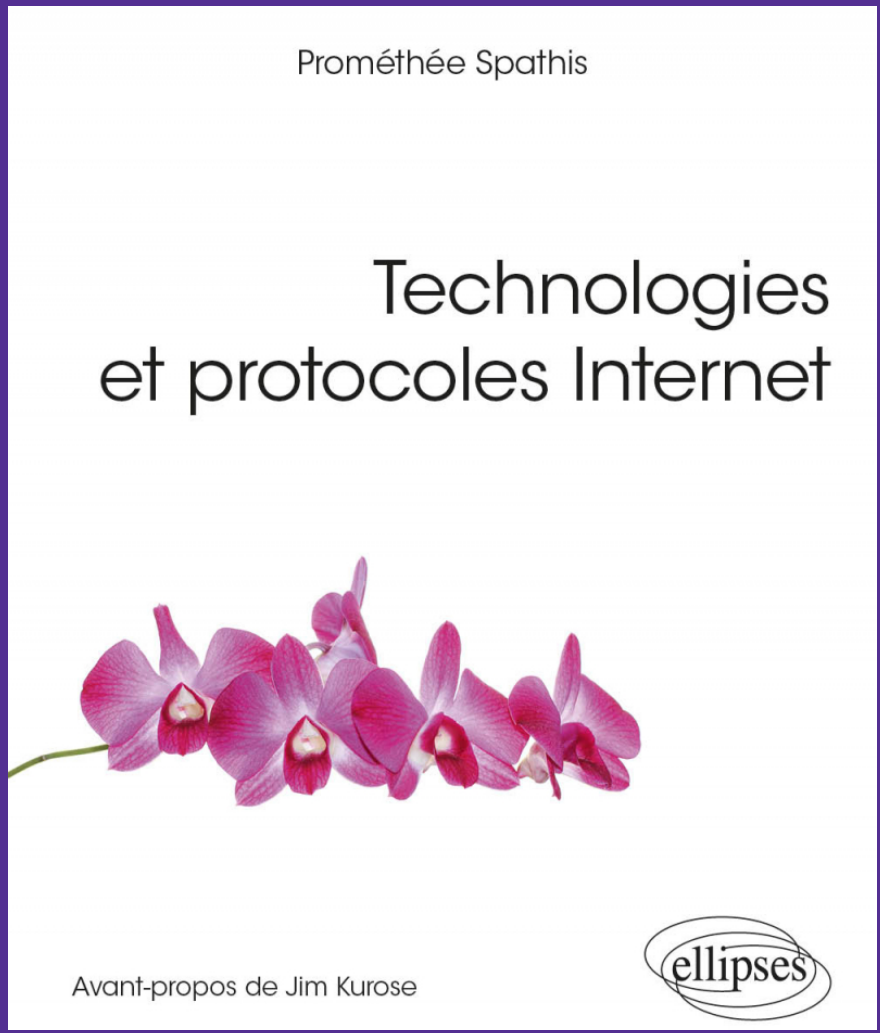
Chap 4 - UDP & TCP

Ces transparents sont mis à disposition de tous (étudiants, enseignants, lecteurs).

En contrepartie, merci de bien vouloir :

- mentionner leur source,
- préciser la mention de copyright.

Merci et bon cours !



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

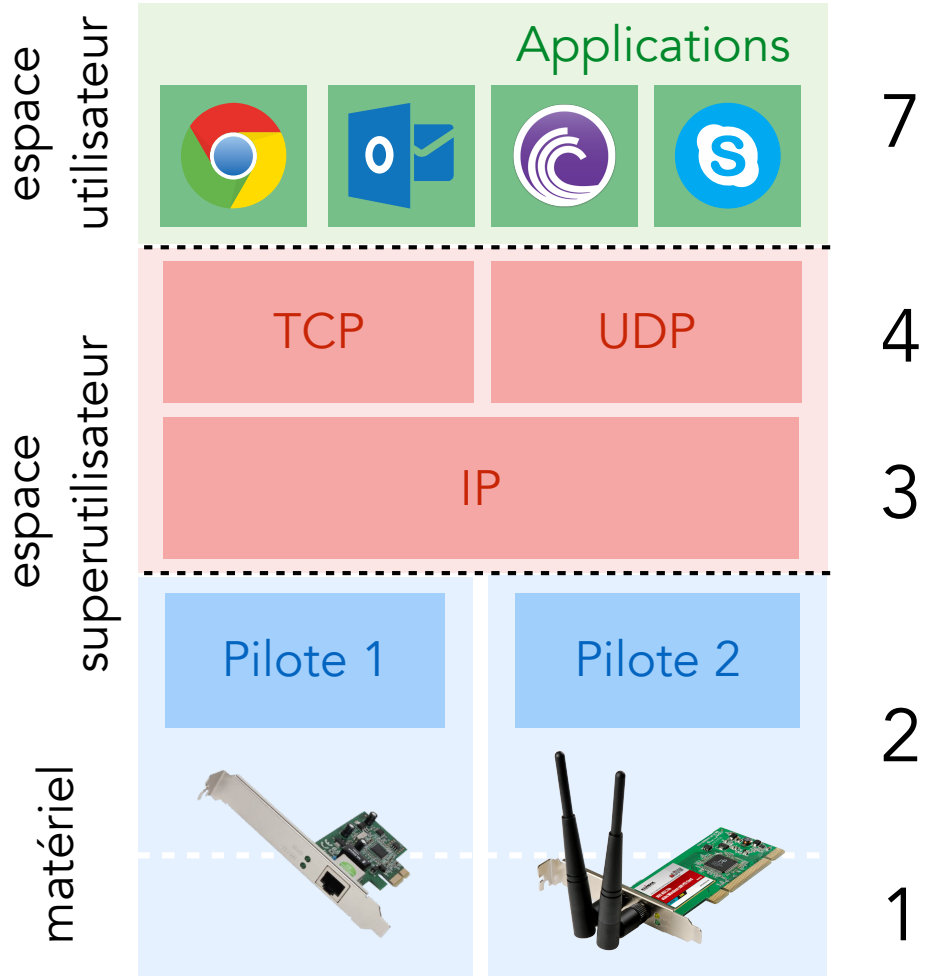
Conception bout en bout de l'Internet

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
   - les paquets peuvent être perdus, dupliqués, reçus en erreur ou hors séquence

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

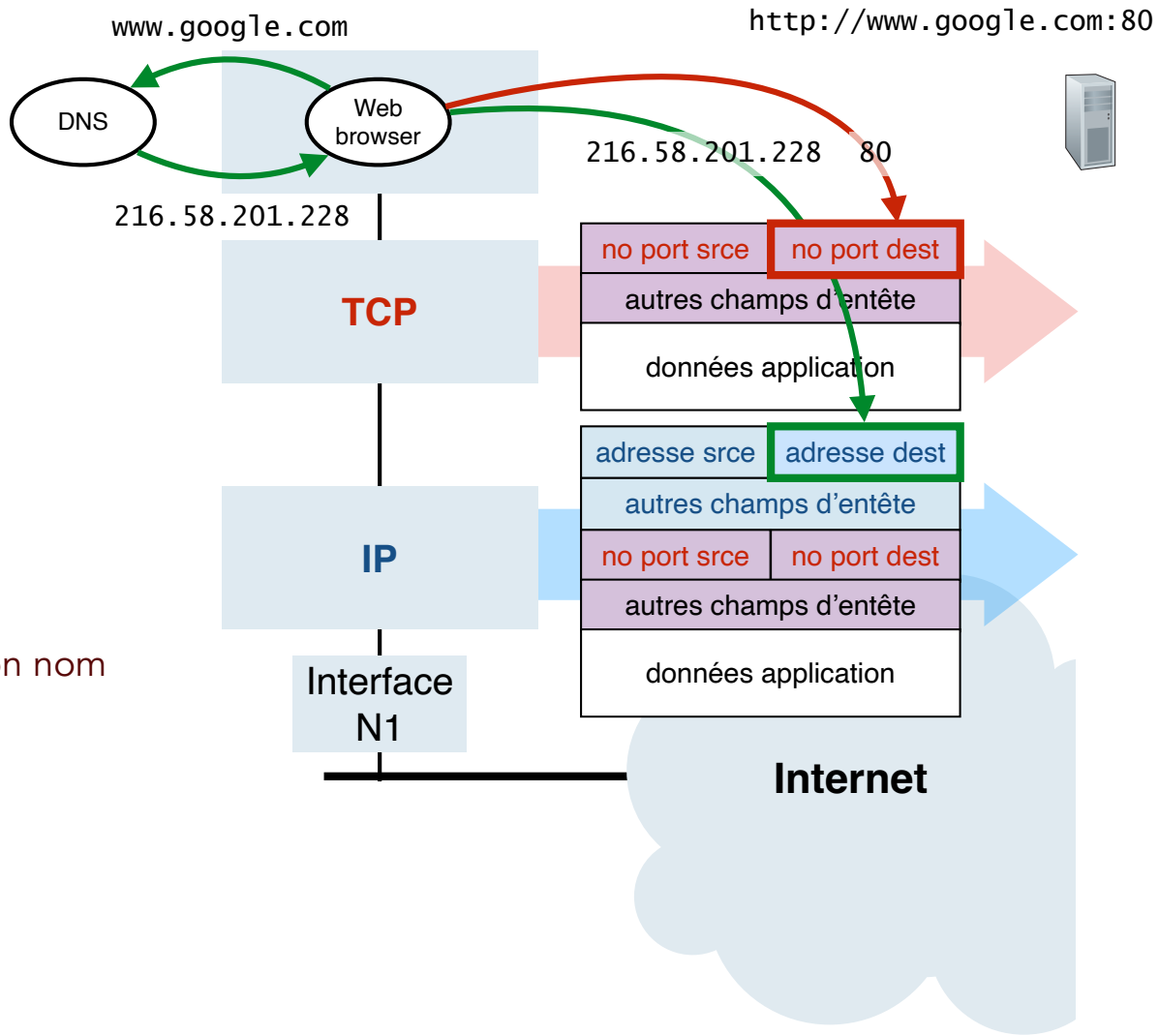
- Les clients :
  - ont une adresse IP dynamique
    - qui change en fonction du temps
    - qui change en fonction de leur localisation
  - utilisent un numéro de port arbitraire supérieur à 1024
    - le choix est généralement laissé au système d'exploitation
- Adresse IP d'un serveur :
  - les clients 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



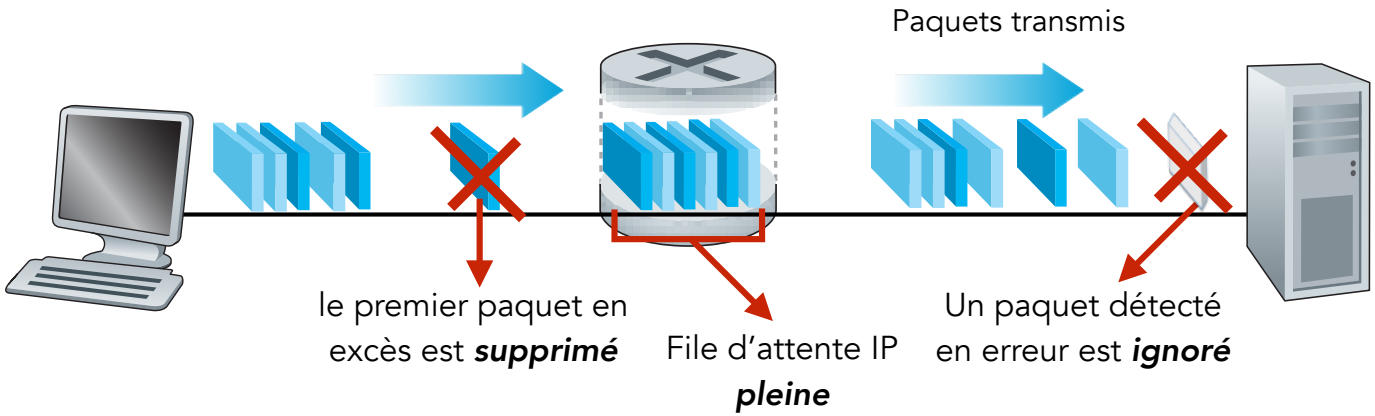
Conception bout en bout de l'Internet

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 ?

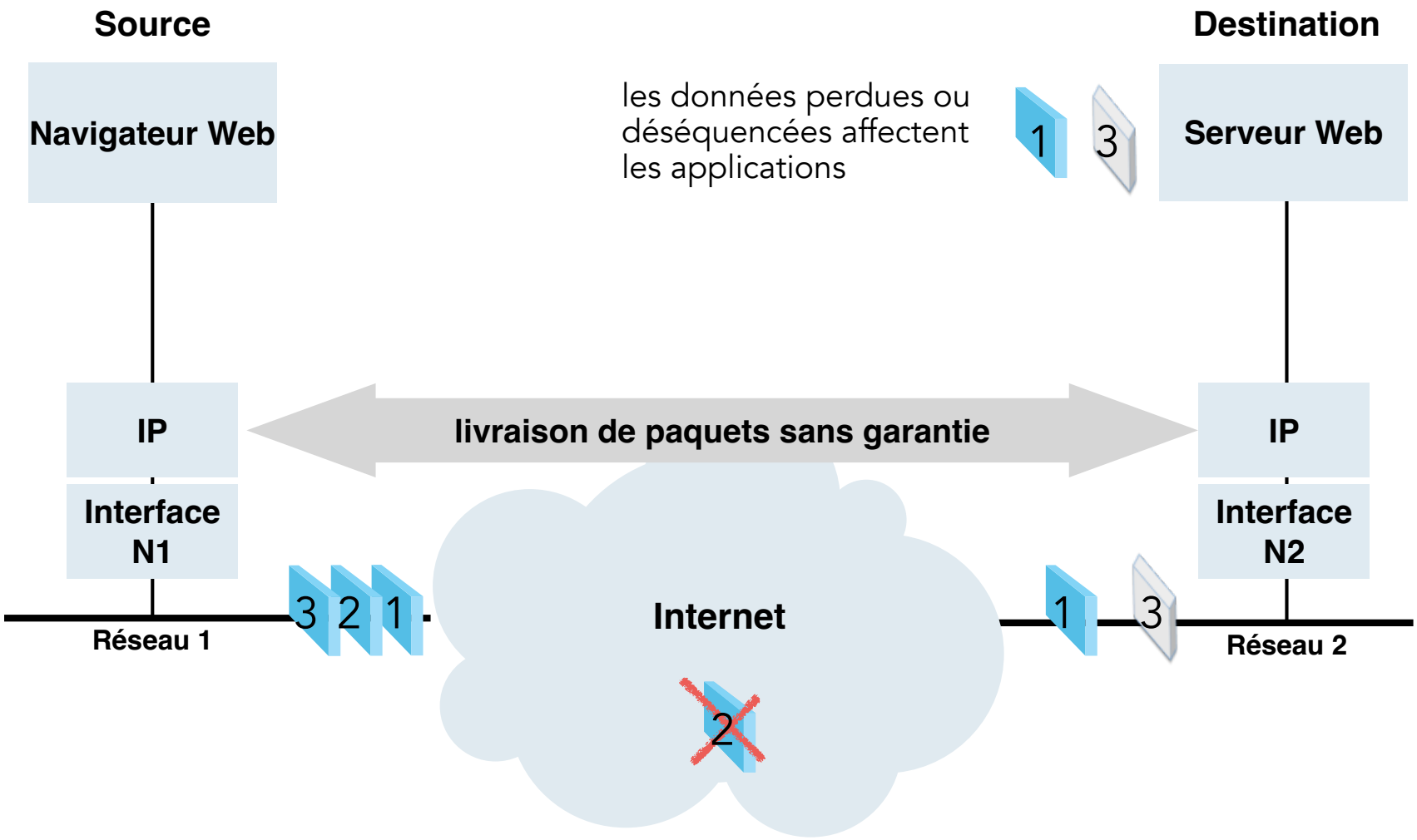
Classification des applications (2)

