Ministère de l'enseignement supérieur et de la recherche scientifique

École supérieure en informatique de Sidi Bel Abbès

Module: RÉSEAUX

Chapitre : Protocoles de transport

PROTOCOLES DE TRANSPORT

1. RÔLE DU TRANSPORT
2. LE PROTOCOLE UDP
3. LE PROTOCOLE TCP

PROTOCOLES DE TRANSPORT - PLAN

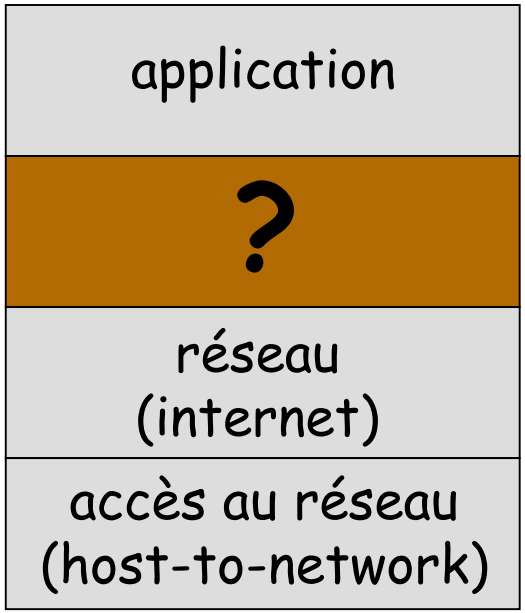
1. **RÔLE DU TRANSPORT**
2. LE PROTOCOLE UDP
3. LE PROTOCOLE TCP

PROBLÉMATIQUE

- DIFFÉRENTES TECHNOLOGIES POSSIBLES POUR CONNECTER UN ENSEMBLE DE MACHINES
  - LAN
  - WAN
  - INTER-RÉSEAUX

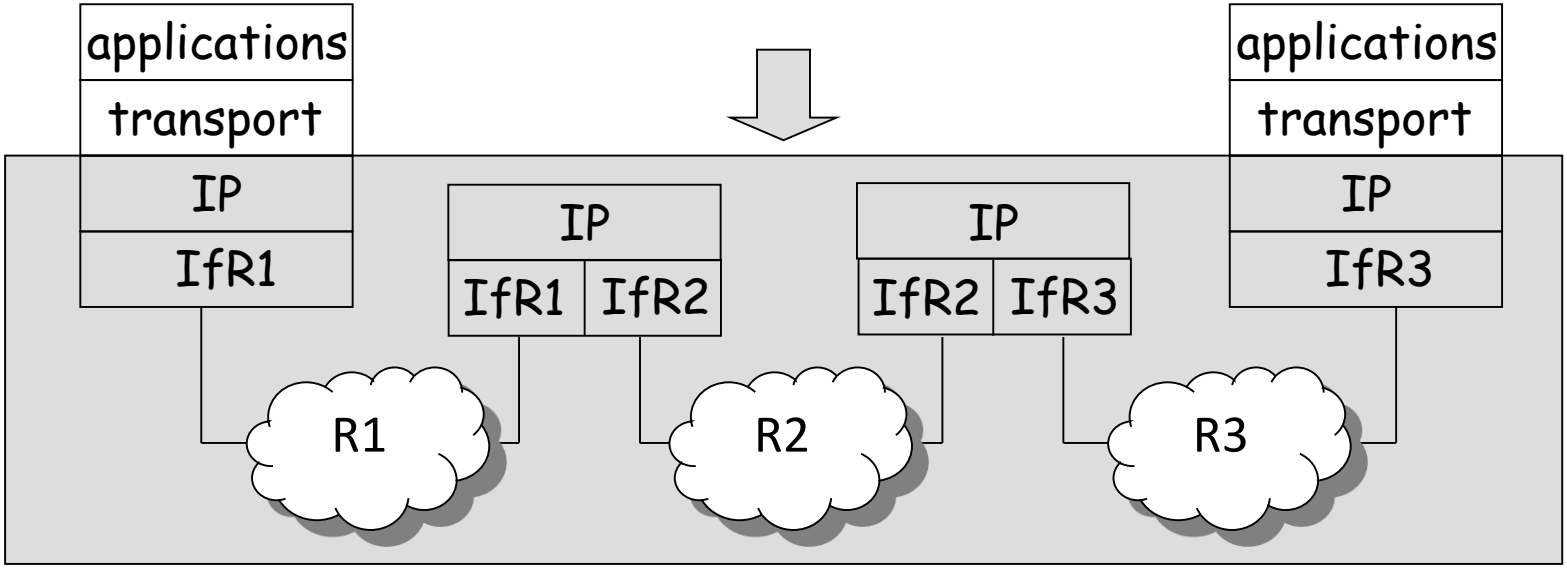
↳SERVICE DE REMISE DE PAQUETS DE MACHINE À MACHINE

- COMMENT PASSER DE CE SERVICE À UN CANAL DE COMMUNICATION DE PROCESSUS À PROCESSUS ?



système de communication
vu par les protocoles de transport

LE SYSTÈME DE COMMUNICATION



- SERVICE FOURNI PAR IP
  - ROUTAGE À TRAVERS UNE INTERCONNEXION DE RÉSEAUX
  - FRAGMENTATION/RÉASSEMBLAGE
  - SERVICE NON CONNECTÉ, *BEST EFFORT*

CHALLENGES

- PRENDRE EN COMPTE LE SERVICE FOURNI PAR LA COUCHE RÉSEAU
  - PERTES DE PAQUETS
  - DÉSÉQUENCEMENTS
  - DUPLICATIONS
  - ERREURS
  - MTU
  - TEMPS DE TRAVERSÉE IMPRÉVISIBLES
  - …
- PRENDRE EN COMPTE LES BESOINS DES APPLICATIONS
  - GARANTIE DE REMISE DES MESSAGES
  - SÉQUENCEMENT
  - ABSENCE DE DUPLICATIONS
  - ABSENCE DE MESSAGES ERRONÉS
