

La couche transport dans Internet (suite)

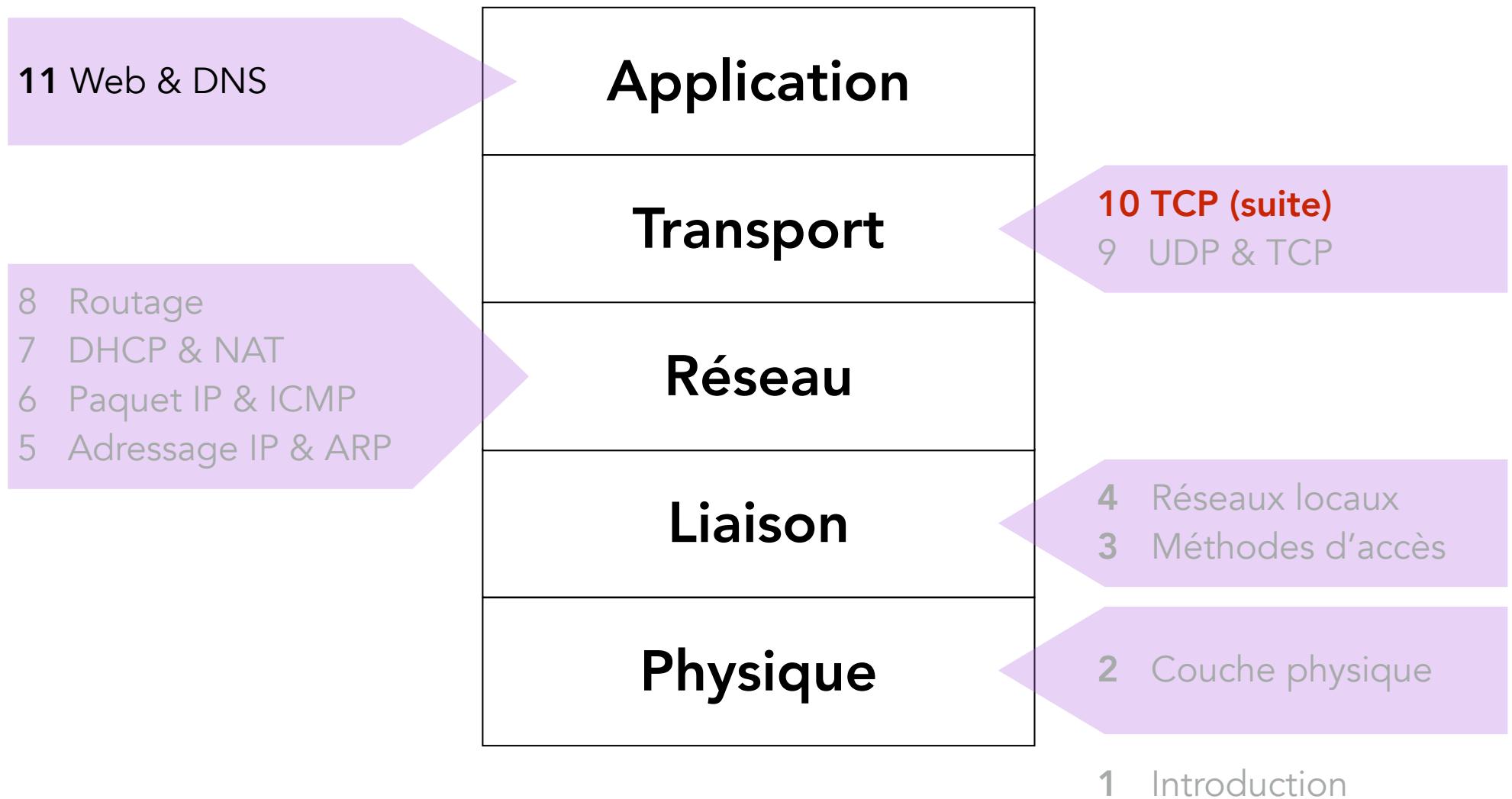
UE LU3IN033 Réseaux
2024-2025

Bruno Baynat

Bruno.Baynat@sorbonne-universite.fr



Programme de l'UE LU3IN033



Plan du cours

- Fiabilité et efficacité de TCP
- Transmission en mode flux d'octets
- Service en mode connecté
- Types et format des segments
- Numérotation des octets de données et accusés de réception
- Temporisation et retransmissions
- Contrôle de flux
- Octets de données urgentes et pushées
- Options TCP

TCP Transmission Control Protocol

TCP : *Transmission Control Protocol*

- Service en **mode flux d'octets**
 - TCP considère les données de l'application comme un flux ordonné et ininterrompu d'octets
 - Coté émetteur : l'écriture des octets à envoyer par l'application (dans le buffer d'émission) et l'envoi de segments par TCP ne sont pas corrélés
 - Coté récepteur : la réception de segments TCP et la lecture des octets reçus par l'application (dans le buffer de réception) ne sont pas corrélés
- Service **orienté connexion**
 - Un connexion TCP est constituée de 3 phases
 - phase d'établissement (ouverture) de la connexion
 - phase de transfert de données
 - phase de libération (fermeture) de la connexion
- Service de livraison **fiable** en séquence d'octets
 - Somme de contrôle (checksum)
 - 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
- Mécanisme de **contrôle de flux**
 - Pour éviter l'engorgement des récepteurs
- Mécanisme de **contrôle de congestion**
 - Pour éviter l'engorgement du réseau

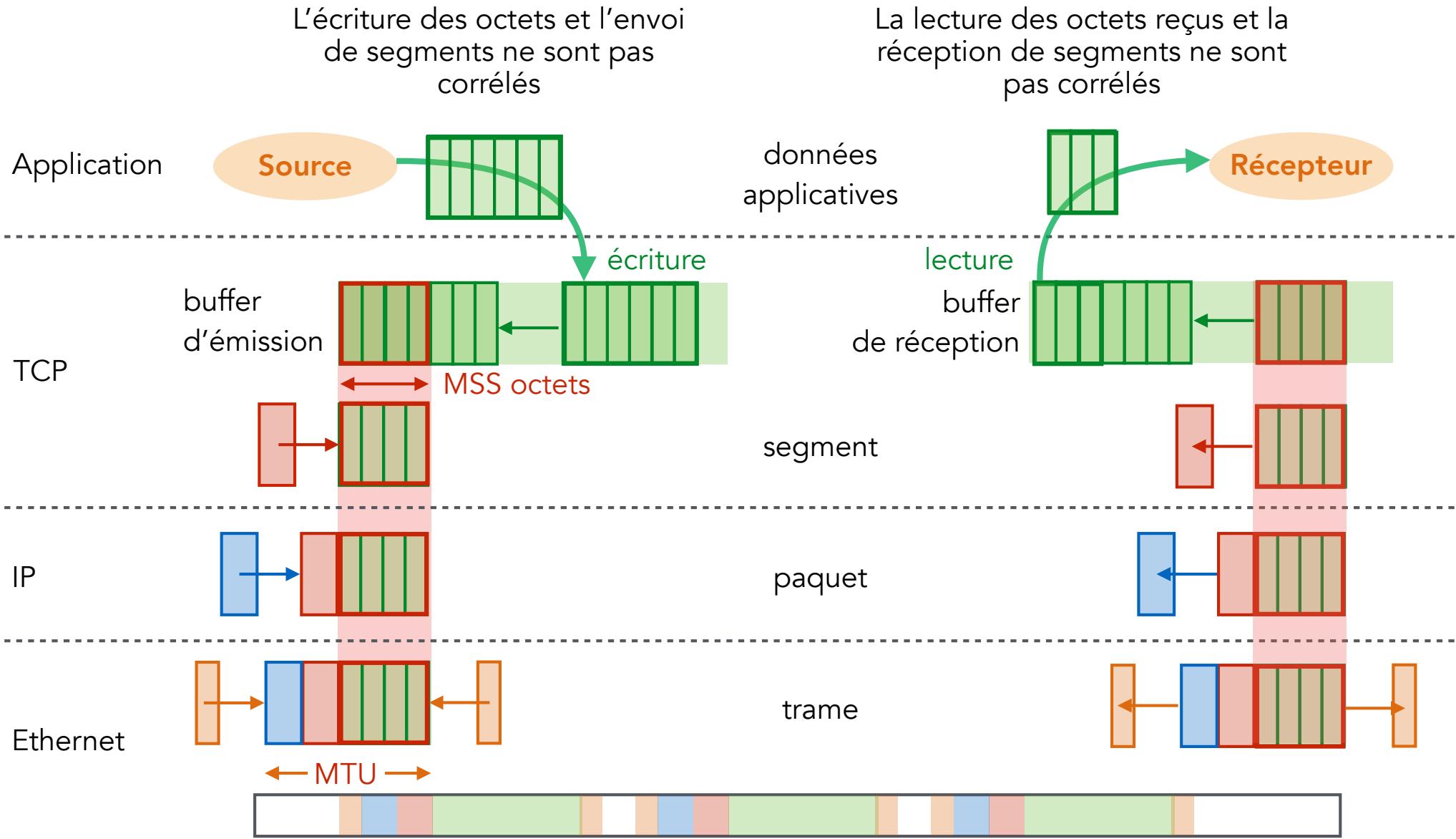
TCP : transfert fiable

- TCP (comme UDP) détecte les erreurs
 - somme de contrôle (checksum)
- Mais à l'encontre d'UDP...
- TCP détecte les pertes et les déséquencements
 - numéros de séquences
- TCP répare les erreurs et les pertes
 - buffer d'émission
 - acquittements des données
 - retransmissions
 - temporiseurs
- TCP réordonne les données déséquencées avant de les donner à l'application
 - buffer de réception

TCP : transfert efficace

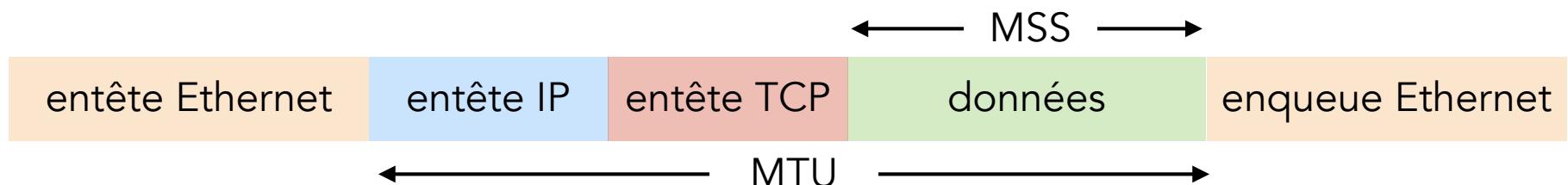
- TCP évite le gâchis de bande passante
 - en envoyant (dès que possible) des segments de taille maximum (MSS)
 - pour éviter l'envoi de trames à moitié vide
 - pour éviter la fragmentation IP
- TCP évite l'engorgement des buffers (des routeurs et des hôtes)
 - en adaptant le débit d'émission de la source
 - à la capacité d'absorption du récepteur : contrôle de flux
 - à la capacité de transfert du réseau : contrôle de congestion
- TCP assure un partage équitable de la bande passante
 - en équilibrant le débit d'émission des différentes sources qui empruntent le même chemin
 - contrôle de congestion

TCP : Mode Flux d'octets



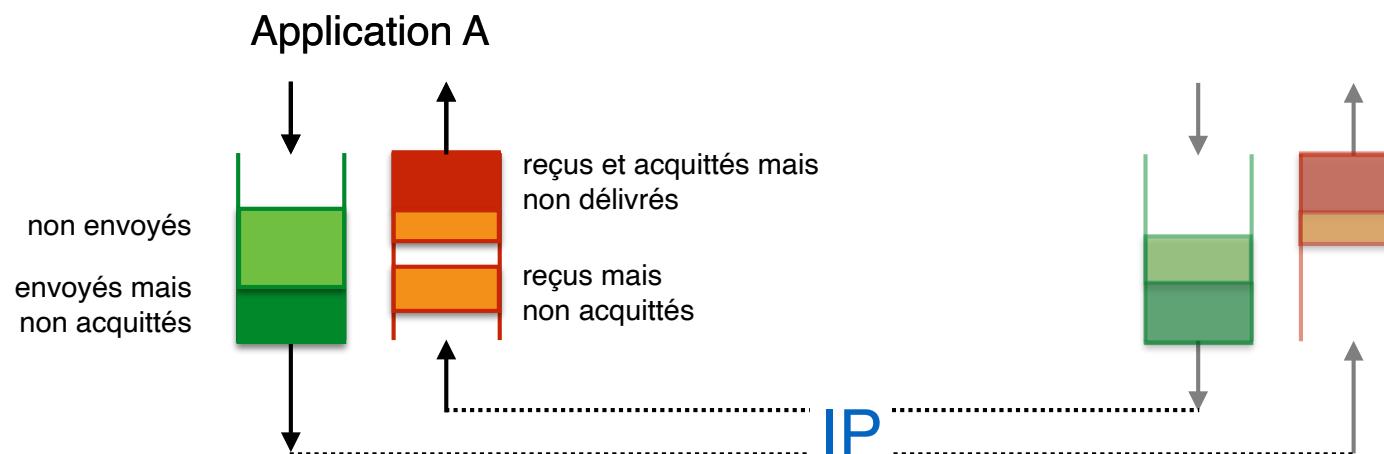
Envoi d'un segment TCP

- L'efficacité d'une transmission est corrélée au taux de remplissage des trames
 - MTU (*Maximum Transmission Unit*) : taille max du champ données d'une trame
 - Ex : MTU Ethernet = 1500 octets
- TCP évite l'envoi de trames à moitié pleines
 - TCP attend que l'application ait écrit MSS octets dans le buffer d'émission
 - *MSS : Maximum Segment Size*
 - $MSS = MTU - (\text{taille en-tête IP} + \text{taille en-tête TCP})$
 - Ex : 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
 - ou sur expiration d'un temporisateur
 - pour éviter d'attendre trop longtemps



