La couche transport dans Internet

UE LU3IN033 Réseaux
2020-2021

Prométhée Spathis
promethee.spathis@sorbonne-universite.fr

Plan du cours

- Répartition des tâches dans Internet
  - Machines hôtes versus routeurs
- Conception des applications réseau
  - Modèle client-serveur vs modèle P2P
- Classification des besoins des applications
  - Fiabilité, bande passante, délai, sécurité
- Les protocoles de la couche transport dans Internet
  - User Datagram Protocol (UDP)
  - Transmission Control Protocol (TCP)
- Les principes sous-jacent aux services de la couche transport
  - (Dé)multiplexage
  - Détection des erreurs et des pertes
  - Livraison fiable
  - Controle de flux

Conception de l'Internet et de ses applications

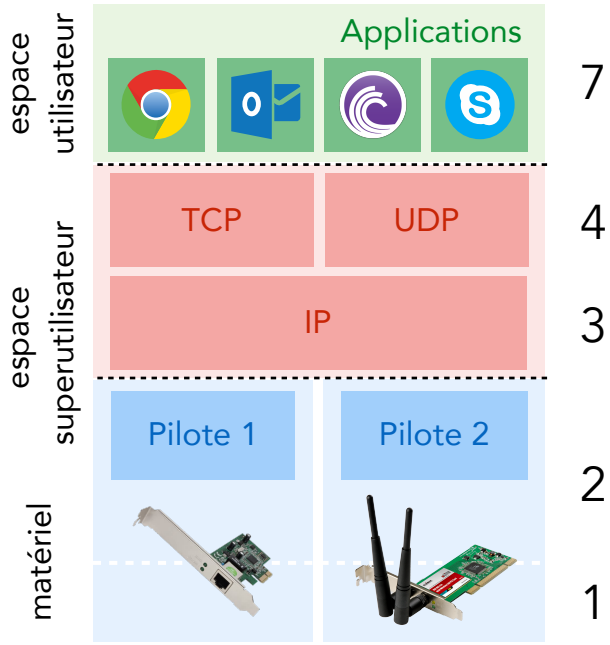
Conception bout en bout de l'Internet

Périphérie versus cœur du réseau

1. Les machines hôtes exécutent les applications
   - les système d'exploitation implémentent une interface de programmation standardisée qui permet :
     - aux développeurs de concevoir des applications…
     - …que les utilisateurs téléchargent et installent
   - les applications communiquent selon un protocole de couche 7
   - les données qu'elles génèrent sont passées aux couches inférieures
2. Les routeurs acheminent les données pour le compte des applications
   - les données sont mises dans des paquets IP adressées aux machines hôtes hébergeant l'application destinatrice
   - chaque paquet est traité indépendamment les uns des autres

Conception des applications Internet

- Architecture des applications réseau
  - Une application réseau est constituée de deux programmes hébergés sur deux machines distantes
  - Ces deux programmes communiquent selon un protocole de couche 7 :
    - HTTP entre navigateur Web et serveur Web
- Conception asymétrique vs symétrique
  - Les applications originelles de l'Internet ont été conçues selon un modèle client-serveur
    - navigateur/serveur Web, …
  - Avènement récent d'applications conçues selon un modèle Pair-à-Pair (P2P)
    - BitTorrent, Skype, …



Client-Serveur vs P2P

Client-Serveur

- Conception asymétrique
  - l'application résulte de l'exécution de deux programmes différents
- Le serveur :
  - s'exécute en permanence
  - attend les requêtes client
  - écoute sur un numéro de port déjà connu des clients
- Les clients :
  - sont connectés par intermittence
  - initient la communication en envoyant leur requête
  - doivent connaitre l'adresse IP et le numéro de port du serveur

Pair-à-Pair

- Conception symétrique
  - tous les noeuds exécutent le même programme
  - les noeuds sont des clients qui peuvent agir comme des serveurs pour d'autres pairs
- Service de découverte des pairs
  - les applications P2P nécessitent un mécanisme de découverte de :
    - l'adresse IP des pairs
    - le numéro de port qu'ils utilisent
  - Un serveur (!), une base de données répartie sur les noeuds, …

Modèle Client-Serveur

Client

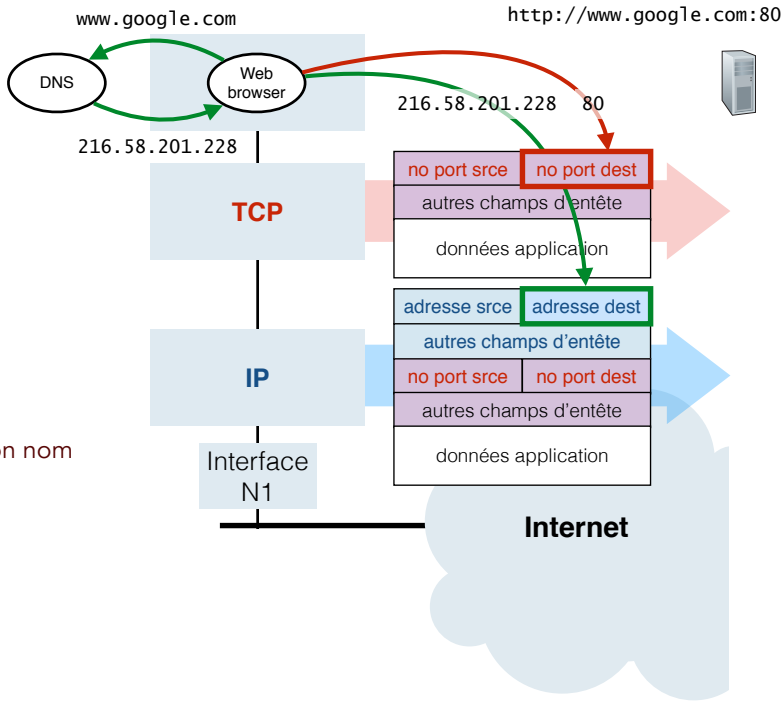
- Les clients :
  - ont une adresse IP dynamique
    - qui change en fonction du temps
    - qui change en fonction de leur localisation
  - utilisent un numéro de port arbitraire supérieur à 1024
    - le choix est généralement laissé au système d'exploitation
- Adresse IP d'un serveur :
  - les clients découvrent l'adresse IP d'un serveur en soumettant son nom (URL) aux serveurs DNS
    - pages blanches de l'Internet