  - MESSAGES DE LG QUELCONQUE
  - SYNCHRONISATION ENTRE L'ÉMETTEUR ET LE RÉCEPTEUR
  - CONTRÔLE DE FLUX PAR LE RÉCEPTEUR SUR L'ÉMETTEUR
  - SUPPORT DE PLUSIEURS APPLICATIONS SUR LE MÊME HÔTE
  - …

RÔLE DE LA COUCHE TRANSPORT

- TRANSFORMER LES PROPRIÉTÉS PAS TOUJOURS DÉSIRABLES DU RÉSEAU EN UN SERVICE DE HAUT NIVEAU SOUHAITÉ PAR LES APPLICATIONS

- PLUSIEURS DÉCLINAISONS DE PROTOCOLES DE TRANSPORT
  - UDP
  - TCP
  - SCTP

PROTOCOLES DE TRANSPORT - PLAN

1. RÔLE DU TRANSPORT
2. **LE PROTOCOLE UDP**
   - **LE (DÉ)MULTIPLEXAGE**
   - **LA NOTION DE PORT**
   - **LE DATAGRAMME UDP**
   - **AUTRES FONCTIONS ?**
3. LE PROTOCOLE TCP

LE PROTOCOLE UDP

- USER DATAGRAM PROTOCOL
- SE CONTENTE D'ÉTENDRE
  - LE SERVICE DE REMISE D'HÔTE À HÔTE À
  - UN SERVICE DE REMISE DE PROCESSUS À PROCESSUS

- PROBLÈME
  - PLUSIEURS APPLICATIONS PEUVENT TOURNER SIMULTANÉMENT SUR UN MÊME HÔTE

  ↳ IL FAUT DONC POUVOIR LES IDENTIFIER DE FAÇON NON AMBIGUË

  ↳ INTRODUCTION D'UN NIVEAU SUPPLÉMENTAIRE DE MULTIPLEXAGE
    - CF. LE CHAMP TYPE D'ETHERNET QUI IDENTIFIE À QUI DOIT ÊTRE DÉLIVRÉ LE CONTENU DU CHAMP DE DONNÉES DE LA TRAME
    - CF. LE CHAMP PROTOCOL DE IP QUI IDENTIFIE À QUI DOIT ÊTRE DÉLIVRÉ LE CONTENU DU CHAMP DE DONNÉES DU DATAGRAMME

MULTIPLEXAGE/DÉMULTIPLEXAGE

- PROBLÈME
  - COMMENT IDENTIFIER UN PROCESSUS (UNE APPLICATION) ?