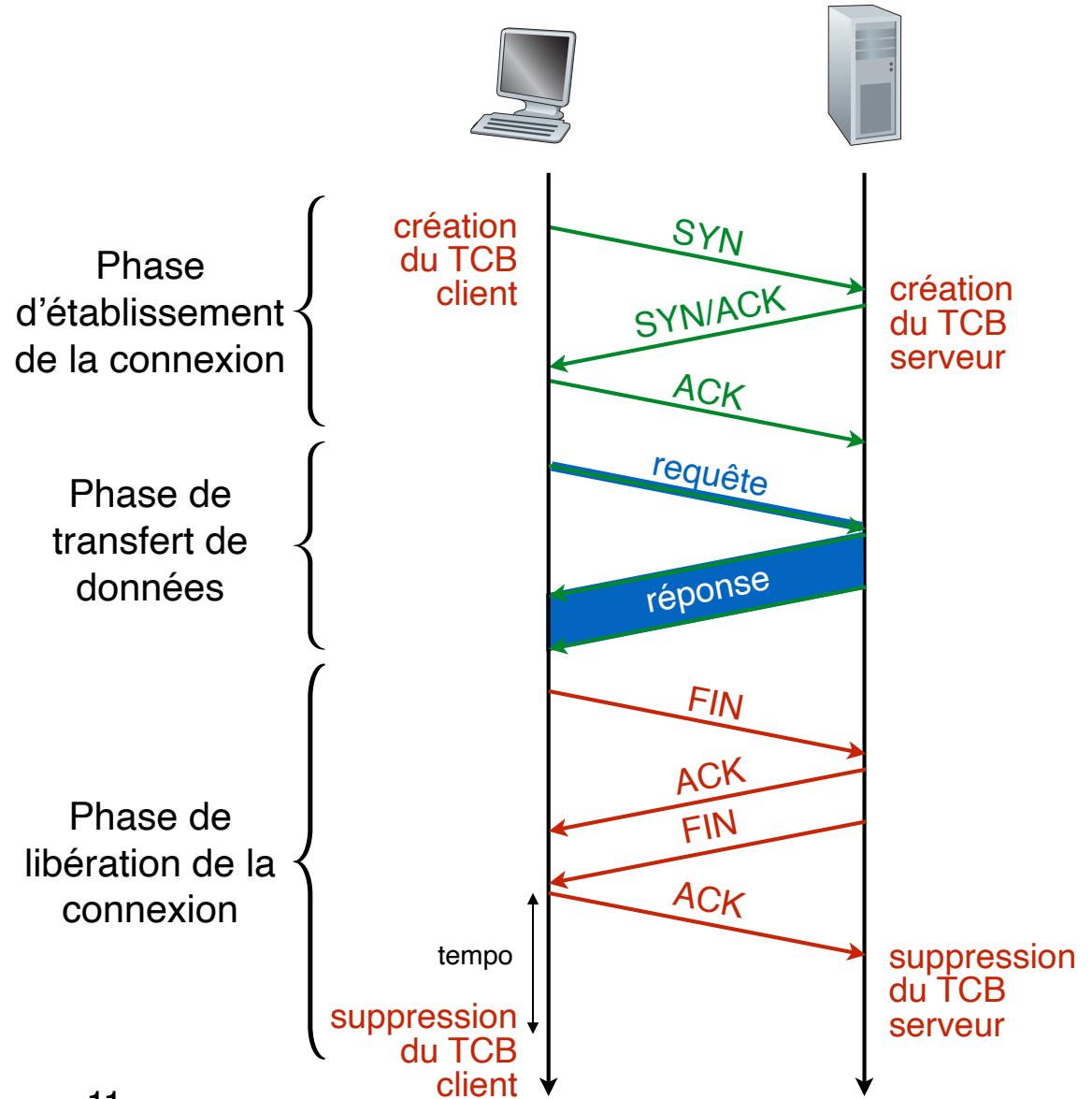
Buffers d'émission et de réception

- TCP offre aux deux applications en communication
 - un **buffer d'émission**
 - où l'application considérée écrit les octets de données que TCP doit envoyer (de façon fiable) à l'application distante
 - un **buffer de réception**
 - contenant les octets de données reçus de l'application distante, que l'application considérée vient lire quand elle le souhaite (dans l'ordre et sans erreur)

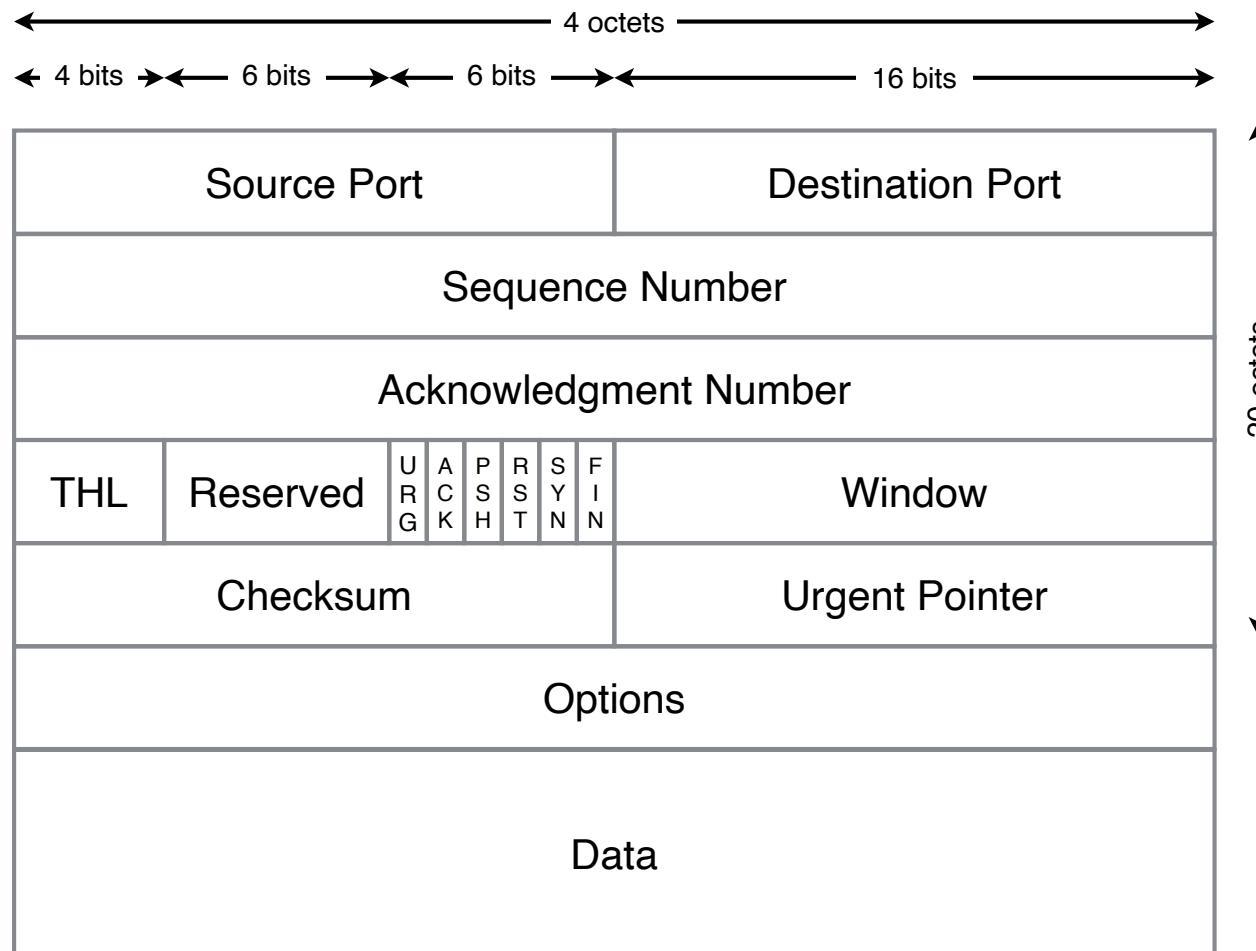


TCP : Mode connecté

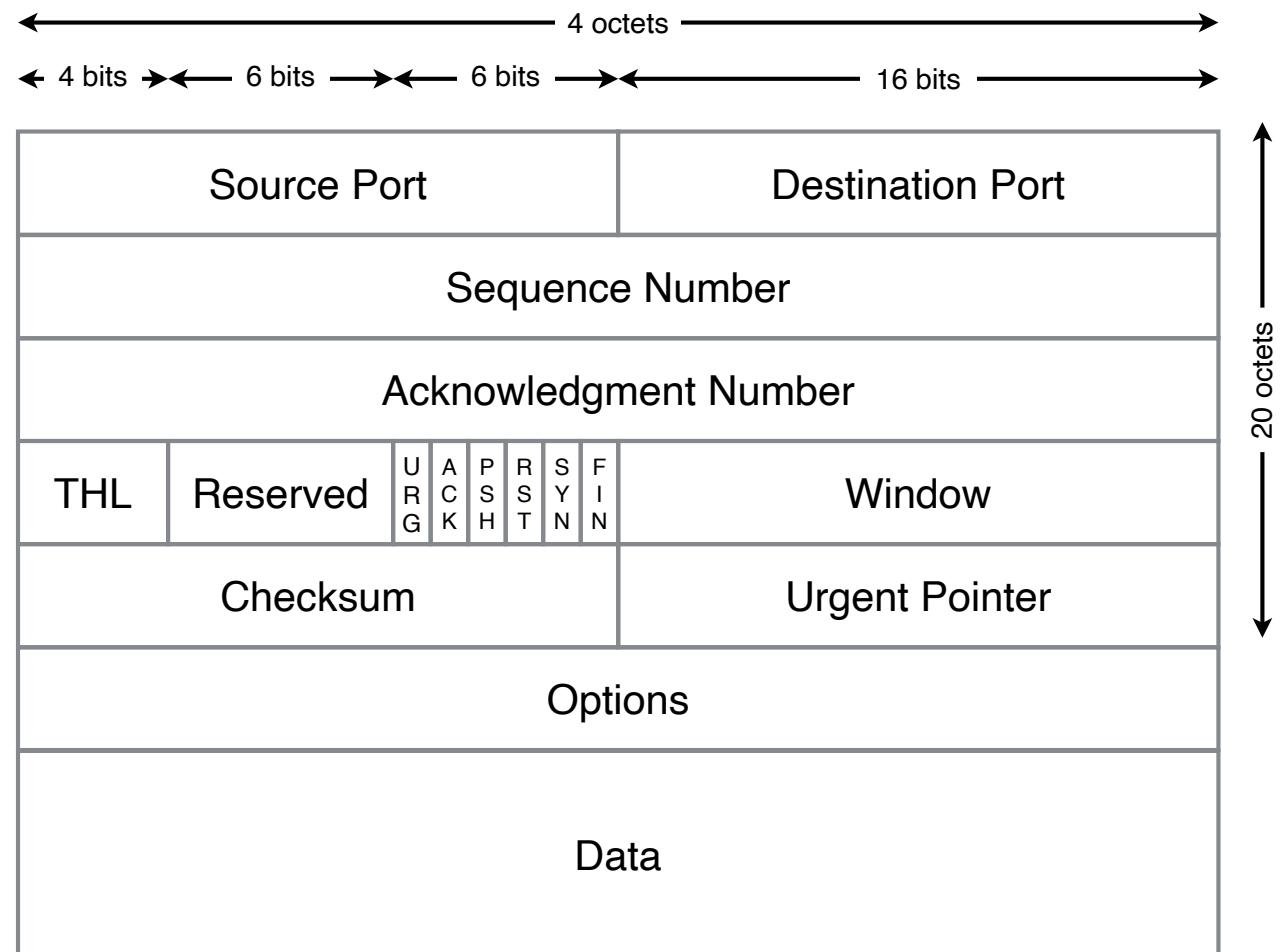
- Connexion TCP
 - 3 phases
 - ouverture : « 3-way handshake »
 - segments SYN, SYN/ACK, ACK
 - fermeture : « 4-way handshake »
 - segments FIN, ACK
- Les TCB (*Transport Control Block*) contiennent les informations d'état caractérisant l'échange
 - identifiants du processus (adresse IP, numéro de port), numéros de séquence des octets reçus en séquence et acquittés, valeurs des fenêtres, etc.
 - créés et initialisés à l'ouverture de la connexion
 - supprimés à la fermeture de la connexion
- Une connexion TCP est caractérisée par la combinaison des informations contenues dans les TCB client et TCB serveur



