

Introduction oo	Transport des données oooooooooooooooo	Protocole TCP oooooooooooooooooooo	Protocole UDP ooooooo
--------------------	---	---------------------------------------	--------------------------

Chapitre 11 : Protocoles & Contrôle de flux

Couche Transport

Mohammed SABER

Département Électronique, Informatique et Télécommunications
 École Nationale des Sciences Appliquées "ENSA"
 Université Mohammed Premier OUJDA

Année Universitaire : 2021-2022



Mohammed SABER	ENSAO	Chapitre 11	AU-2021-2022	1 / 41
----------------	-------	-------------	--------------	--------

Introduction oo	Transport des données oooooooooooooooo	Protocole TCP oooooooooooooooooooo	Protocole UDP ooooooo
--------------------	---	---------------------------------------	--------------------------

Plan de chapitre

1 Introduction

2 Transport des données

3 Protocole TCP : Transmission Control Protocol (RFC 793 corrigée par RFC 1122 et 1323)

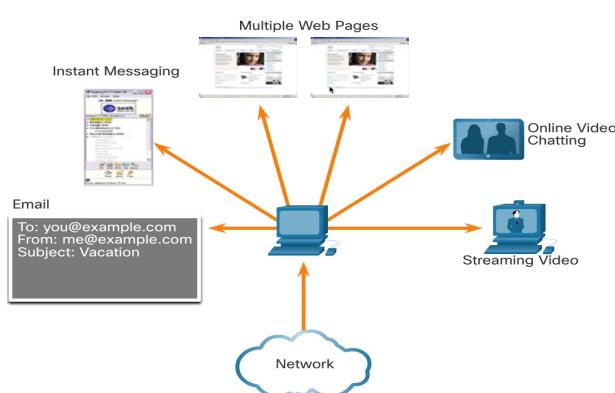
4 Protocole UDP : User Datagram Protocol (RFC 768)



Mohammed SABER	ENSAO	Chapitre 11	AU-2021-2022	2 / 41
----------------	-------	-------------	--------------	--------

Introduction

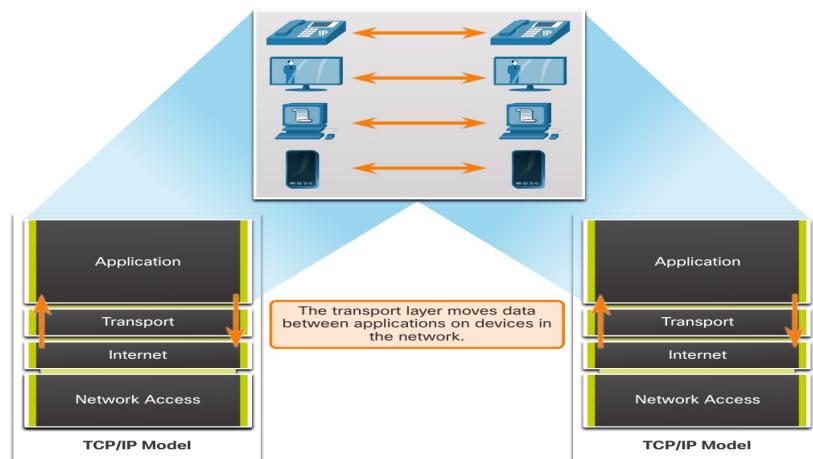
- Les réseaux de données et Internet renforcent le réseau humain en garantissant la fiabilité des communications entre les personnes.
 - Nous pouvons, sur un même périphérique, utiliser des services aussi divers que les messageries électroniques, le web ou les messageries instantanées pour envoyer des messages et recevoir des informations.



- Les données envoyées par ces applications sont empaquetées, transportées et livrées à l'application voulue sur le périphérique de destination.
 - Les processus décrits dans la couche transport acceptent des données provenant de la couche application et les préparent pour les adresser à la couche réseau.
 - Un ordinateur source communique avec un ordinateur destinataire pour décider de la méthode de division des données en segments, de la méthode permettant de s'assurer qu'aucun des segments n'est perdu et de la méthode de vérification permettant de savoir si tous les segments sont arrivés.
 - Pour comprendre la couche transport, pensez à un service des expéditions qui prépare une seule commande composée de plusieurs colis à livrer.

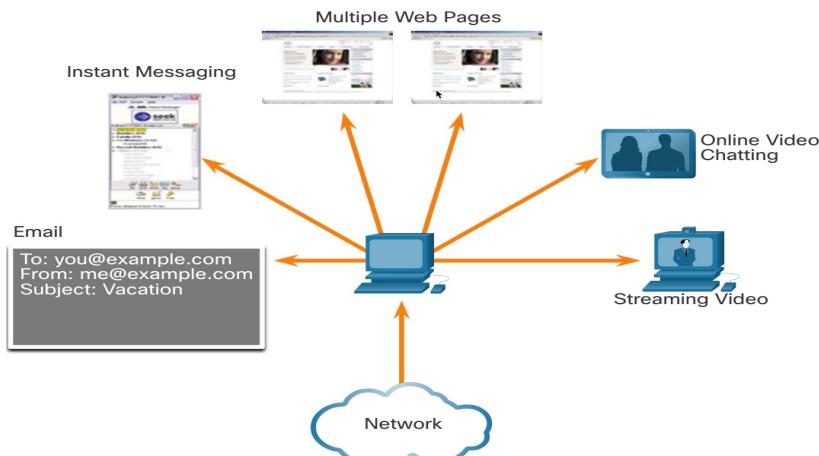
Transport des données

- La couche transport est chargée de l'établissement d'une session de communication temporaire entre deux applications et de l'acheminement des données entre ces deux applications.
 - Une application génère des données qui sont envoyées d'une application située sur un hôte source à une autre application située sur un hôte de destination.
 - Sans se soucier du type de l'hôte de destination (PC, SmartPhone, ...), du type de support que les données doivent emprunter (fil, fibre, sans-fil WIFI, ...), du chemin suivi par ces données, de l'encombrement de la liaison ni de la taille du réseau.
 - La couche transport constitue la liaison entre la couche application et les couches inférieures chargées de la transmission sur le réseau.



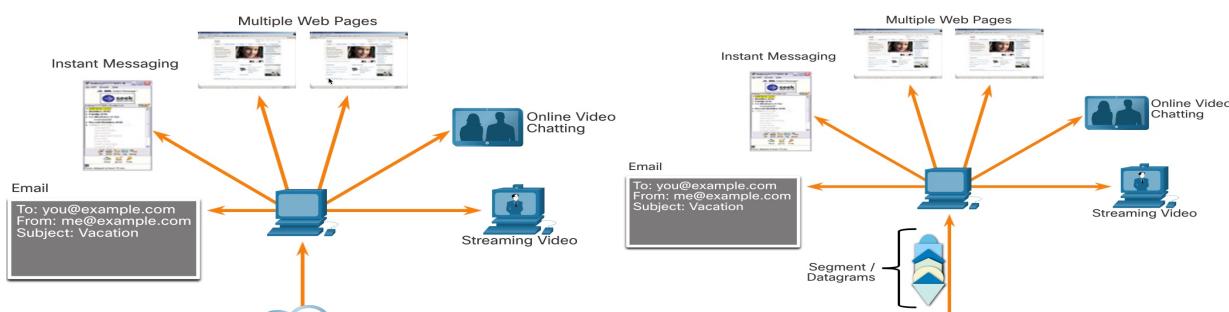
Suivi des conversations individuelles

- Chaque ensemble de données transitant entre une application source et une application de destination est appelé une **conversation**.
 - Un hôte peut héberger plusieurs applications qui communiquent sur le réseau simultanément.
 - Chacune de ces applications communique avec une ou plusieurs applications sur un ou plusieurs hôtes distants.
 - La couche transport est chargée de garantir ces multiples conversations et d'en effectuer le suivi.



