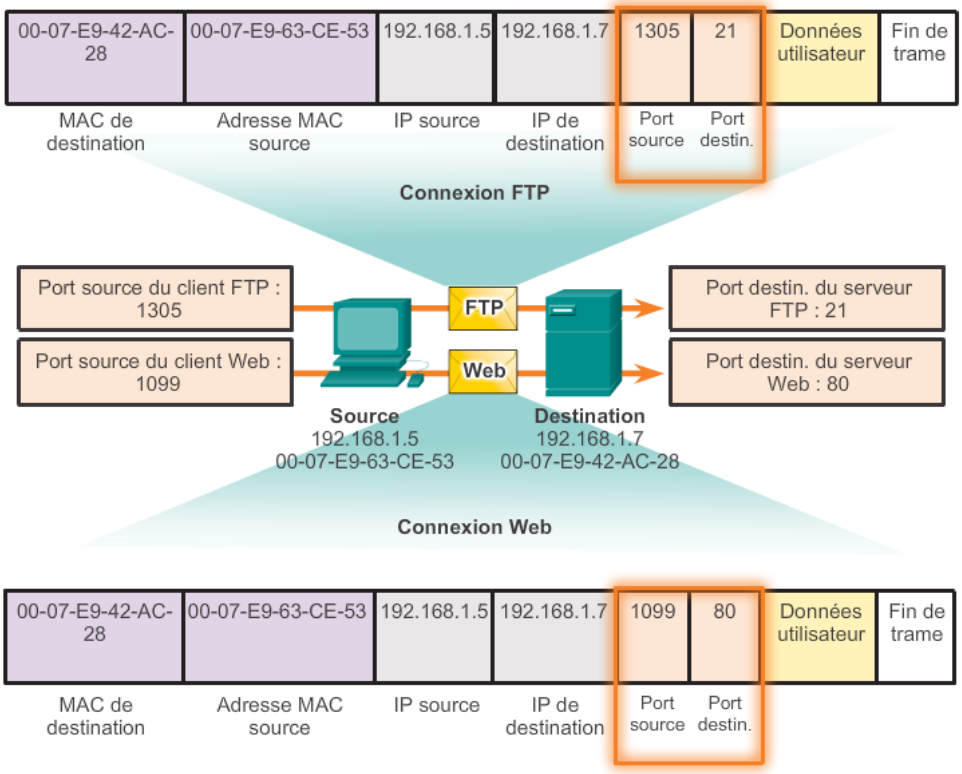
Le socket d'un serveur Web peut avoir la forme suivante : `192.168.1.7:80`

Ensemble, ces deux sockets constituent une paire de sockets : `192.168.1.5:1099, 192.168.1.7:80`

Avec la création de sockets, les points de communication sont connus de sorte que les données peuvent passer d'une application sur un hôte à une application sur un autre. Les sockets permettent à plusieurs processus exécutés sur un client de se différencier les uns des autres, et aux multiples connexions à un processus serveur de se distinguer les unes des autres.

Le port source d'une requête de client est généré aléatoirement. Le numéro de port fait office d'adresse de retour pour l'application envoyant la requête. La couche transport effectue le suivi du

port et de l'application à l'origine de la requête afin que la réponse, quand elle sera envoyée, soit transmise à l'application appropriée. Le numéro de port de l'application envoyant la requête sert de numéro de port de destination dans la réponse renvoyée depuis le serveur.



Plage de numéros de port	Groupe de ports
0à1023	Ports réservés
De 1024 à 49151	Ports inscrits
49152 à 65535	Ports dynamiques et/ou privés

Légende

Ports UDP inscrits :
1812 RADIUS Authentication Protocol
5004 RTP (Voice and Video Transport Protocol)
5040 SIP (VoIP)

Ports UDP réservés :
69 TFTP
520 RIP

Ports TCP inscrits :
1863 MSN Messenger
2000 Cisco SCCP (VoIP)
8008 Alternate HTTP
8080 Alternate HTTP

Ports TCP réservés :
21 FTP
23 Telnet
25 SMTP
80 HTTP
143 IMAP
194 Internet Relay Chat (IRC)
443 Secure HTTP (HTTPS)