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

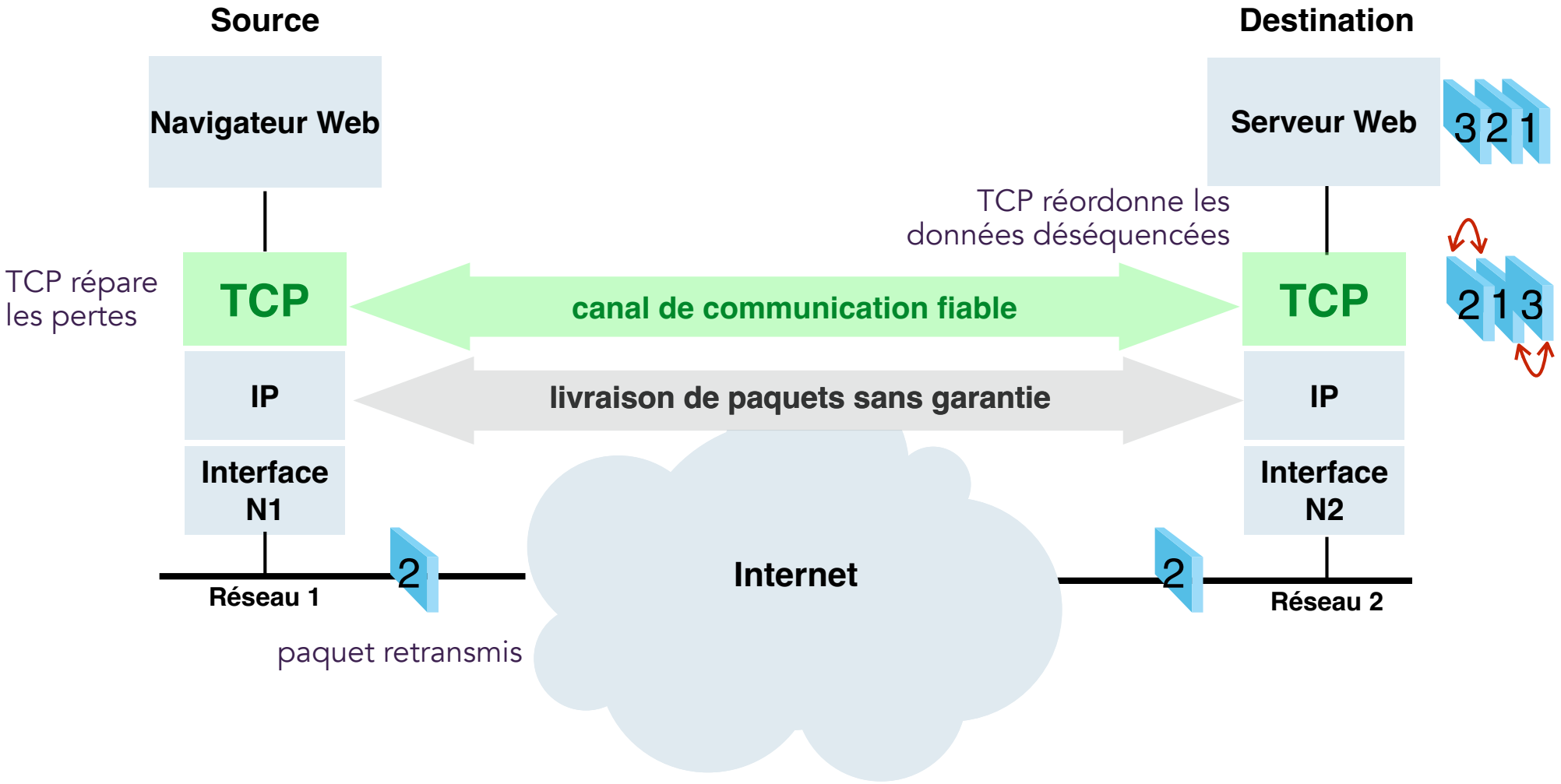
TCP

UDP

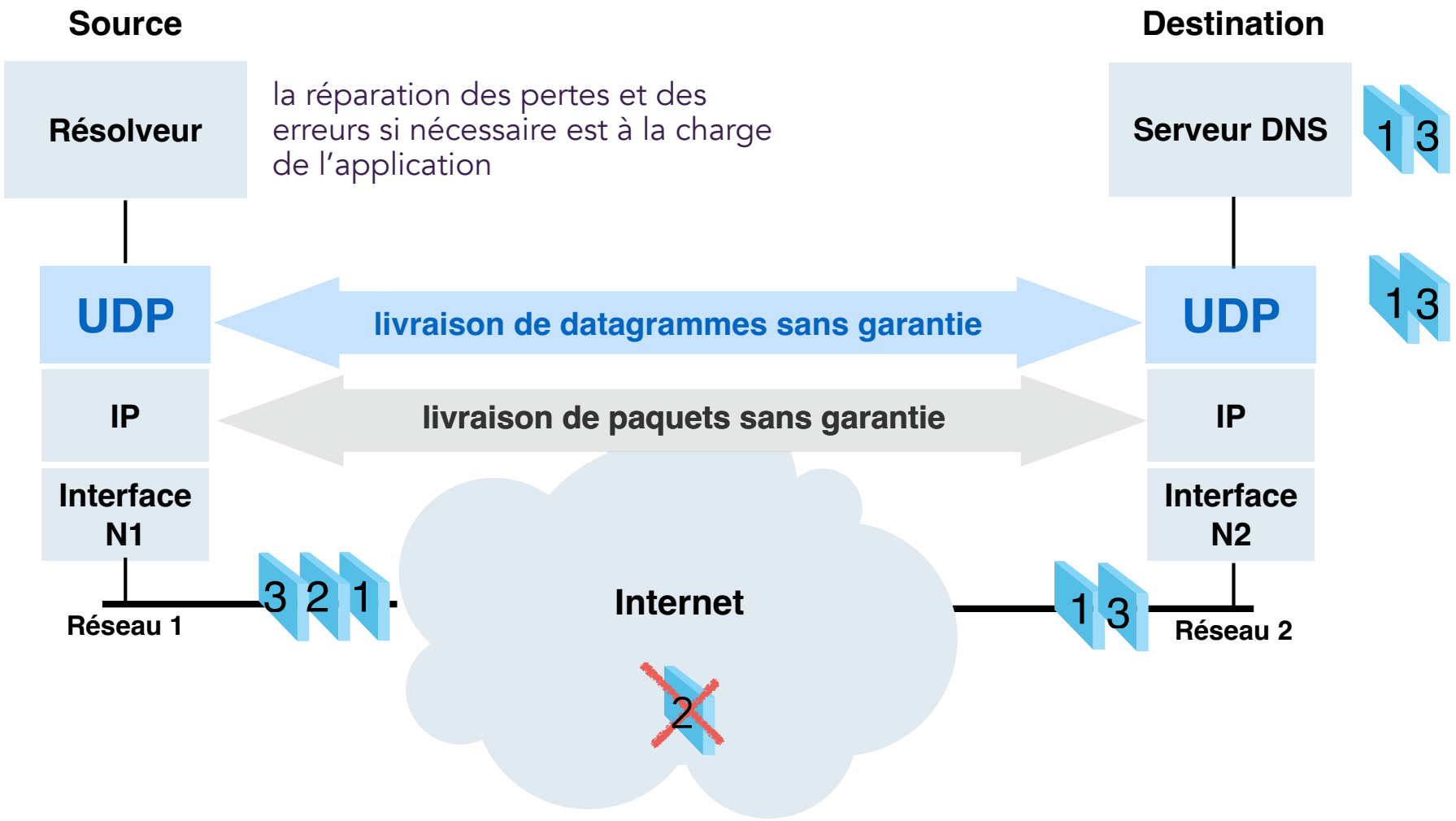
IP: Acheminement proche en proche sans garantie



TCP: livraison fiable bout en bout entre processus distants



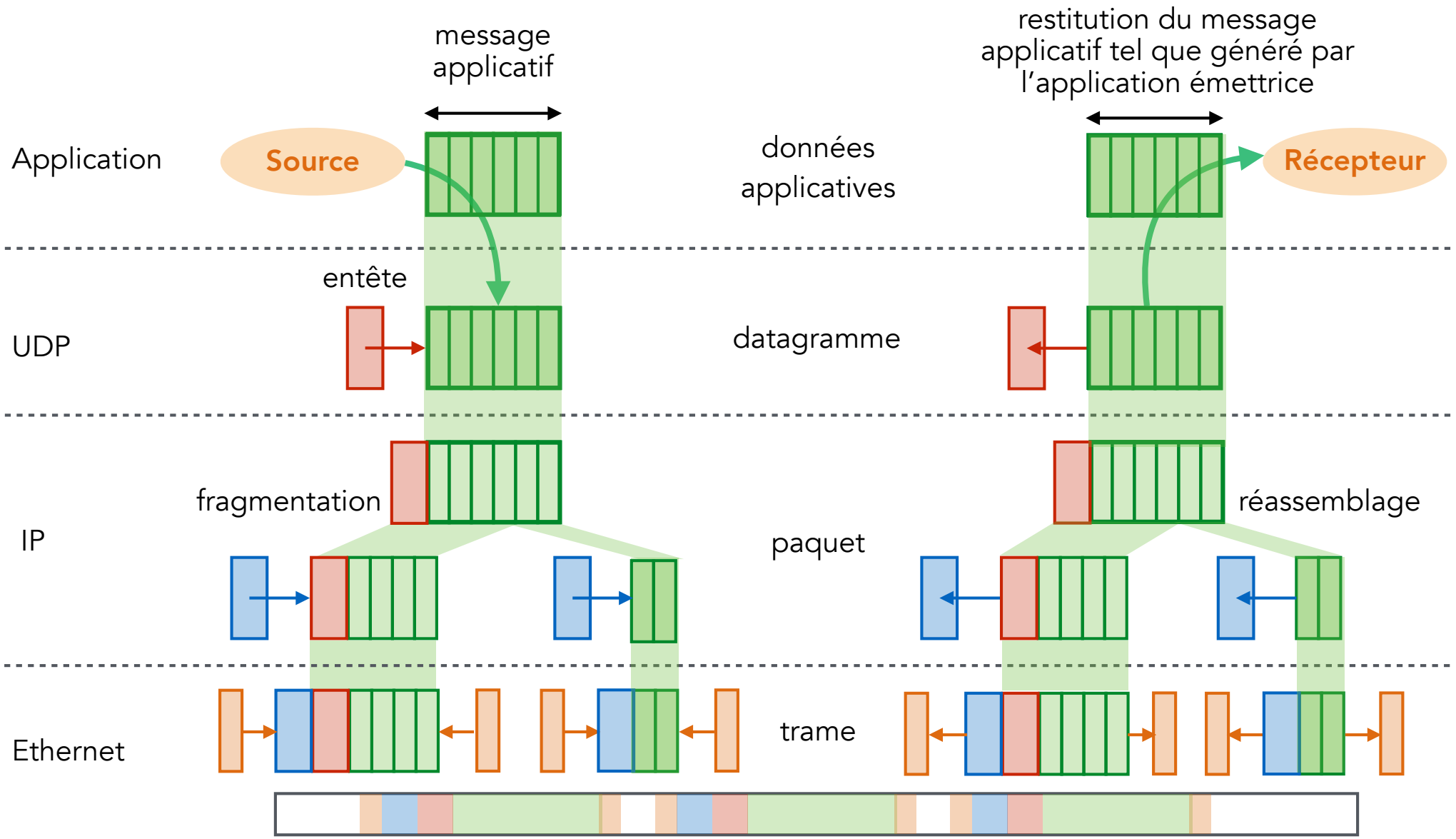
UDP: livraison bout en bout sans garantie entre processus distants



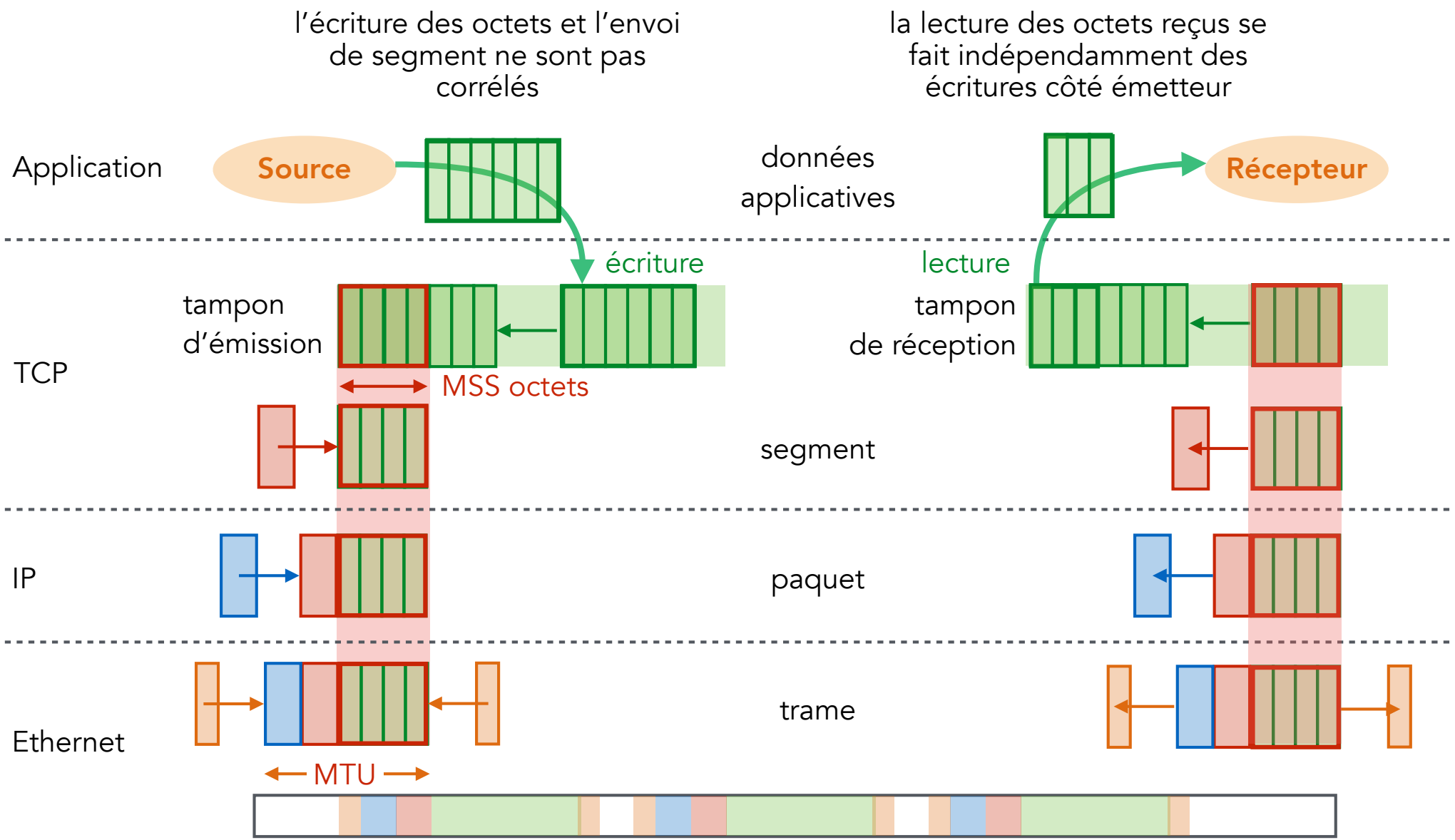
Rôle de la couche Transport (4)

- 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

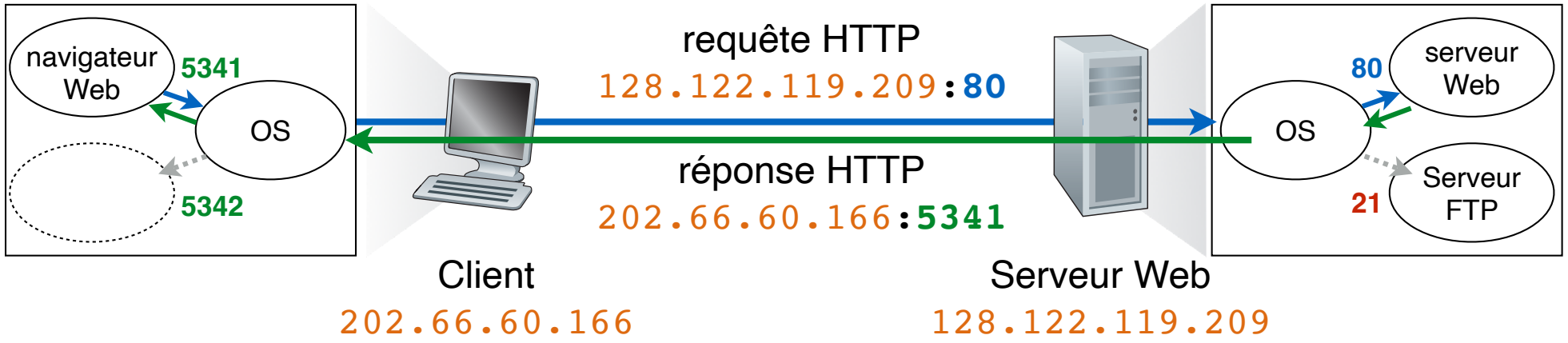
2 protocoles transport : TCP et UDP

Mode Non connecté vs Mode Connecté

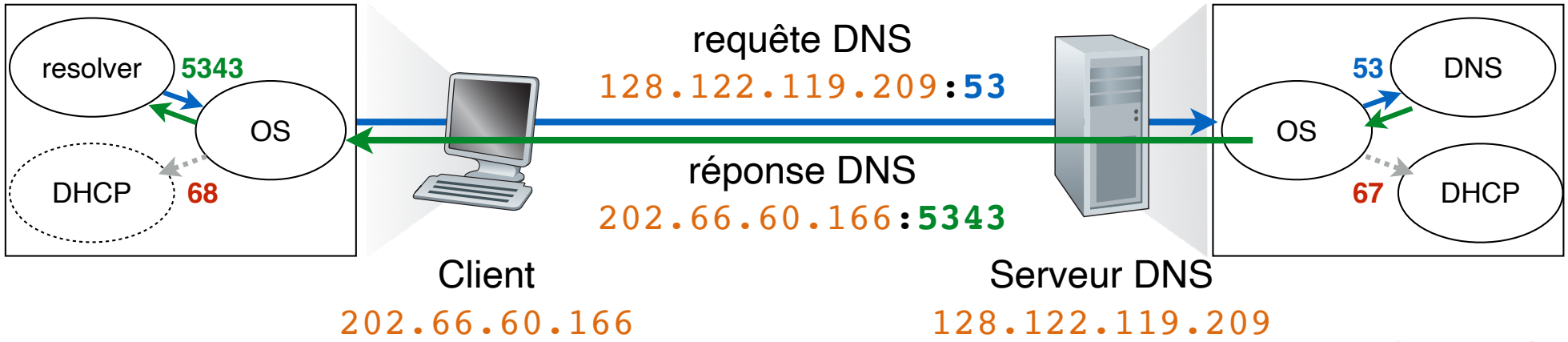
- UDP : mode non connecté
  - Envoi des données sans attendre l'établissement d'une connexion
    - adapté pour des échanges simples de type une requête-une réponse
  - Sans installer d'états
    - pas de tampons, pas de numéros de séquence, pas de fenêtres, pas de temporisateurs
- TCP : mode connecté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