Serveur

- Les serveurs :
  - ont une adresse IP statique
  - ont un nom connu des clients
    - moteurs de recherche
  - écoutent sur un numéro de port déjà connu des clients
- Adresse IP et numéro de port des clients
  - ce sont les clients qui contactent les serveurs !
    - un serveur découvre ainsi les informations du client
    - et peut ainsi leur répondre

Remplissage des entêtes

- Client (source)
  - Adresse IP du client
    - statique (conf manuelle)
    - dynamique (DHCP)
  - Numéro de port du client
    - valeur arbitraire (> 1023)
    - souvent laissé au choix de l'OS
- Serveur (destination)
  - Adresse IP du serveur
    - codée dans l'application
    - découverte par DNS à partir de son nom
  - Numéro de port du serveur
    - valeur définie par convention
    - connue par avance



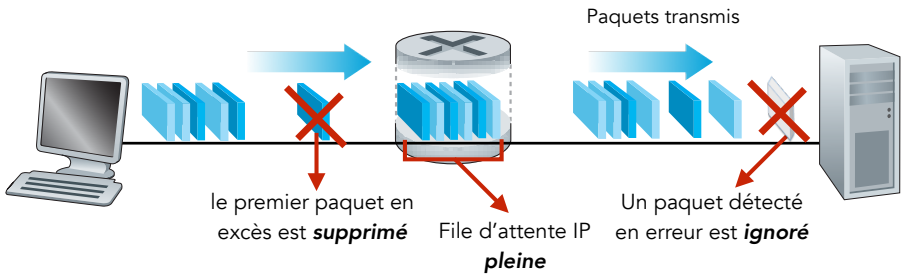
Conception bout en bout de l'Internet

Répartition des tâches dans Internet

1. Les routeurs font de leur mieux pour acheminer les données
   - selon les ressources en présence
   - l'absence de ressources provoque :
     - des pertes, des duplications, des retards, des déséquencements, …
2. Les machines hôtes comblent les manquements de la couche réseau si nécessaire
   - en corrigeant les erreurs
   - en limitant les pertes et en réparant celles qu'ils n'ont pu éviter
   - en éliminant les données dupliquées
   - en réordonnant les données
   - pour satisfaire aux besoins des applications et simplifier leur conception

Comment les paquets sont-ils perdus?

- File d'attente des routeurs
  - Les paquets IP sont mis en attente le temps d'être traités
- Congestion réseau
  - Les paquets IP en excès sont supprimés
  - Ces pertes ne sont pas détectées par la couche Liaison de données



- Paquets IP en erreur
  - Les paquets contenant des bits erronés sont ignorés par leur destinataire

Rôle des couches 3, 4, et 7

- Couche Réseau
  - Identification des extrémités du chemin
    - adresses IP de la source et de la destination
  - Acheminement des données le long de réseaux physiques hétérogènes
  - Un seul protocole : IP
- Couche Transport
  - Identification des processus exécutés sur les machines d'extrémités
    - numéros de port source et destination
  - Multiplexage des flots de données provenant de plusieurs applications
  - Deux protocoles : TCP et UDP
- Couche Application
  - Nommage des serveurs intelligible pour les utilisateurs
    - URL, noms de domaine, …
  - Protocoles : HTTP, POP, IMAP, Skype, Bittorrent, …

Pourquoi deux protocoles transport et lequel choisir ?

Classification des applications (1)

- Les applications ont des besoins classifiables en termes de :
  - Fiabilité
    - tolérance aux pertes de données ?
  - Bande passante
    - quantité minimale nécessaire au bon fonctionnement de l'application ?
  - Contraintes temporelles
    - délai d'acheminement borné ?
  - Sécurité
    - authentification, encryption, signature ?

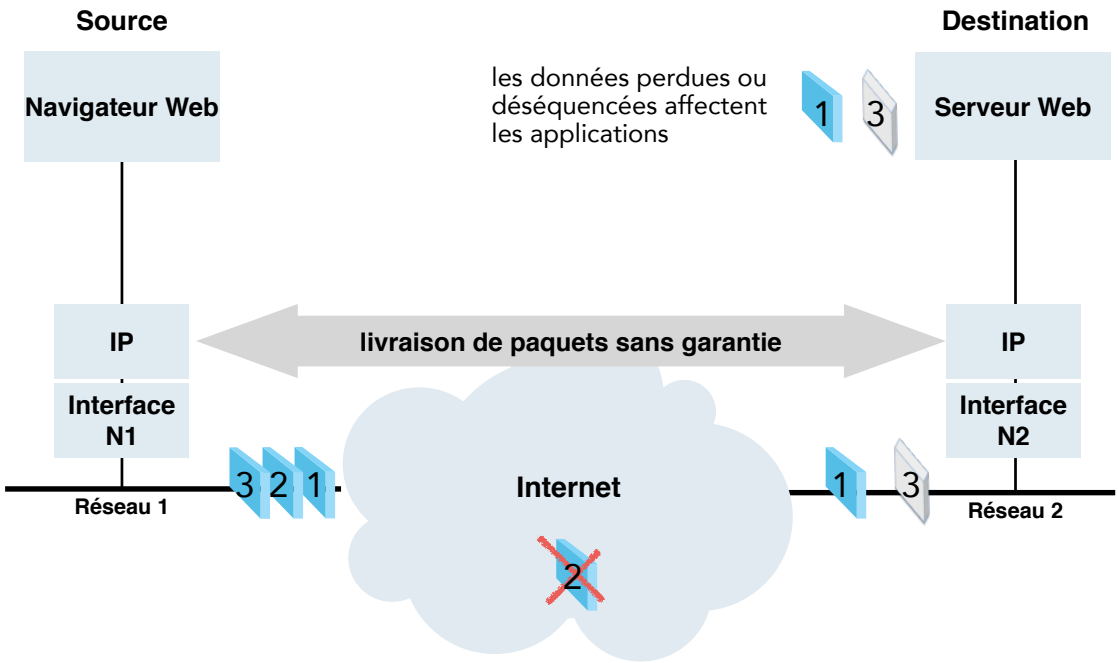
Classification des applications (2)