- SOLUTION 0
  - ON PEUT IDENTIFIER DIRECTEMENT UN PROCESSUS SUR UNE MACHINE PAR SON *PID* (*PROCESS IDENTIFIER*)

MULTIPLEXAGE/DÉMULTIPLEXAGE

- SOLUTION
  - ON PEUT IDENTIFIER INDIRECTEMENT UN PROCESSUS PAR UNE RÉFÉRENCE ABSTRAITE (*ABSTRACT LOCATER*) APPELÉE PORT
    - UN PROCESSUS SOURCE ENVOIE UN MESSAGE SUR SON PORT
    - UN PROCESSUS DESTINATAIRE REÇOIT LE MESSAGE SUR SON PORT
    - PORT ~ BOÎTE AUX LETTRES

- RÉALISATION DE LA FONCTION DE (DÉ)MULTIPLEXAGE
  - CHAMP *PORT* (DE LA) *SOURCE* DU MESSAGE
  - CHAMP *PORT* (DE LA) *DESTINATION* DU MESSAGE

MULTIPLEXAGE/DÉMULTIPLEXAGE

- CHAMPS *PORT* CODÉS SUR 16 BITS
  - 65 535 VALEURS DIFFÉRENTES ➔ INSUFFISANT POUR IDENTIFIER TOUS LES PROCESSUS DE TOUS LES HÔTES DE L'INTERNET
  - LES PORTS N'ONT PAS UNE SIGNIFICATION GLOBALE
    - SIGNIFICATION RESTREINTE À UN HÔTE

    ↳ UN PROCESSUS EST IDENTIFIÉ PAR SON PORT SUR UNE MACHINE DONNÉE

  ↳ CLÉ DE DÉMULTIPLEXAGE DE UDP = (PORT, HÔTE)

MULTIPLEXAGE/DÉMULTIPLEXAGE

- COMMENT UN PROCESSUS CONNAÎT-IL LE PORT DE CELUI À QUI IL SOUHAITE ENVOYER UN MESSAGE ?
  - MODÈLE DE COMMUNICATION CLIENT/SERVEUR
  - UNE FOIS QUE LE CLIENT A CONTACTÉ LE SERVEUR, LE SERVEUR CONNAÎT LE PORT DU CLIENT

- COMMENT LE CLIENT CONNAÎT-IL LE PORT DU SERVEUR ?
- LE SERVEUR ACCEPTE DES MESSAGES SUR UN PORT CONNU DE TOUS (*WELL-KNOWN PORT*)
  - EX : DNS : 53, TELNET : 23, HTTP : 80