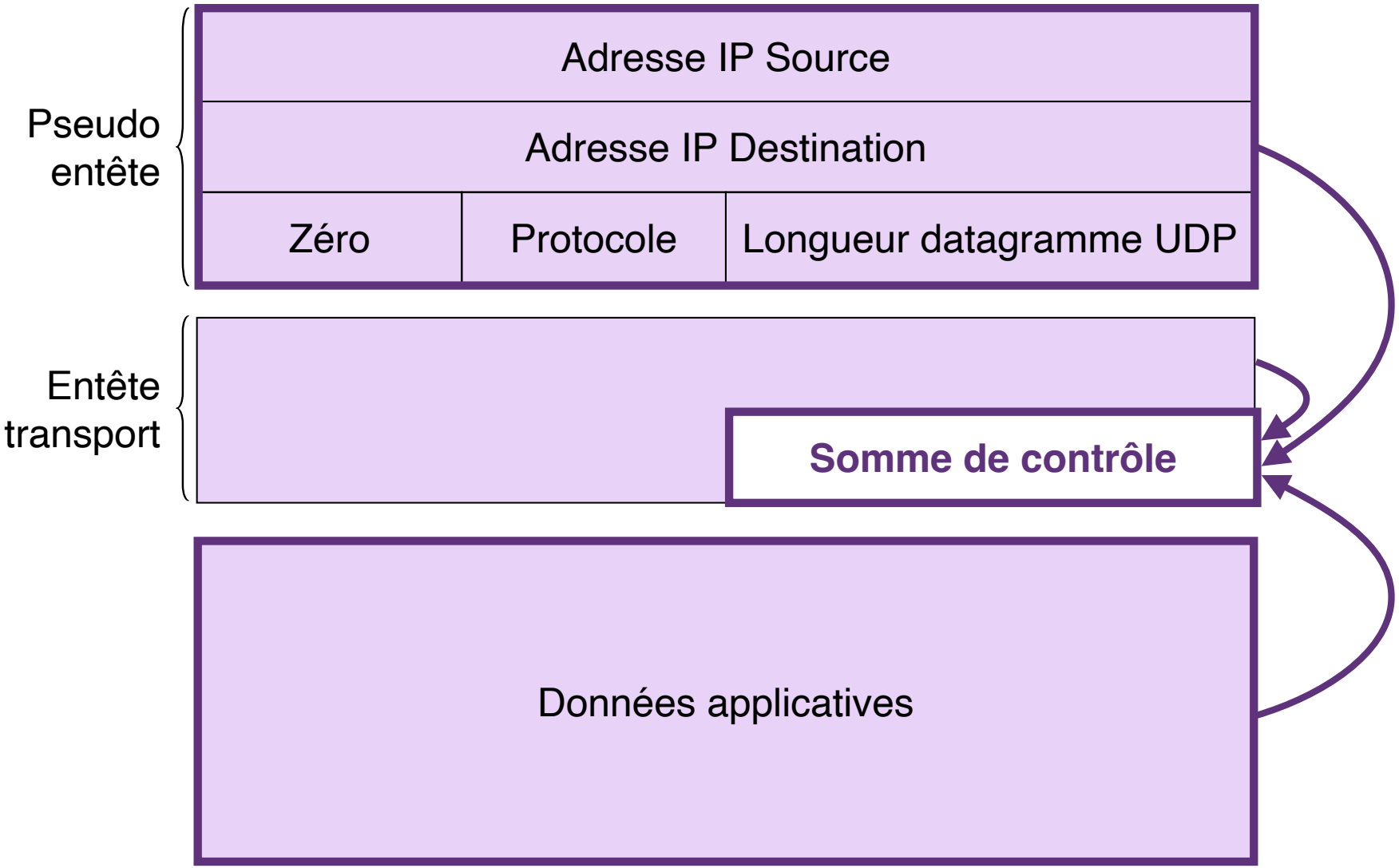
UDP



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, …

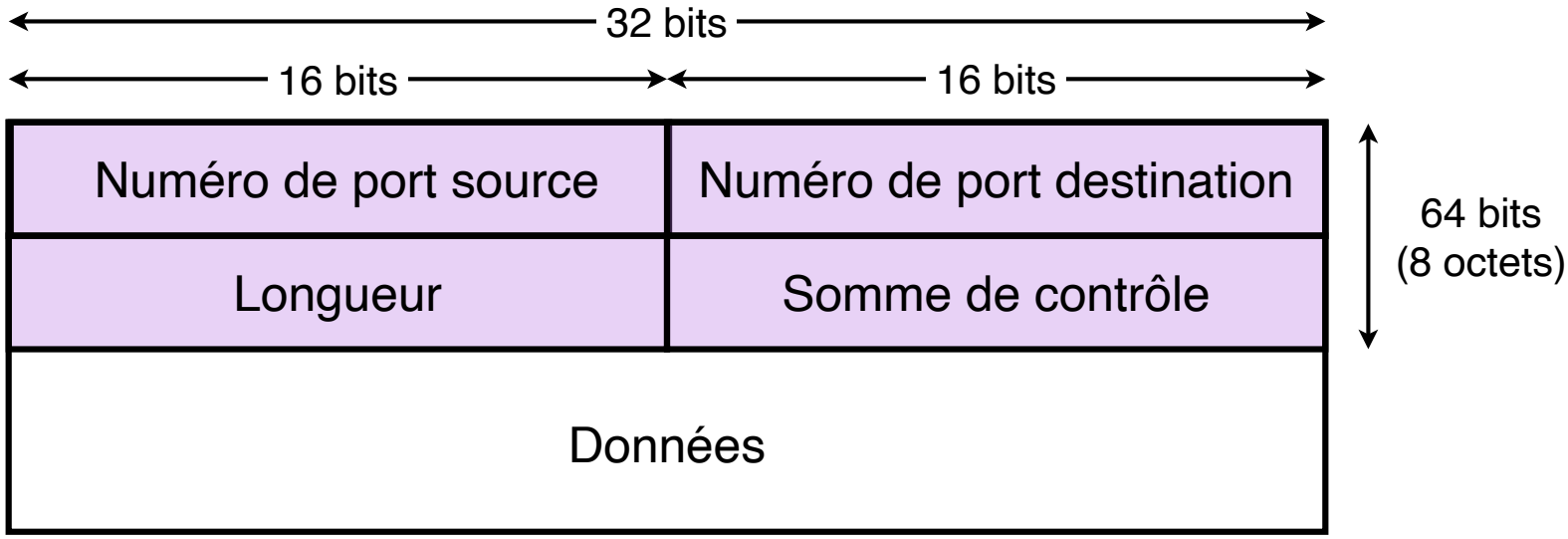
Somme de contrôle



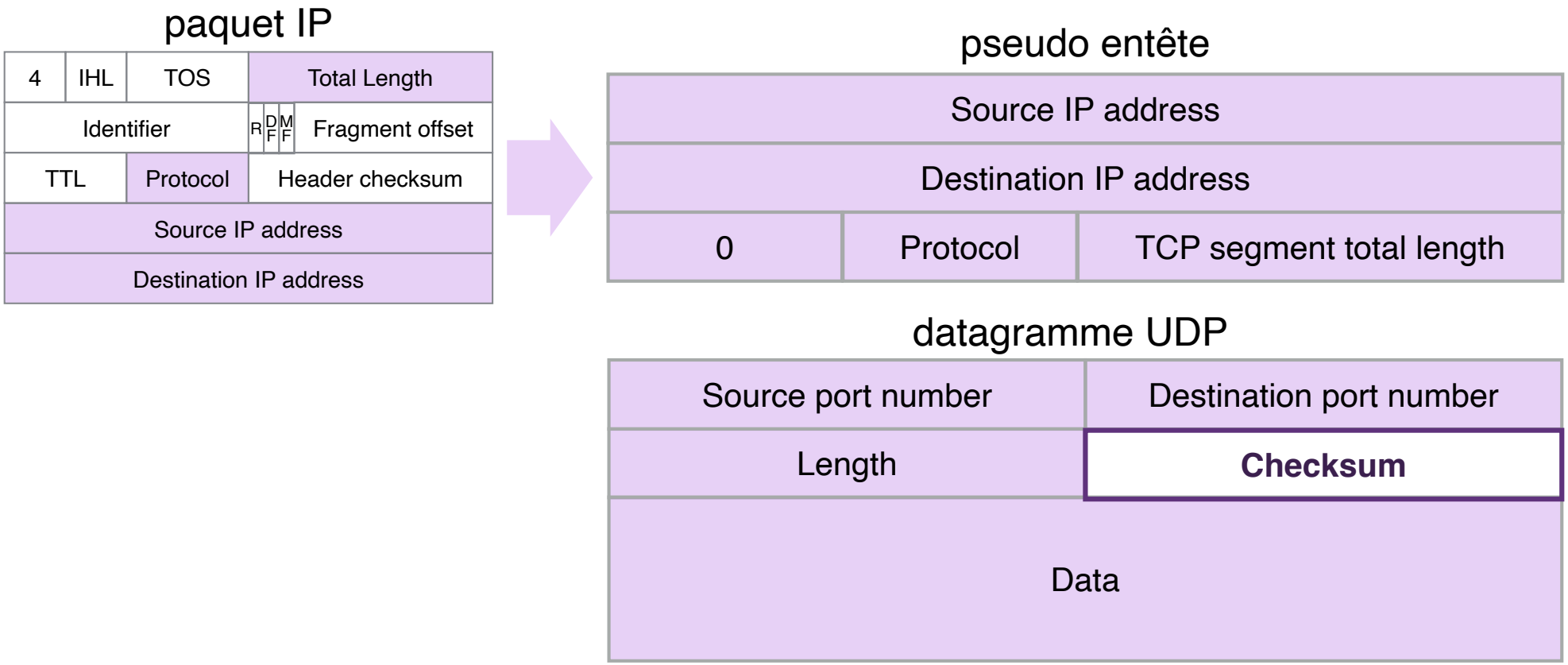
UDP
User Datagram Proctocol

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

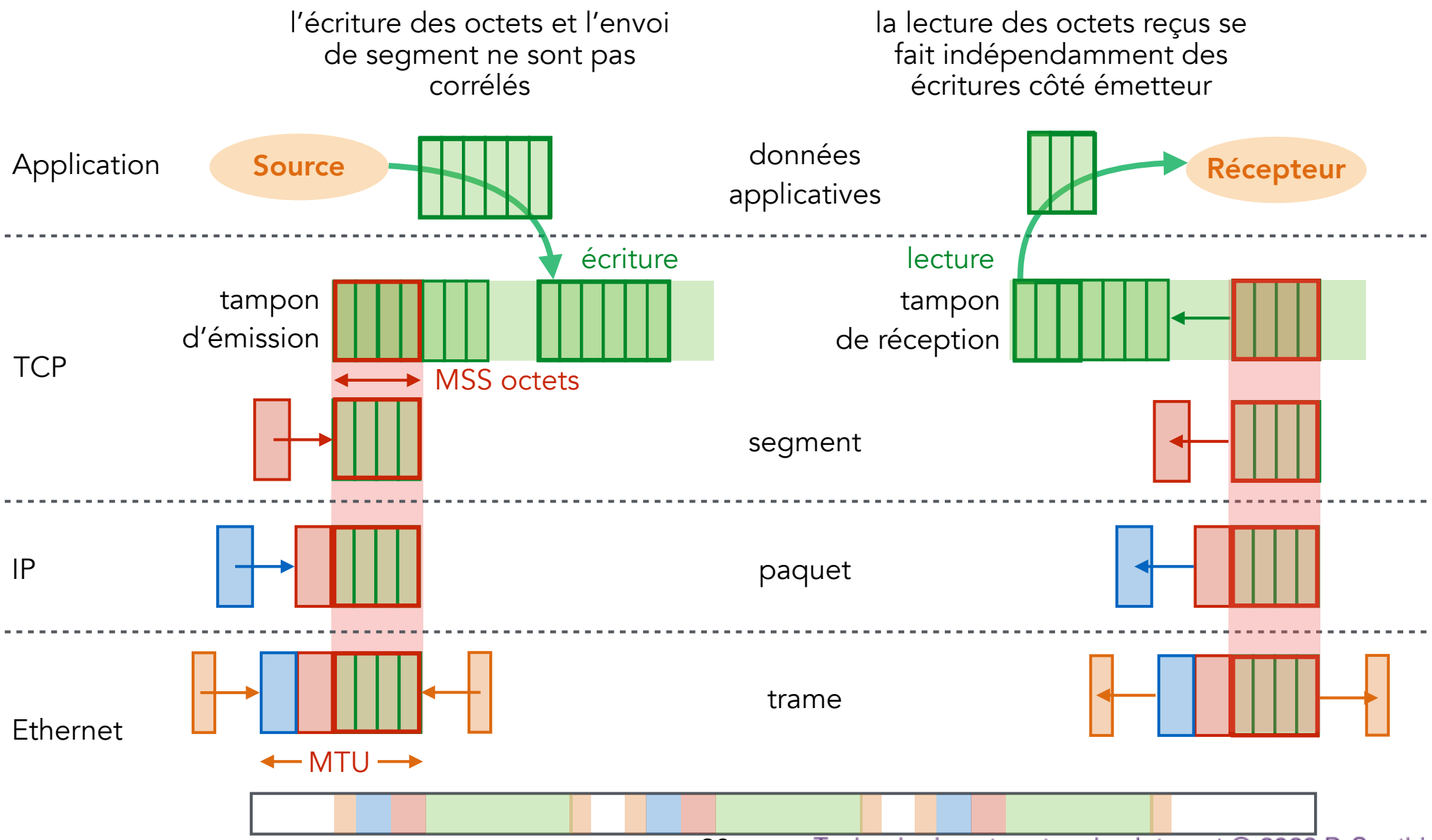
Transfert fiable

- 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

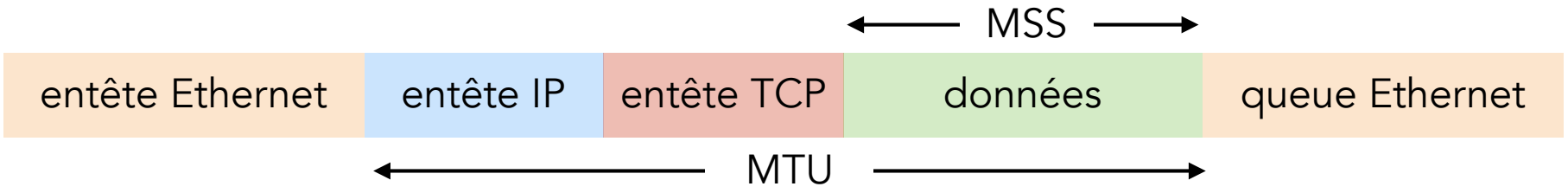
- TCP construit un segment si :
  - la charge utile des segments (MSS) est déterminée en fonction de la MTU locale
    - pour éviter la fragmentation IP
    - pour éviter l'envoi de trames à moitié pleine
- TCP évite les pertes :
  - le nombre d'octets de données envoyés est déterminé par :
    - la 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

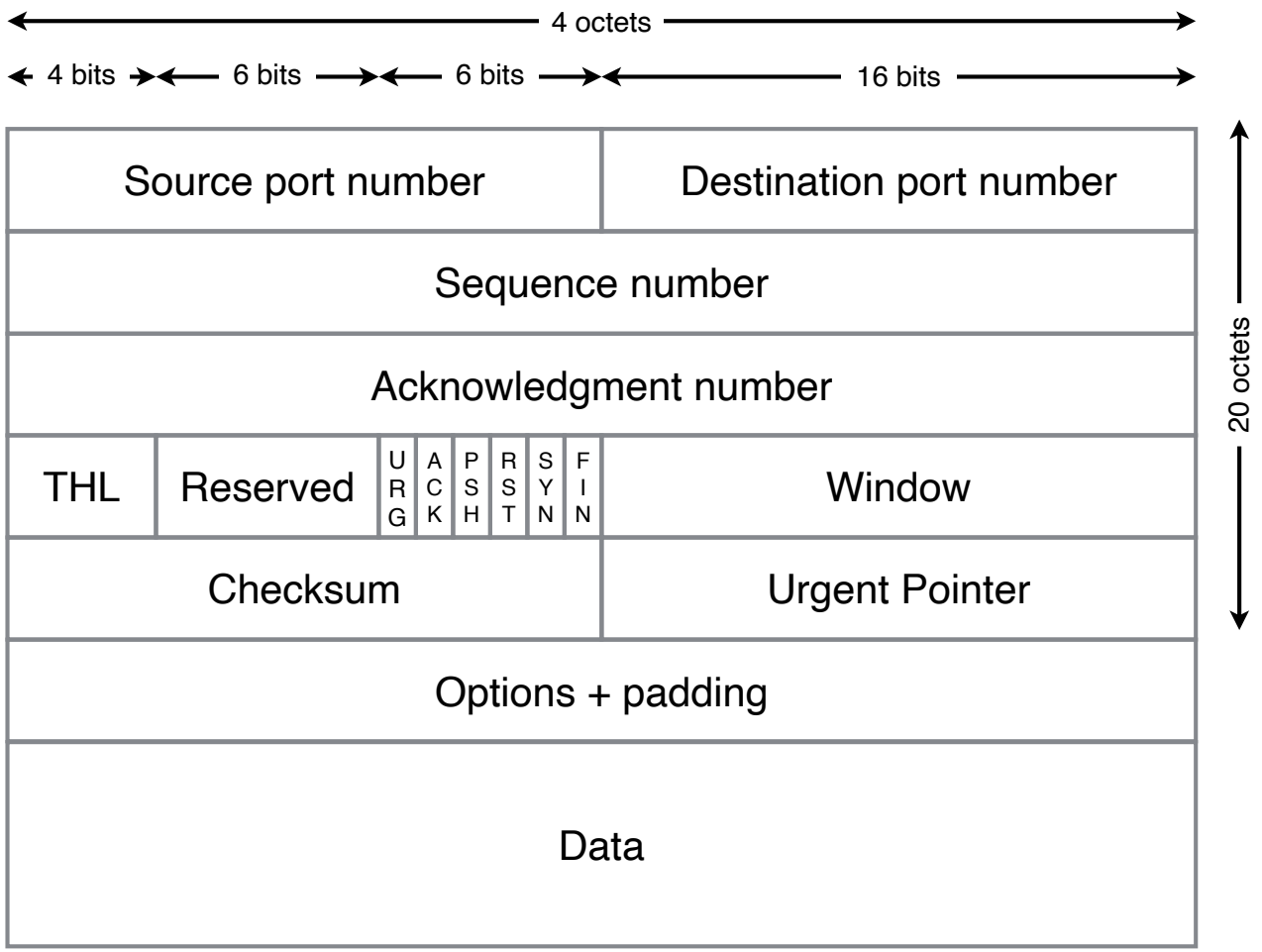


Constitution d'un segment TCP

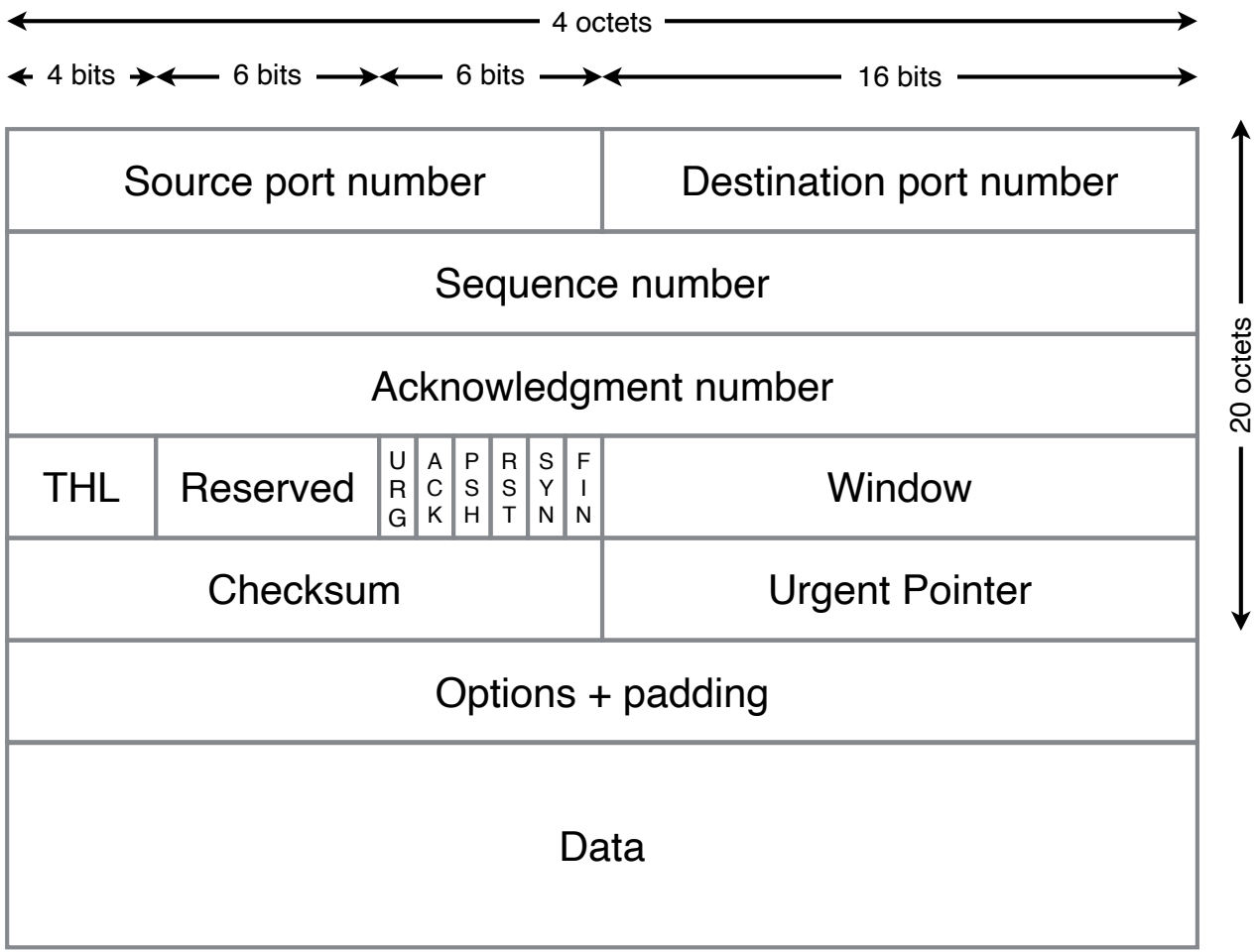
- L'efficacité d'une transmission se mesure selon le remplissage des trames
  - La MTU représente la taille max du champ données des trames
    - La MTU des trames Ethernet est de 1500 octets
- TCP évite l'envoi de trames à moitié pleine
  - TCP attend que l'application ait écrit MSS octets
    - MSS : 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