Ports TCP/UDP inscrits courants :
1433 MS SQL
2948 WAP (MMS)

Ports TCP/UDP réservés courants :
53 DNS
161 SNMP
531 AOL Instant Messenger, IRC

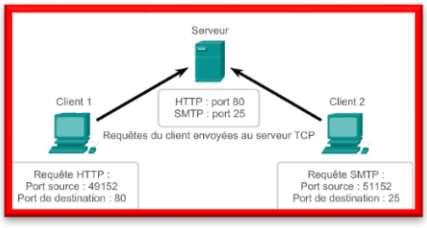
Il est parfois nécessaire de savoir quelles connexions TCP actives sont ouvertes et s'exécutent sur un hôte en réseau. L'utilitaire `netstat` est un utilitaire de réseau important qui peut être utilisé pour vérifier ces connexions. Il répertorie le protocole utilisé, l'adresse et le numéro de port locaux, l'adresse et le numéro de port distants, et l'état de la connexion.

```
C:\> netstat

Active Connections

Proto   Local Address    Foreign Address          State
TCP     kenpc:3126       192.168.0.2:netbios-ssn  ESTABLISHED
TCP     kenpc:3158       207.138.126.152:http     ESTABLISHED
TCP     kenpc:3159       207.138.126.169:http     ESTABLISHED
TCP     kenpc:3160       207.138.126.169:http     ESTABLISHED
```

Si vous avez encore besoin de précisions, regardez cette animation

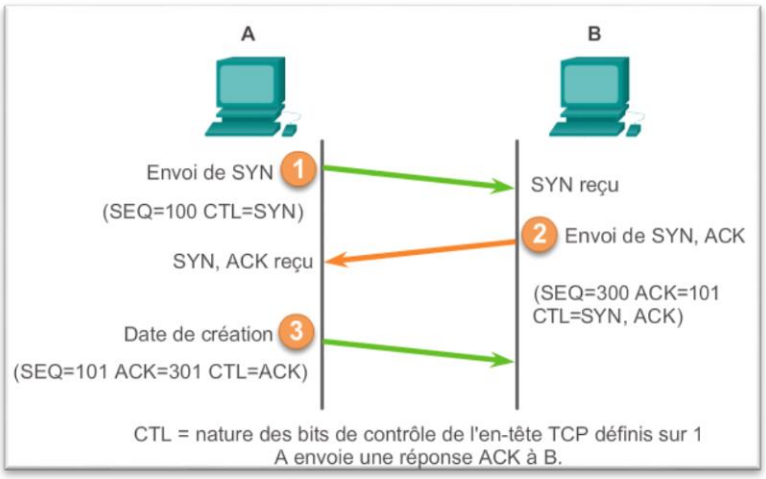


8 Analyse détaillée du protocole TCP

8.1 Etablissement d'une connexion TCP

La première étape de la connexion nécessite la synchronisation. La deuxième étape consiste à accuser réception de la requête de synchronisation initiale et à synchroniser les paramètres de connexion dans la direction opposée. La troisième étape de connexion consiste à envoyer un accusé réception indiquant à la destination que la connexion peut être établie des deux côtés.

Lorsque deux hôtes communiquent à l'aide du protocole TCP, une connexion est établie avant que les données ne puissent être échangées. Une fois la communication terminée, les sessions sont fermées et il est mis fin à la connexion. Les mécanismes de connexion et de session permettent l'activation de la fonction de fiabilité du protocole TCP. Consultez la figure ci-contre pour découvrir les étapes de l'établissement d'une connexion TCP.