Application	Pertes	Bande passante	Délai
Transfert de fichiers	pas de tolérance	élastique	non sensible
email	pas de tolérance	élastique	non sensible
Documents Web	pas de tolérance	élastique	non sensible
Audio/video temps réel	tolérance	audio: 5kb/s-1Mb/s video:10kb/s-5Mb/s	100aines msec
Streaming audio/video préenregistré	tolérance	audio: 5kb/s-1Mb/s video:10kb/s-5Mb/s	qques sec
Jeux interactifs	tolérance	qques kbps	100aines msec

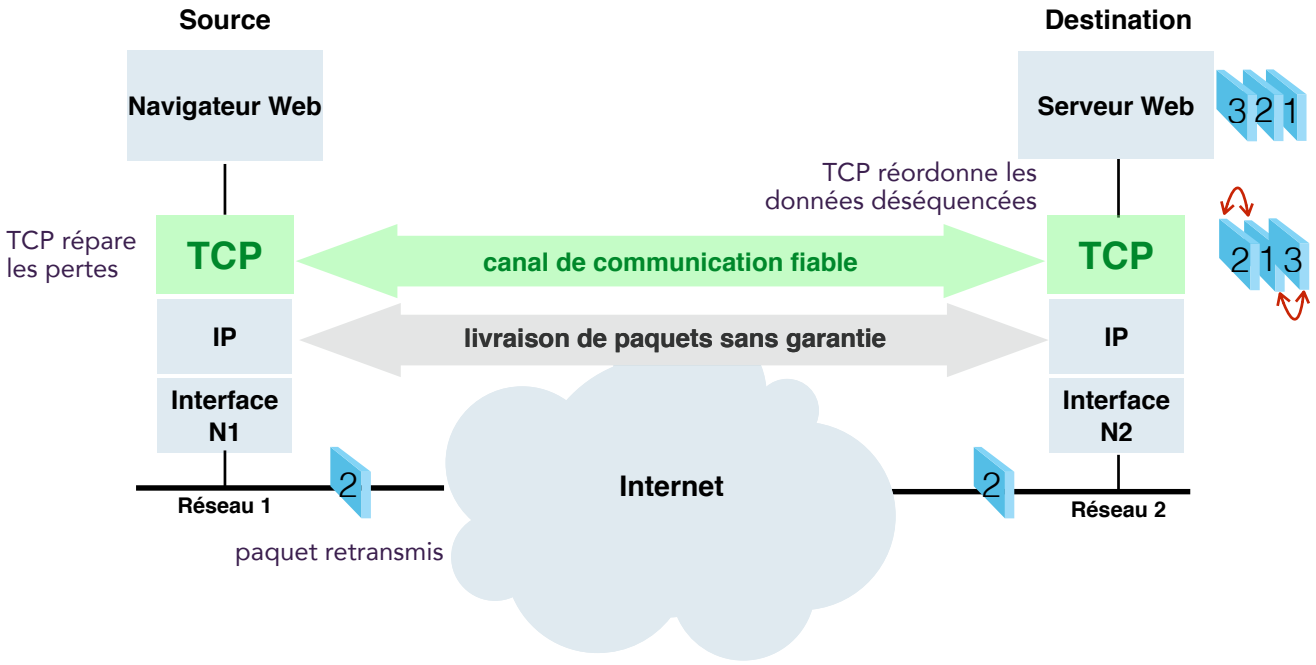
TCP

UDP

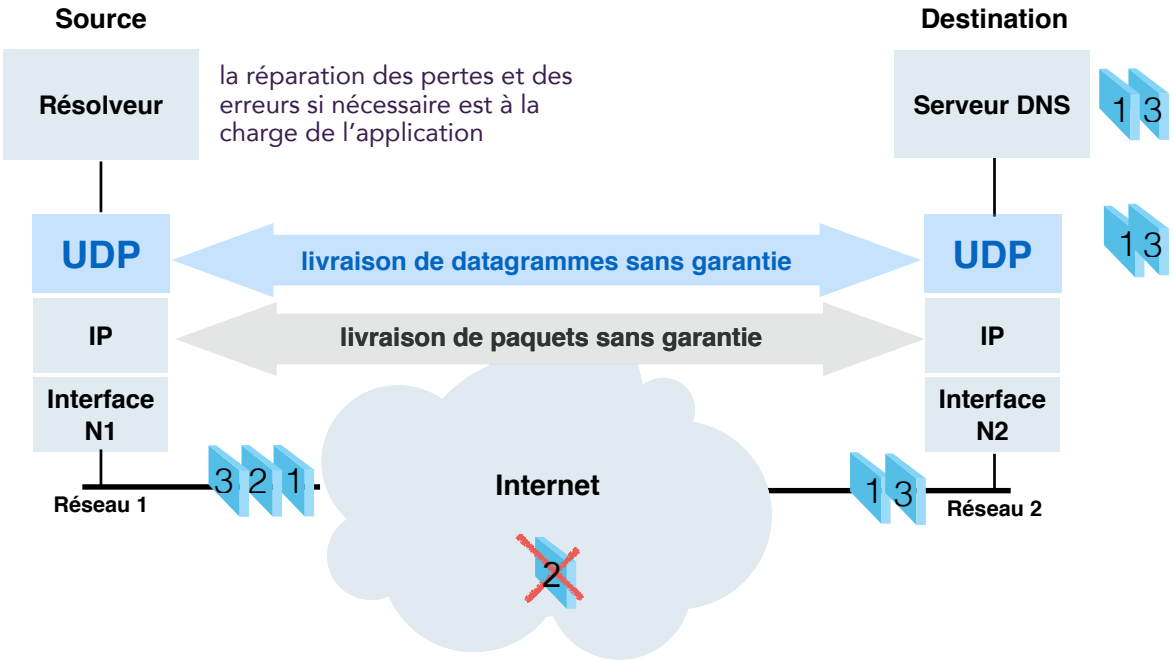
IP: Acheminement proche en proche sans garantie