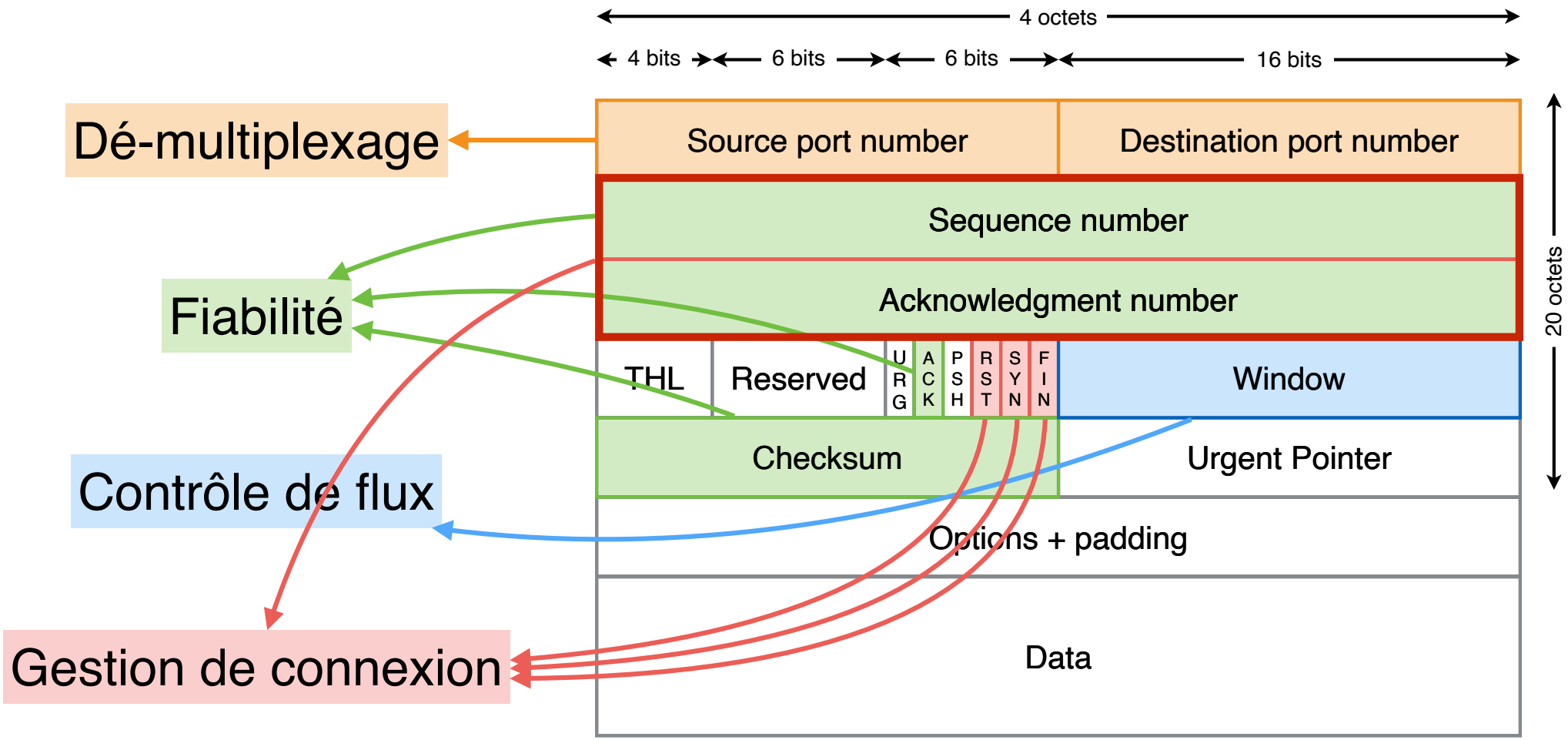
Entête TCP



Entête TCP

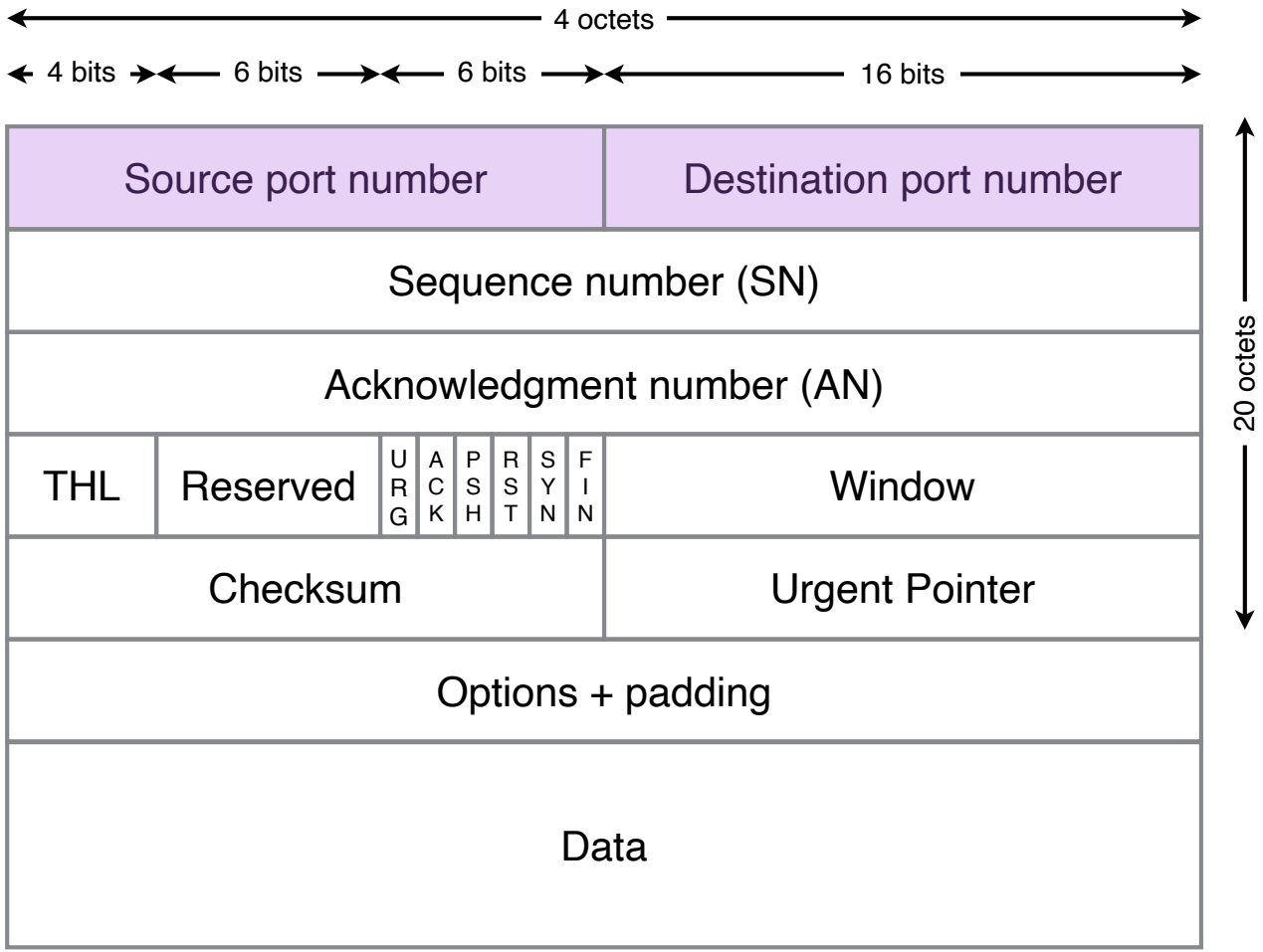

4 octets
4 bits
6 bits
6 bits
16 bits
Source port number
Destination port number
Sequence number
Acknowledgment number
THL
Reserved
URG
ACK
PSH
RST
SYN
FIN
Window
Checksum
Urgent Pointer
Options + padding
Data
20 octets

Entête TCP


4 octets
4 bits
6 bits
6 bits
16 bits
Dé-multiplexage
Source port number
Destination port number
Sequence number
Fiabilité
Acknowledgment number
THL
Reserved
URG
ACK
PSH
RST
SYN
FIN
Window
Checksum
Urgent Pointer
Contrôle de flux
Options + padding
Data
Gestion de connexion
20 octets

Numéros de port

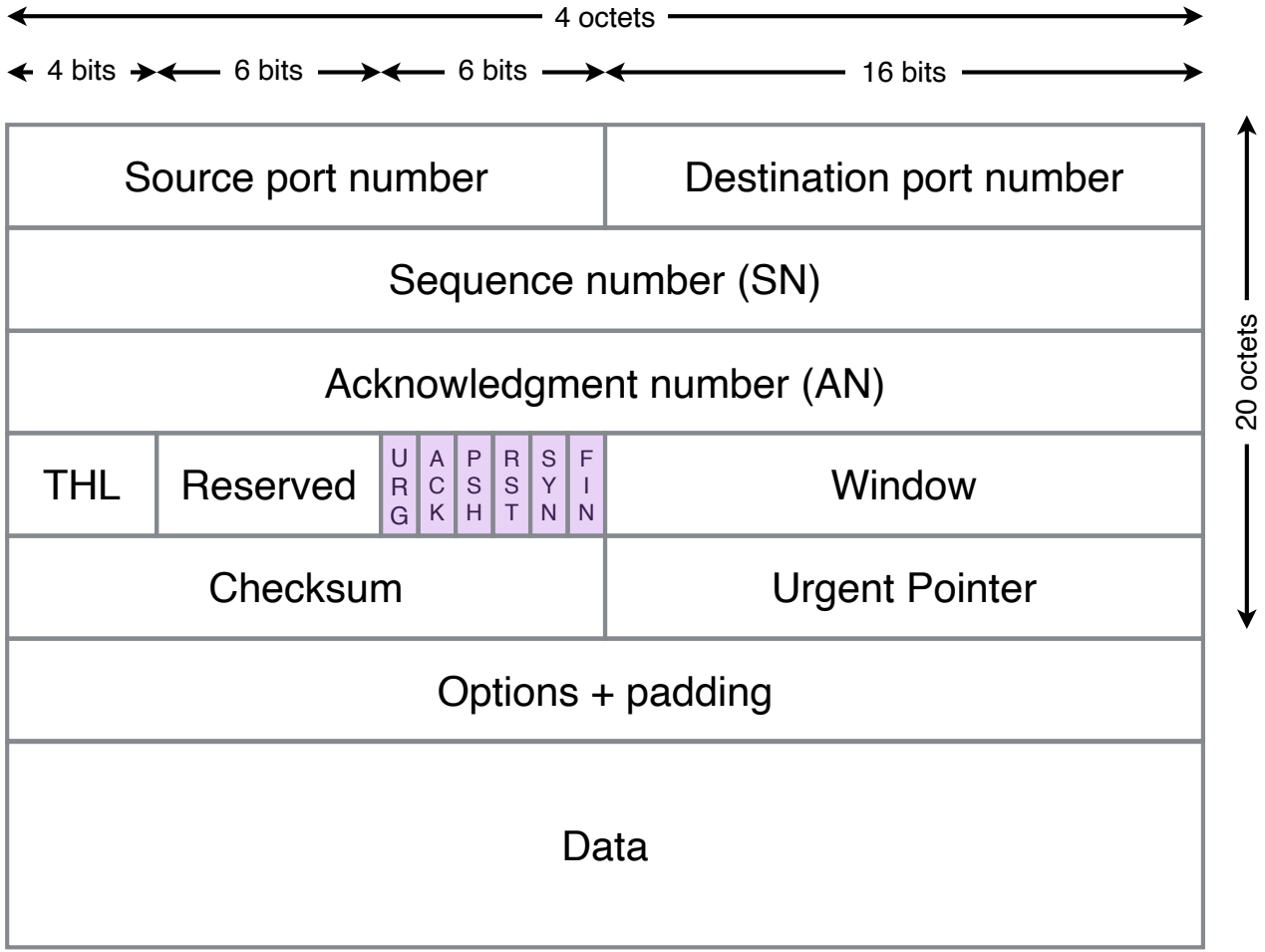
- Numéro de port source
  - identifie le processus qui émet le segment
- Numéro de port destination
  - identifie le processus à qui est destiné le segment
- Numéro de port client
  - valeur arbitraire
  - laissé au choix de l'OS
- Numéro de port serveur
  - selon le type de service
    - 80 : Serveur Web
    - …



Drapeaux TCP

- SYN, FIN, RST, et ACK
  - identifient 4 types de segment
  - les segments de données n'ont pas de drapeau dédié
- SYN
  - ouverture de connexion
- FIN
  - fermeture de connexion
- RST
  - réinitialisation de la connexion
- ACK
  - le champ AN est valide

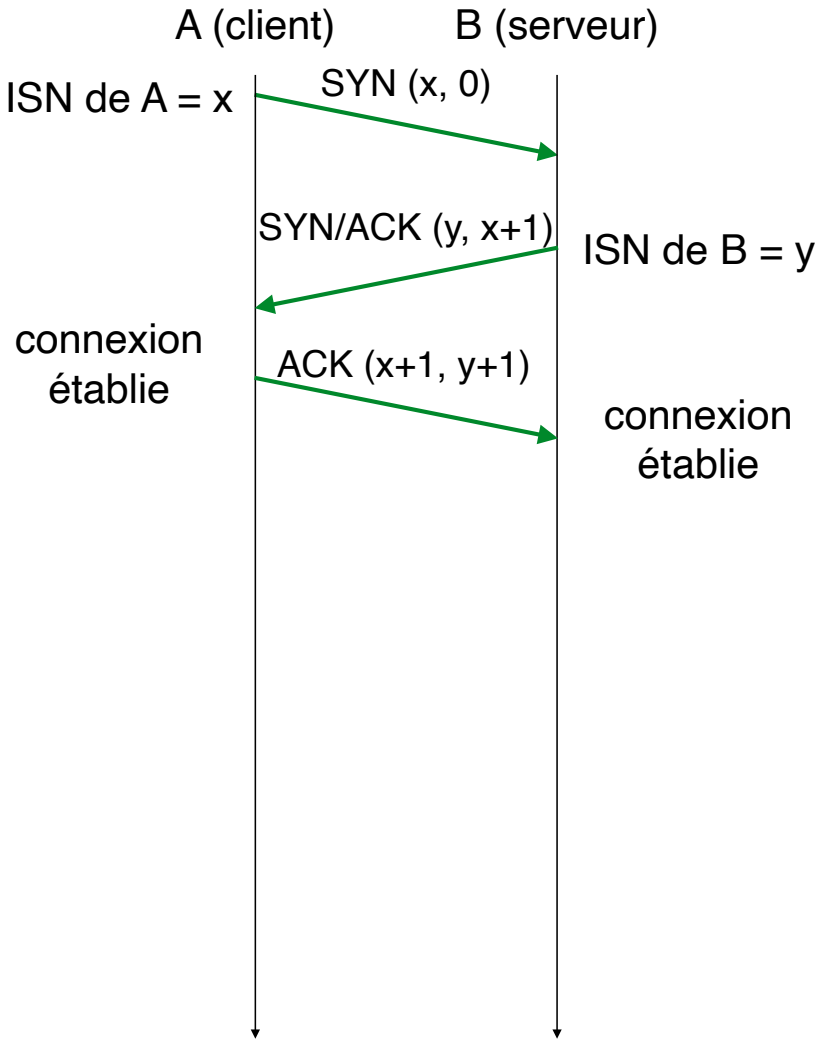
- PSH
  - les données doivent être lues au plus vite par l'application côté récepteur
- URG
  - nombre d'octets urgents



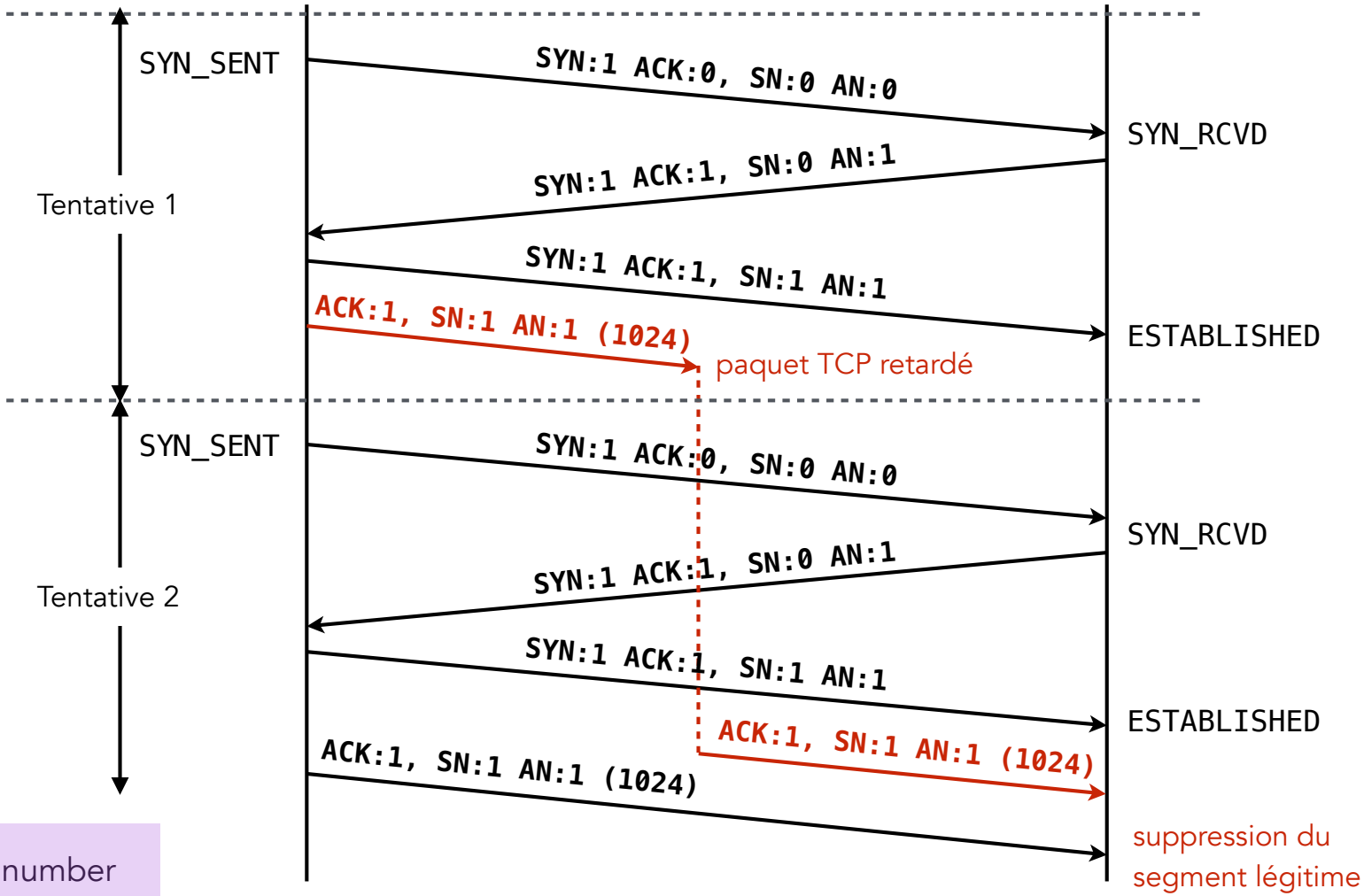
Types de segments TCP: SYN et FIN

Drapeaux SYN et ACK

- Segment SYN
  - Ouverture bilatérale d'une connexion
    - le client envoie un premier SYN auquel le serveur répond par un segment SYN-ACK
    - la connexion TCP est établie une fois le SYN du serveur acquitté par le client
  - Synchronisation des numéros de séquence
    - le champ Sequence Number contient la valeur de l'ISN Initial sequence number
    - choisie aléatoirement
  - Paramétrage de la connexion (options TCP de l'entête des SYN)
- Segments ACK
  - accusent la bonne réception des SYN :
    - en incrémentant le SN du SYN