Étape 1. Le client demande l'établissement d'une session de communication client-serveur avec le serveur (SYN)

Étape 2. Le serveur accuse réception de la session de communication client-serveur et demande l'établissement d'une session de communication serveur-client (SYN,ACK)

Étape 3. Le client accuse réception de la session de communication serveur

Pour comprendre le processus de connexion en trois étapes, examinez les différentes valeurs échangées par les deux hôtes. Dans l'en-tête du segment TCP se trouvent six champs de 1 bit contenant des informations de contrôle qui servent à gérer les processus TCP. Il s'agit des champs :

- URG : pointeur de données urgentes valide
- ACK : champ d'accusé de réception valide
- PSH : fonction de livraison des données sans attendre le remplissage des tampons (Push)
- RST : réinitialisation de la connexion
- SYN : synchronisation des numéros d'ordre
- FIN : arrêt de l'envoi de données par l'expéditeur

8.2 Fin de la communication TCP

Pour mettre fin à une connexion, l'indicateur de contrôle FIN (Finish) doit être défini dans l'en-tête de segment. Pour mettre fin à chaque session TCP unidirectionnelle, on utilise un échange en deux étapes, constitué d'un segment FIN et d'un segment ACK. Par conséquent, pour mettre fin à une seule conversation TCP, quatre échanges sont nécessaires pour mettre fin aux deux sessions.

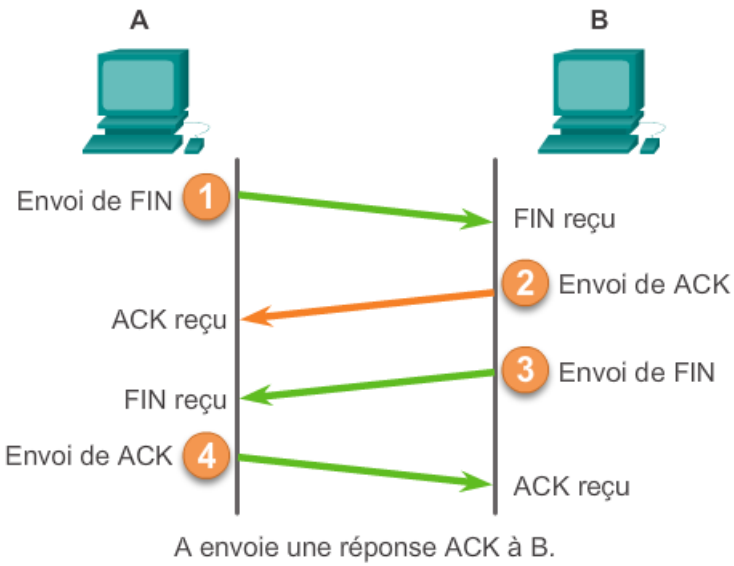
Remarque : les termes client et serveur sont utilisés ici pour simplifier l'explication, mais le processus d'interruption peut être initié par n'importe lequel des deux hôtes ayant une session ouverte :

Étape 1 : quand le client n'a plus de données à envoyer dans le flux, il envoie un segment dont l'indicateur FIN est défini.

Étape 2 : le serveur envoie un segment ACK pour informer de la bonne réception du segment FIN, afin de fermer la session du client au serveur.

Étape 3 : le serveur envoie un segment FIN au client pour mettre fin à la session du serveur au client.

Étape 4 : le client répond à l'aide d'un segment ACK pour accuser réception du segment FIN envoyé par le serveur.