TCP: livraison fiable bout en bout entre processus distants



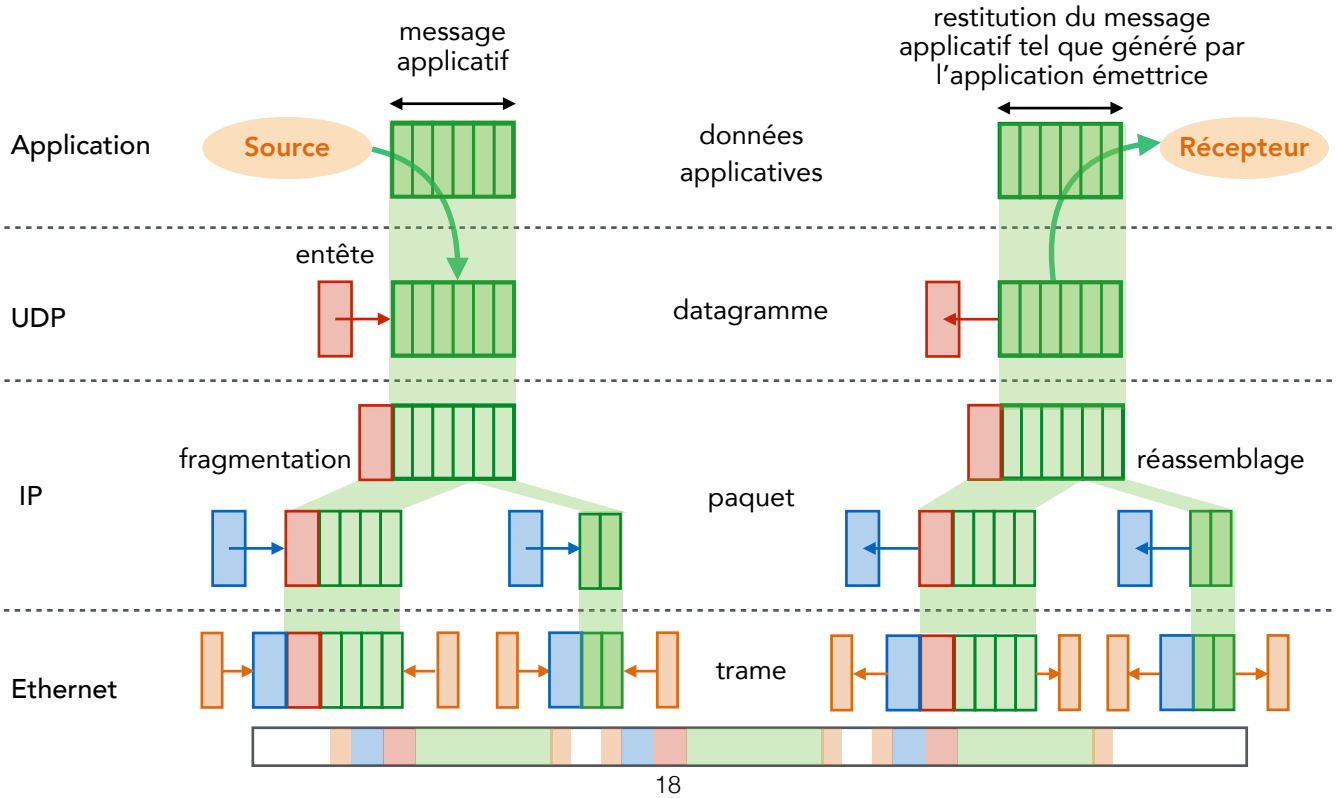
UDP: livraison bout en bout sans garantie entre processus distants



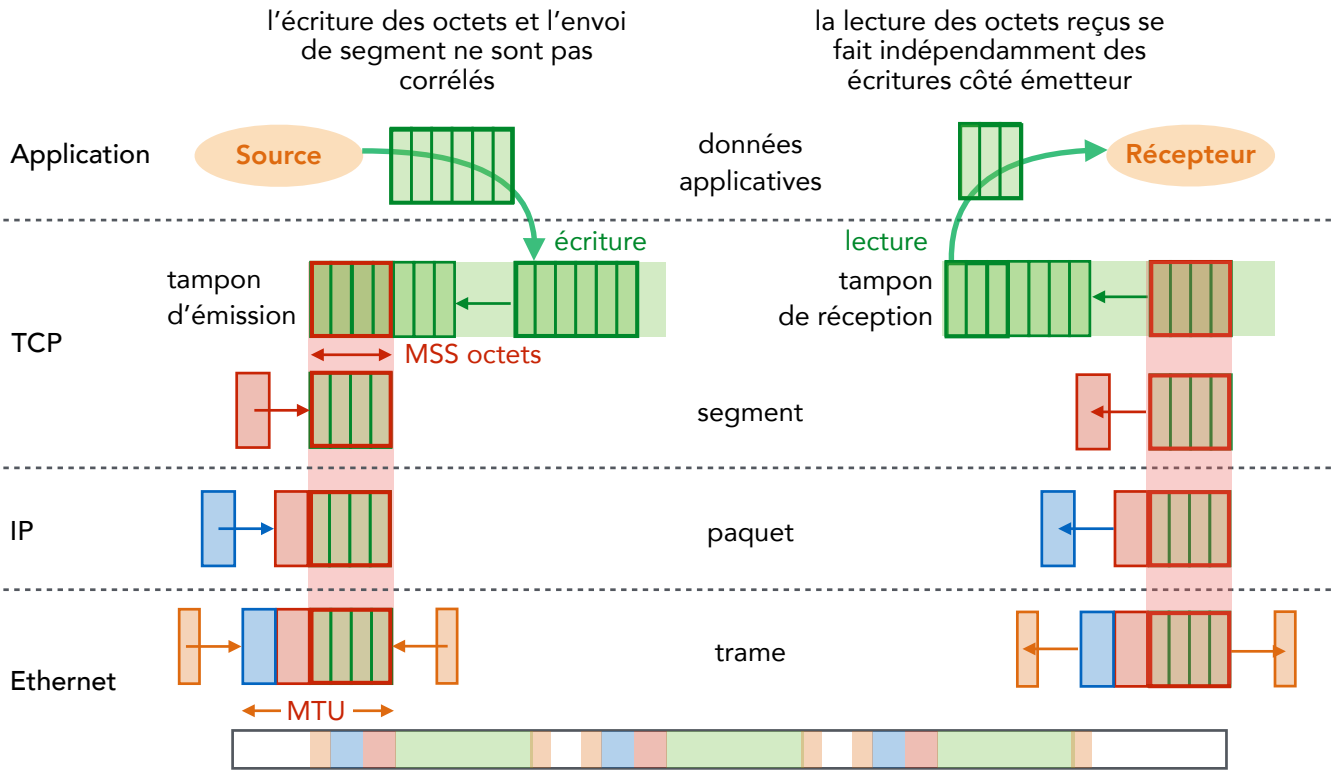
Rôle de la couche Transport (4)