Initial Sequence Number



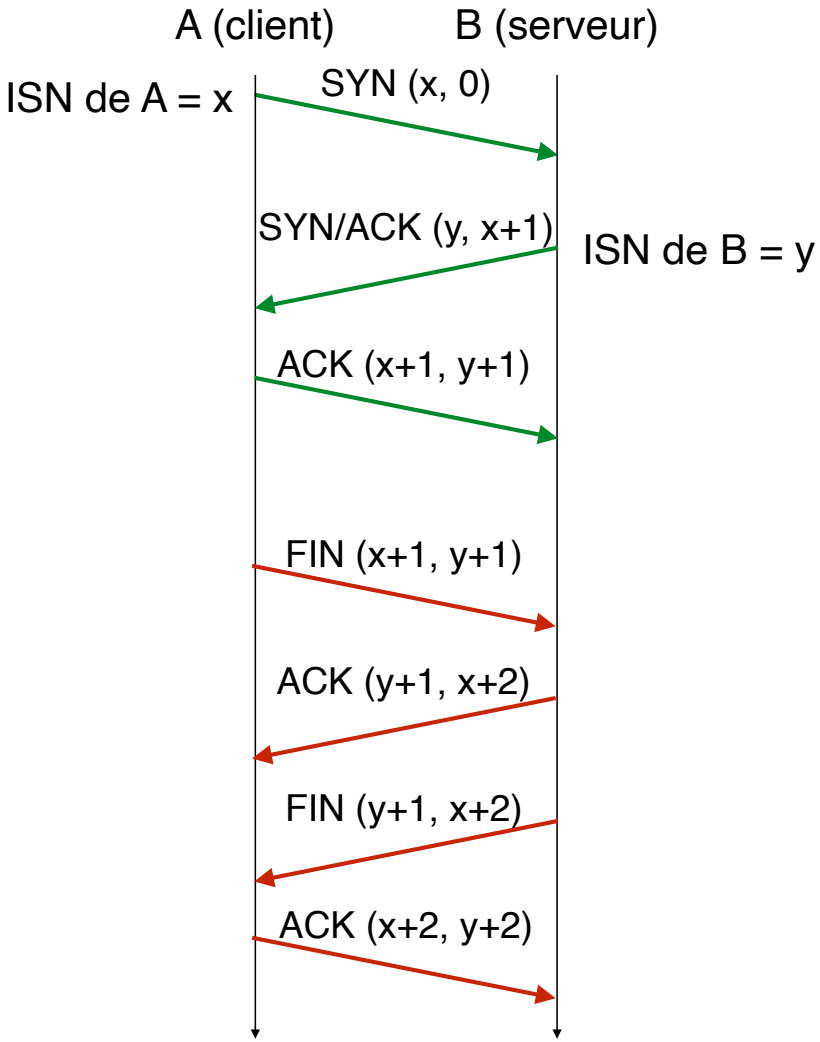
SN : sequence number

AN : ACK number

Types de segments TCP: FIN et ACK

Drapeaux FIN et ACK

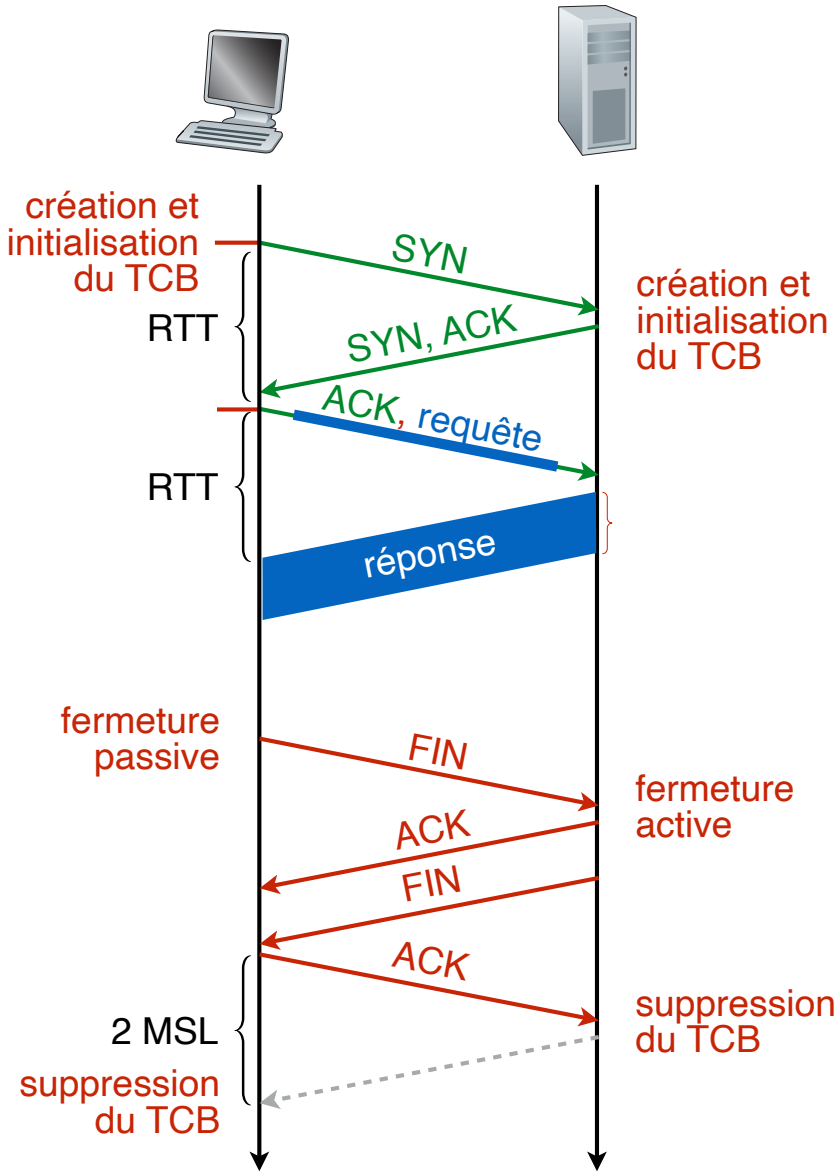
- Segment FIN
  - Fermeture bilatérale de connexion
    - serveur et client envoient un segment FIN
    - la connexion est fermée une fois les deux FIN acquittés
- Segments ACK
  - accusent la bonne réception des FIN :
    - en incrémentant le SN du FIN



Connexion TCP

- Connexion TCP
  - ouverture : poignée de main à 3 voies
    - segments SYN, SYN/ACK, ACK
  - fermetures : poignée de main à 4 voies
    - segments FIN, ACK
- Création et initialisation des TCB côté client et serveur
  - Les TCB Transport Control Block contiennent les informations d'état caractérisant l'échange
    - les identifiants du processus (adresse IP, numéro de port)
    - numéros de séquence des octets reçus en séquence et acquittés
    - les valeurs des fenêtres
- Les TCB sont supprimés après réception des FIN segments (four-way handshake)

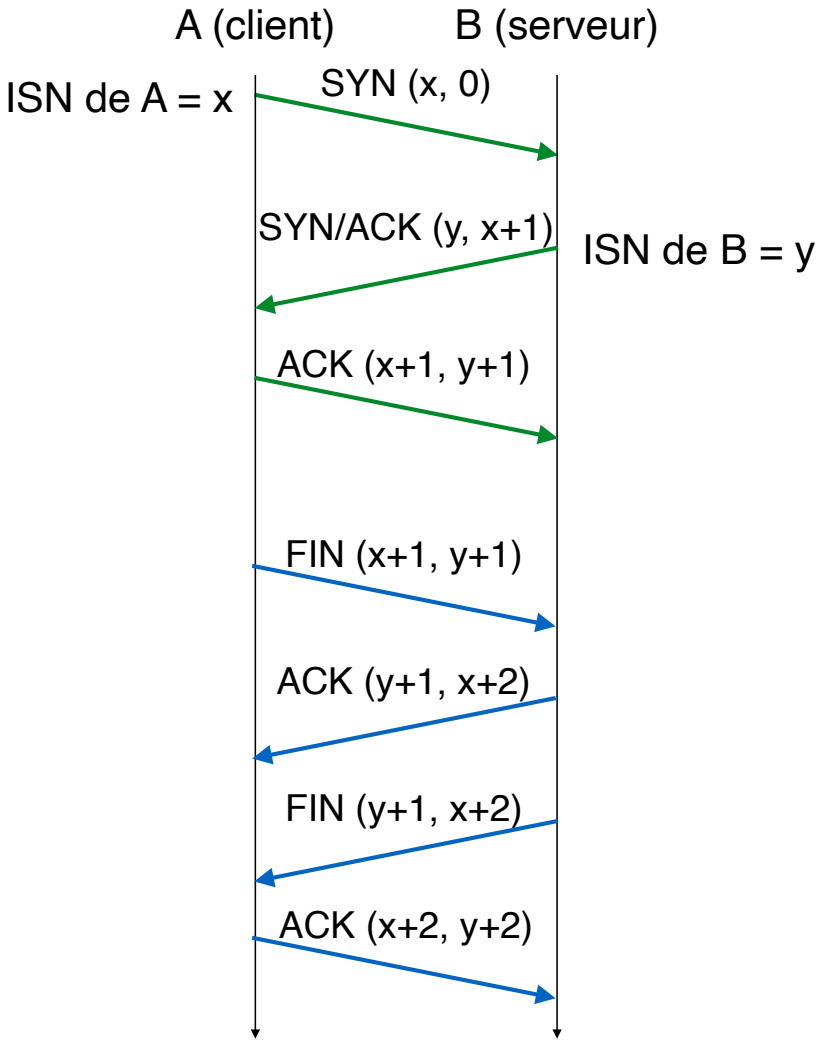
Une connexion TCP est la combinaison des TCB client et serveur et les informations d'état qu'ils contiennent



ISN aléatoire

Drapeaux SYN et FIN

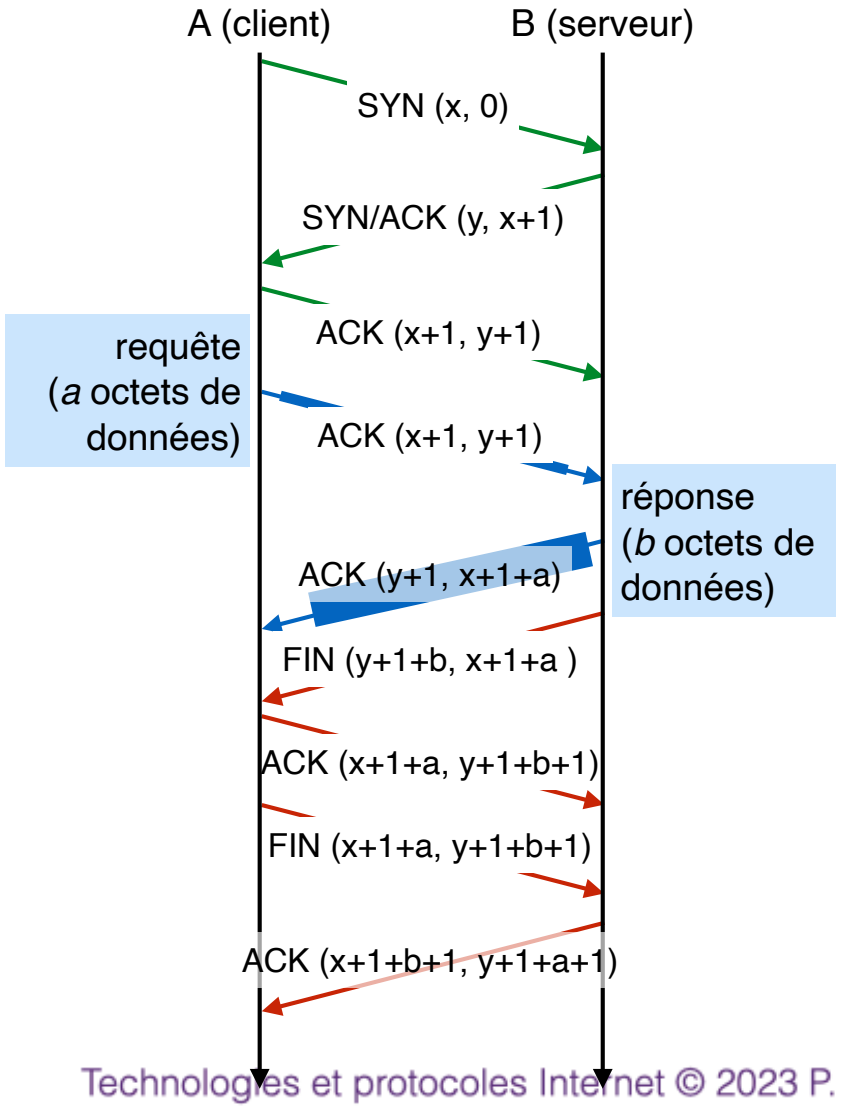
- Segment SYN
  - Ouverture unilatérale d'une connexion
    - le client envoie un premier SYN auquel le serveur répond par un segment SYN-ACK
    - la connexion TCP est établie une fois le SYN du serveur acquitté par le client
  - Synchronisation des numéros de séquence
  - Paramétrage de la connexion (options TCP de l'entête des SYN)
- Segment FIN
  - Fermeture unilatérale de connexion
    - serveur et client envoient un segment FIN
    - la connexion est fermée une fois les deux FIN acquittés



Types de segments TCP: ACK et données

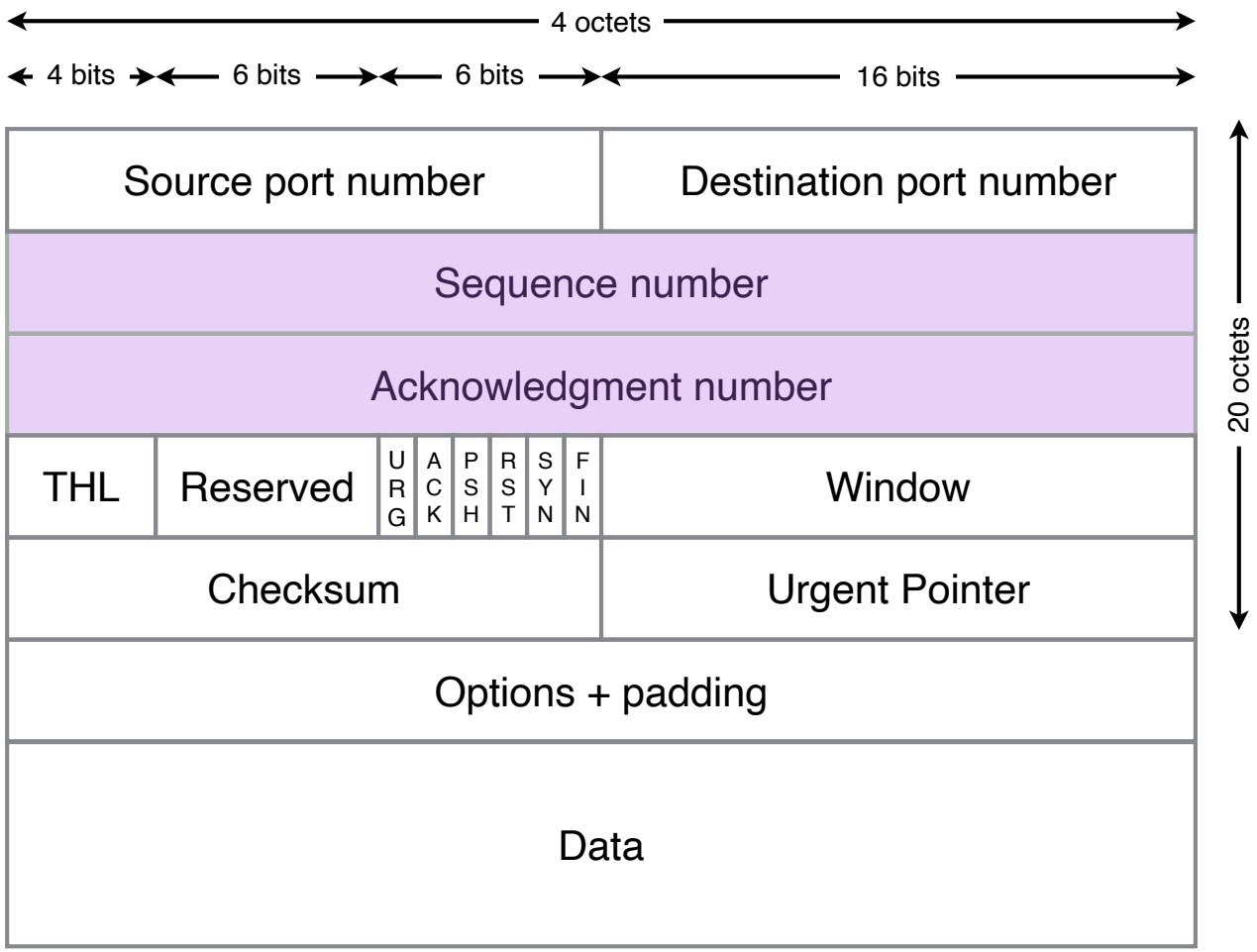
Drapeau ACK

- Segments ACK
  - accusent la bonne réception :
    - des SYN et des FIN
    - des octets de données correctement reçus
  - éventuellement "à cheval sur" (piggyback)
    - des segments SYN, FIN
    - des segments de données
- Segments de données
  - pas de drapeau explicite
  - drapeaux éventuellement positionnés :
    - ACK, PSH, et/ou URG



Numérotation des segments (1)

- Numéro de séquence
  - signification différente selon le type de segment
- Numéro d'acquittement
  - accuse la réception d'un segment en incrémentant la valeur du NS :
    - de 1 si segment SYN, FIN ou RST
    - du nombre d'octets reçus si segment de données



Numérotation des segments (2)