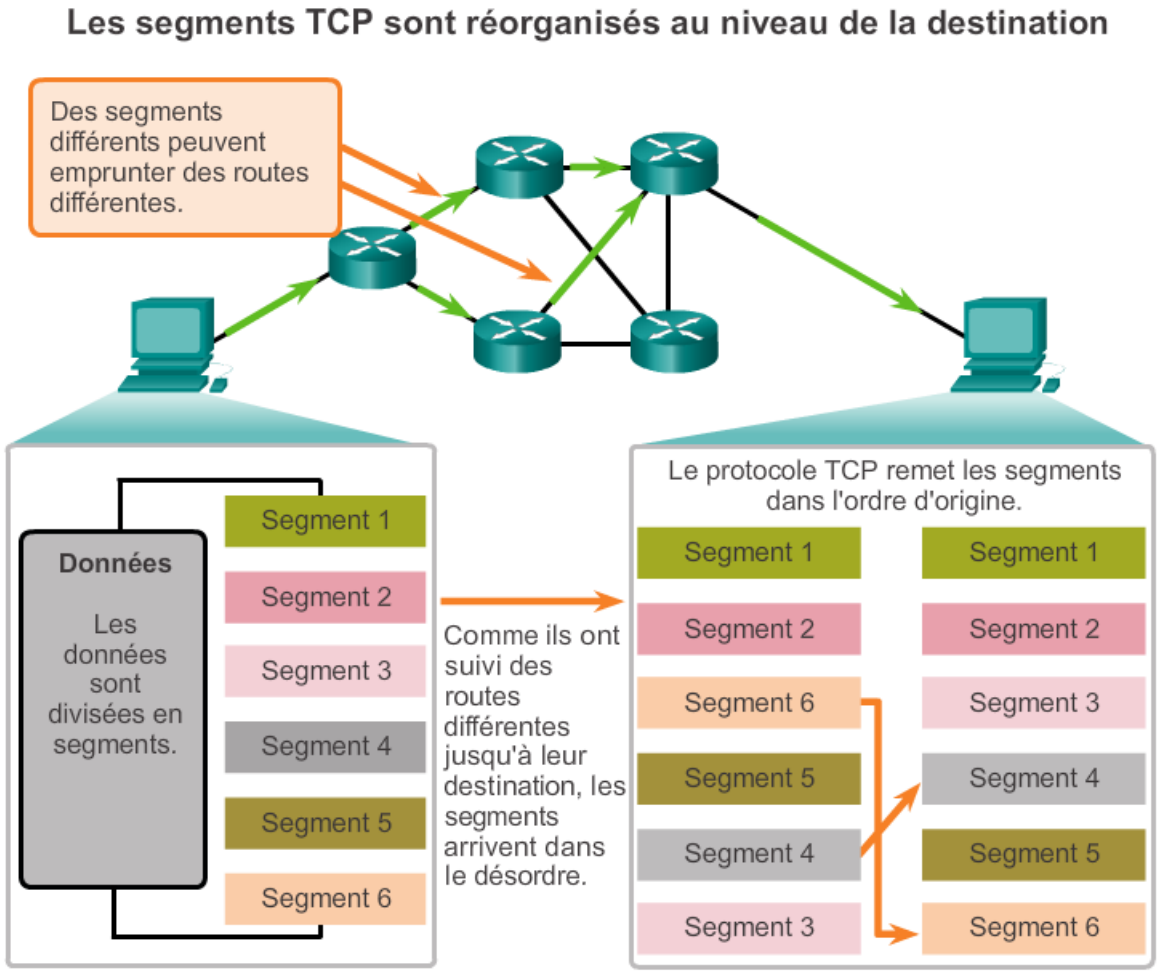
La session dans l'autre sens est fermée selon le même processus. Le récepteur indique qu'il n'y a plus de données à envoyer en définissant l'indicateur FIN dans l'en-tête d'un segment envoyé à la source. Un accusé de réception confirme que tous les octets de données ont été reçus et que cette session, à son tour, se ferme.

TP d'application avec wireshark

8.3 Réorganisation des segments TCP

Quand des services envoient des données à l'aide du protocole TCP, il arrive que les segments parviennent à destination dans le désordre. Pour que le destinataire puisse comprendre le message d'origine, il faut que les données contenues dans ces segments soient réagencées dans leur ordre d'origine. Pour cela, des numéros d'ordre sont affectés à l'en-tête de chaque paquet.

Lors de la configuration de la session, un numéro d'ordre initial, ou **ISN**, est défini. L'ISN représente la valeur de début des octets de cette session qui est transmise à l'application destinataire. Lors de la transmission des données pendant la session, le numéro d'ordre est incrémenté du nombre d'octets ayant été transmis. Ce suivi des octets de données permet d'identifier chaque segment et d'en accuser réception individuellement. Il est ainsi possible d'identifier les segments manquants.



Le processus TCP récepteur place les données d'un segment dans une mémoire tampon de réception. Les segments sont remis dans l'ordre correct et sont transmis à la couche application une fois qu'ils ont été réassemblés. Tous les segments reçus dont les numéros d'ordre ne sont pas contigus sont conservés en vue d'un traitement ultérieur. Ensuite, ces segments sont traités dans l'ordre quand les segments contenant les octets manquants sont reçus.

8.4 Confirmation de la réception des segments

L'une des fonctions du protocole TCP consiste à garantir que chaque segment atteigne sa destination. Les services TCP sur l'hôte de destination accusent réception des données reçues à l'application source.