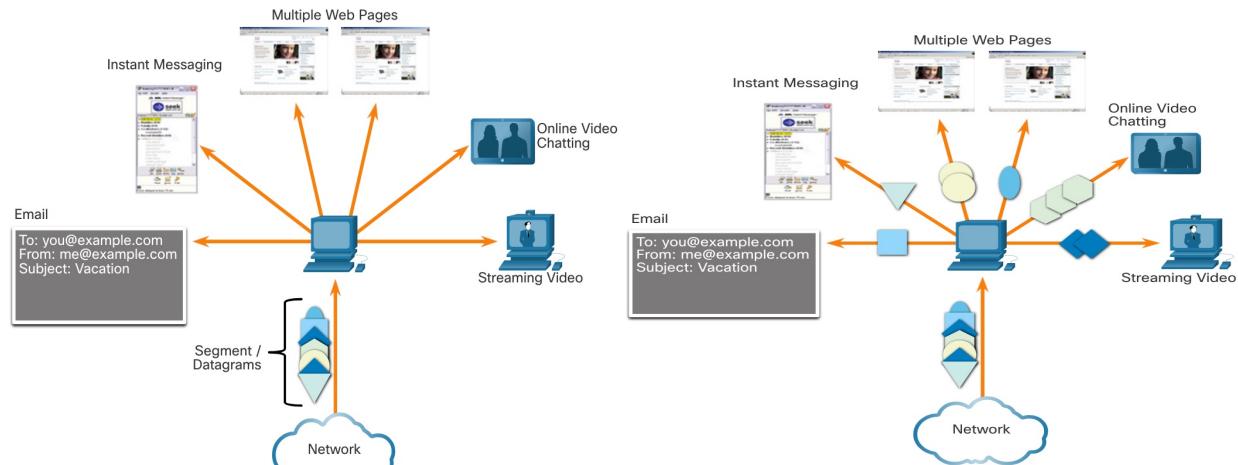
Segmentation des données et reconstitution des segments

- Les données doivent être préparées à être envoyées sur le support sous forme de blocs faciles à gérer.
 - La plupart des réseaux limitent la quantité de données pouvant être incluses dans un paquet.
 - Les protocoles de transport disposent de services qui segmentent les données d'application en blocs (**segments, datagrammes**) de taille appropriée.
 - Un en-tête est utilisé pour la réorganisation, est ajouté à chaque bloc de données, est utilisé pour suivre le flux de données.
 - Au destinataire, la couche transport doit pouvoir reconstituer un flux de données complet et utilisable par la couche application, à partir des blocs de données.



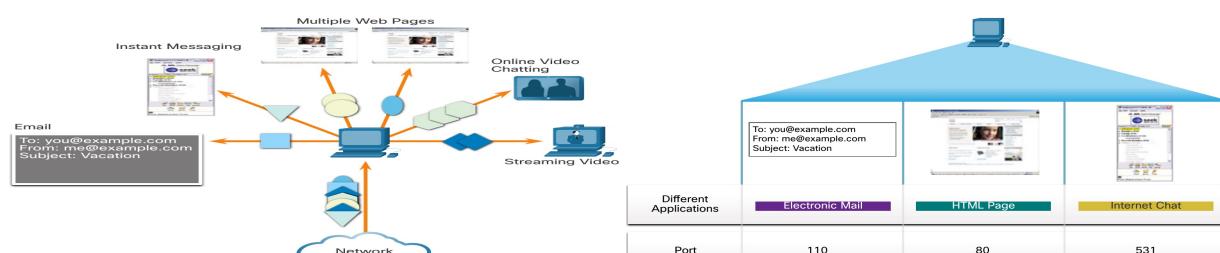
Identification des applications

- Pour que les flux de données atteignent les applications auxquelles ils sont destinés, la couche transport doit identifier l'application cible.
- La couche transport garantit que même si plusieurs applications sont exécutées sur un même appareil, toutes les applications reçoivent les données correctes.
- Les protocoles de transport gèrent ces conversations simultanées multiples au moyen de champs d'en-tête identifiant ces applications de façon unique.



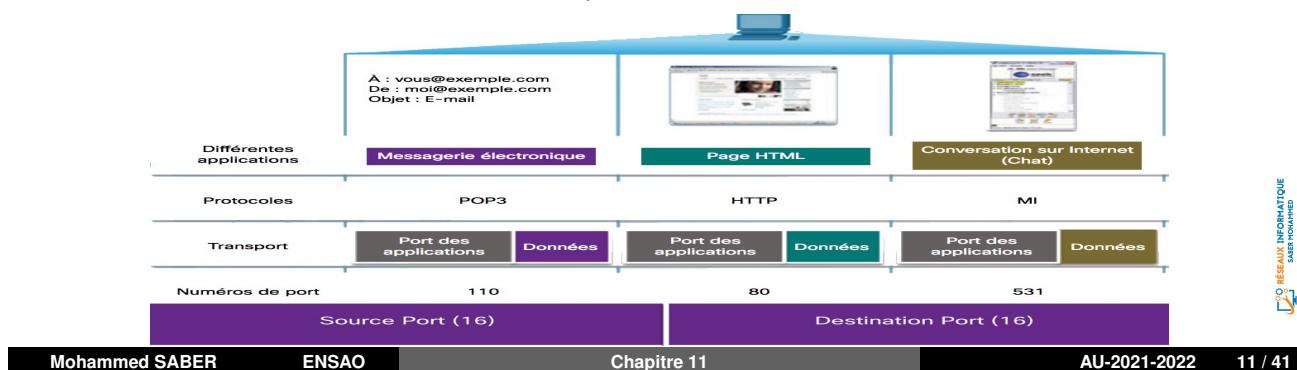
Gestion des conversations simultanées

- La couche transport doit pouvoir segmenter et gérer plusieurs communications ayant des exigences différentes en matière de transport.
- Les utilisateurs s'attendent à pouvoir recevoir et envoyer simultanément des e-mails et des messages instantanés, afficher des sites web et passer un appel téléphonique par voix sur IP (VoIP).
- Chacune de ces applications envoie des données sur le réseau et en reçoit simultanément, et ce, malgré leurs différents besoins en termes de fiabilité.
- De plus, les données de l'appel téléphonique ne sont pas dirigées vers le navigateur web et le texte des messages instantanés n'apparaît pas dans un e-mail.
- Pour gérer les conversations simultanées multiples, les protocoles de transport utilisent un identifiant unique pour chacune des applications. Ce identificateur unique est le **numéro de port**.



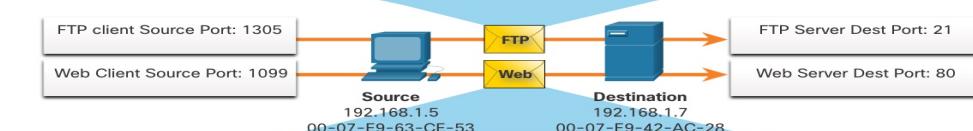
Numéros de port

- Le **numéro du port source** est associé à l'application d'origine sur l'hôte local.
 - Le numéro de port source est généré de manière dynamique par le périphérique émetteur pour identifier une conversation entre deux périphériques.
 - Ainsi, plusieurs conversations peuvent s'effectuer simultanément.
 - Un périphérique peut envoyer plusieurs requêtes de service HTTP à un serveur web en même temps.
 - Un suivi des différentes conversations HTTP est effectué sur la base des ports sources.
 - Le **numéro du port de destination** est associé à l'application de destination sur l'hôte distant.
 - Le client place un numéro de port de destination dans le segment pour informer le serveur de destination du service demandé.
 - Par exemple, 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 offrir plusieurs services simultanés, comme des services web sur le port 80 et une connexion FTP sur le port 21.



Numéros de port - Socket (Interface) de connexion

- 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 de l'adresse IP source et du numéro de port source, ou de l'adresse IP de destination et du numéro de port de destination, est appelée **interface de connexion** ou **socket de connexion**.



Numéros de port

Ports TCP	
Plage de numéros de port	Groupe de ports
0 à 1023	Ports réservés
De 1024 à 49151	Ports inscrits
49152 à 65535	Ports dynamiques et/ou privés

Ports UDP	
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)
5060	SIP (VoIP)

Ports UDP réservés:

69	TFTP
520	RIP

Légende

PortsTCP 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 communs aux protocoles TCP et UDP

Plage de numéros de port	Groupe de ports
0 à 1023	Ports réservés
De 1024 à 49151	Ports inscrits
49152 à 65535	Ports dynamiques et/ou privés

Légende

Ports TCP/UDP inscrits courants:

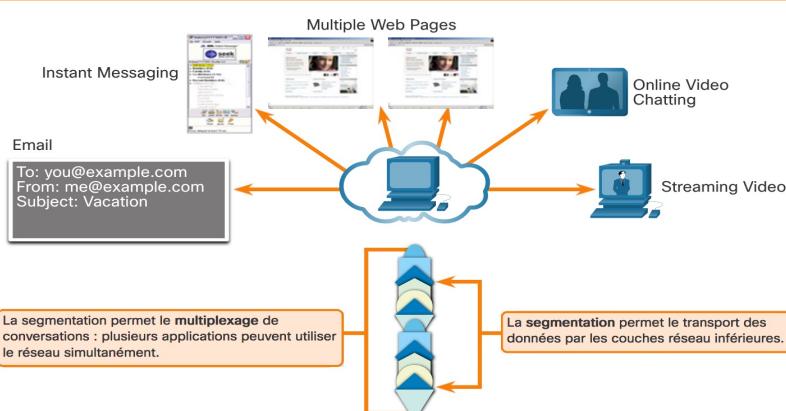
1433	MS SQL
2948	WAP (MMS)