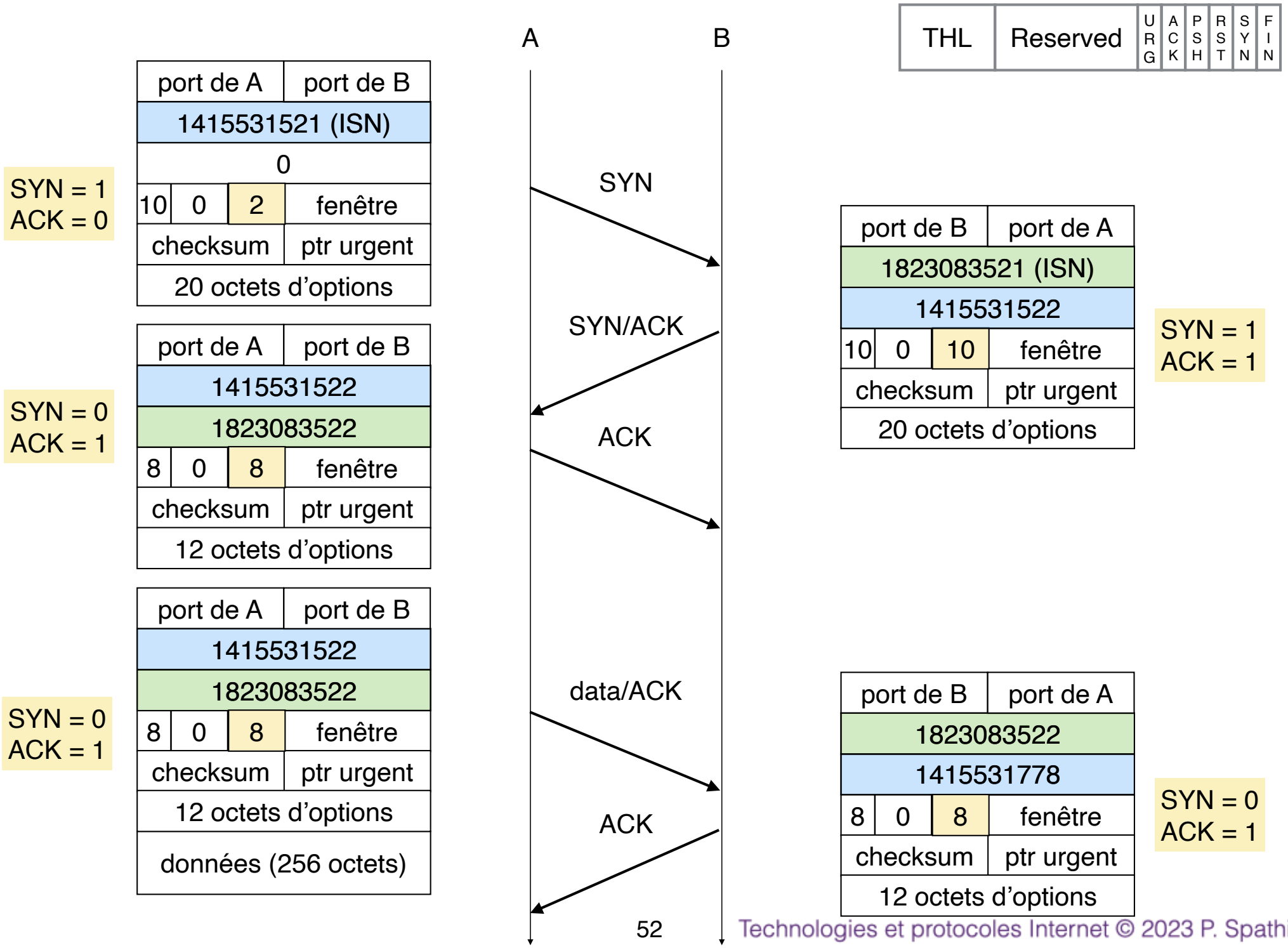
Champ Sequence Number

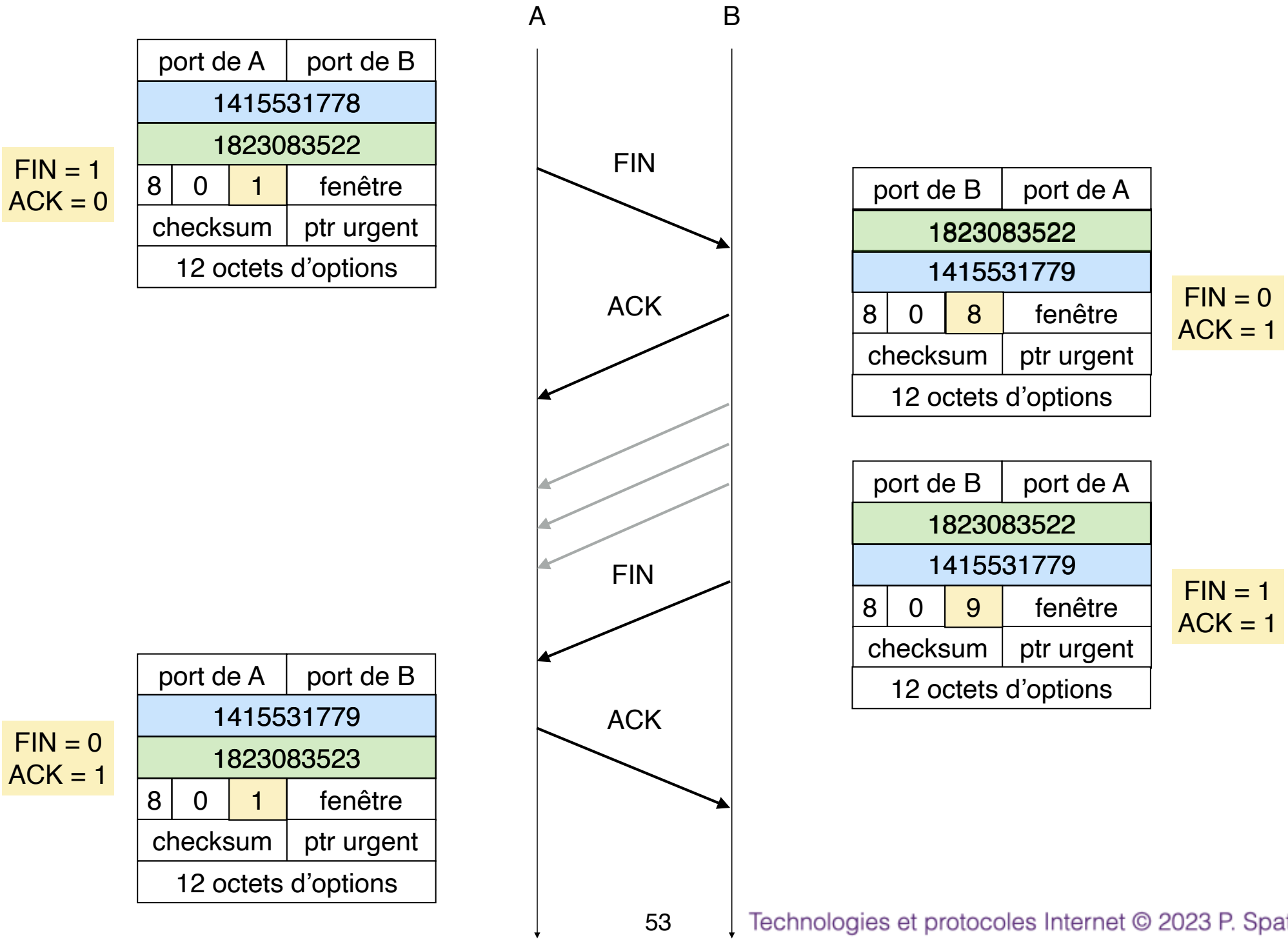
- TCP numérote les octets de données en séquence
  - la valeur initiale est choisie aléatoirement
- Segments SYN
  - le champ Sequence Number contient la valeur du numéro de séquence initiale (ISN)
    - Un SYN est acquitté par un segment ACK dont le AN incrémente la valeur de l'ISN
- Segments de données
  - le champ Sequence Number contient le numéro de séquence du premier octet de données transporté
- Segments FIN
  - la valeur du champ Sequence Number d'un segment FIN permet d'identifier le segment ACK retourné pour accuser la bonne réception du segment FIN
    - Un FIN est acquitté par un segment ACK dont l'AN incrémente le SN du segment FIN

Numérotation des segments (3)

Champ Acknowledgment Number

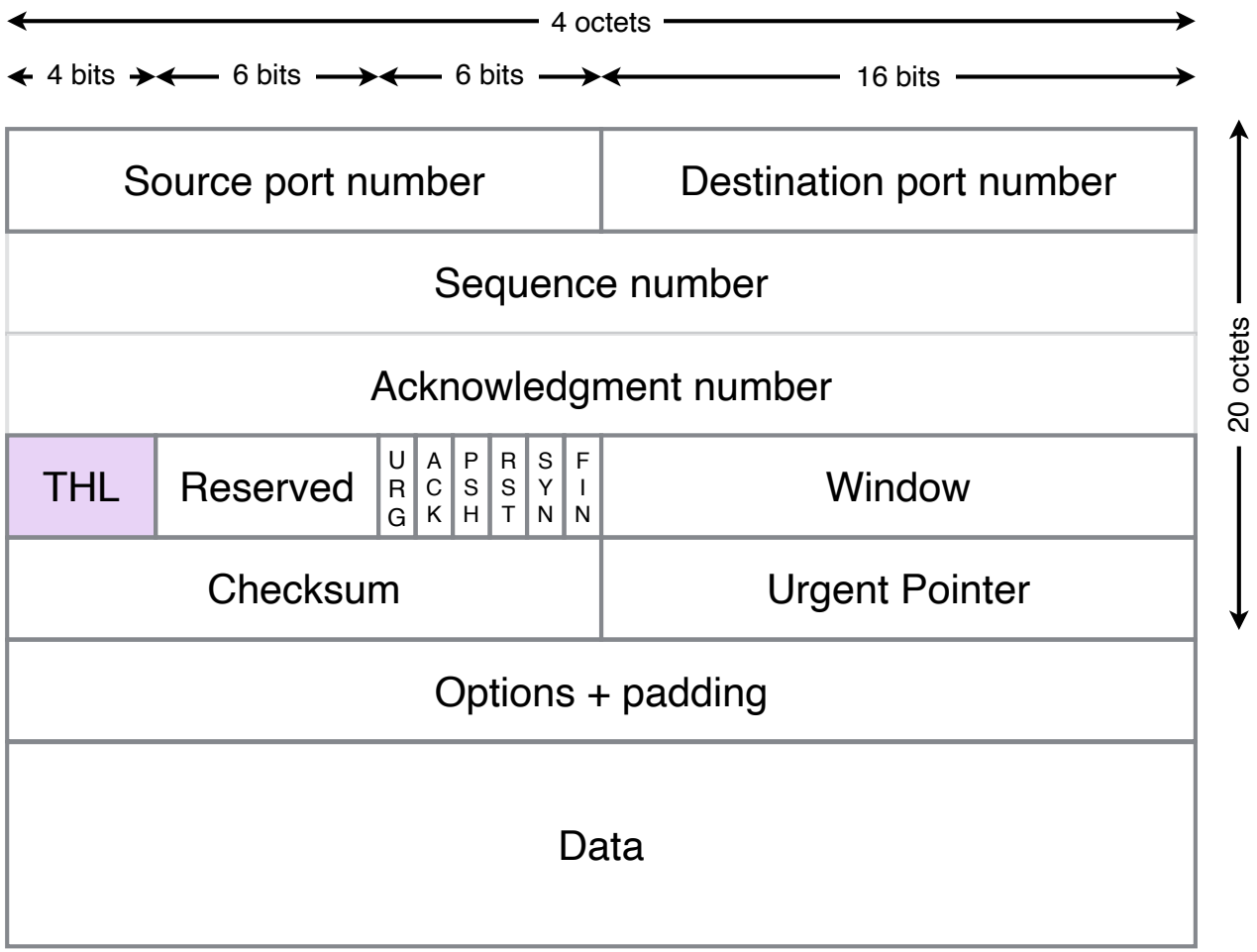
- Un segment TCP accuse les segments reçus si le drapeau ACK est positionné
- La valeur du Acknowledgment Number est calculée en incrémentant la valeur du SN du segment reçu :
  - de 1, si le segment reçu est un SYN, un FIN ou un RST
  - du nombre d'octets de données correctement reçus, si le segment reçu est un segment de données
- Les ACK sont éventuellement piggybackés
  - aux segments SYN, FIN et aux segments de données
- Le SN d'un segment ACK vide non piggybacké n'est pas pertinent
  - sa valeur reprend la valeur du prochain SN attendu le récepteur sans le consommer
  - le segment de données suivant reprend cette valeur pour son SN





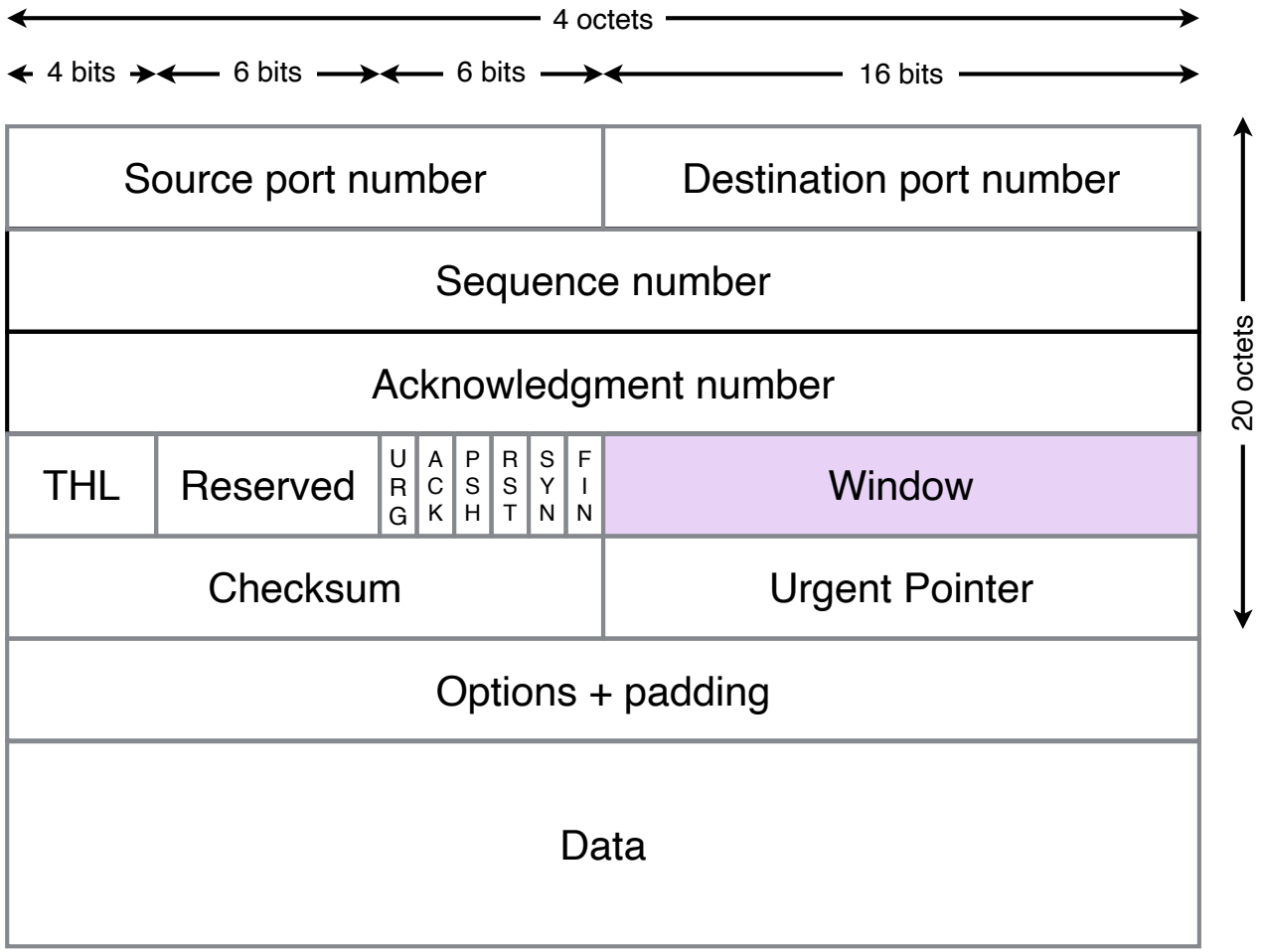
Champ THL Transport Header Length

- THL
  - taille de l'entête exprimée en mots de 32 bits (4 octets)
- Taille fixe
  - THL = 0x5 (0101) : 5
    - 20 octets
    - entête sans option
- Taille max
  - THL = 0xF (1111) : 15
    - taille totale 60 octets
    - options TCP 40 octets



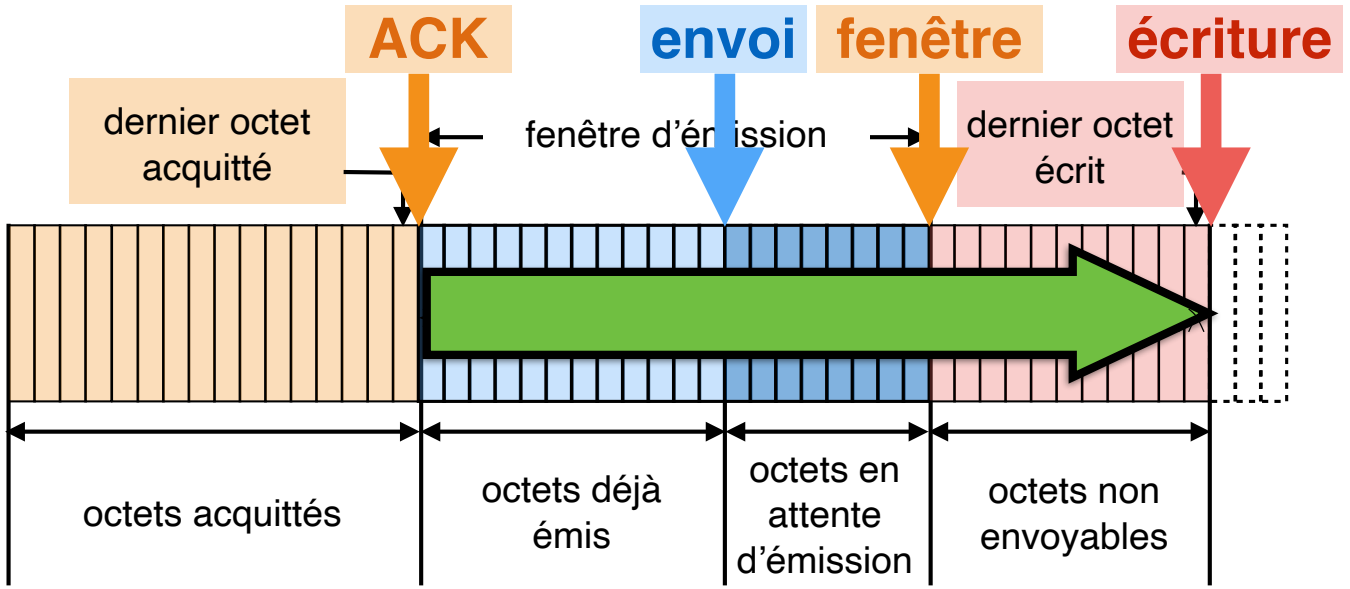
Champ Window

- Window
  - utilisé pour le contrôle de flux
  - indique la taille des tampons libres en réception de l'émetteur du segment