Le numéro d'ordre (SEQ) et le numéro d'accusé de réception (ACK) sont utilisés ensemble pour confirmer la réception des octets de données contenus dans les segments envoyés. Le numéro SEQ indique le nombre relatif d'octets qui ont été transmis dans cette session, y compris les octets dans le segment actuel. Le protocole TCP utilise le numéro ACK renvoyé à la source pour indiquer l'octet suivant que le destinataire s'attend à recevoir. C'est ce que l'on appelle un accusé de réception prévisionnel.

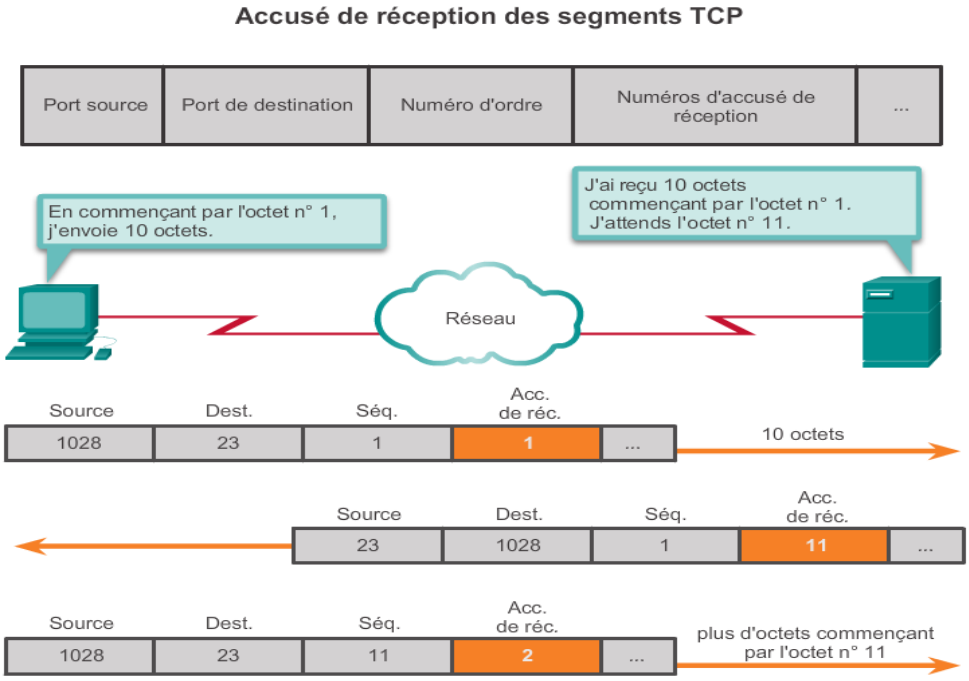
La source est informée que la destination a reçu tous les octets de ce flux de données jusqu'à l'octet indiqué par le numéro ACK, mais sans inclure ce dernier. L'hôte expéditeur est censé envoyer un segment qui utilise un numéro d'ordre égal au numéro ACK.

Souvenez-vous qu'en fait chaque connexion est composée de deux sessions unidirectionnelles. Les numéros SEQ et ACK sont échangés dans les deux sens.

Dans l'exemple de la figure, l'hôte de gauche envoie des données à l'hôte de droite. Il envoie un segment contenant 10 octets de données pour cette session et un numéro d'ordre égal à 1 dans l'en-tête. L'hôte récepteur reçoit le segment au niveau de la couche 4 et détermine que le numéro d'ordre est 1 et qu'il y a 10 octets de données. L'hôte renvoie alors un segment à l'hôte de gauche pour accuser la réception de ces données. Dans ce segment, l'hôte définit le numéro ACK sur 11 pour indiquer que le prochain octet de données qu'il prévoit de recevoir dans cette session est l'octet numéro 11. Quand l'hôte expéditeur reçoit cet accusé de réception, il peut envoyer le segment suivant contenant des données pour cette session commençant par l'octet numéro 11.