- Deux protocoles :
  - UDP : livraison sans garantie en mode datagramme
  - TCP : restitution fiable et en séquence en mode flux d'octets
- Services de base commun à UDP et TCP
  - Multiplexage / Démultiplexage
    - identification des processus exécutés sur les machines d'extrémités
    - numéros de port source et destination
  - Détection d'erreurs
    - somme de contrôle sur l'entête et les données
- Services spécifiques à TCP
  - Correction des octets en erreur ou perdus
  - Remise en séquence des octets reçus, suppression des octets dupliqués
  - Contrôle de flux
  - Contrôle de congestion

UDP: Mode Datagramme



TCP: Mode Flux d'octets



2 protocoles transport : TCP et UDP

Mode datagramme vs. Mode flux d'octets

- UDP : mode datagramme
  - les messages applicatifs sont encapsulés tel quel dans un datagramme
  - les datagrammes sont envoyés immédiatement
  - IP fragmente éventuellement les datagrammes en fonction de la MTU locale
  - l'application côté récepteur reçoit les messages applicatifs tel que générés côté émetteur
- TCP : mode flux d'octets
  - Côté émetteur, TCP attend :
    - que l'application génère MSS octets pour remplir une trame complète (MTU)
    - ou l'expiration d'un temporisateur pour former un 'petit' segment
  - TCP encapsule ces octets dans un segment et attend le moment opportun pour les envoyer
    - sans congestionner le réseau
    - sans engorger les récepteurs
  - IP n'a pas de raison de fragmenter les paquets encapsulant des segments TCP
  - Côté récepteur, TCP écrit les octets reçus dans un tampon le temps que l'application les lise

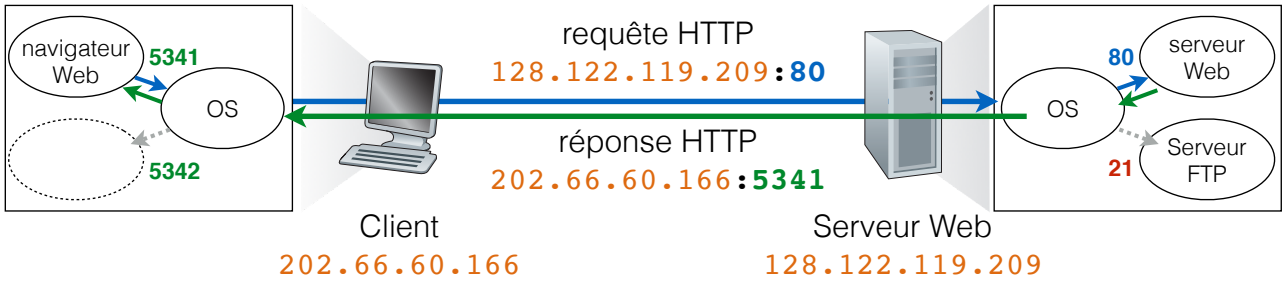
2 protocoles transport : TCP et UDP

Mode Non connecté vs Mode Connecté

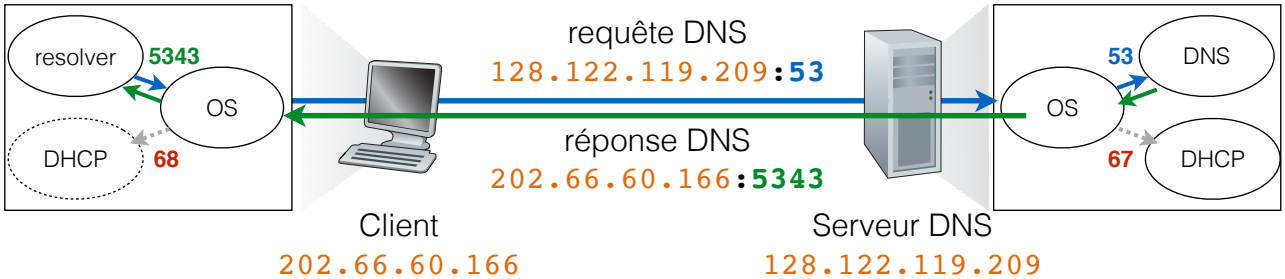
- UDP : mode non connecté
  - Envoi des données sans attendre l'établissement d'une connexion
    - adapté pour des échanges simples de type une requête-une réponse
  - Sans installer d'états
    - pas de tampons, pas de numéros de séquence, pas de fenêtres, pas de temporisateurs
- TCP : mode connecté
  - Délai d'établissement d'une connexion avant de prendre en charge les données
  - Installation et maintien d'états
    - tampons, numéros de séquence, fenêtres, temporisateurs