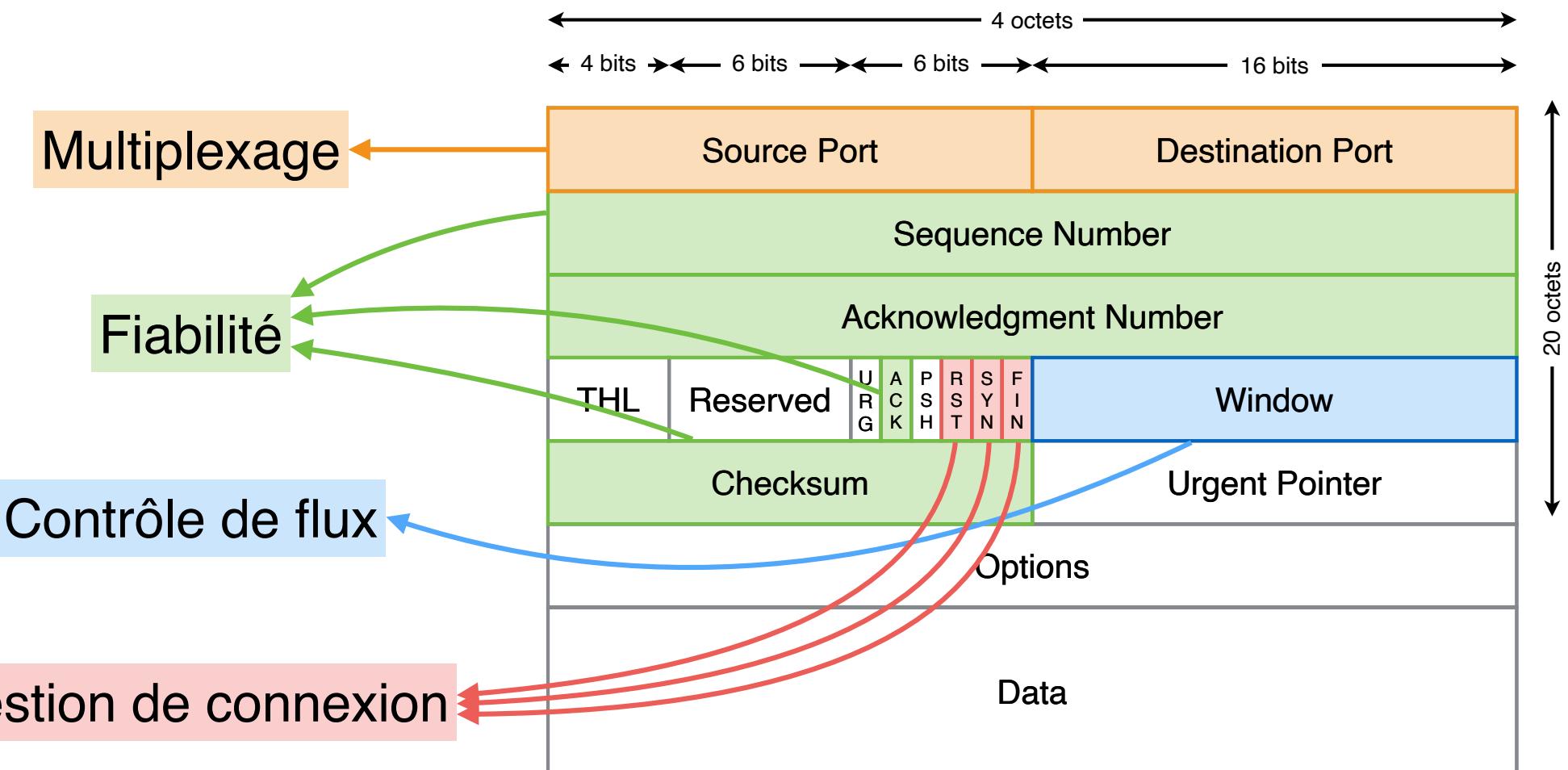
Entête TCP



Entête TCP

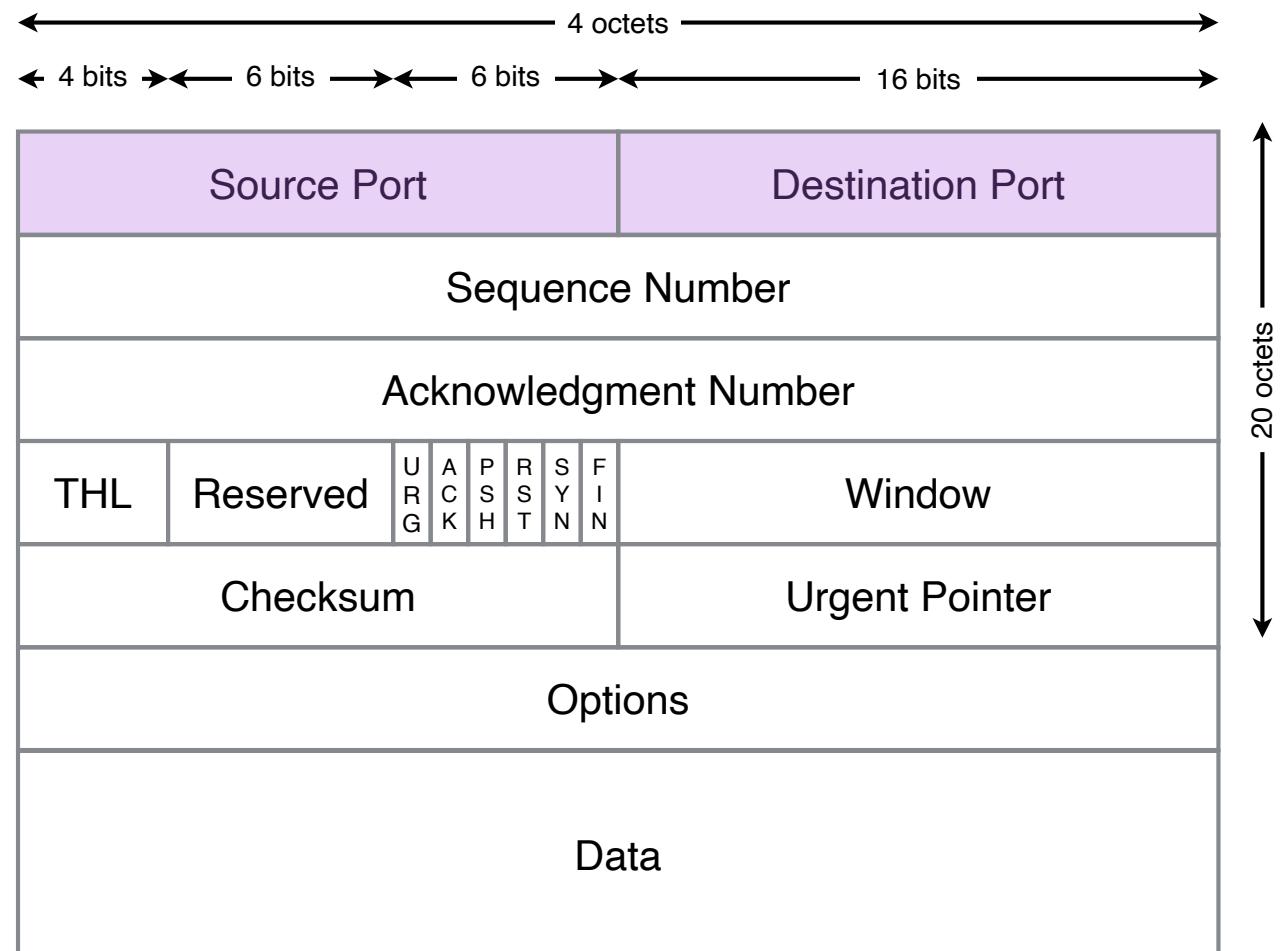


Entête TCP



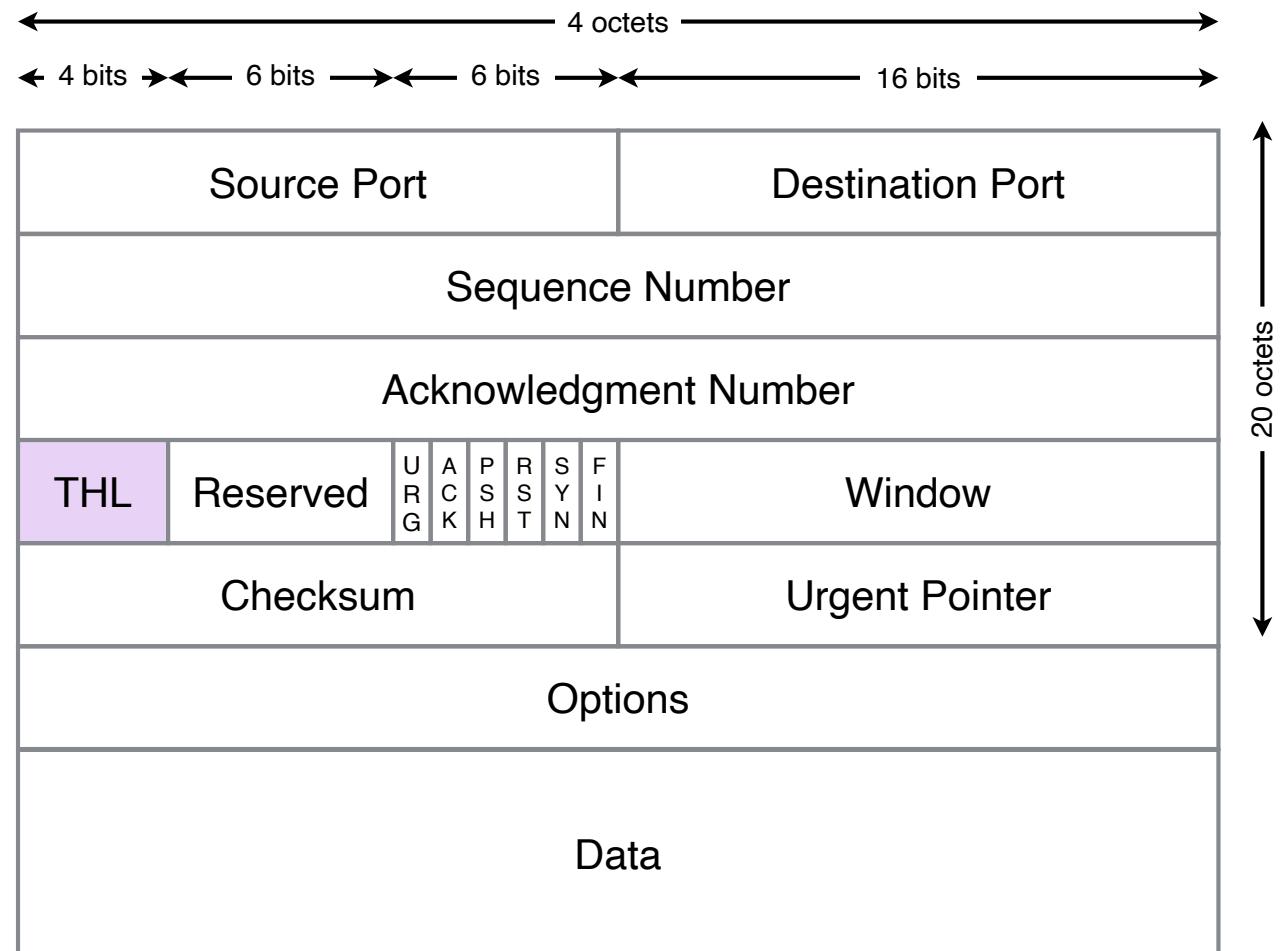
Numéros de port

- Numéro de port source
 - identifie l'application qui émet les données incluses dans le segment
- Numéro de port destination
 - identifie le processus à qui sont destinées les données du segment
- Numéro de port client
 - valeur arbitraire ≥ 1024
 - laissé au choix de l'OS
- Numéro de port serveur
 - valeur connue < 1024
 - Ex : 80 pour un Serveur web



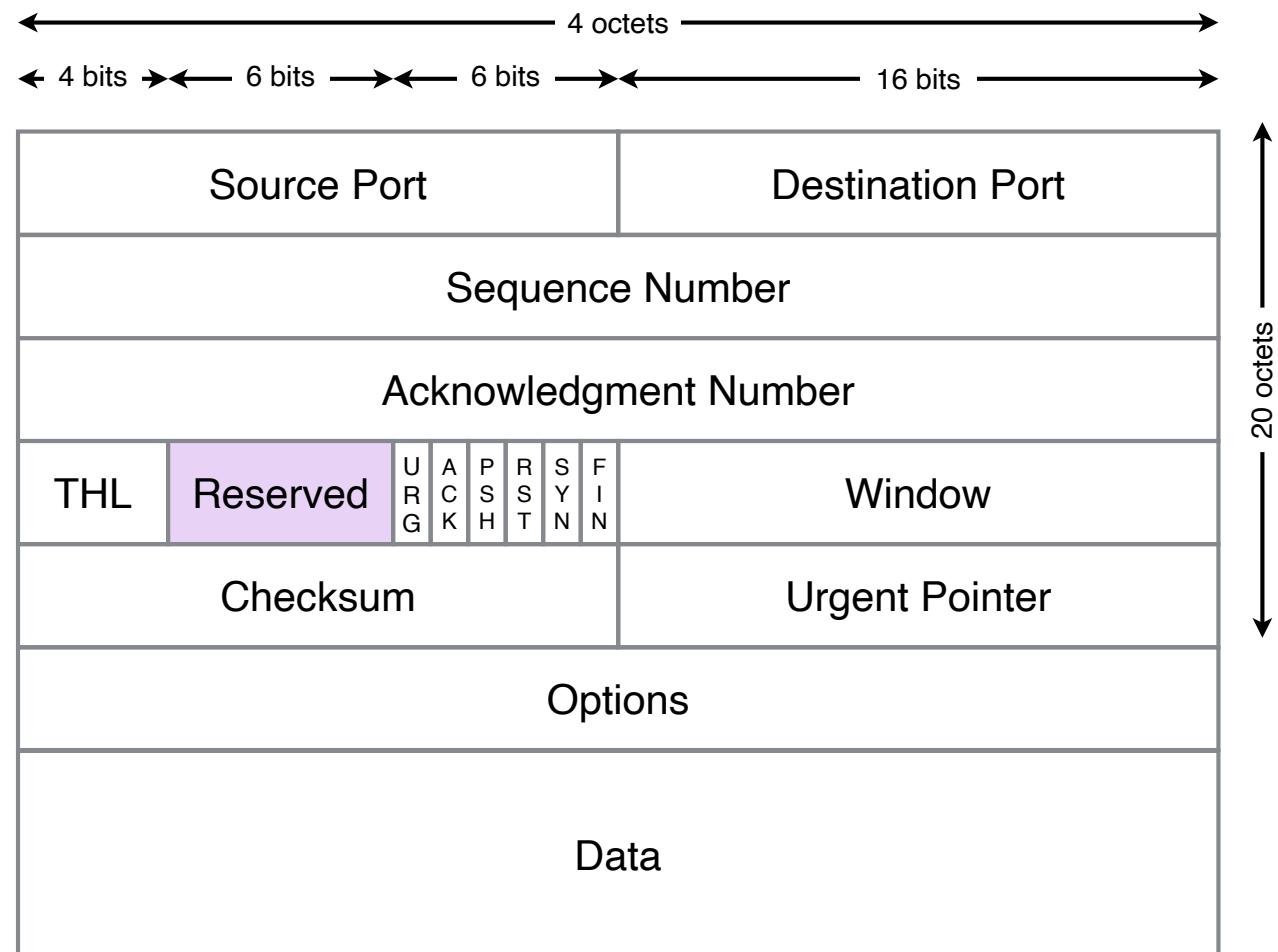
Champ THL

- THL (*TCP Header Length*)
 - taille de l'entête exprimée en mots de 32 bits (4 octets)
- Taille min
 - THL = 0x5 (0101) : 5
 - entête de 20 octets
 - pas d'options TCP
- Taille max
 - THL = 0xF (1111) : 15
 - entête de 60 octets
 - 40 octets d'options TCP