Dans notre exemple, si l'hôte expéditeur devait attendre un accusé de réception tous les 10 octets, le réseau subirait une forte surcharge. Pour réduire la surcharge due à ces accusés de réception, plusieurs segments de données peuvent être envoyés et faire l'objet d'un accusé de réception grâce à un seul message TCP en retour. Cet accusé de réception contient un numéro ACK basé sur le nombre total d'octets reçus dans la session. Prenons l'exemple d'un numéro d'ordre de début égal à 2000. Si 10 segments de 1 000 octets chacun étaient reçus, le numéro ACK 12001 serait renvoyé à la source.

La quantité de données qu'une source peut transmettre avant qu'un accusé de réception soit reçu est la « taille de fenêtre », qui est un champ de l'en-tête TCP qui permet de gérer les données perdues et le contrôle de flux.



8.5 Traitement des pertes de segments

Qu'un réseau soit bien conçu ou non, il arrive que des données se perdent. Par conséquent, le protocole TCP fournit des méthodes de gestion des pertes de segments. Parmi elles se trouve un mécanisme de retransmission des segments contenant des données sans accusé de réception.

En général, un service sur l'hôte de destination utilisant le protocole TCP ne génère d'accusé de réception que pour les séquences contiguës d'octets. Si un ou plusieurs segments sont manquants, seules les données dans la première séquence contiguë d'octets sont reconnues. Par exemple, si les segments avec des numéros allant de 1500 à 3000 et de 3400 à 3500 sont reçus, le numéro ACK est 3001. Cela est dû au fait qu'il existe des segments avec les numéros SEQ 3001 à 3399 qui n'ont pas été reçus.

Lorsque le protocole TCP sur l'hôte source ne reçoit pas d'accusé de réception après un délai prédéterminé, il revient au dernier numéro ACK reçu et retransmet les données à partir de ce point. Le processus de retransmission n'est pas spécifié par le document RFC, mais il incombe à l'implémentation particulière du protocole TCP de le déterminer.

Dans une implémentation TCP classique, un hôte peut transmettre un segment, placer une copie du segment dans une file d'attente de retransmission et lancer un minuteur. Quand l'accusé de réception des données est reçu, le segment est supprimé de la file d'attente. Si l'accusé de réception n'est pas reçu avant l'écoulement du délai prévu, le segment est retransmis.

Observez l'animation ci-dessous pour bien comprendre le mécanisme de traitement des pertes de segments.