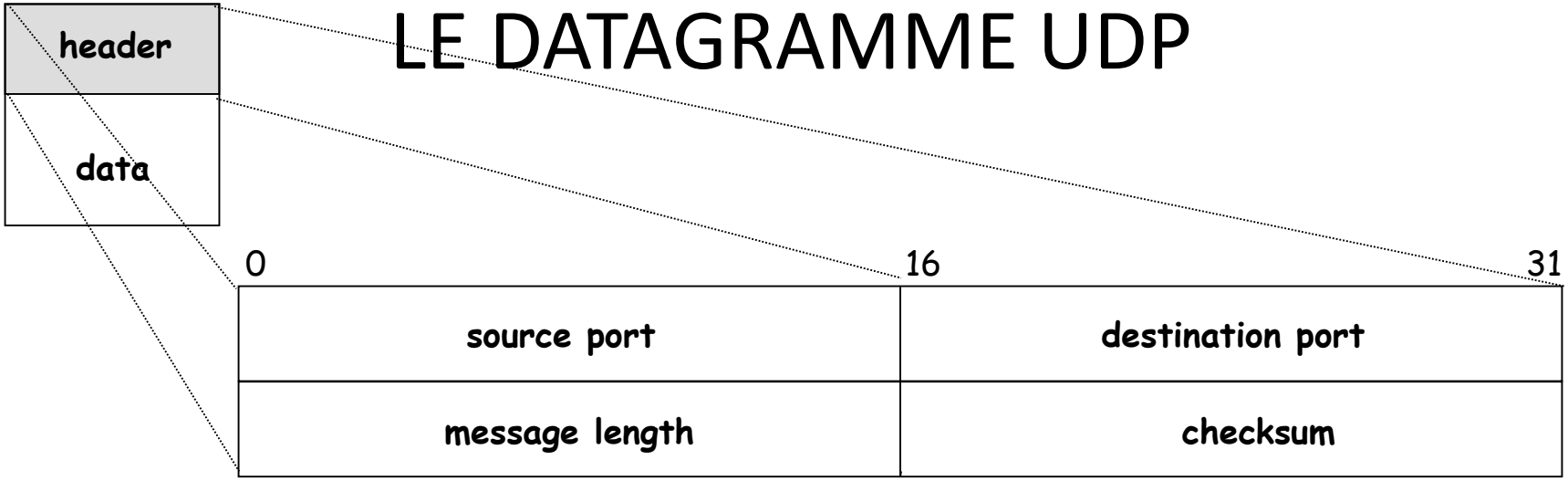
NUMÉROS DE PORT

- 3 CATÉGORIES
  - PORTS WELL-KNOWN : DE 0 À 1023
    - ALLOUÉS PAR L'IANA
    - SUR LA PLUPART DES SYSTÈMES, NE PEUVENT ÊTRE UTILISÉS QUE PAR DES PROCESSUS SYSTÈME (OU ROOT) OU DES PROGRAMMES EXÉCUTÉS PAR DES UTILISATEURS PRIVILÉGIÉS
  - PORTS REGISTERED : DE 1024 À 49 151
    - LISTÉS PAR L'IANA
    - SUR LA PLUPART DES SYSTÈMES, PEUVENT ÊTRE UTILISÉS PAR DES PROCESSUS UTILISATEUR ORDINAIRES OU DES PROGRAMMES EXÉCUTÉS PAR DES UTILISATEURS ORDINAIRES
  - PORTS DYNAMIC/PRIVATE : DE 49 152 À 65 535
    - ALLOUÉS DYNAMIQUEMENT

LES PORTS *WELL-KNOWN*

NO PORT	MOT-CLÉ	DESCRIPTION
7	ECHO	ECHO
11	USERS	ACTIVE USERS
13	DAYTIME	DAYTIME
37	TIME	TIME
42	NAMESERVER	HOST NAME SERVER
53	DOMAIN	DOMAIN NAME SERVER
67	BOOTPS	BOOT PROTOCOL SERVER
68	BOOTPC	BOOT PROTOCOL CLIENT
69	TFTP	TRIVIAL FILE TRANSFERT PROTOCOL
123	NTP	NETWORK TIME PROTOCOL
161	SNMP	SIMPLE NETWORK MANAGEMENT PROTOCOL

LE DATAGRAMME UDP



- FONCTIONNALITÉS AUTRES QUE LE (DÉ)MULTIPLEXAGE ?
  - PAS DE CONTRÔLE DE FLUX
  - PAS DE FIABILITÉ
  - DÉTECTION D'ERREURS ?
  - FRAGMENTATION ?

LE CHECKSUM UDP

- CALCUL OPTIONNEL AVEC IPV4, OBLIGATOIRE AVEC IPV6

- PORTÉE
  - L'EN-TÊTE UDP
  - LE CHAMP DE DONNÉES UDP
  - UN *PSEUDO-HEADER*
    - CHAMP IP *PROTOCOL* (8 BITS CADRÉS À DROITE SUR 16 BITS)
    - CHAMP IP *@SOURCE* (32 BITS)
    - CHAMP IP *@DEST*. (32 BITS)
    - CHAMP UDP *LENGTH* (16 BITS)

↳ UDP EST INDISSOCIABLE DE IP !