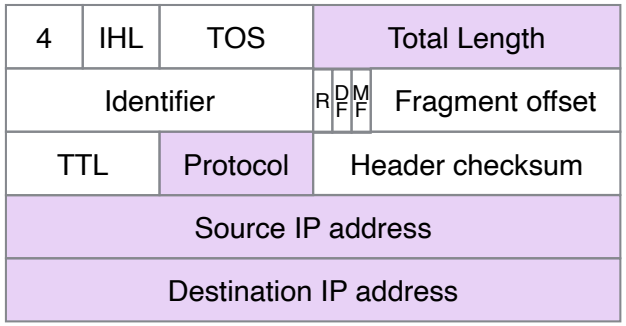
Contrôle de flux

- Le contrôle de flux évite les pertes découlant de l'engorgement des récepteurs
  - Un récepteur TCP stocke les octets reçus dans un tampon mémoire …
  - … le temps que l'application vienne les lire
  - L'absence de tampon en réception provoque la suppression des octets en excès
- Mécanisme de fenêtrage
  - La quantité d'octets que peut envoyer un émetteur est déterminée par sa fenêtre d'émission
  - La taille de la fenêtre d'émission est déterminée :
    - par la valeur du champ window contenu dans l'entête des segments que retourne le récepteur
    - cette valeur représente la quantité de ses tampons libres en réception

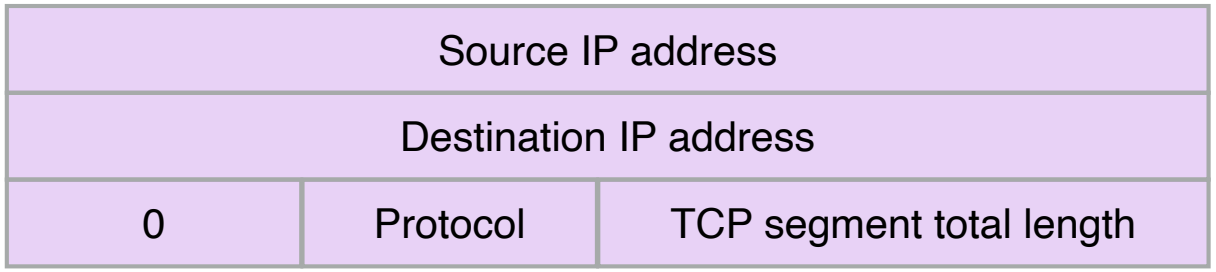


Champ Checksum

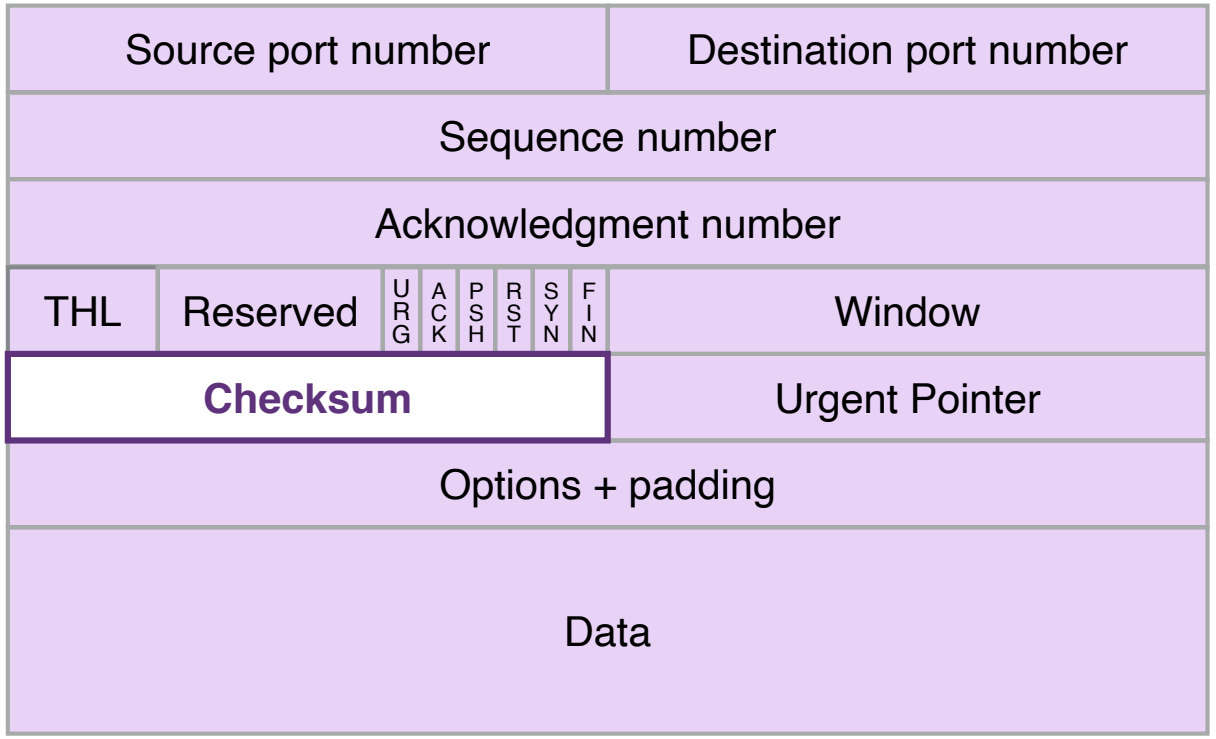
paquet IP



pseudo entête



segment TCP



Champs d'entête TCP (1)

- Source / destination port numbers (16 bits)
  - identifient les processus source et destination
- Sequence number (32 bits)
  - numéro de séquence du premier octet de données véhiculé dans le champ données du segment
- Acknowledgment number (32 bits)
  - valide si le drapeau ACK est positionné dans l'entête
  - numéro de séquence du dernier octet reçu correctement et en séquence + 1
  - identifié le numéro de séquence du prochain octet que l'émetteur du segment s'attend à recevoir
- THL (4bits)
  - longueur de l'entête exprimé en mots de 4 octets (32 bits)
  - 5 si pas d'option TCP, 40 octets max d'options TCP
- Reserved (6 bits)
  - champ réservé pour de futures utilisations

Champs d'entête TCP (2)

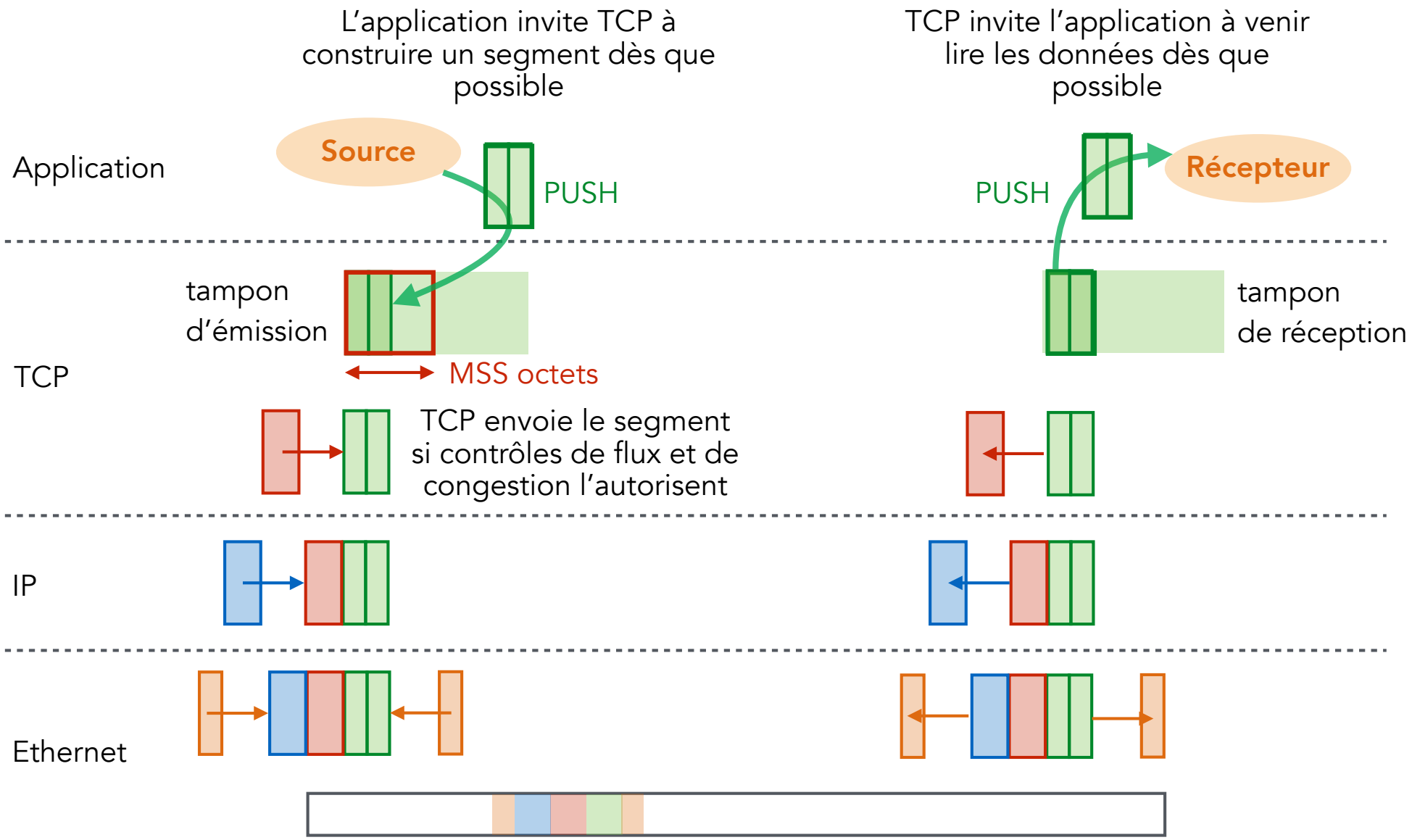
- Flags (6 bits) de gauche à droite
  - URG champ pointeur urgent valide
  - ACK champ numéro d'acquittement valide
  - RST réinitialisation de la connexion
  - PSH côté émetteur : l'application invite TCP à construire un segment sans attendre d'octets supplémentaires

    côté récepteur : l'application est invitée à lire les octets en attente dès que possible
  - SYN synchronisation des numéros de séquence
  - FIN l'émetteur du segment a fini d'envoyer des données et ferme son côté de la connexion

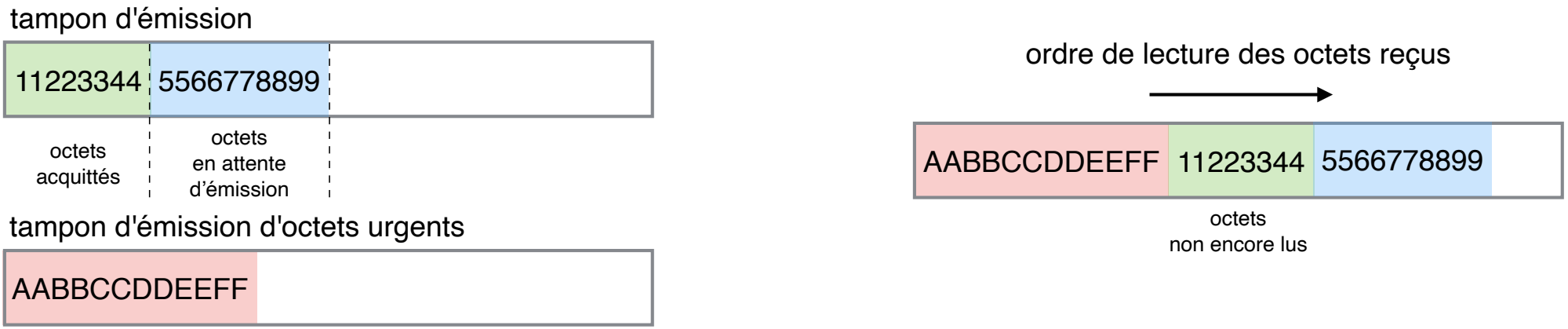
Drapeaux PUSH et URG

- Drapeau PUSH
  - côté émetteur :
    - l'application invite TCP à construire un segment sans attendre d'octets supplémentaires
  - côté récepteur :
    - l'application est invitée à lire les octets reçus dès que possible
- Drapeau URG et champ Urgent pointer (16 bits)
  - nombre d'octets urgents transportés dans la charge utile du segment (et les suivants)
  - ces octets sont lus en priorité par l'application réceptrice …
  - … sans attendre que les octets précédemment reçus soient préalablement lus

Drapeau PUSH

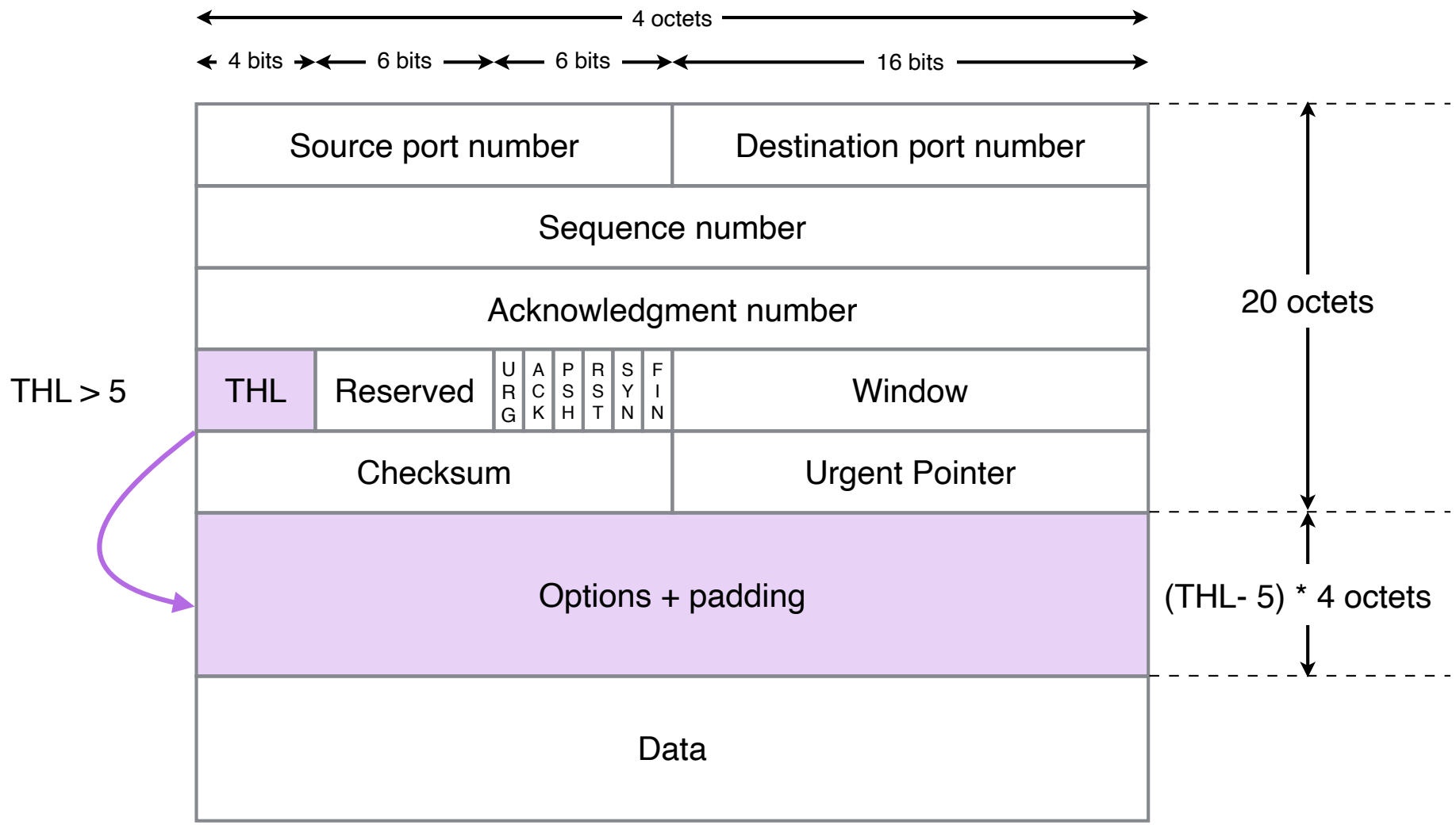


Drapeau URG et champ Urgent Pointer



Source port number								Destination port number	
Sequence number									
Acknowledgment number									
THL	Reserved			URG	ACK	PSH	RST	SYN	FIN	Window	
Checksum								Urgent Pointer: 0x0005	
A	A	B	B	C	C	D	D		
E	E	F	F	5	5	6	6		
7	7	8	8	9	9				

Options TCP (1)



TCP Options (2)

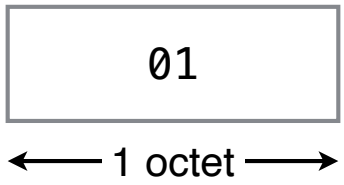
Kind	Length	Meaning	Reference
0x00	–	End of Option List	RFC 793
0x01	–	No-Operation	RFC 793
0x02	0x04	Maximum Segment Size	RFC 793
0x03	0x03	WSOPT - Window Scale	RFC 1323
0x04	0x02	SACK Permitted	RFC 2018
0x05	N	SACK (Selective ACK)	RFC 2018
0x06	0x06	Echo (obsoleted by option 8)	RFC 1072
0x07	0x06	Echo Reply (obsoleted by option 8)	RFC 1072
0x08	0x0A	TSOPT - Time Stamp Option	RFC 1323
0x09	0x02	Partial Order Connection Permitted	RFC 1693
0x0A	0x03	Partial Order Service Profile	RFC 1693
0x0B	–	CC	RFC 1644
0x0C	–	CC.NEW	RFC 1644
0x0D	–	CC.ECHO	RFC 1644
0x0E	0x03	TCP Alternate Checksum Request	RFC 1146
0x0F	N	TCP Alternate Checksum Data	RFC 1146

http://www.iana.org/assignments/tcp-parameters

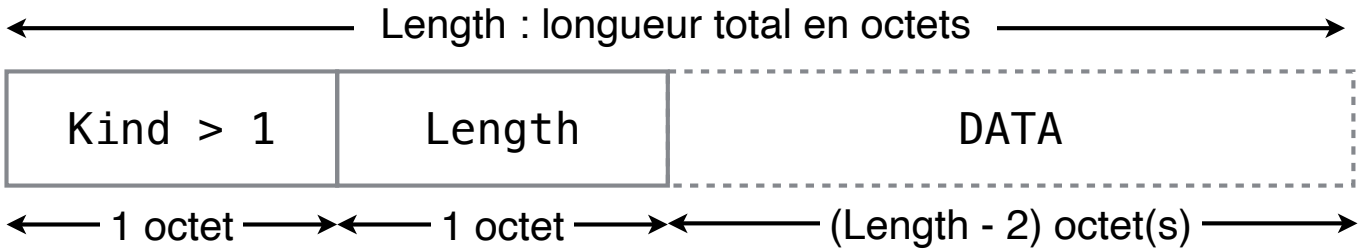
Options TCP (3)