8.6 Contrôle de flux

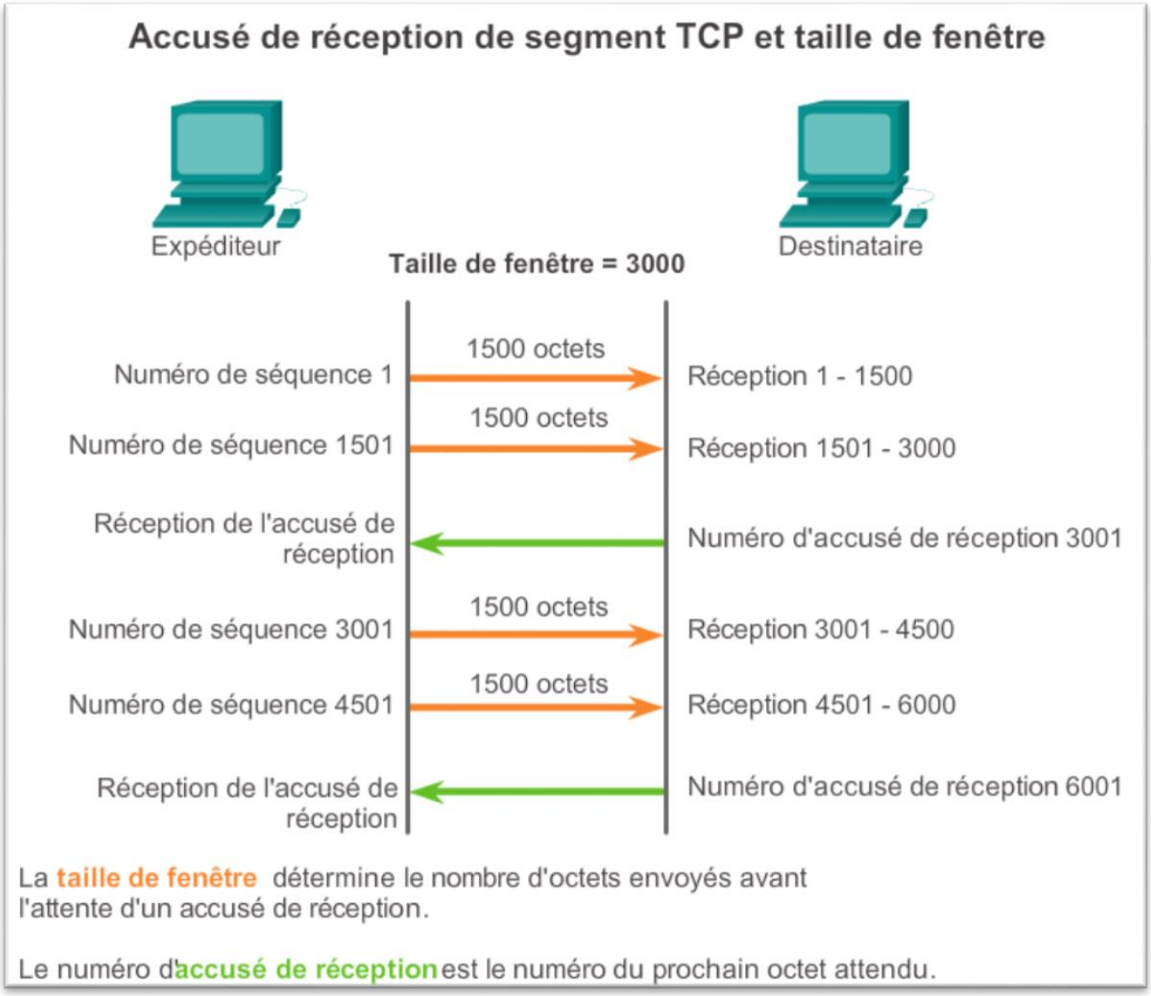
Le protocole TCP inclut également des mécanismes de contrôle de flux. Le contrôle de flux aide à maintenir la fiabilité des transmissions TCP en réglant le flux de données entre la source et la destination pour une session donnée. Le contrôle de flux consiste à limiter la quantité de données de segments transférées en une seule fois et à demander des accusés de réception avant de transmettre davantage de données.

Pour effectuer le contrôle de flux, la première chose que le protocole TCP doit déterminer est la quantité de données de segments que le périphérique de destination peut accepter. L'en-tête TCP comprend un champ de 16 bits appelé la taille de fenêtre. Il s'agit du nombre d'octets que le périphérique de destination d'une session TCP peut accepter et traiter en une seule fois. La taille de fenêtre initiale est convenue lors du démarrage de la session, via la connexion en trois étapes entre la source et la destination. Une fois cette taille approuvée, le périphérique source doit limiter la quantité de données de segments envoyées au périphérique de destination en fonction de la taille de fenêtre. Une fois que le périphérique source a reçu un accusé de réception l'informant que les segments de données ont été reçus, il peut continuer à envoyer des données pour la session.

Pendant le délai d'attente de l'accusé de réception, l'expéditeur n'envoie pas de segment supplémentaire. Quand le réseau est encombré ou que les ressources de l'hôte récepteur subissent une forte pression, le délai peut augmenter. Plus ce délai s'allonge, plus le taux de transmission effectif des données de cette session diminue. Le ralentissement de la transmission de données de chaque session permet de réduire les conflits d'utilisation des ressources sur le réseau et sur le périphérique de destination lorsque plusieurs sessions sont en cours.

La figure ci-desous présente une représentation simplifiée de la taille de fenêtre et des accusés de réception. Dans cet exemple, la taille de fenêtre initiale d'une session TCP représentée est définie à 3000 octets. Lorsque l'expéditeur a transmis 3 000 octets, il attend l'accusé de réception de ces octets avant de transmettre d'autres segments de cette session. Une fois que l'expéditeur a reçu l'accusé de réception provenant du destinataire, il peut transmettre 3 000 octets supplémentaires.