Ports TCP/UDP réservés courants:

53	DNS
161	SNMP
531	AOL Instant Messenger, IRC

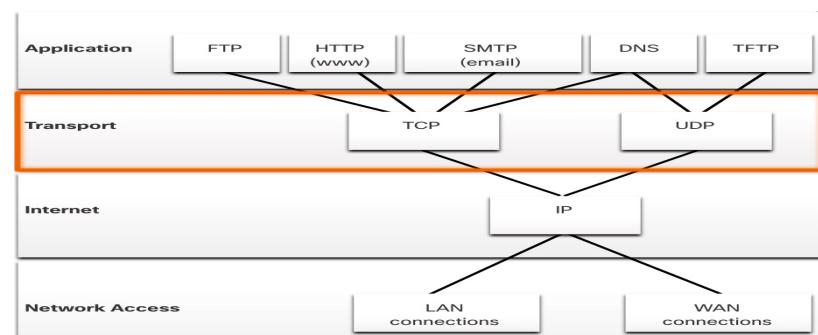
Multiplexage de conversations

- 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 nécessiter d'utiliser toute la bande passante disponible.
 - De fait, cela empêche d'autres communications d'avoir lieu en même temps.
 - En outre, cela rend également difficiles la reprise sur erreur et la retransmission des données endommagées.
 - La segmentation des données en éléments plus petits permet à plusieurs communications différentes, provenant de nombreux utilisateurs, d'être **multiplexées** (imbriquées) sur le même réseau.



Multiplexage de conversations

- 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.
 - 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**.
 - Le protocole IP utilise ces protocoles de transport pour permettre aux hôtes de communiquer et de transmettre des données.

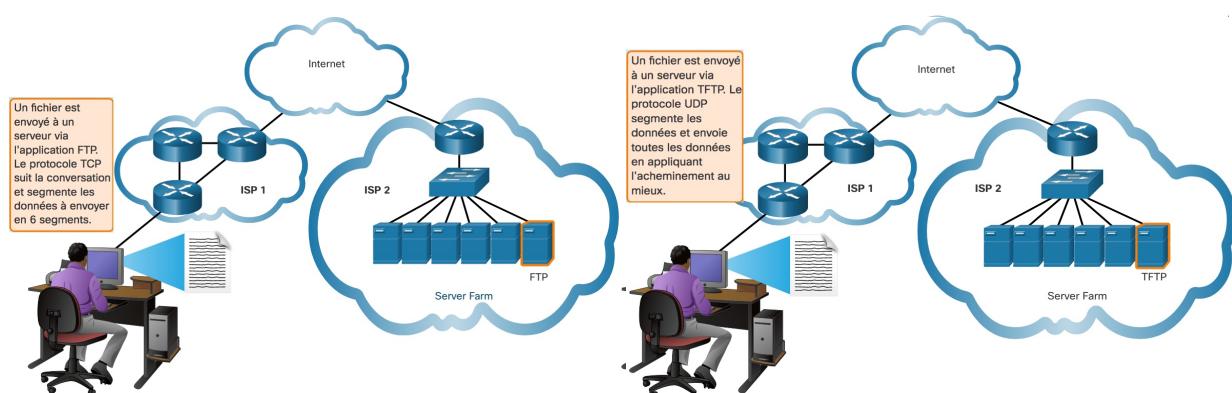


Protocole TCP

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

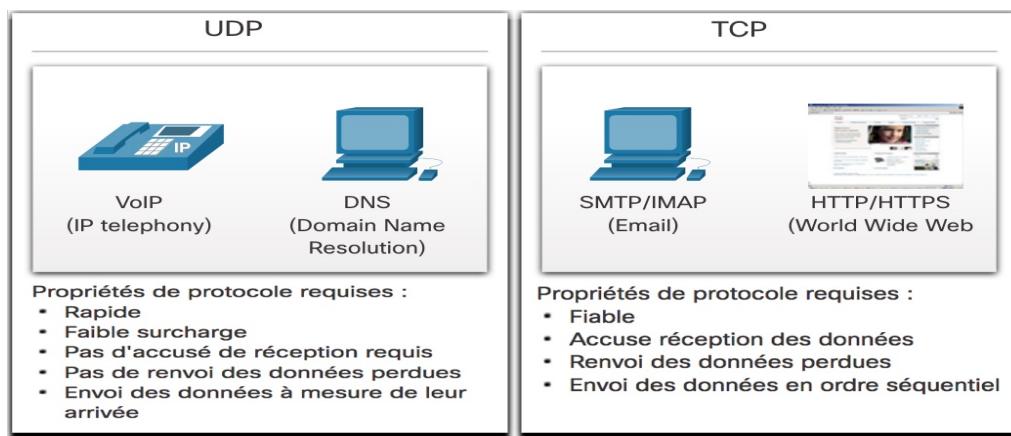
Protocole UDP

est un protocole de couche transport plus simple, qui ne permet pas de garantir la fiabilité du transport.



- Pour certaines applications, les segments doivent arriver dans un ordre donné pour être traités correctement.
 - Pour d'autres applications, toutes les données doivent être entièrement reçues pour être utilisées.
 - Dans les deux cas, le protocole TCP est utilisé comme protocole de transport.
 - Les développeurs d'applications doivent déterminer quel type de protocole de transport est approprié en fonction des exigences des applications.
 - Par exemple, les applications telles que les bases de données, les navigateurs web et les clients de messagerie ont besoin que toutes les données envoyées arrivent à destination dans leur état d'origine. Toute donnée manquante risque de corrompre la communication en la rendant incomplète ou illisible. Ces applications sont conçues pour utiliser le protocole TCP.
 - Dans d'autres cas, une application peut tolérer une certaine perte lors de la transmission de données sur le réseau, mais pas les retards de transmission.
 - Le protocole UDP est alors le choix idéal pour ces applications, car il implique moins de surcharge sur le réseau.
 - Le protocole UDP est à préférer, notamment pour la lecture audio en continu, la vidéo et la voix sur IP (VoIP). Les accusés de réception et la retransmission ralentissent la livraison.
 - Par exemple, si au cours d'une lecture vidéo, un ou deux segments n'arrivent pas, cela entraîne une interruption momentanée du flux. Cela peut se traduire par une distorsion de l'image ou du son, que l'utilisateur ne remarque peut-être même pas. Si par contre le périphérique de destination doit tenir compte de la perte de données, il se peut que le flux soit retardé à cause des retransmissions qu'il attend, causant ainsi une forte dégradation de la qualité de l'image ou du son.

- Les applications qui diffusent du contenu audio et vidéo enregistré utilisent le protocole TCP.
 - Par exemple, si votre réseau ne peut soudainement plus prendre en charge la bande passante nécessaire pour regarder un film à la demande, l'application interrompt la lecture. Pendant cette interruption, un message de mise en tampon peut apparaître tandis que le protocole TCP s'efforce de rétablir le flux de données. Une fois tous les segments en ordre et un niveau minimum de bande passante à nouveau atteint, votre session TCP se rétablit et la lecture du film peut reprendre.



- Pour comprendre les différences entre les protocoles TCP et UDP, il est important de comprendre comment chaque protocole utilise des fonctions spécifiques de fiabilité et comment il effectue le suivi des conversations.

Protocole TCP

Établissement d'une session

- TCP est un protocole orienté connexion.
- Un protocole orienté connexion est un protocole qui négocie et établit une connexion permanente (ou session) entre la source et la destination avant de transmettre du trafic.
- Grâce à l'établissement de la session, les périphériques négocient la quantité de trafic pouvant être transmise à un moment donné et les données de communication peuvent être étroitement gérées.

Livraison dans le même ordre

- Les réseaux peuvent fournir plusieurs routes dont les débits de transmission varient, il se peut que les données arrivent dans le désordre.
- En numérotant et en ordonnant les segments, le protocole TCP s'assure que ces segments sont remis dans le bon ordre.