Format de l'option EOL End of Options List (kind = 0)

Format de l'option NOP No OPeration (kind = 1)

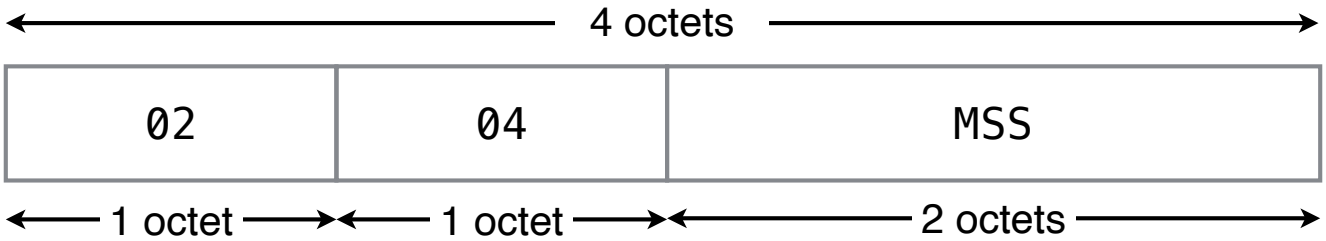


Format des options de type kind > 1

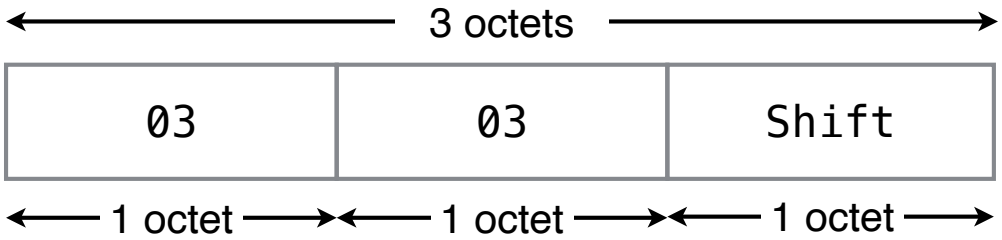


Options TCP (4)

Maximum Segment Size (MSS)

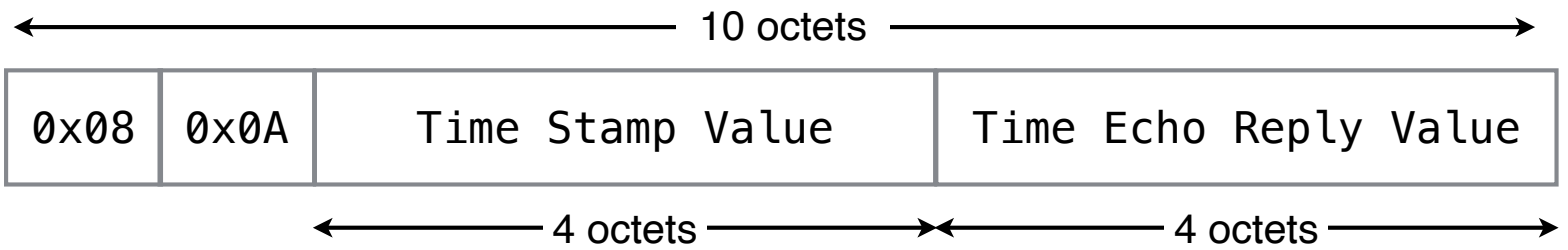


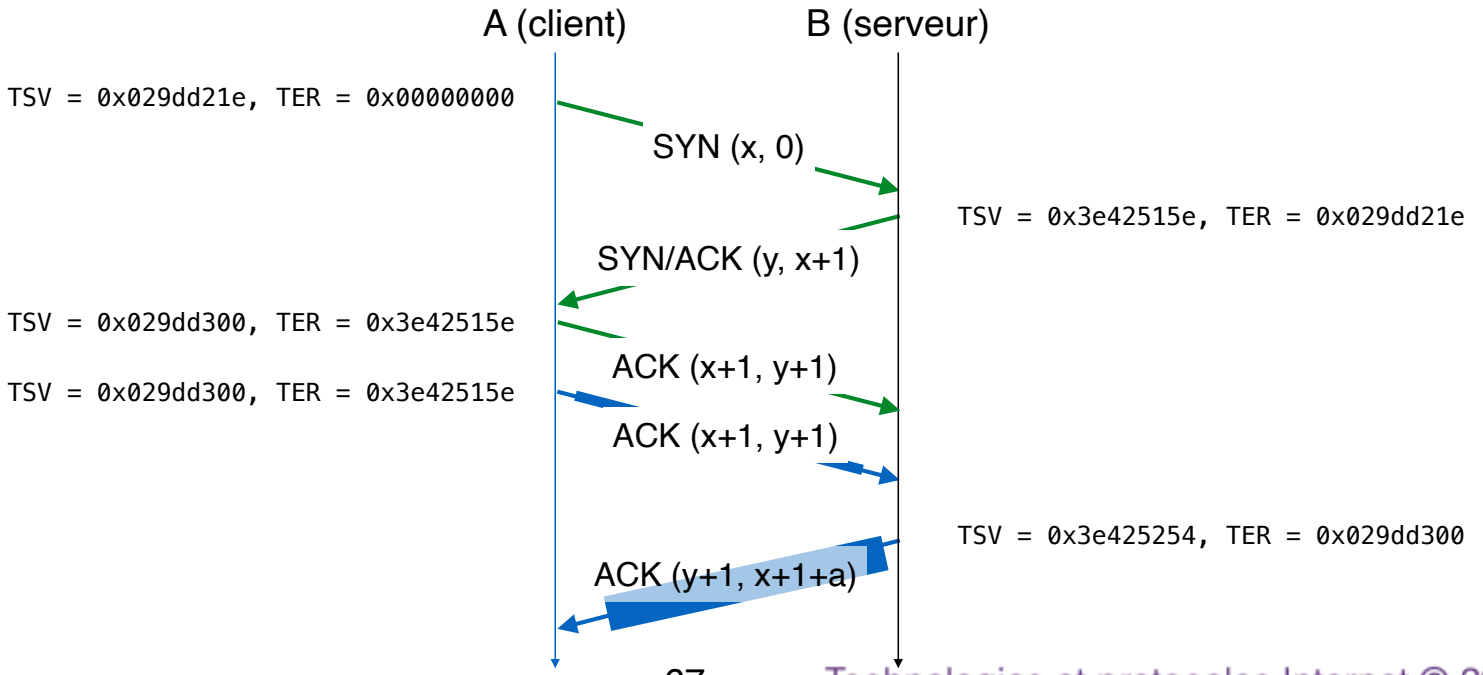
Windows Scale WSopt



Options TCP (5)

Time Stamp (TS)





```
ff ad 13 89 5c 3e 06 6a
00 00 00 00 a0 02 40 00
9c 2e 00 00 02 04 05 a0
01 03 03 00 01 01 08 0a
00 9e 7b c4 00 00 00 00
```

Port source : 0xFFAD, Port destination : 0X1389

Numéro de séquence : 0x5C3E066A

Numéro d'acquittement : 0x00000000

THL : 0xA (40 octets), SYN : 1, Fenêtre : 0x4000

Somme de contrôle : 0x9C2E, Pointeur urgent : 0x0000

Option 1 : Type : 0x02 (MSS), MSS : 0x05A0 (1440)
Option 2 : 0x01 (NOP),
Option 3 : Type : 0x03 (WScale), Décalage : 0x00
Option 4 : Type : 0x01 (NOP)
Option 5 : Type : 0x01 (NOP)
Option 6 : Type : 0x08 (Time Stamp), TSV : 0x9E7BC4, TERV : 0x000000

0xFFAD		0x1389	
0x5C3E066A			
0x00000000			
0xA002		0x4000	
0x9C2E		0x0000	
0x02	0x04	0x05A0	
0x01	0x03	0x03	0x00
0x01	0x01	0x08	0x0A
0x009E		0x7BC4	
0x0000		0x0000	

Efficacité des transmissions (1)

Génération des segments de données

- Objectif : émission de trames pleines (MTU octets de données)
  - Mise en mémoire des écritures (retard à l'émission)
- Si MSS (Maximum Segment Size) octets de données à envoyer
  - envoi de segments pleins
- Sinon :
  - demande explicite d'émission de l'application
    - fonction push
  - sur expiration d'un temporisateur
    - pour éviter d'attendre trop longtemps MSS octets
- Découplage écriture - émission des octets de données
  - une écriture d'octets : plusieurs envois de segments de données
  - plusieurs écritures d'octets : envoi d'un segment de données

Efficacité des transmissions (2)

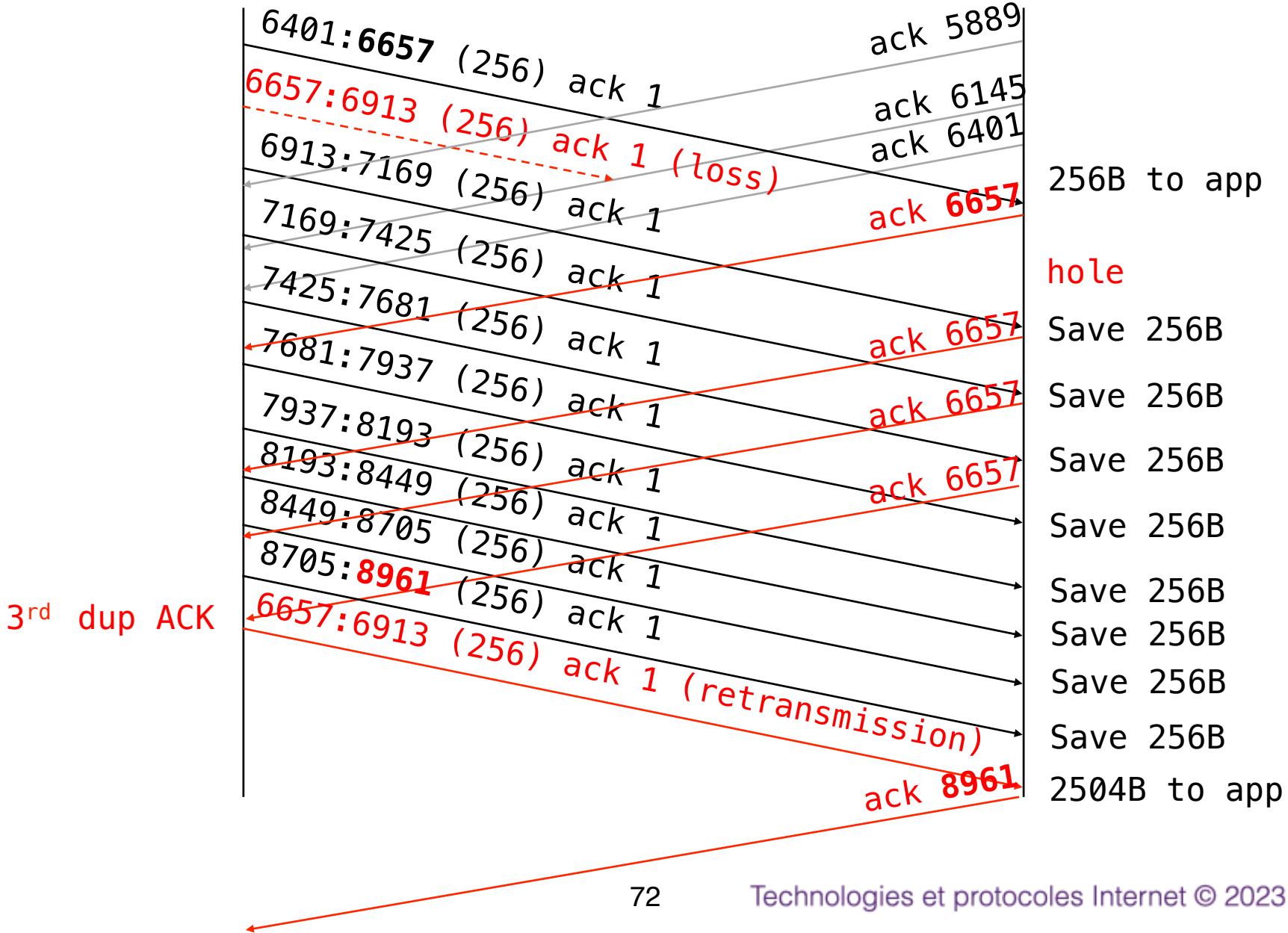
Génération des segments ACK

- "Dumb receiver"
  - un récepteur accuse le dernier des octets de données réçus en séquence
    - en indiquant le numéro du prochain octet attendu
  - la détection des pertes et leur réparation est à la charge de l'émetteur
    - pas de NAK dans TCP
- Acquittements retardés ou émis immédiatement :
  - ACK retardés : après un délai maximum d'attente
  - ACK cumulatifs : après réception de 2 MSS
  - ACK immédiat : sur réception d'un segment hors séquence ou corrigeant une perte

Génération des segments ACK

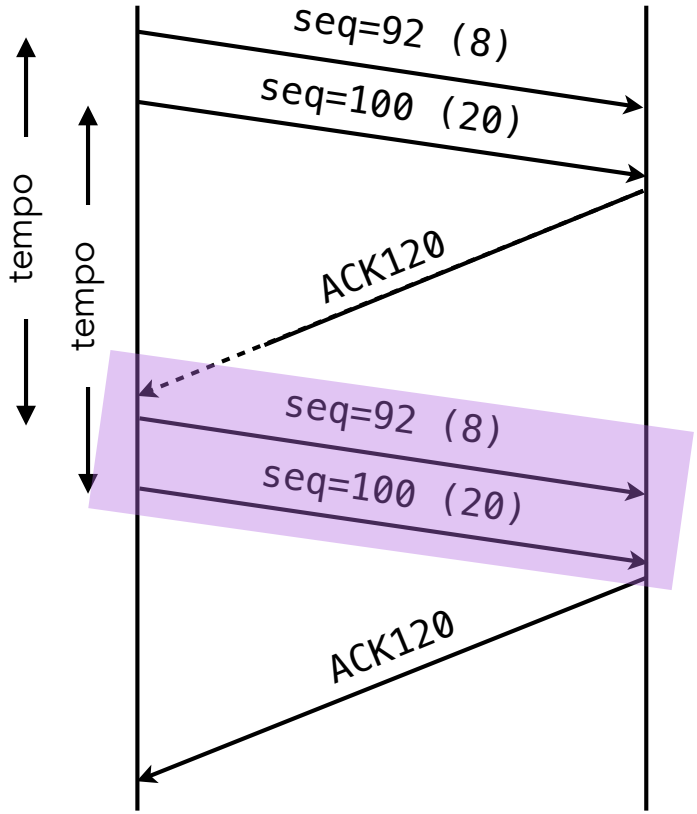
Evènement	Action du récepteur
Réception d'un segment en séquence : • le numéro de séquence du segment correspond au numéro de séquence attendu • les octets précédents ont déjà été acquittés	Attendre 500ms : • Si réception d'un second segment, alors envoi immédiat d'un **ACK cumulatif** • Sinon envoi d'un **ACK retardé** après 500ms
Réception d'un segment hors séquence : • un trou de séquence est détecté • le numéro de séquence du segment est supérieur au numéro de séquence attendu	Envoi immédiat d'un **ACK dupliqué** : • **l**e numéro d'acquittement indique à nouveau le numéro de séquence de l'octet attendu
Réception d'un segment qui comble le début d'un trou de séquence • partiellement ou complètement	Envoi immédiat d'un ACK : • qui accuse les octets reçus en séquence après le trou de séquence si le segment comble **complètement** le trou de séquence • qui accuse les octets de ce segment **sinon**

Fast Retransmission



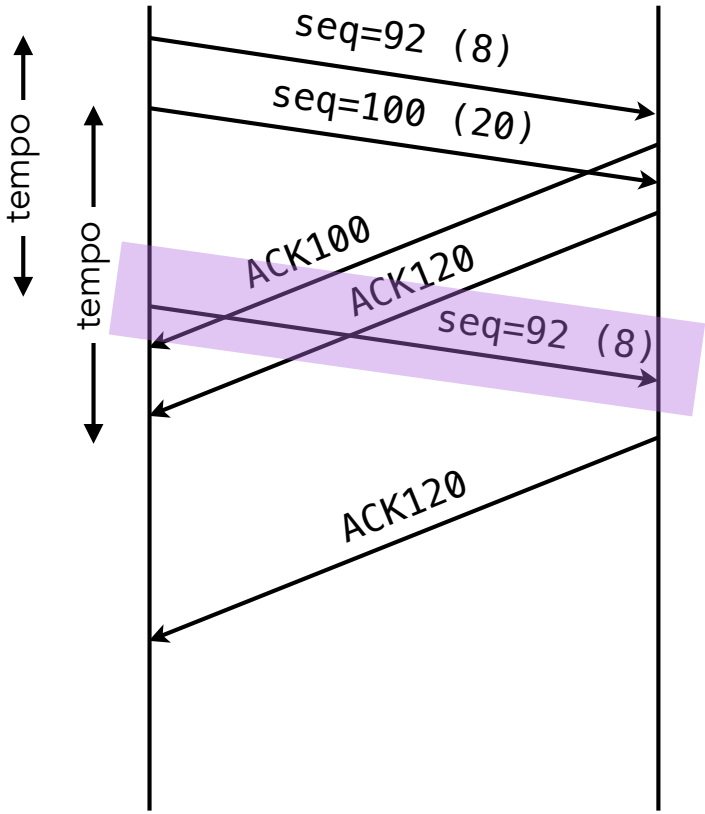
Quand retransmettre ? (2)

Perte de ACK



paquets TCP dupliqués

Expiration de temporisateur prématurée



paquet TCP et ACK dupliqués

Durée du temporisateur de retransmission

- La source arme un temporisateur dans l'attente d'un ACK
  - trop court : retransmissions inutiles
  - trop long : attente trop longue avant réparation d'une perte
- TCP calcule la durée du temporisateur en fonction du délai aller-retour (RTT)
  - les émetteurs mesurent la durée qui s'écoule entre l'envoi d'un segment de données
  - et la réception de l'ACK correspondant
    - Valeur moyennée du RTT
  - la source mesure RTT : le temps qu'a mis le dernier ACK à lui revenir
  - la source calcule SRTT : la valeur moyenne pondérée des RTT mesurés

$$SRTT = a * SRTT + (1 - a) * RTT$$

où a est un facteur de lissage

- Valeur du temporisateur de retransmission :

$$RTO = 2 * SRTT$$

  - valeur initiale du RTO = 1 seconde

Conclusions

- TCP et UDP
  - Multiplexage et démultiplexage
  - Détection d'erreur
    - somme de contrôle
- TCP
  - Connexion et états
  - Réparation des pertes et correction d'erreur
    - numéros de séquence
    - estimation du RTT
    - expiration de temporisateur et retransmission
  - Contrôle de flux
    - fenêtre glissante
  - Contrôle de congestion
- Cours prochain
  - Couche réseau : Adresses IP