Champ Reserved

- Champ réservé à l'origine pour une utilisation future
 - aujourd'hui non utilisé
 - doit être laissé à zéro (000000)



Champ Checksum

paquet IP

4	IHL	TOS	Total Length
Identifier	R	D M F F	Fragment offset
TTL	Protocol	Header checksum	
Source IP address			
Destination IP address			



pseudo entête

Source IP address		
Destination IP address		
0	Protocol	TCP length

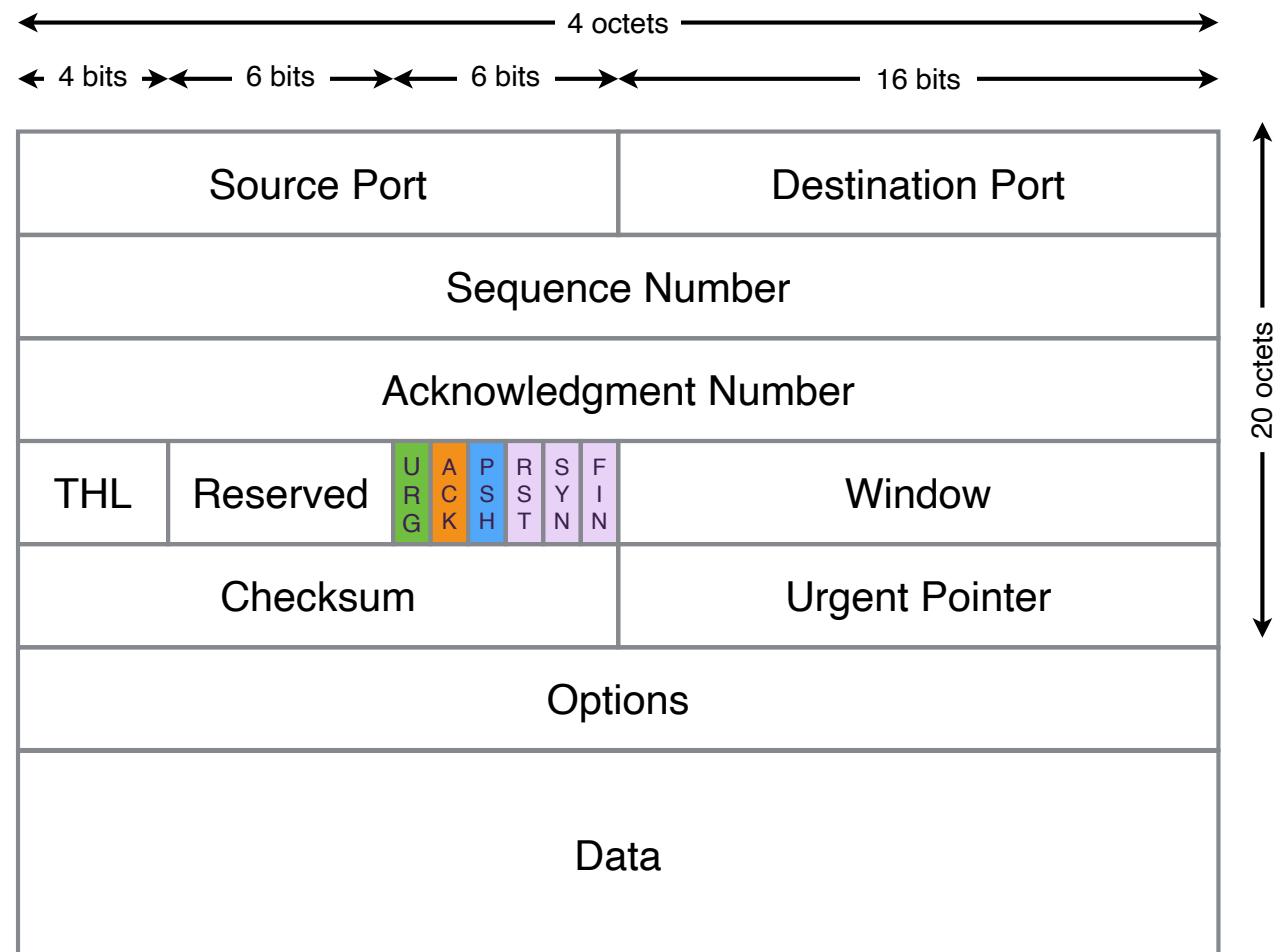
segment TCP

Source Port	Destination Port
Sequence Number	
Acknowledgment Number	
THL	Reserved
URG	A CK P SH R ST S Y N F
Checksum	
Urgent Pointer	
Options	
Data	

- *Checksum* : somme de contrôle portant
 - sur l'en-tête du segment
 - sur les données transportées
 - sur un « pseudo-entête »

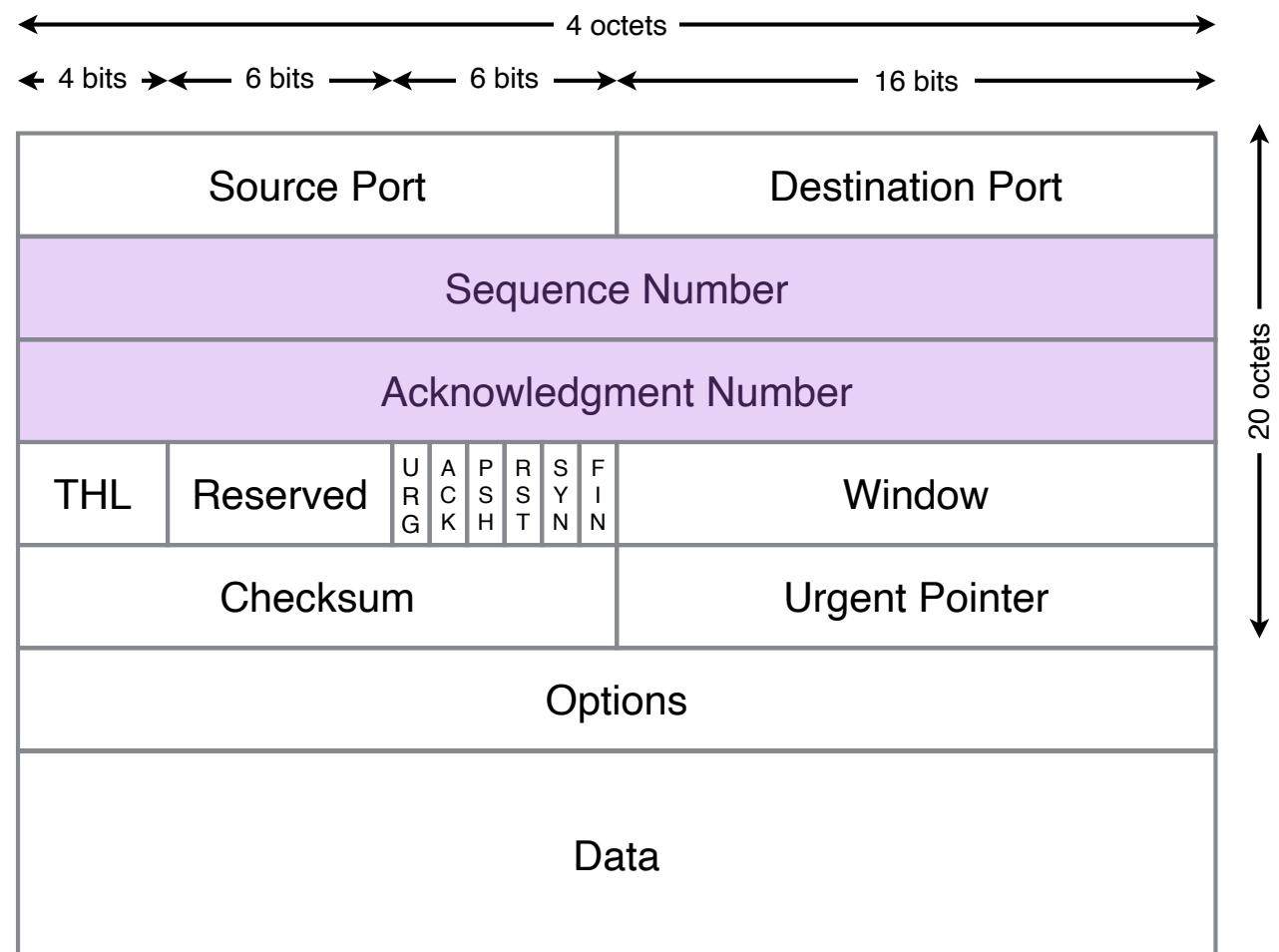
Drapeaux TCP

- SYN, FIN et RST
 - identifie 3 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
 - valide le champ AN
- PSH
 - les données doivent être lues au plus vite par l'application côté récepteur
- URG
 - le segment contient des données urgentes



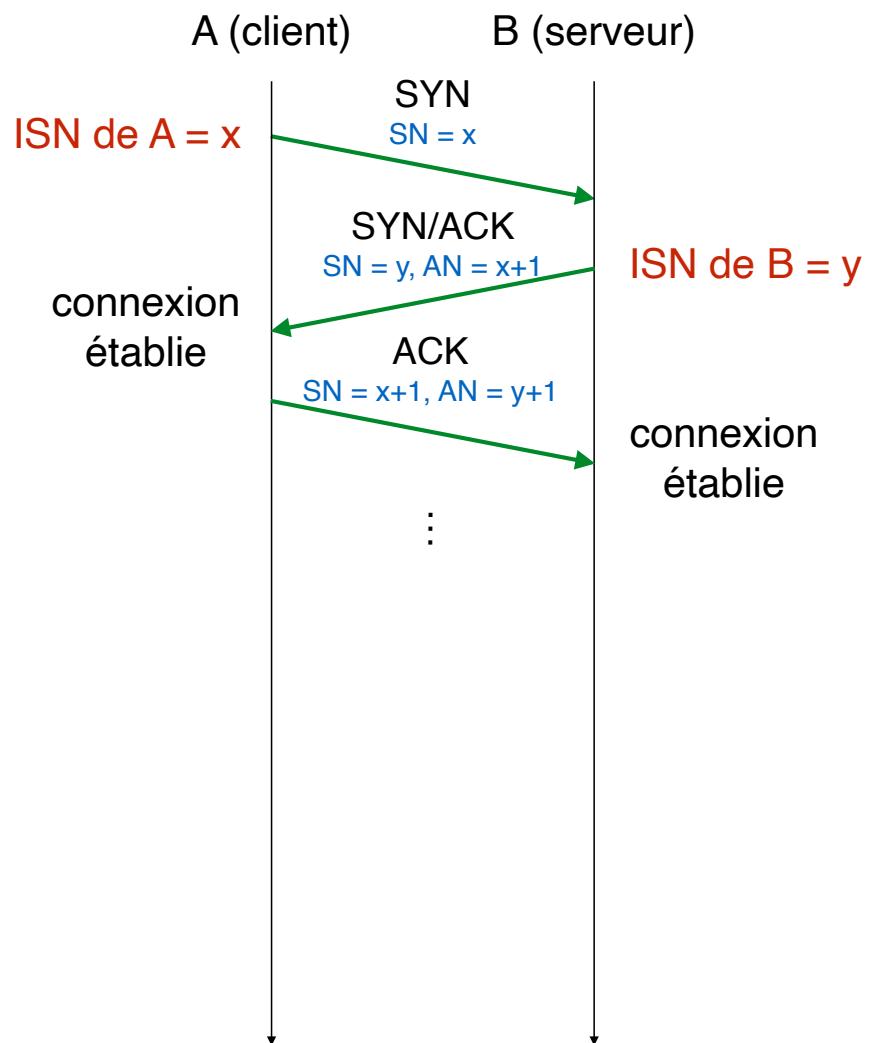
Numérotation des segments

- Numéro de séquence
 - pour un segment SYN : ISN
 - pour un segment de données : numéro du premier octet transporté par le segment
 - pour un segment FIN : numéro du dernier octet transporté + 1
- Numéro d'acquittement
 - valide uniquement lorsque le drapeau ACK est positionné
 - accuse la réception des segments SYN et FIN
 - accuse la réception des octets de données envoyés par l'application distante