Contrôle de flux

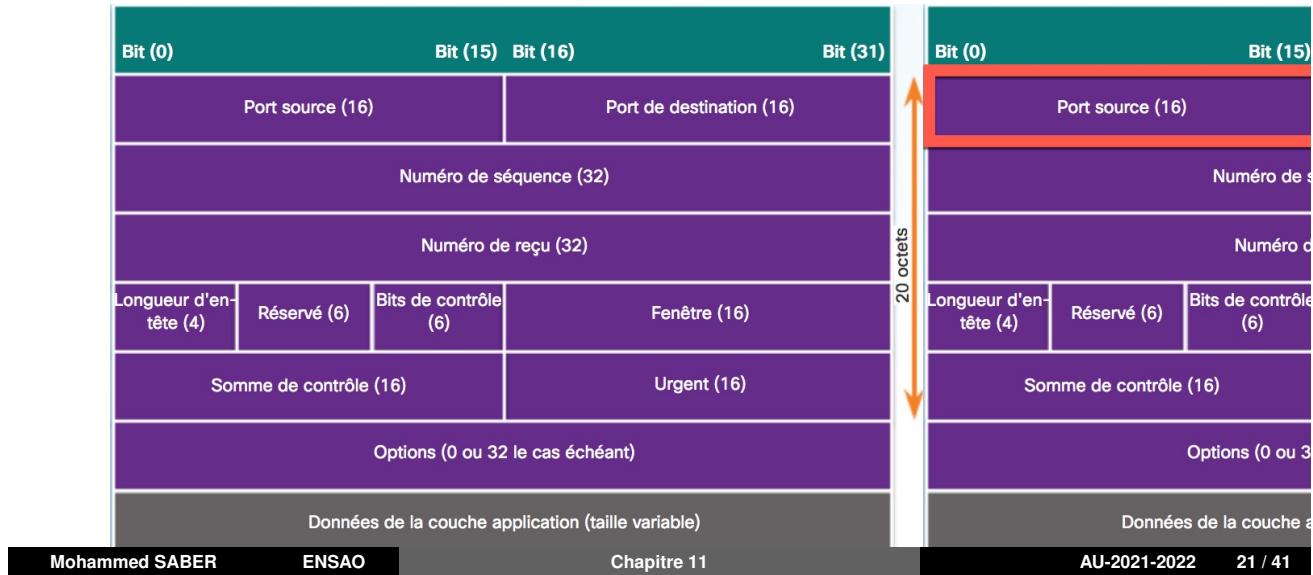
- Permet de s'assurer que le récepteur est capable de traiter les données reçues.
- Cette opération consiste à réguler la quantité de données transmises par la source.
- Le contrôle du flux contribue à rendre inutile la retransmission des données lorsque les ressources de l'hôte de réception sont saturées.

Acheminement fiable

- La fiabilité consiste à veiller à ce que chaque segment envoyé par la source parvienne à destination.
- Signifie que les segments perdus sont renvoyés afin que les données soient reçues dans leur intégralité.

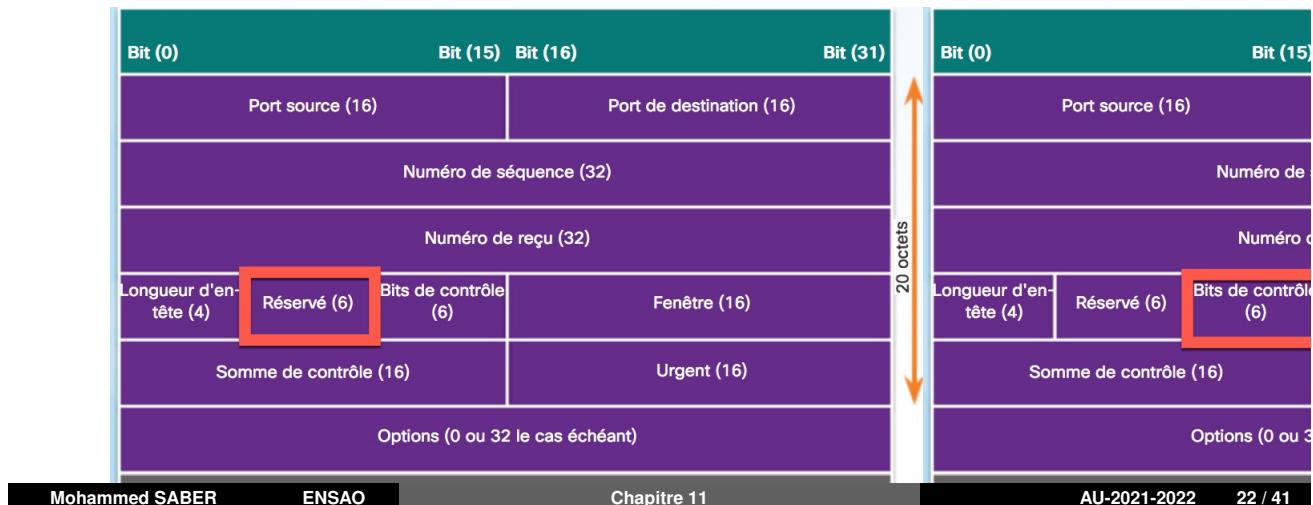
Chaque segment TCP utilise 20 octets de surcharge dans l'en-tête pour encapsuler les données de la couche application :

- **Port source (16 bits) et port de destination (16 bits)** : utilisés pour identifier l'application.
 - **Numéro d'ordre (32 bits)** : utilisé pour réorganiser les données.
 - **Numéro d'accusé de réception (32 bits)** : indique que les données ont été reçues et que le prochain
 - **Longueur d'en-tête (4 bits)** : indique la longueur de l'en-tête du segment TCP.



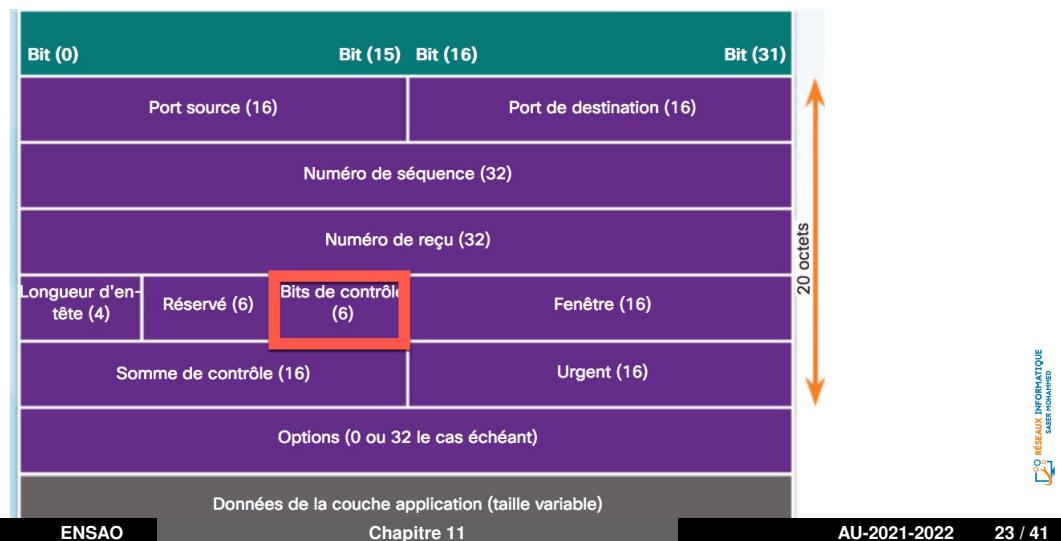
Chaque segment TCP utilise 20 octets de surcharge dans l'en-tête pour encapsuler les données de la couche application (suite) :

- **Réserve (6 bits)** : champ réservé pour les futures évolutions.
 - **Bits de contrôle (6 bits)** : comprennent des codes de bits, ou indicateurs, indiquant l'objectif et la fonction du segment TCP.
 - **Taille de fenêtre (16 bits)** : indique le nombre de segments pouvant être acceptés en même temps.
 - **Somme de contrôle (16 bits)** : utilisée pour le contrôle des erreurs dans l'en-tête et les données de segment.
 - **Urgent (16 bits)** : indique si les données sont urgentes.