(Dé-)multiplexage: numéros de port

TCP



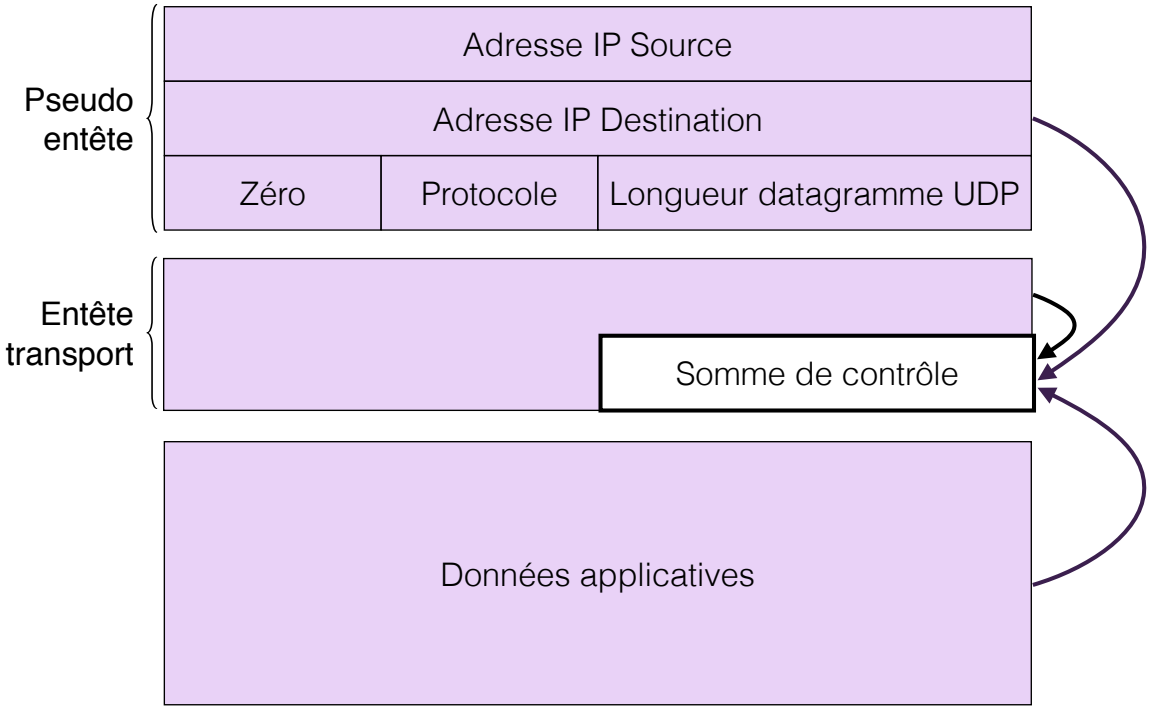
UDP



Numéros de port

- Un numéro de port est codé sur 16 bits
  - valeur max : 65,535
- Identification des processus de la couche 7
  - permet l'envoi (la réception) simultané(e) d'octets provenant de plusieurs applications
  - portée locale
- Numéros de port réservés (connus de tous) : < 1024
  - utilisation de ports bien connus côté serveur
    - les communications étant initiées à l'initiative des clients…
    - …un client doit connaitre au préalable le numéro de port du serveur
    - un serveur Web utilise toujours 80
- Numéros de port côté client
  - valeur sans signification pour l'application client
    - un navigateur Web utilise une valeur quelconque supérieure à 1023
  - généralement laissée au choix du système d'exploitation
- Utiliser pour filtrer les trafics Internet
  - Firewall, NAT, …

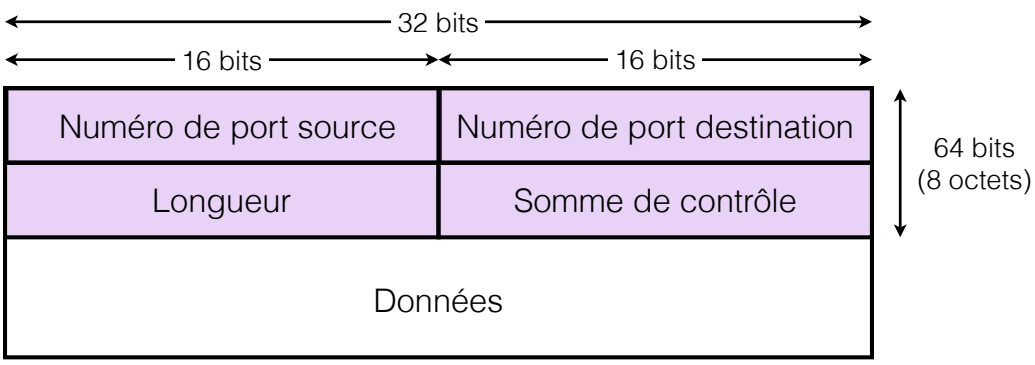
Somme de contrôle



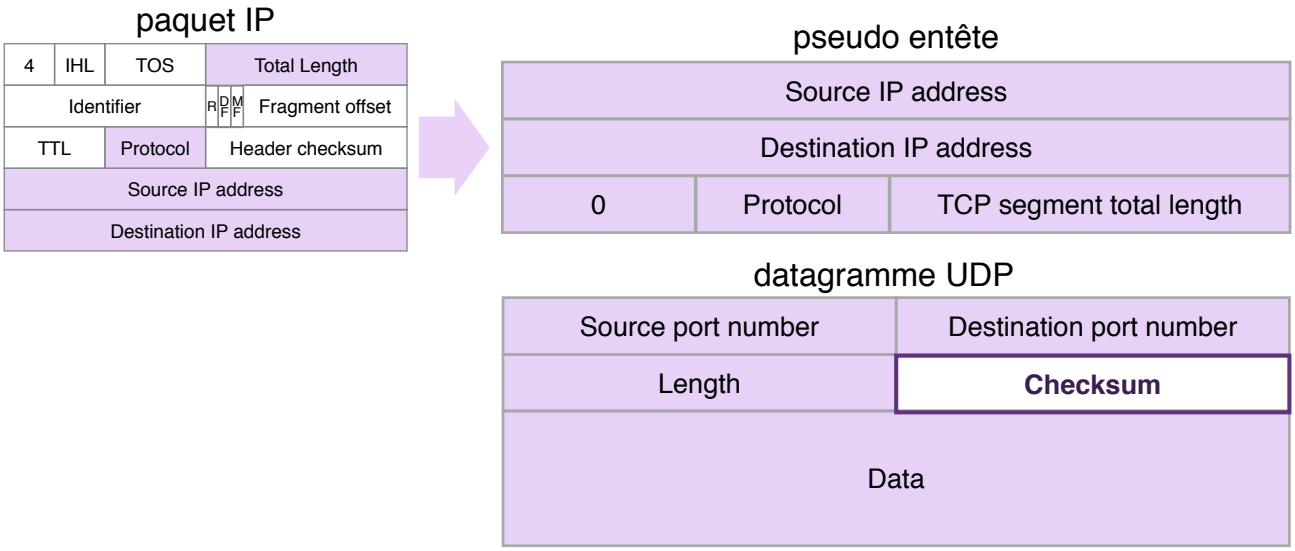
UDP
User Datagram Proctocol

UDP : User Datagram Protocol

- Service de livraison de datagrammes simple
  - Numéros de port : (dé)-multiplexage des datagrammes
  - Somme de contrôle : détection des datagrammes en erreur
  - Longueur : pour restituer les messages aux applications



Pseudo entête et somme de contrôle



Le pseudo-entête UDP (et TCP) permet la double-vérification des informations IP

UDP : service sans garantie

- Service simple de livraison de datagrammes
  - Démultiplexage des datagrammes : numéros de port
  - Détection des datagrammes en erreur : somme de contrôle
- Communication basique entre processus distants
  - UDP ajoute à IP la gestion des numéros de port et la détection des bits en erreur
    - les datagrammes si reçus peuvent l'être en désordre
  - Evite la complexité et les délais nécessaires à la fiabilisation des échanges
    - pas de connexion, pas d'états
  - Ne limite pas l'utilisation de la bande passante
    - pas de contrôle de flux ou de congestion
- Protocole destiné aux applications :
  - qui privilégie la rapidité de remise des données …
  - … à la fiabilité
  - mieux vaut jamais que tard