Numéros de séquence et d'acquittement

- Segments SYN et SYNC/ACK
 - Ouverture bilatérale d'une connexion
 - le client envoie un segment SYN
 - le serveur répond par un segment SYNC/ACK
 - le client confirme l'établissement en envoyant un segment ACK
 - Synchronisation des numéros de séquence
 - le champ SN (Sequence Number) contient la valeur de l'ISN (Initial Sequence Number)
 - choisie aléatoirement de chaque côté
 - Paramétrage de la connexion (options TCP dans l'entête du SYN et du SYNC/ACK)
- Segments ACK
 - accusent la bonne réception des SYN
 - le champ AN (Acknowledgment Number) contient le SN du SYN incrémenté de 1



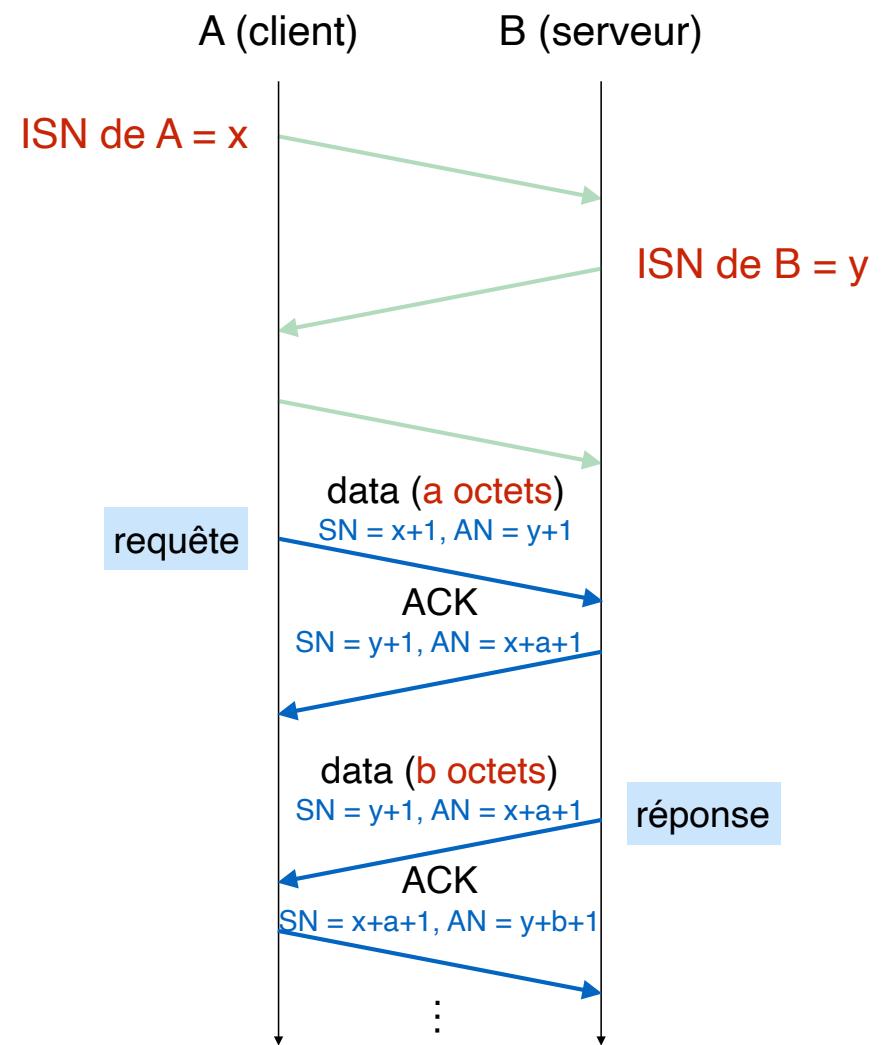
Numéros de séquence et d'acquittement

- Segments de données

- Drapeaux
 - SYN, FIN et RST à 0
 - ACK à 1
 - PUSH et URG à 0 ou à 1
- le champ SN (Sequence Number) contient le numéro de séquence du premier octet de données transporté dans le segment
- le champ AN (Acknowledgment Number) contient le numéro de séquence du prochain octet attendu de l'application distante
 - acquittement dans les données : « piggybacking »

- Segments ACK

- ne transportent pas de données
- accusent les octets correctement reçu
 - le champ AN contient le numéro de séquence du prochain octet attendu de l'application distante



Numéros de séquence et d'acquittement

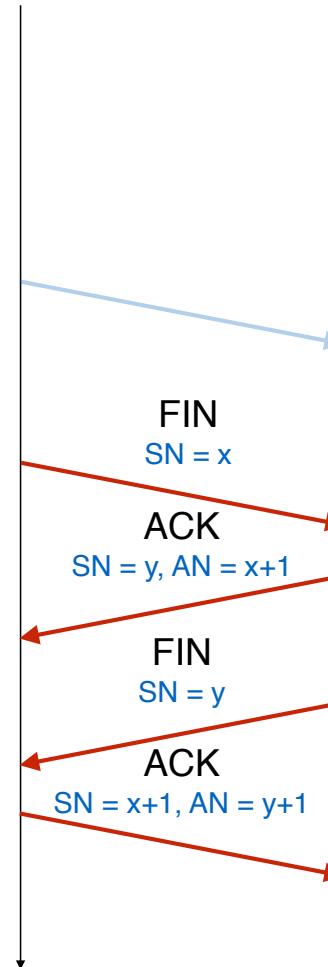
- Segment FIN

- Fermeture bilatérale de connexion
 - serveur et client envoient chacun un segment FIN
 - la connexion est fermée une fois que les deux FIN sont acquittés

- Segments ACK

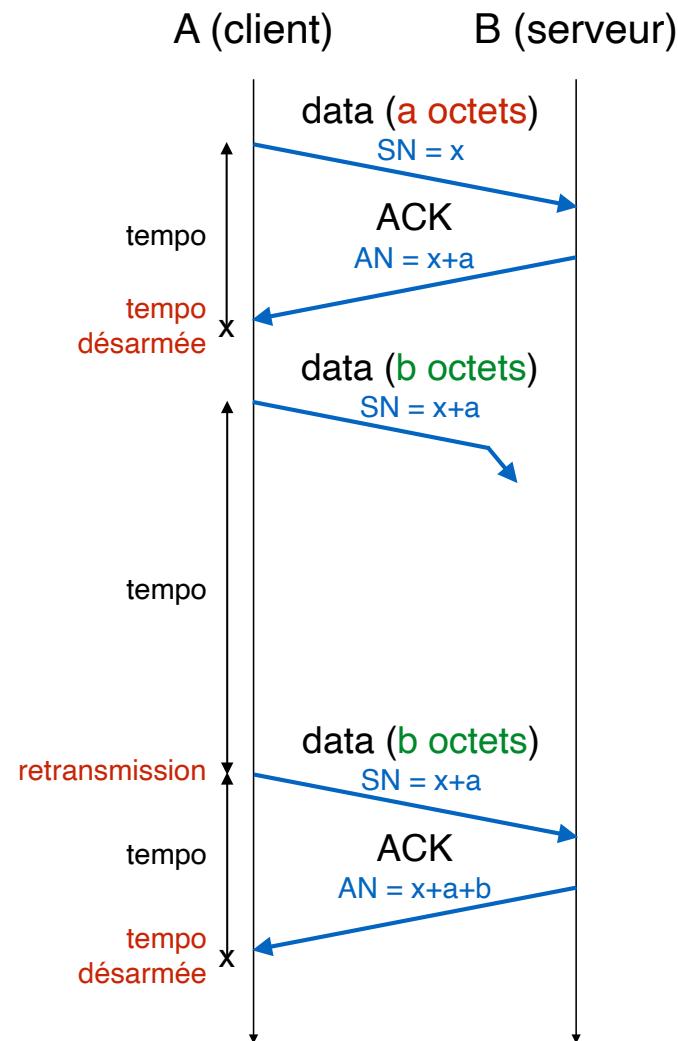
- accusent la bonne réception des FIN
 - en incrémentant de 1 le SN du FIN

A (client) B (serveur)

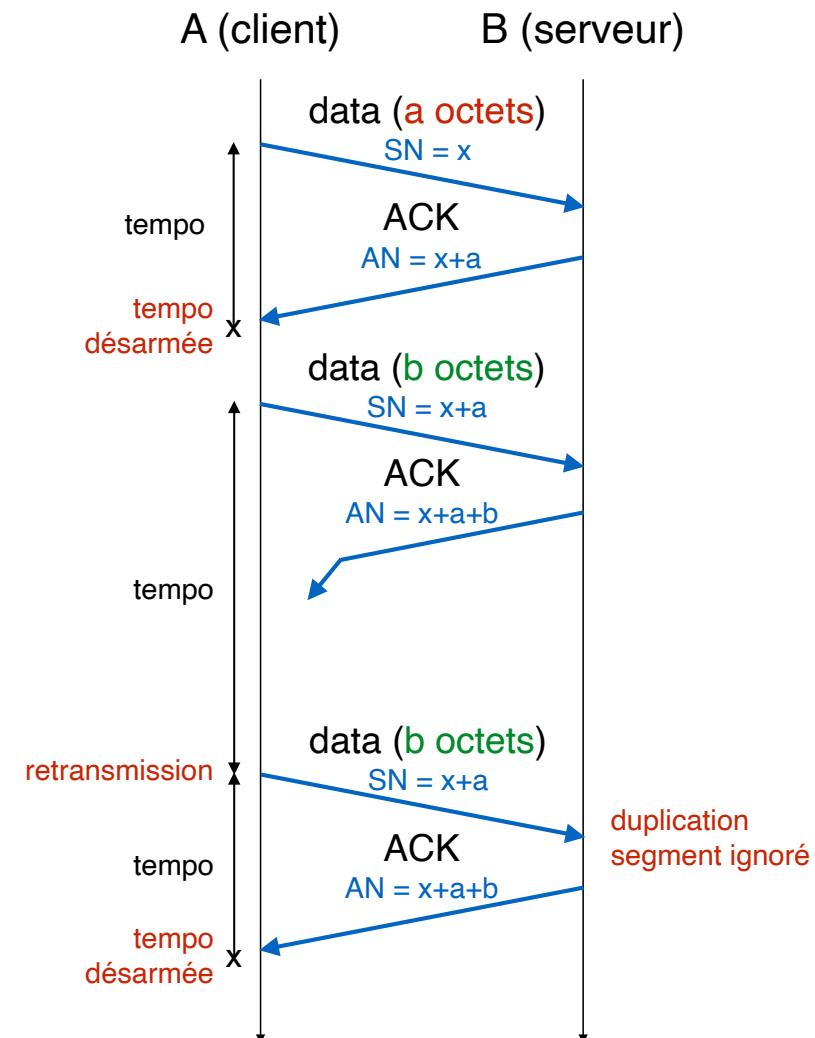
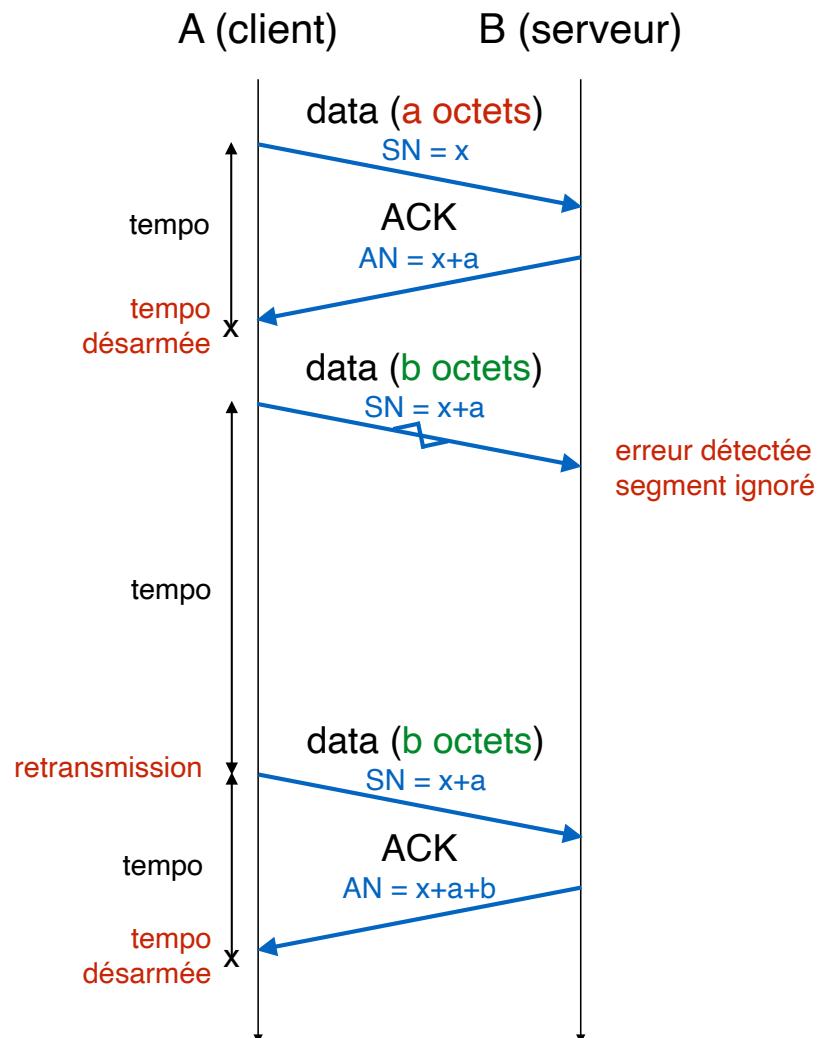