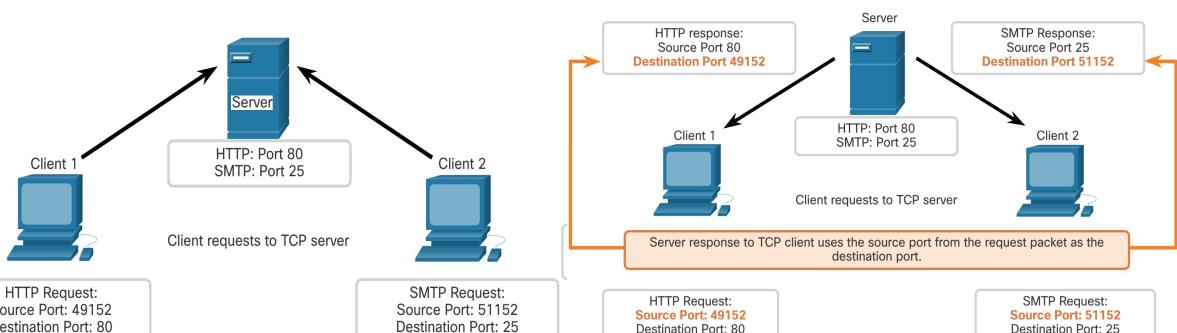
Bits de contrôle (6 bits) : 6 Bits pour définir la fonction des message et la validité de certains champs :

- **ACK** : Indique la validité du champ acquittement.
- **SYN** : Indique l'ouverture d'une connexion.
- **FIN** : Fermeture normale d'une connexion.
- **RST (Reset)** : Réinitialisation de la connexion suite à une erreur irrécupérable.
- **URG** : Message urgent. Les données doivent être traitées sans attendre que le récepteur ait traité les octets envoyés dans le flux.
- **PSH** : Les données reçues doivent être immédiatement remises à la couche supérieure.



Processus serveur TCP

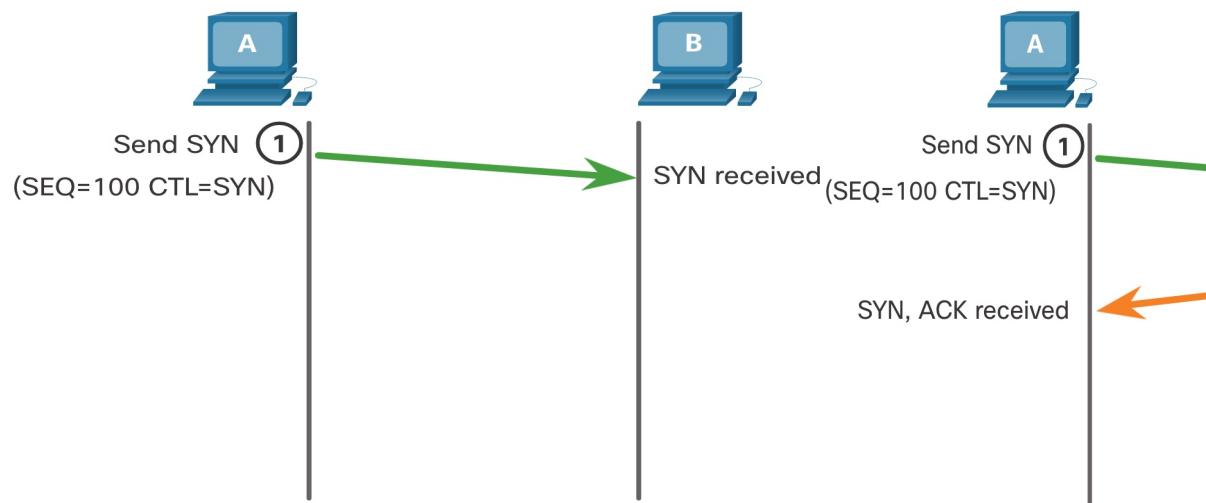
- Chaque processus applicatif qui s'exécute sur le serveur est configuré par défaut, ou manuellement par un administrateur système, pour utiliser un numéro de port.
- Deux services ne peuvent pas être affectés au même numéro de port d'un serveur particulier au sein des mêmes services de la couche transport.
- Une application de serveur active affectée à un port spécifique est considérée comme étant ouverte, ce qui signifie que la couche transport accepte et traite les segments adressés à ce port.
- Toute demande entrante d'un client qui est adressée à l'interface de connexion correcte est acceptée et les données sont transmises à l'application de serveur.
- De nombreux ports peuvent être ouverts simultanément sur un serveur, chacun étant destiné à une application de serveur active.



Établissement d'une connexion TCP

Une connexion TCP s'établit en trois étapes :

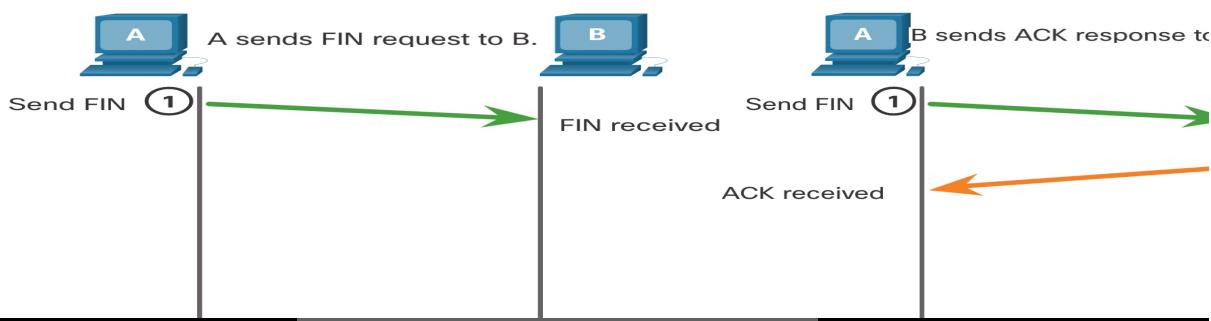
- **Étape 1** : le client demande l'établissement d'une session de communication client-serveur avec le serveur.
 - **Étape 2** : le serveur accuse réception de la session de communication client-serveur et demande l'établissement d'une session de communication serveur-client.
 - **Étape 3** : le client accuse réception de la session de communication serveur-client.



No.	Time	Source	Destination	Protocol	Info
10	16.303490	10.1.1.1	192.168.254.254	TCP	kiosk > http [SYN] Seq=0 Win=16
11	16.304896	192.168.254.254	10.1.1.1	TCP	http > kiosk [SYN, ACK] Seq=1 Win=16
12	16.304925	10.1.1.1	192.168.254.254	TCP	kiosk > http [ACK] Seq=1 Win=16
13	16.305153	10.1.1.1	192.168.254.254	HTTP	GET / HTTP/1.1
14	16.307875	192.168.254.254	10.1.1.1	TCP	http > kiosk [ACK] Seq=1 Win=16
Frame 10: 62 bytes on wire (496 bits), 62 bytes captured (496 bits)					
Ethernet II, Src: vmware_be:62:88 (00:50:56:be:62:88), Dst: Cisco_63:74:a0 (00:0f:24:63:74:a0)					
Internet Protocol Version 4, Src: 10.1.1.1 (10.1.1.1), Dst: 192.168.254.254 (192.168.254.254)					
Transmission Control Protocol, Src Port: kiosk (1061), Dst Port: http (80), Seq: 0, Len: 62					
Source port: kiosk (1061)					
Destination port: http (80)					
[Stream index: 0]					
Sequence number: 0 (relative sequence number)					
Header length: 28 bytes					
Flags: 0x02 (SYN)					
000 = Reserved: Not set					
...0 = Nonce: Not set					
...0 = Congestion Window Reduced (CWR): Not set					
...0 = ECN-Echo: Not set					
...0 = Urgent: Not set					
...0 = Acknowledgement: Not set					
...0 = Push: Not set					
...0 = Reset: Not set					
...1 = Sync: Set					
...0 = Fin: Not set					
Frame 11: 62 bytes on wire (496 bits), 62 bytes captured (496 bits)					
Ethernet II, Src: Cisco_63:74:a0 (00:0f:24:63:74:a0), Dst: vmware_be:62:88 (00:50:56:be:62:88)					
Internet Protocol Version 4, Src: 192.168.254.254 (192.168.254.254), Dst: 10.1.1.1 (10.1.1.1)					
Transmission Control Protocol, Src Port: http (80), Dst Port: kiosk (1061), Seq: 0, Ack: 1					
Source port: http (80)					
Destination port: kiosk (1061)					
[Stream index: 0]					
Sequence number: 0 (relative sequence number)					
Acknowledgement number: 1 (relative ack number)					
Header length: 28 bytes					
Flags: 0x12 (SYN, ACK)					
000 = Reserved: Not set					
...0 = Nonce: Not set					
...0 = Congestion Window Reduced (CWR): Not set					
...0 = ECN-Echo: Not set					
...0 = Urgent: Not set					
...1 = Acknowledgement: Set					
...0 = Push: Not set					
...0 = Reset: Not set					
...0 = Sync: Not set					
Frame 12: 54 bytes on wire (432 bits), 54 bytes captured (432 bits)					
Ethernet II, Src: vmware_be:62:88 (00:50:56:be:62:88), Dst: Cisco_63:74:a0 (00:0f:24:63:74:a0)					
Internet Protocol Version 4, Src: 10.1.1.1 (10.1.1.1), Dst: 192.168.254.254 (192.168.254.254)					
Transmission Control Protocol, Src Port: kiosk (1061), Dst Port: http (80), Seq: 1, Ack: 1					
Source port: kiosk (1061)					
Destination port: http (80)					
[Stream index: 0]					
Sequence number: 1 (relative sequence number)					
Acknowledgement number: 1 (relative ack number)					
Header length: 20 bytes					
Flags: 0x10 (ACK)					
000 = Reserved: Not set					
...0 = Nonce: Not set					
...0 = Congestion Window Reduced (CWR): Not set					
...0 = ECN-Echo: Not set					
...1 = Acknowledgement: Set					
...0 = Push: Not set					
...0 = Reset: Not set					
...0 = Sync: Not set					
Frame 13: 54 bytes on wire (432 bits), 54 bytes captured (432 bits)					
Ethernet II, Src: vmware_be:62:88 (00:50:56:be:62:88), Dst: Cisco_63:74:a0 (00:0f:24:63:74:a0)					
Internet Protocol Version 4, Src: 10.1.1.1 (10.1.1.1), Dst: 192.168.254.254 (192.168.254.254)					
Transmission Control Protocol, Src Port: kiosk (1061), Dst Port: http (80), Seq: 1, Ack: 1					
Source port: kiosk (1061)					
Destination port: http (80)					
[Stream index: 0]					
Sequence number: 1 (relative sequence number)					
Acknowledgement number: 1 (relative ack number)					
Header length: 20 bytes					
Flags: 0x10 (ACK)					
000 = Reserved: Not set					
...0 = Nonce: Not set					
...0 = Congestion Window Reduced (CWR): Not set					
...0 = ECN-Echo: Not set					
...0 = Urgent: Not set					
...0 = Acknowledgement: Set					
...0 = Push: Not set					
...0 = Reset: Not set					
...0 = Sync: Not set					