Avantages de UDP

- Protocole en mode non connecté
  - UDP envoie un datagramme dès que l'application génère des données…
  - …sans établir de connexion
- Protocole sans état
  - pas de tampons mémoire, paramètres, numéros de séquence, etc.
- Envoi immédiat des données applicatives
  - ajout d'un entête UDP aux données telles que passées par l'application …
  - … quelle que soit leur taille
  - … sans établir de connexion
  - … sans se soucier de la congestion ou de l'engorgement du récepteur
- Surcoût modique de l'entête
  - l'entête UDP est longue de 8 octets

Applications populaires qui utilisent UDP

- Streaming multimédia
  - La fiabilité n'est pas compatible avec les applications interactives
    - les retransmissions retardent la réception des données
  - Les utilisateurs d'applications multimédia ne sont pas sensibles aux pertes :
    - appels téléphoniques, téléconférences, jeux en ligne, IPTV, …
- Protocoles simples de type ``une requête-une réponse''
  - Les états nécessaires par connexion ne sont pas justifiés ou possibles à maintenir
    - si l'adresse des clients n'est pas encore connue
    - si les clients se connectent en grand nombre au même serveur
  - DHCP, Domain Name System (DNS),…
- Mais, les fonctions des box Internet ne sont pas compatibles avec UDP
  - Les pares-feux bloquent le trafic UDP
  - NAT ne sait pas laisser passer les datagrammes UDP

TCP
Transmission Control Protocol

Plan

- Fiabilité et efficacité de TCP
- Transmission en mode flux d'octets
- Taille des segments TCP
- Types et format des segments
- Numérotation des octets de données et accusés de réception
- Ouverture et fermeture de connexion TCP
- Contrôle de flux
- Octets de données urgentes et pushées
- Options TCP

Transmission Control Protocol (TCP)

- Service orienté flux d'octets
  - L'application écrit ses octets dans un tampon en émission
  - TCP y prélève une quantité d'octets appelée MSS indépendante des blocs écrits par l'application
    - cette quantité dépend de la MTU locale
- Orienté connexion
  - Ouverture et fermeture d'une connexion
    - avec installation et libération d'états
- Livraison fiable, en séquence d'octets
  - Somme de contrôle
    - pour détecter les octets en erreur
  - Numérotation en séquence des octets de données
    - pour détecter leur perte et les réordonner
  - Accusés de réception, temporisateurs et retransmissions
    - pour réparer les pertes ou les erreurs
- Contrôle de flux
  - Pour éviter l'engorgement des récepteurs (tampons de réception limités)
- Contrôle de congestion
  - Pour adapter le débit d'émission à la charge du réseau

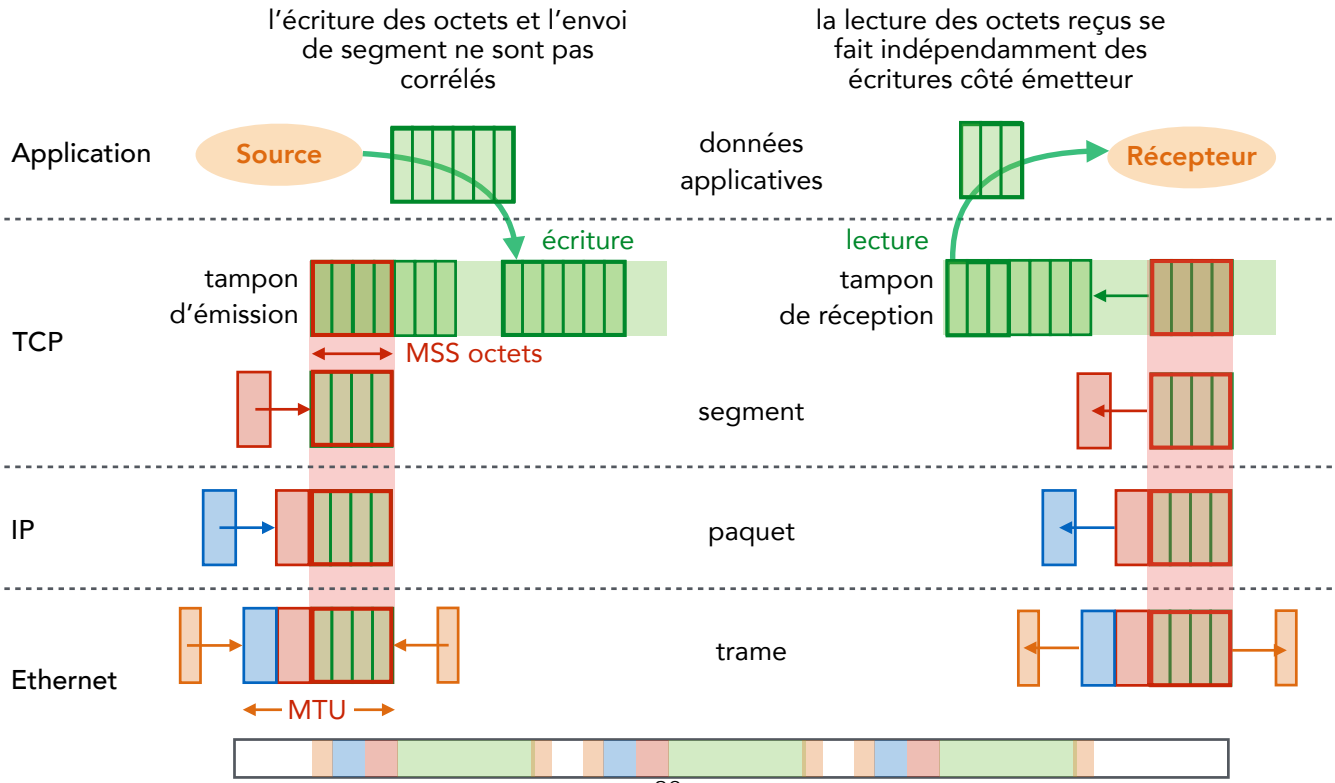
Transfert fiable

- Etablissement/libération de connexion
  - création et maintien d'états
    - tampons, numéros de séquence, fenêtres, temporisateurs
- Détection des erreurs
  - somme de contrôle portant sur :
    - sur le segment entier (entête et données) et
    - sur l'entête IP partiel (pseudo-entête)
- Détection des pertes et remise en ordre des octets
  - numérotation des octets
  - tampons de réception
- Correction des pertes et des erreurs
  - acquittement positif cumulatif (ACK)
  - temporisateur de retransmission chez l'émetteur