Retransmissions

- Pour chaque segment de données envoyé une temporisation est déclenchée
 - si un acquittement couvrant les octets transportés dans le segment revient avant la fin de la tempo
 - la tempo est désarmée
 - sinon
 - un segment contenant les mêmes octets de données est retransmis

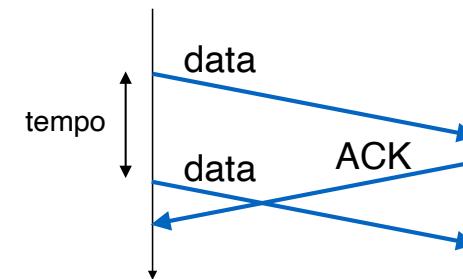


Retransmissions

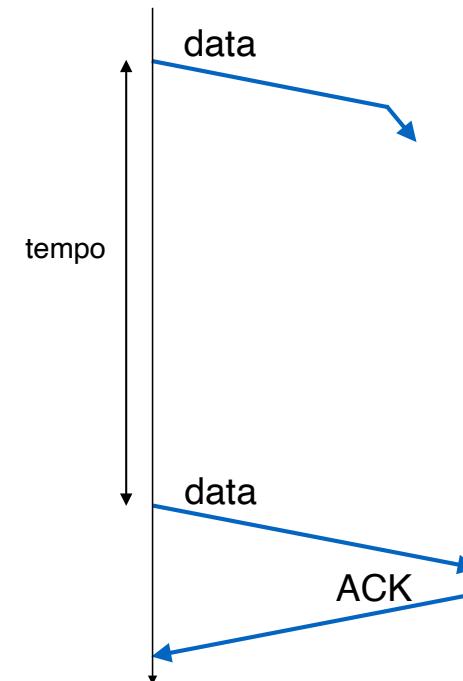


Temporisation de retransmission

- Si elle est trop courte, elle risque de se déclencher avant le retour de l'acquittement
 - Retransmissions inutiles

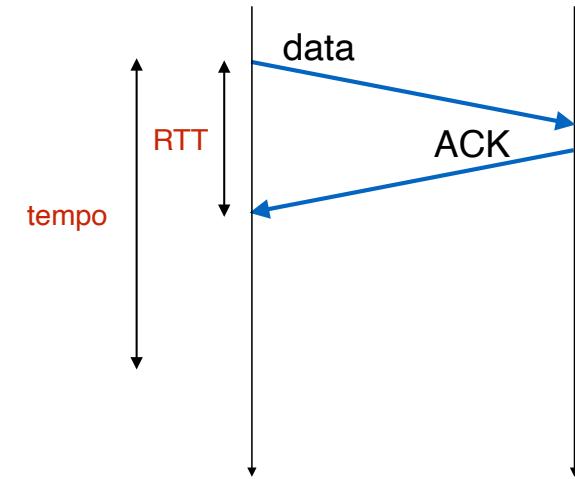


- Si elle est trop longue, elle génère une attente trop longue avant la réparation d'une perte
 - Retards inutiles



Temporisation de retransmission

- La temporisation de retransmission doit être dimensionnée de façon à laisser à l'acquittement le temps de revenir
 - Tempo > RTT (*Round-Trip Time*)
- Le RTT sur une connexion TCP
 - dépend de la distance entre la source et la destination
 - dépend de la bande passante du chemin emprunté
 - dépend de la charge du réseau
 - varie d'un segment à l'autre
- Le RTT doit être estimé par une moyenne sur des valeurs mesurées
 - adaptée à la connexion
 - variable dans le temps



Temporisation de retransmission

- A chaque nouvel envoi d'un segment, TCP mesure le délai nécessaire au retour de l'acquittement correspondant : **RTT** (*Round-Trip Time*)
- TCP calcule une moyenne glissante du RTT appelée **SRTT** (*Smoothed Round-Trip Time*) :

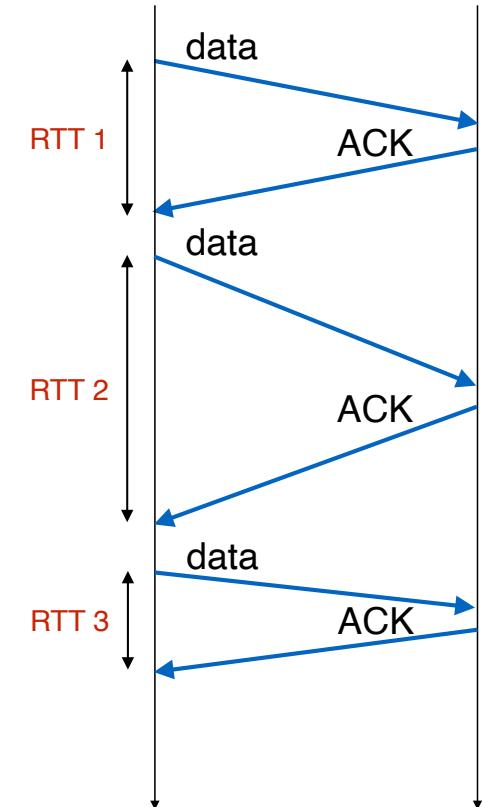
$$\text{SRTT} = \alpha * \text{SRTT} + (1 - \alpha) * \text{RTT}$$

où α est un facteur de lissage (Ex : $\alpha = 0,9$)

- et en déduit la valeur courante du temporisateur de retransmission appelée **RTO** (*Retransmission Time Out*) :

$$\text{RTO} = \beta * \text{SRTT}$$

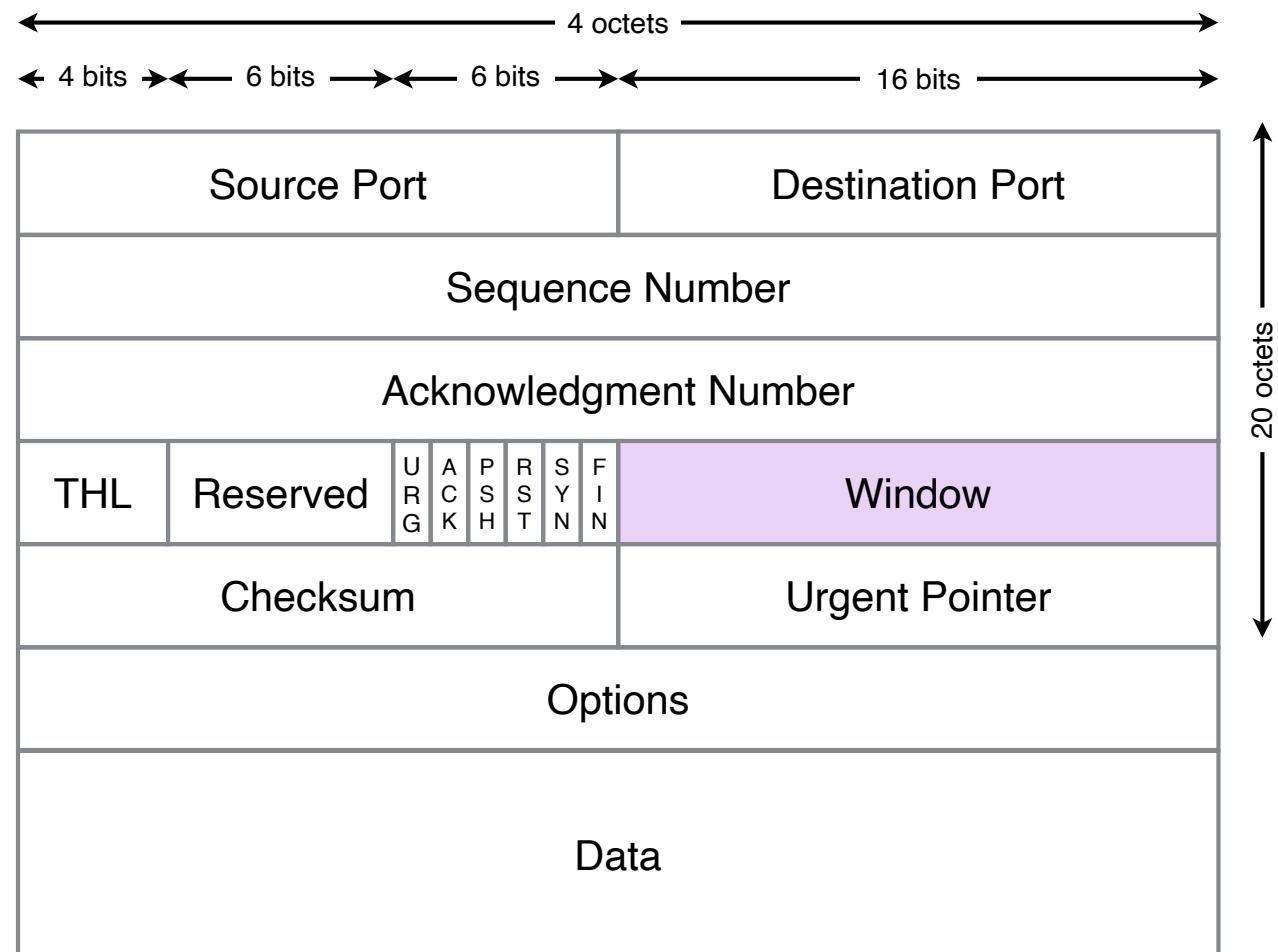
où β est un facteur de pondération (Ex : $\beta = 2$)



Champ Window

- Champ Window

- utilisé pour le contrôle de flux
- indique le nombre d'octets prêts à être reçu à partir du numéro AN



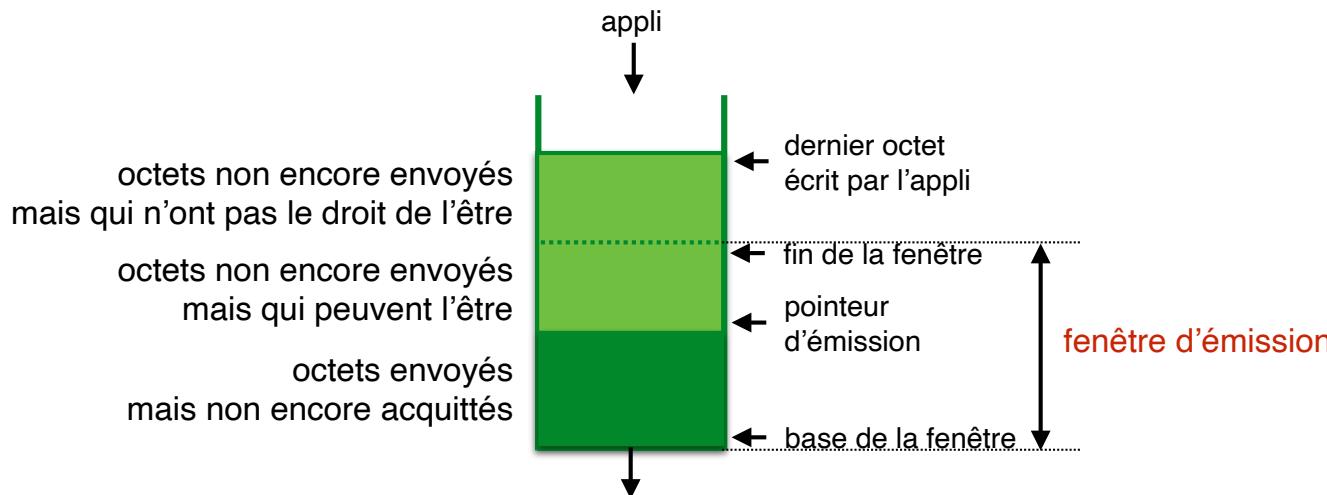
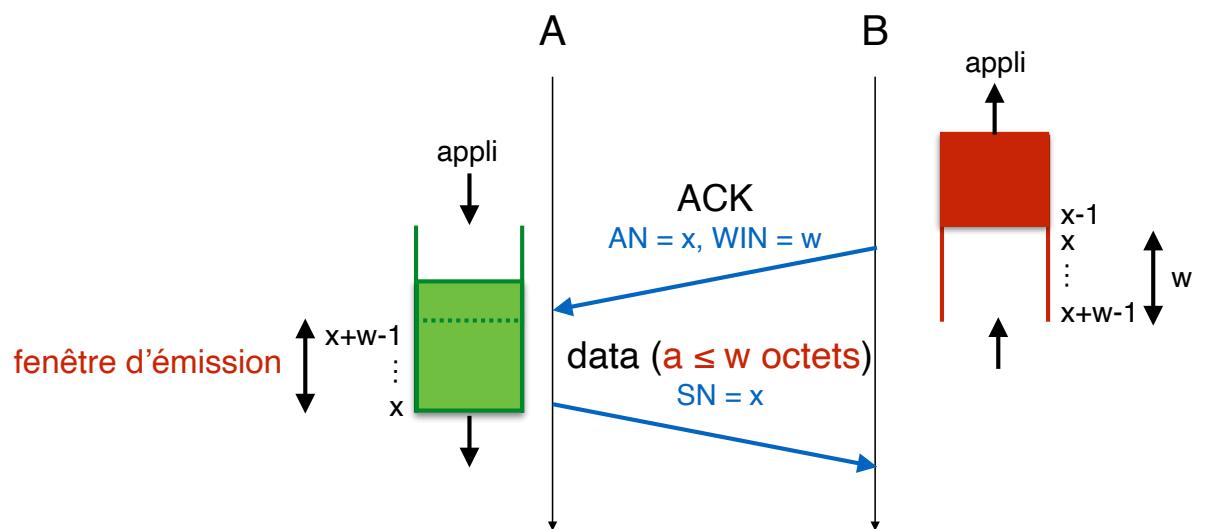
Contrôle de flux