Le protocole TCP utilise les tailles de fenêtre pour maintenir le plus haut débit de transmission possible sur le réseau et le périphérique de destination, tout en réduisant la perte et les retransmissions de données.



9 Analyse détaillée du protocole UDP

Le protocole UDP est un protocole simple offrant des fonctions de couche transport de base. Il crée beaucoup moins de surcharge que le protocole TCP car il n'est pas orienté connexion et ne propose pas de mécanismes sophistiqués de fiabilité (retransmission, séquençage et contrôle de flux).

Cela ne signifie pas que les applications utilisant le protocole UDP ne sont jamais fiables, ni que le protocole UDP n'est pas efficace. Cela signifie simplement que ces fonctions ne sont pas fournies par le protocole de couche transport et qu'elles doivent être implémentées à un autre niveau, le cas échéant.

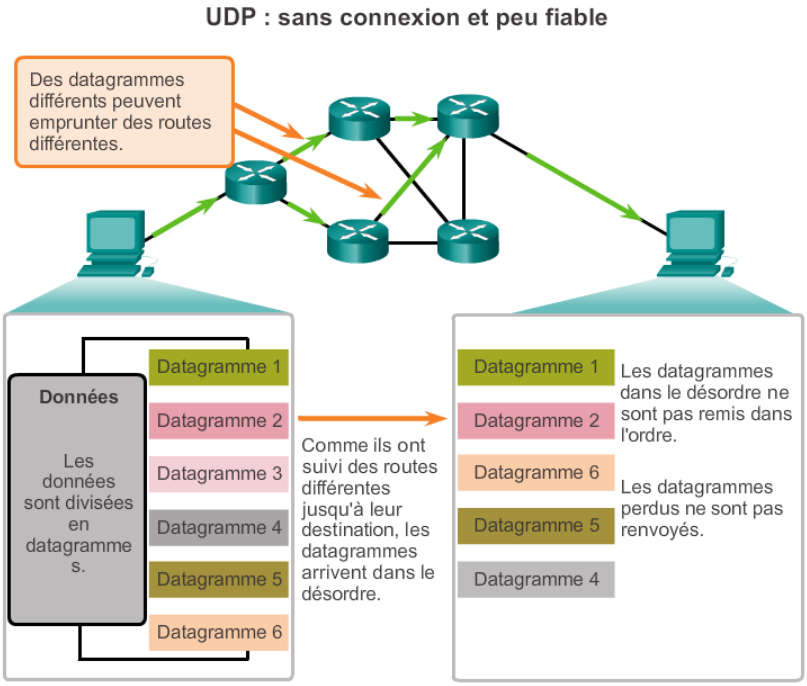
Bien que le volume total de trafic UDP d'un réseau standard soit relativement faible, des protocoles importants de couche application utilisent le protocole UDP, notamment :

- Système de noms de domaine (DNS)
- SNMP (Simple Network Management Protocol)
- Protocole DHCP (Dynamic Host Configuration Protocol)
- Protocole RIP (Routing Information Protocol)
- TFTP (Trivial File Transfer Protocol)
- Téléphonie IP ou voix sur IP (VoIP)
- Jeux en ligne

Certaines applications, comme les jeux en ligne ou la VoIP, peuvent tolérer la perte d'une certaine quantité de données. Si ces applications utilisaient le protocole TCP, elles risqueraient d'être confrontées à des retards importants lorsque le protocole TCP détecterait les pertes de données et retransmettrait les données. Ces délais seraient plus préjudiciables à l'application que la perte d'une petite quantité de données. Certaines applications, comme le système DNS, renvoient simplement la requête si aucune réponse n'est reçue. Par conséquent, elles n'ont pas besoin du protocole TCP pour garantir l'acheminement des messages.

La faible surcharge qu'engendre le protocole UDP rend celui-ci très intéressant pour de telles applications.

Comme le protocole UDP n'est pas orienté connexion, les sessions ne sont pas établies avant que la communication n'ait lieu comme c'est le cas avec le protocole TCP. On dit que le protocole UDP est basé sur les transactions : en d'autres termes, quand une application doit envoyer des données, elle les envoie tout simplement.



TP d'application avec Wireshark

TP de synthèse avec Wireshark