Transfert efficace

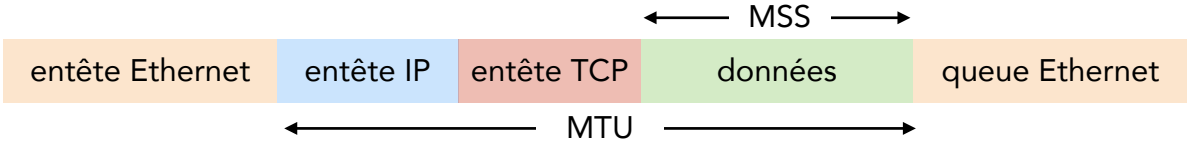
- TCP construit un segment si :
  - la charge utile des segments (MSS) est déterminée en fonction de la MTU locale
    - pour éviter la fragmentation IP
    - pour éviter l'envoi de trames à moitié pleine
- TCP évite les pertes :
  - le nombre d'octets de données envoyés est déterminé par :
    - la fenêtre de contrôle de flux : égale aux tampons libres du récepteur
    - la fenêtre de contrôle de congestion : calculée en fonction de la bande passante disponible le long du chemin emprunté par les segments
- TCP assure un partage équitable de la bande passante :
  - le débit d'émission des sources qui émettent de segments empruntant le même chemin :
    - résulte du calcul de la taille de leur fenêtre de congestion
    - ce calcul assure une répartition équitable de la bande passante entre ces sources

TCP: Mode flux d'octets

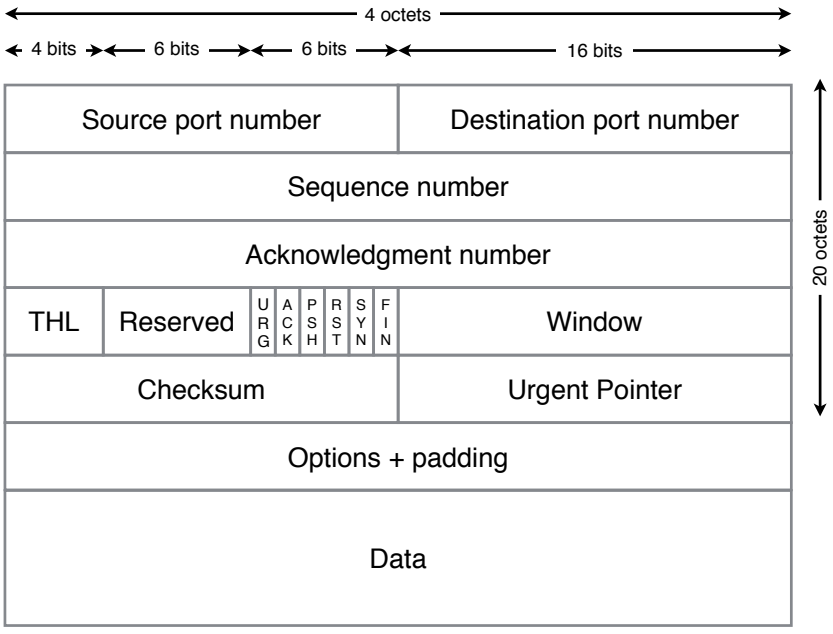


Constitution d'un segment TCP

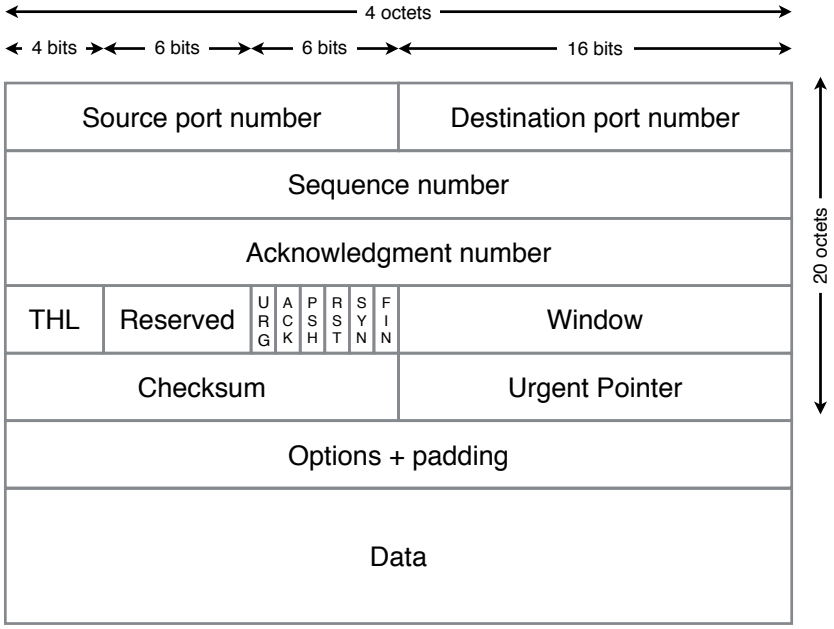
- L'efficacité d'une transmission se mesure selon le remplissage des trames
  - La MTU représente la taille max du champ données des trames
    - La MTU des trames Ethernet est de 1500 octets
- TCP évite l'envoi de trames à moitié pleine
  - TCP attend que l'application ait écrit MSS octets
    - MSS : Maximum Segment Size
    - MSS = MTU – (taille en-tête IP + taille en-tête TCP)
    - MSS au-dessus d'Ethernet : 1460 octets si entêtes IP et TCP sans options
  - Sauf si l'application demande explicitement à TCP d'envoyer un 'petit' segment
    - fonction push (voir drapeaux de l'entête TCP)
  - Ou l'expiration d'un temporisateur
    - pour éviter d'attendre trop longtemps MSS octets



Entête TCP



Entête TCP



Entête TCP