- TCP offre un mécanisme de contrôle de flux
 - permettant à l'émetteur d'adapter sa vitesse d'émission à la capacité de traitement du récepteur
 - au travers d'une **fenêtre d'émission** appelée « fenêtre de contrôle de flux » (*Flow Control Window*)
 - plage d'octets que l'émetteur a le droit d'émettre
 - afin ne pas saturer le récepteur
 - corrélée à la place libre dans le buffer de réception du récepteur



Contrôle de flux

- Lorsque B envoie à A un ACK avec
 - $AN = x$
 - $WIN = w$
- B donne le droit à A d'envoyer
 - w octets de données
 - dont les numéros de séquence sont compris entre x et $x+w-1$



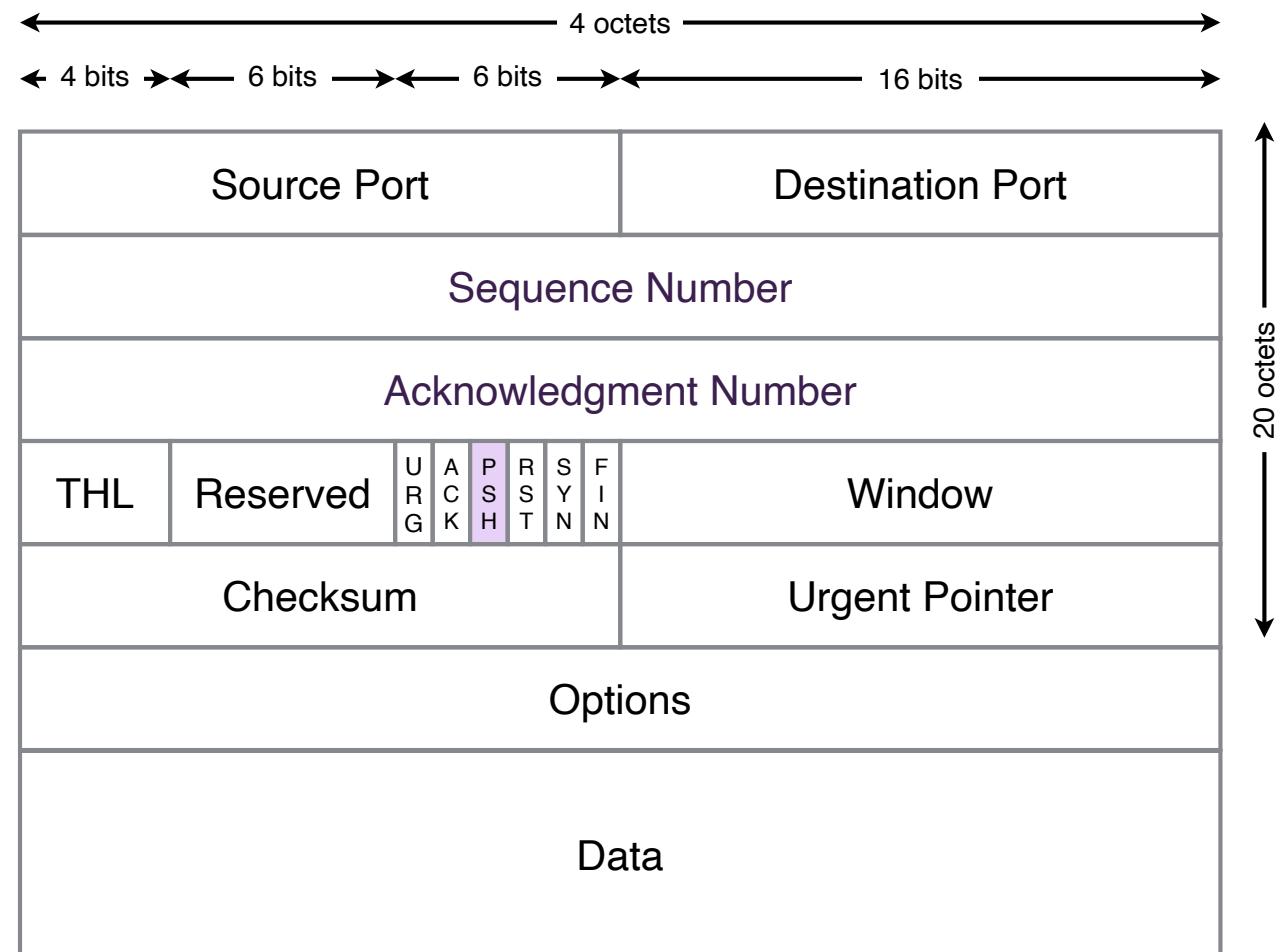
Contrôle de congestion

- TCP offre un mécanisme de contrôle de congestion
 - permettant à l'émetteur d'adapter sa vitesse d'émission à la capacité de transfert du réseau
 - au travers d'une deuxième fenêtre d'émission appelée « fenêtre de congestion » (Congestion Window)
 - plage d'octets que l'émetteur à le droit d'émettre
 - afin ne pas saturer le réseau
 - calculée en fonction des segments de données envoyés et des segments perdus (sans ACK)
- La fenêtre d'émission utilisée par TCP est en réalité l'intersection des deux fenêtres
 - fenêtre d'émission = fenêtre de contrôle de flux \cap fenêtre de contrôle de congestion

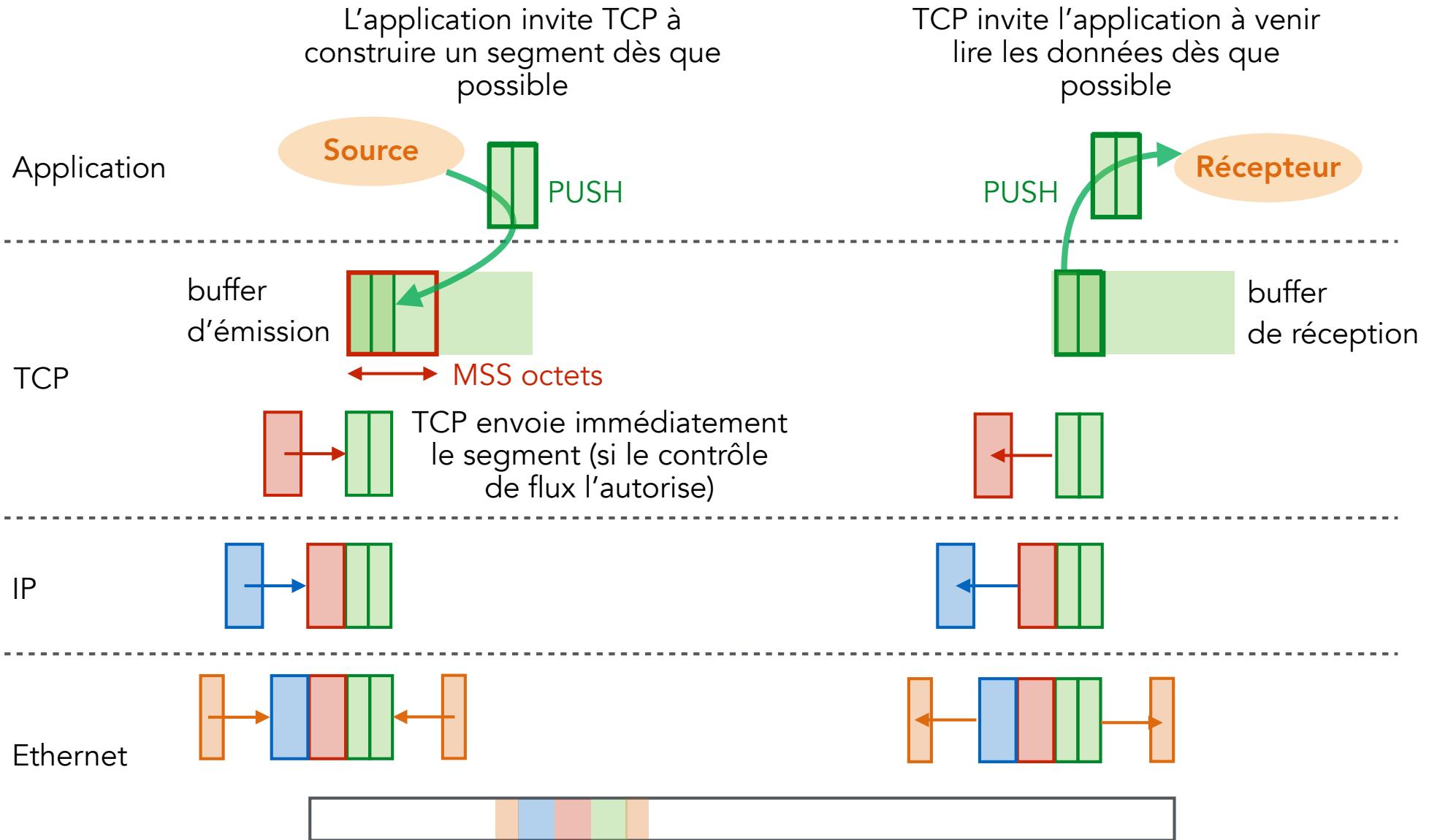
Drapeau PUSH

- 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

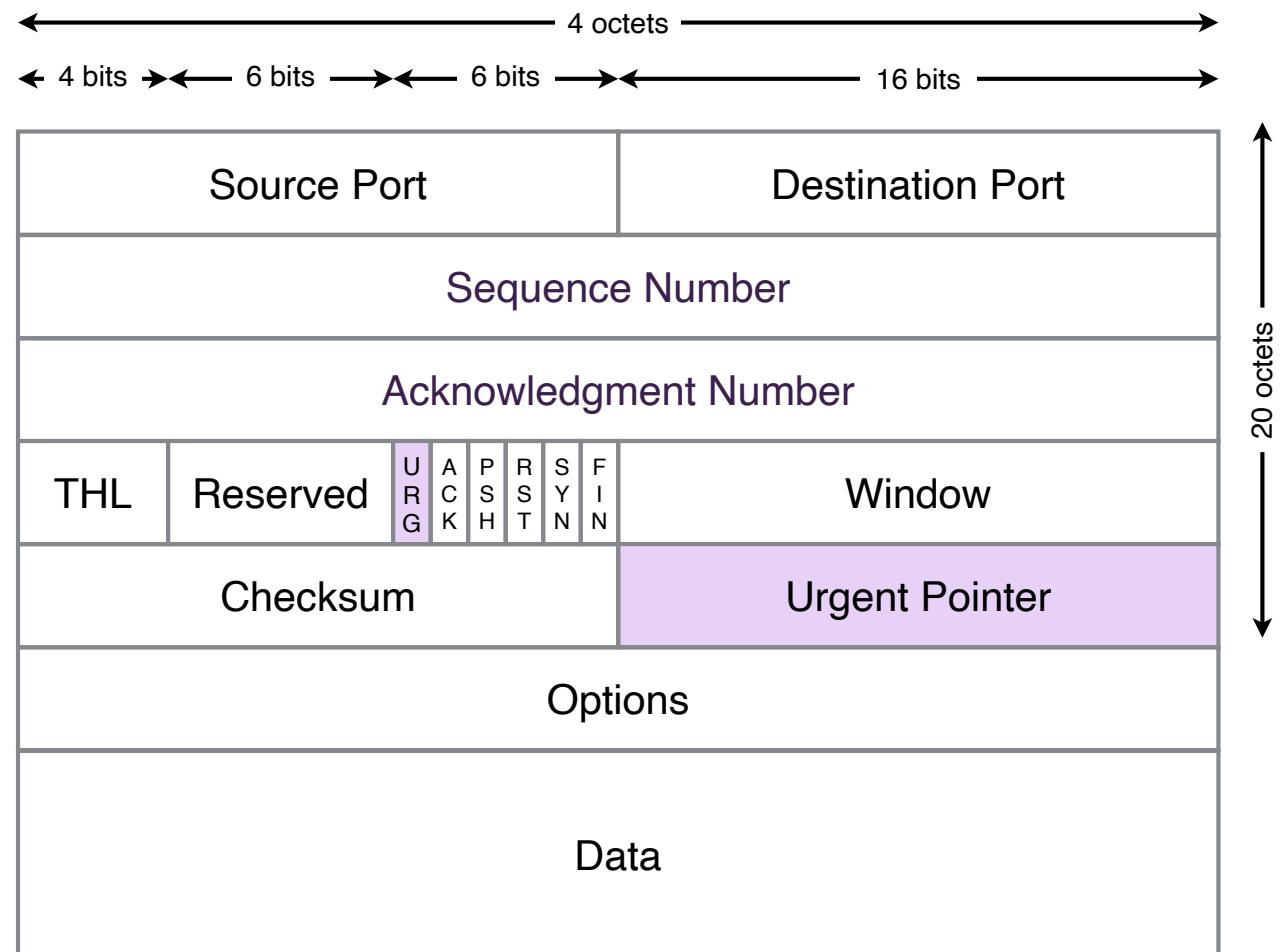


Drapeau PUSH



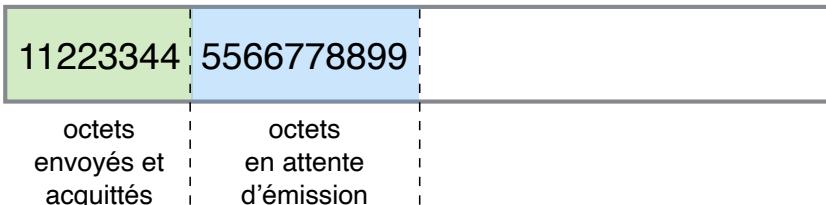
Drapeau URG

- Drapeau URG
 - indique que le segment contient des données « urgentes » (prioritaires ou critiques)
 - ces données sont placées au début du champ de données
- Champ *Urgent Pointer* (16 bits)
 - indique la fin des données urgentes
 - ces données sont lues en priorité par l'application réceptrice