Fermeture d'une session 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**.
 - Pour mettre fin à une seule conversation TCP, quatre échanges sont nécessaires pour mettre fin aux deux sessions :
 - **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 indiquer la bonne réception du segment FIN afin de fermer la session du client au serveur.
 - **Étape 4** : le serveur envoie un segment FIN au client pour mettre fin à la session du serveur au client.
 - **Étape 5** : le client répond à l'aide d'un segment ACK pour accuser réception du segment FIN envoyé par le serveur.

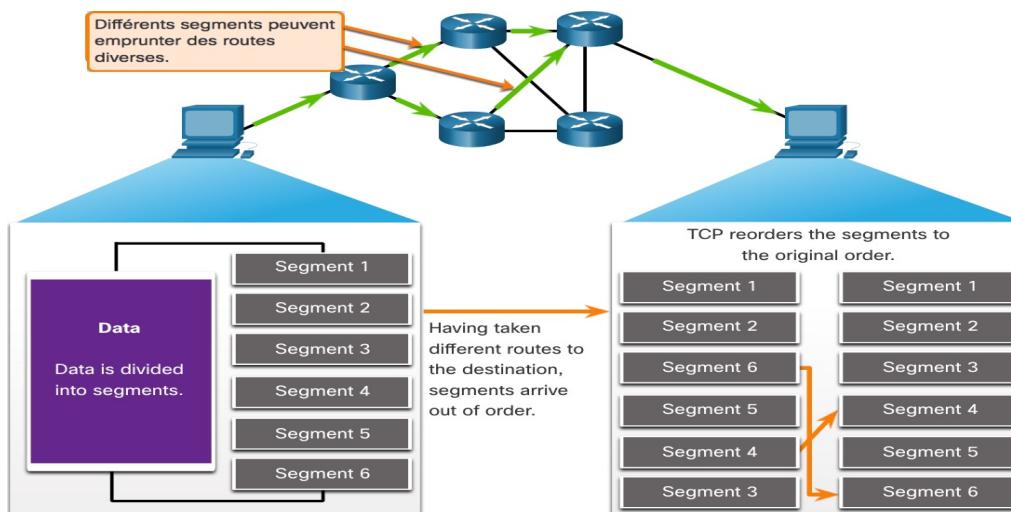


Fermeture d'une session TCP

No.	Time	Source	Destination	Protocol	Info
15	16.308976	192.168.254.254	10.1.1.1	HTTP	HTTP/1.1 304 Not Modified
16	16.309088	192.168.254.254	10.1.1.1	TCP	kiosk > http [FIN, ACK] Seq=374 Ack=146
17	16.309140	10.1.1.1	192.168.254.254	TCP	kiosk > http [ACK] Seq=374 Ack=146
18	16.309268	10.1.1.1	192.168.254.254	TCP	kiosk > http [FIN, ACK] Seq=374 Ack=146
19	16.310277	192.168.254.254	10.1.1.1	TCP	http > kiosk [ACK] Seq=146
# Frame 16: 60 bytes on wire (480 bits), 60 bytes captured (480 bits)					
# Ethernet II, Src: Cisco_63:74:a0 (00:0f:24:63:74:a0), Dst: VMware_be:62:88 (00:50:56:be:62:88)					
# Internet Protocol Version 4, Src: 192.168.254.254 (192.168.254.254), Dst: 10.1.1.1 (10.1.1.1)					
Transmission Control Protocol, Src Port: http (80), Dst Port: kiosk (1061), Seq: 145, Ack: 146, Flags: 0x10 (FIN, ACK)					
Source port: http (80)					
Destination port: kiosk (1061)					
[Stream index: 0]					
Sequence number: 145 (relative sequence number)					
Acknowledgement number: 374 (relative ack number)					
Header length: 20 bytes					
Flags: 0x10 (FIN, ACK)					
000..... = Reserved: Not set					
...0..... = Nonce: Not set					
....0.... = Congestion Window Reduced (CWR): Not set					
....0.... = ECN-Echo: Not set					
....0.... = Urgent: Not set					
....1.... = Acknowledgement: Set					
....0... = Push: Not set					
....0... = Reset: Not set					
....0... = Syn: Not set					
....1.... = Fin: Set					
window size value: 64096					
# Frame 17: 54 bytes on wire (432 bits), 54 bytes captured (432 bits)					
# Ethernet II, Src: Cisco_63:74:a0 (00:0f:24:63:74:a0), Dst: Cisco_63:74:a0 (00:0f:24:63:74:a0)					
# Internet Protocol Version 4, Src: 10.1.1.1 (10.1.1.1), Dst: 192.168.254.254 (192.168.254.254)					
Transmission Control Protocol, Src Port: kiosk (1061), Dst Port: http (80), Seq: 374, Ack: 146, Flags: 0x0 (SYN)					
Source port: kiosk (1061)					
Destination port: http (80)					
[Stream index: 0]					
Sequence number: 374 (relative sequence number)					
Acknowledgement number: 146 (relative ack number)					
Header length: 20 bytes					
Flags: 0x10 (ACK)					
000..... = Reserved: Not set					
...0..... = Nonce: Not set					
....0.... = Congestion Window Reduced (CWR): Not set					
....0.... = ECN-Echo: Not set					
....0.... = Urgent: Not set					
....1.... = Acknowledgement: Set					
....0... = Push: Not set					
....0... = Reset: Not set					
....0... = Syn: Not set					
....1.... = Fin: Not set					