Champ Urgent Pointer

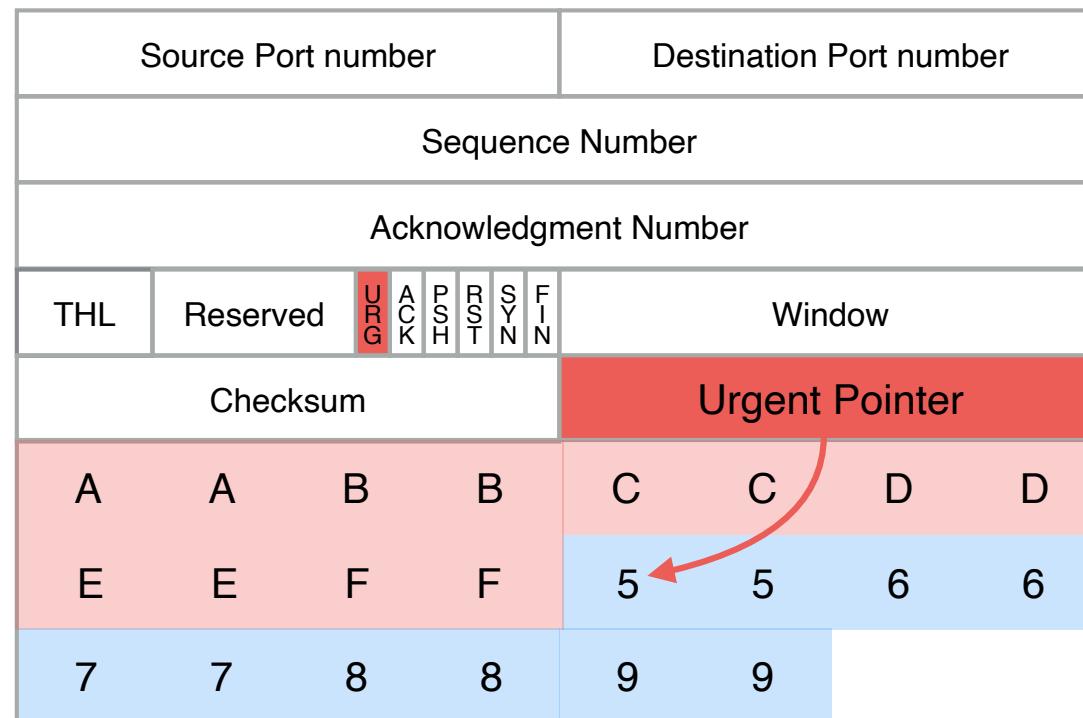
tampon d'émission



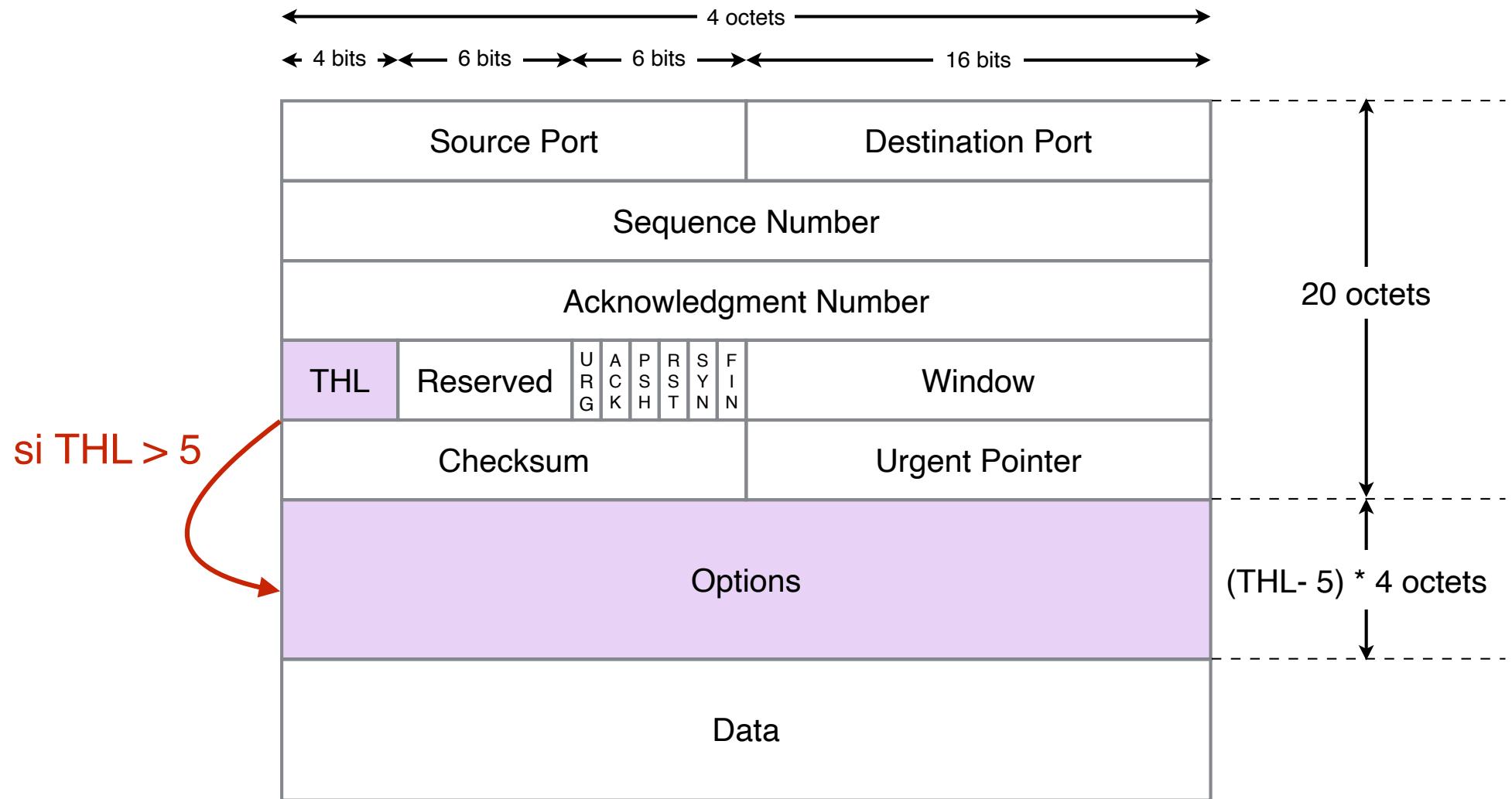
ordre de lecture des octets reçus



tampon d'émission d'octets urgents



Options TCP



Options TCP

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

Options TCP

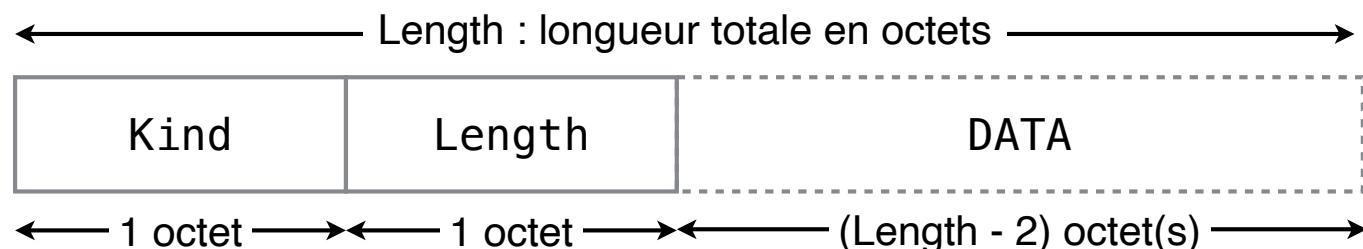
Format de l'option *End of Options List* (EOL) : type = 0



Format de l'option *No OPeration* (NOP) : type = 1

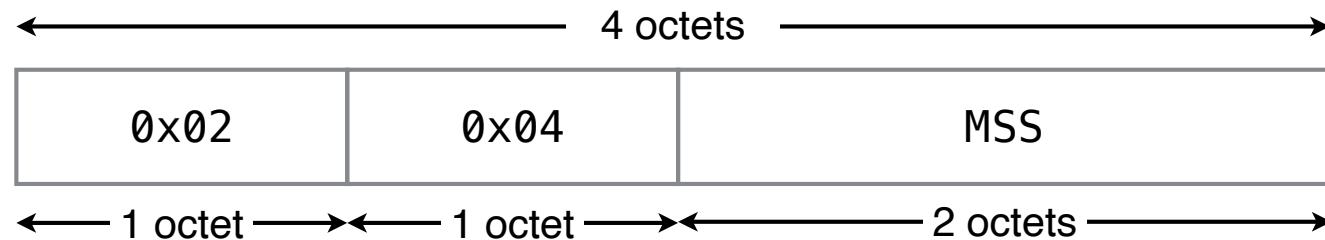


Format des options de type > 1

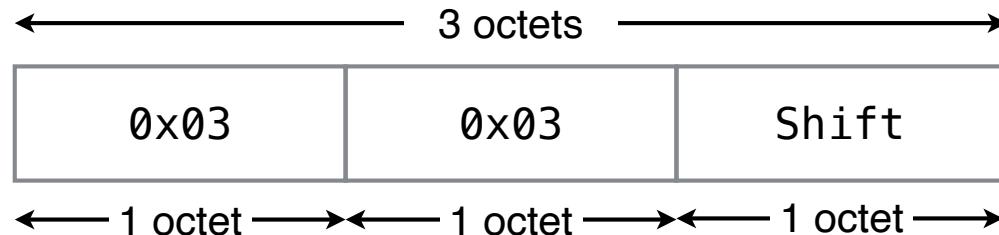


Options TCP

Maximum Segment Size (MSS) : type = 2

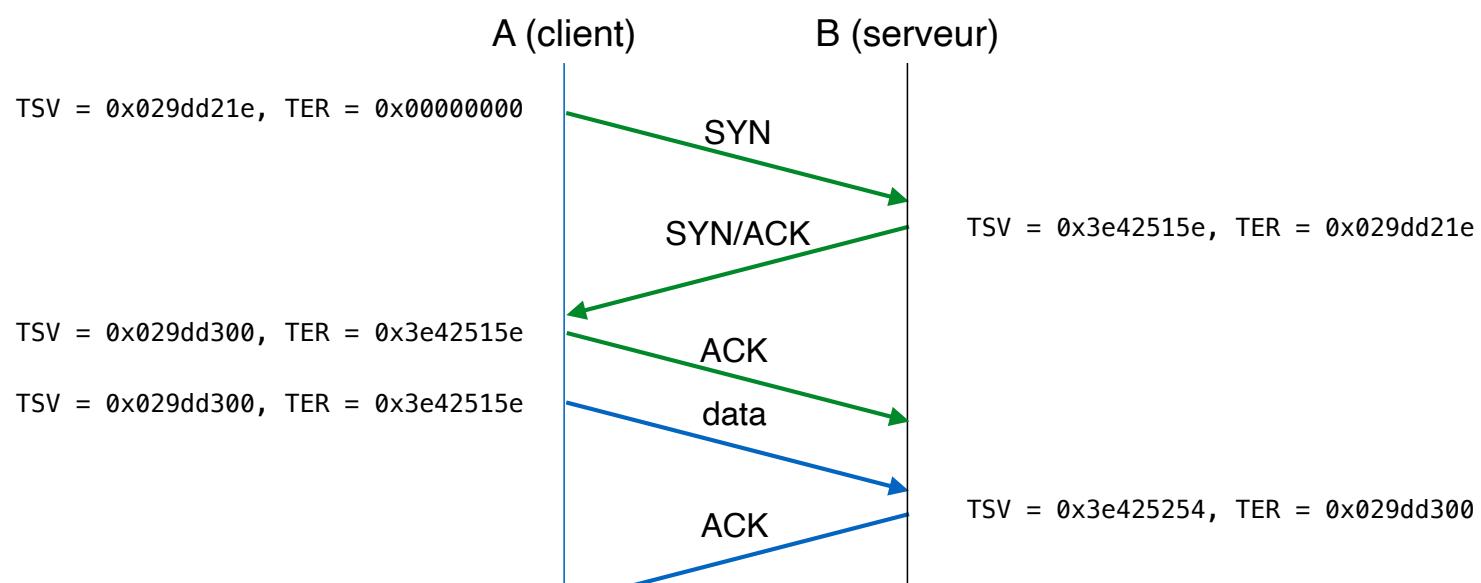


Windows Scale (WSopt) : type = 3



Options TCP

Time Stamp (TS) : type = 8



Conclusion

- TCP comme UDP
 - Multiplexage et démultiplexage
 - Détection d'erreur
 - somme de contrôle
- TCP à l'encontre d'UDP
 - 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 d'émission
 - Contrôle de congestion

A faire

- Cours 10
 - à relire attentivement
- Devoir 10 sur Moodle
 - date de rendu : dimanche 24 novembre