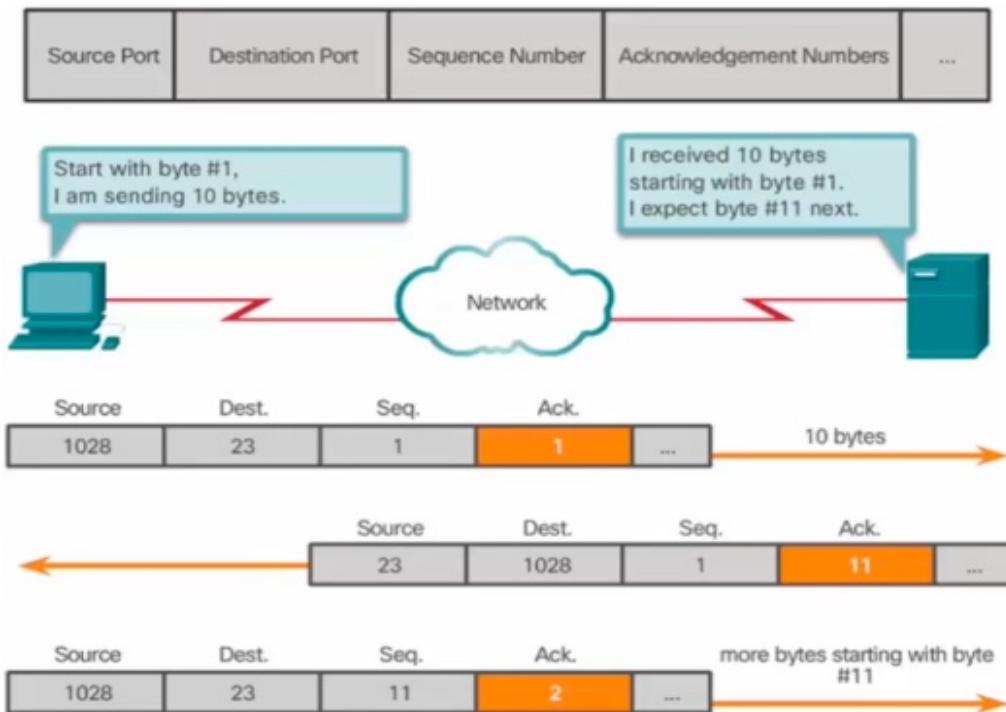
Livraison ordonnée

- Il se peut que les segments TCP parviennent à leur 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.
- Pour cela, des **numéros d'ordre** sont affectés à l'en-tête de chaque paquet.
- Les numéros d'ordre des segments indiquent comment réassembler et réordonner les segments reçus



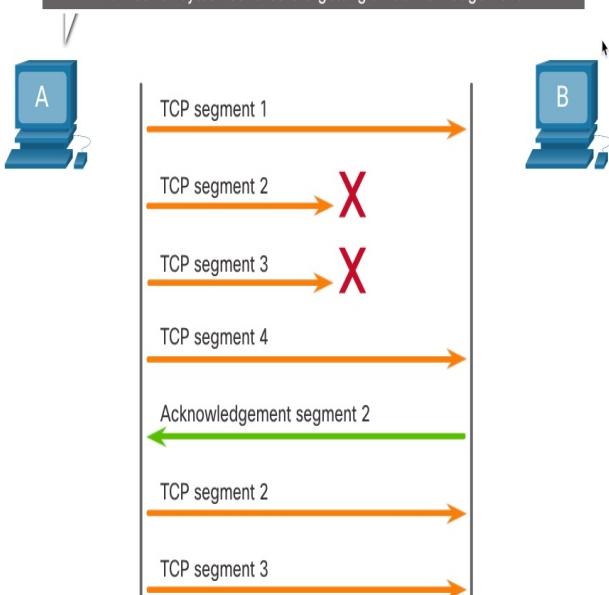
Livraison ordonnée

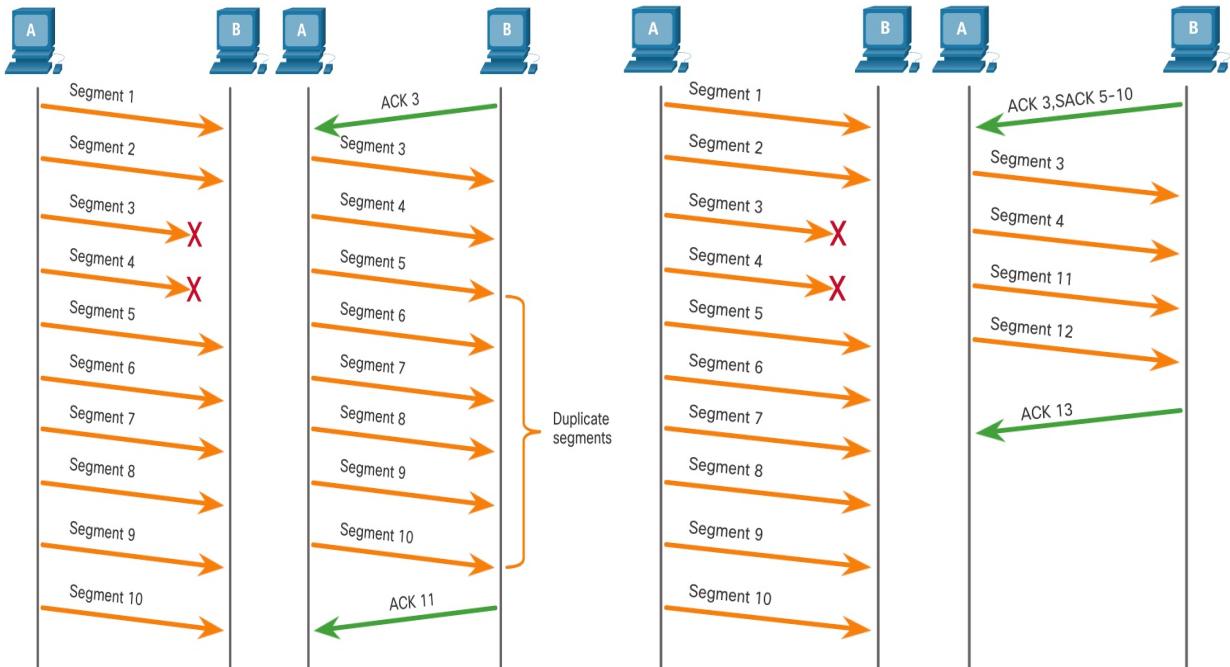
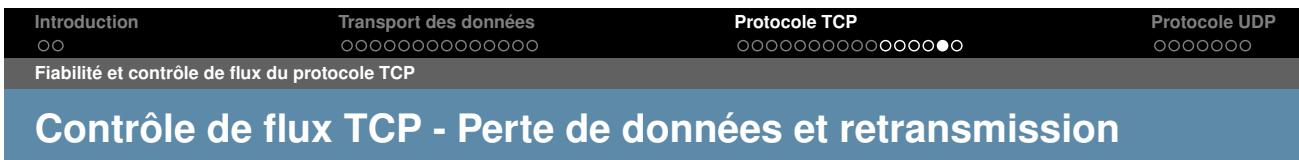
- Pour que le destinataire puisse comprendre le message d'origine, il faut que les données contenues dans ces segments soient réagencées dans leur ordre d'origine. Pour cela, des numéros d'ordre sont affectés à l'en-tête de chaque paquet.
- Lors de la configuration de la session, un numéro d'ordre initial, ou **ISN**, est défini.
- Cet ISN représente la valeur de départ 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 qui arrivent en désordre sont conservés en vue d'un traitement ultérieur.
- Ces segments sont ensuite traités dans l'ordre lorsque les segments contenant les octets manquants sont reçus.



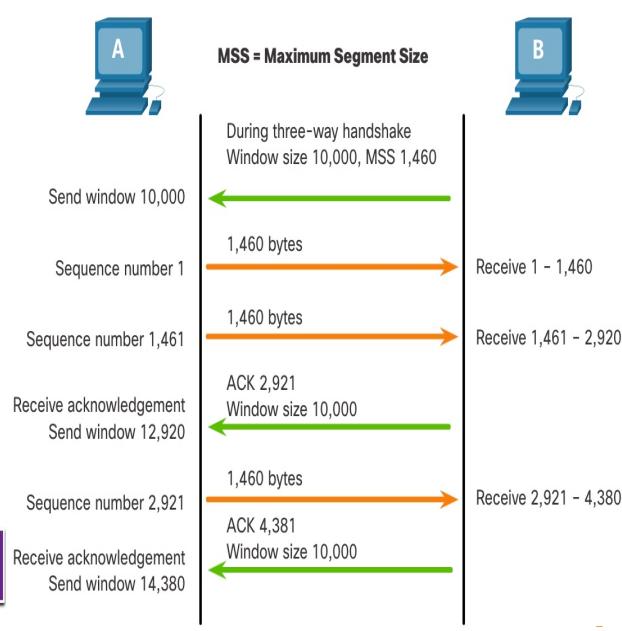
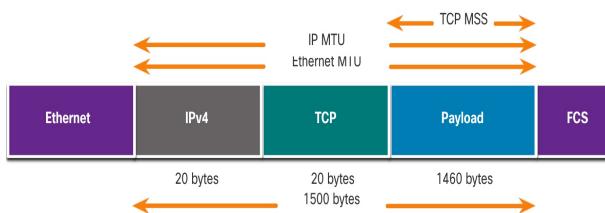
- Chaque fois qu'il y a encombrement, les segments TCP perdus sont retransmis par la source.
- Si la retransmission n'est pas correctement contrôlée, une retransmission supplémentaire des segments TCP peut aggraver encore le niveau d'encombrement du réseau.
- Non seulement de nouveaux paquets contenant des segments TCP sont introduits sur le réseau, mais l'effet de rétroaction des segments TCP perdus et retransmis encombre encore davantage le réseau.
- Afin d'éviter et de contrôler l'encombrement du réseau, le protocole TCP utilise divers mécanismes, minuteurs et algorithmes de gestion des encombrements.

I'm not getting the acknowledgments I expect from PC B so I will reduce the number of bytes I send before getting an acknowledgement.





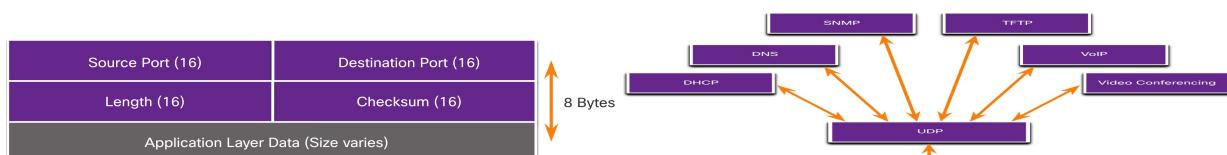
- Le protocole TCP offre également des mécanismes relatifs au contrôle de flux, à la quantité de données que la destination peut recevoir et à la fiabilité du processus.
- Le contrôle de flux aide à maintenir la fiabilité des transmissions TCP en réglant le flux de données entre la source et la destination pour une session donnée.
- Pour cela, l'en-tête TCP inclut un champ de 16 bits appelé taille de fenêtre.



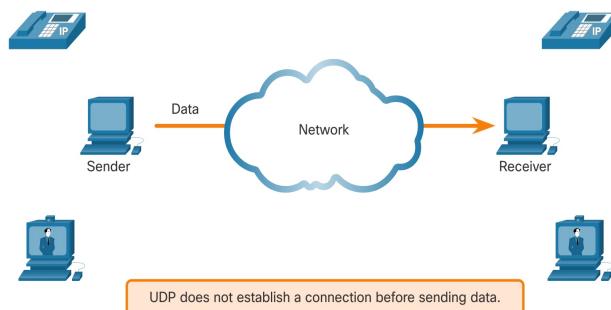
Protocole UDP

Fonctionnalités et en-tête du protocole UDP

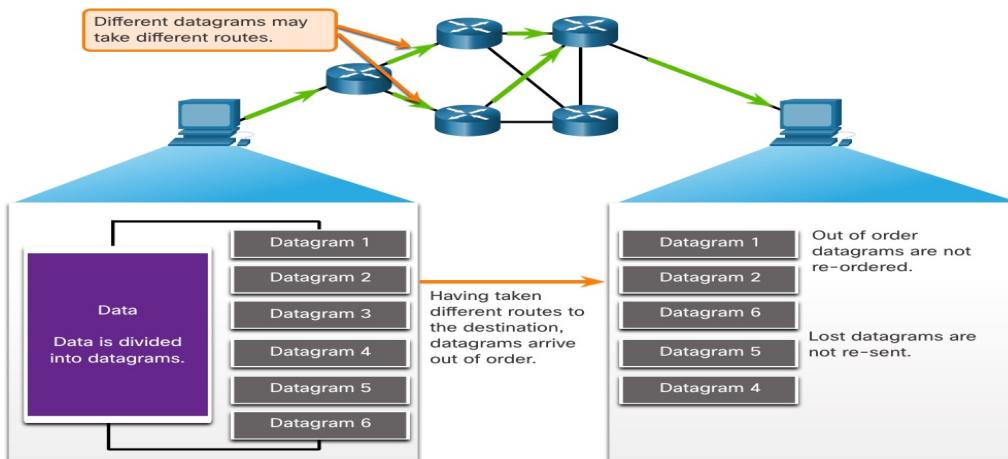
- Le protocole UDP est considéré comme un protocole d'acheminement au mieux.
- Le protocole UDP est un protocole de transport léger qui offre les mêmes fonctions de segmentation et de réorganisation des données que le protocole TCP, mais sans la fiabilité et le contrôle de flux du protocole TCP.
- C'est un protocole simple, qui est généralement décrit en indiquant ce qu'il ne fait pas par rapport au protocole TCP.
- Les fonctionnalités du protocole UDP sont :
 - Les données sont reconstituées selon l'ordre de reception.
 - Les segments perdus ne sont pas renvoyés.
 - Pas d'établissement de session.
 - La source n'est pas informée de la disponibilité des ressources.
- Le protocole 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.
- 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.
- 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 protocole UDP présente une surcharge faible de 8 octets.



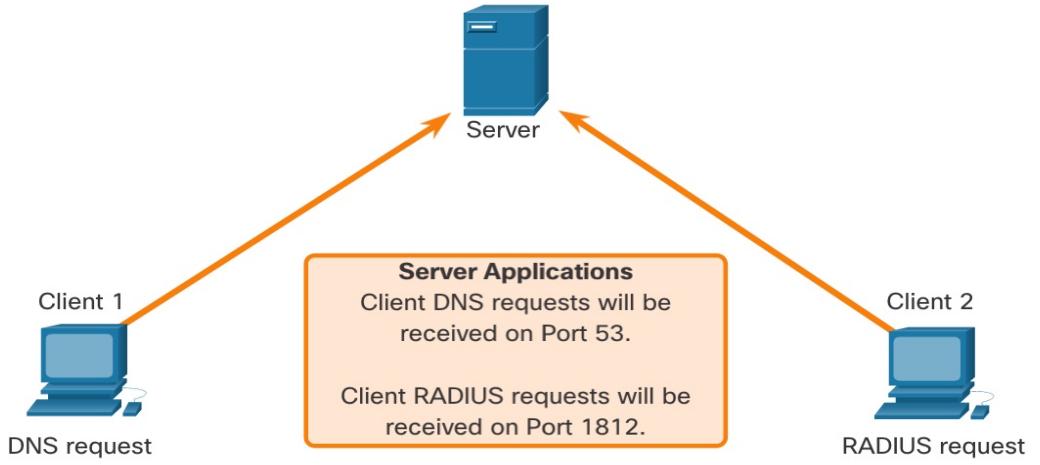
- Le protocole UDP est un protocole simple offrant des fonctions de couche transport de base.
 - Il crée beaucoup moins de surcharge que le protocole TCP, car il n'est pas orienté connexion et ne propose pas de mécanismes sophistiqués de fiabilité (retransmission, séquençage et contrôle de flux).
 - 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 la couche transport et qu'elles doivent être implémentées à un autre niveau, le cas échéant.
 - La faible surcharge qu'engendre le protocole UDP rend celui-ci très intéressant pour les cas où les transactions de requête et de réponse sont simples.
 - Le protocole UDP fournit un transport de données à faible surcharge car il utilise de petits en-têtes de datagrammes et n'offre pas de gestion du trafic réseau.



- Lorsque des datagrammes UDP sont envoyés vers une destination, ils peuvent souvent emprunter des chemins différents et arriver dans le désordre.
 - Le protocole UDP n'effectue pas le suivi des numéros d'ordre comme le fait le protocole TCP.
 - Le protocole UDP ne peut pas réassembler les datagrammes dans leur ordre de transmission.
 - Le protocole UDP se contente donc de réassembler les données dans l'ordre dans lequel elles ont été reçues, puis de les transmettre à l'application.
 - Si l'ordre des données est important pour l'application, cette dernière doit identifier l'ordre correct et déterminer le mode de traitement des données.



- Comme pour les applications basées sur le protocole TCP, des numéros de ports réservés sont affectés aux applications serveur basées sur le protocole UDP.
 - Lorsque ces applications ou processus s'exécutent sur un serveur, elles ou ils acceptent les données correspondant au numéro de port attribué.
 - Quand le protocole UDP reçoit un datagramme destiné à l'un de ces ports, il transmet les données applicatives à l'application appropriée d'après son numéro de port.



- La communication entre le client et le serveur est initiée par une application cliente qui demande des données à un processus serveur.
 - Le processus client UDP sélectionne dynamiquement un numéro de port dans une plage de numéros de ports et il l'utilise en tant que port source pour la conversation.
 - Le port de destination est généralement le numéro de port réservé affecté au processus serveur.