FRAGMENTATION

- EN THÉORIE
  - LES MESSAGES UDP PEUVENT ÊTRE FRAGMENTÉS PAR IP

- EN PRATIQUE
  - LA PLUPART DES APPLICATIONS UTILISANT UDP LIMITENT LEURS MESSAGES À 512 OCTETS

  ↳ PAS DE FRAGMENTATION

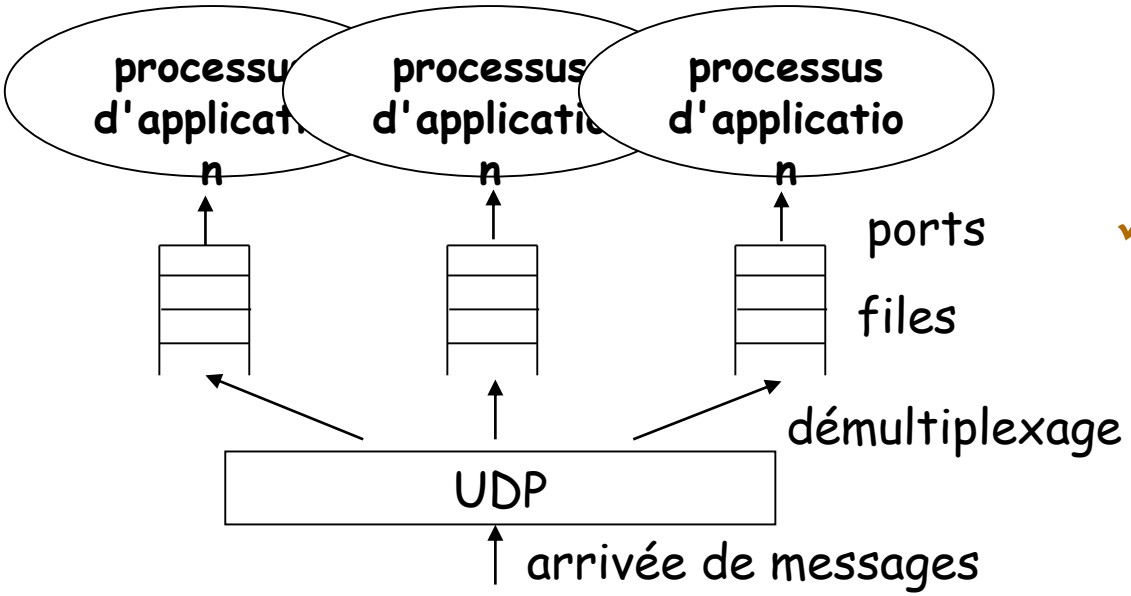
  ↳ PAS DE RISQUE DE MESSAGE INCOMPLET

IMPLÉMENTATION DES PORTS

- PORT = RÉFÉRENCE *ABSTRAITE*

  ↳ L'IMPLÉMENTATION DIFFÈRE D'UN OS À L'AUTRE !

- EN GÉNÉRAL, UN PORT = UNE FILE DE MESSAGES



- ✓ quand un msg arrive, UDP l'insère en fin de file
  - si la file est pleine, le msg est rejeté
- ✓ quand le processus veut recevoir un msg, il le retire de la tête de la file
  - si la file est vide, le processus se bloque jusqu'à ce qu'un msg soit disponible

PROTOCOLES DE TRANSPORT

1. RÔLE DU TRANSPORT
2. LE PROTOCOLE UDP
3. **LE PROTOCOLE TCP**
   - **TRANSPORT VS. LIAISON**
   - **FLUX D'OCTETS ET SEGMENTS**
   - **LE SEGMENT TCP**
   - **L'ÉTABLISSEMENT DE CONNEXION**
   - **LA LIBÉRATION DE CONNEXION**
   - **LE CONTRÔLE DE FLUX**
   - **LE CONTRÔLE D'ERREUR**

LE PROTOCOLE TCP

- TRANSMISSION CONTROL PROTOCOL
- OFFRE UN SERVICE DE REMISE
  - EN MODE CONNECTÉ
  - FIABLE
  - FULL-DUPLEX
  - DE FLUX D'OCTETS
- MET EN ŒUVRE DES MÉCANISMES DE
  - (DÉ)MULTIPLEXAGE
  - GESTION DE CONNEXIONS
  - CONTRÔLE D'ERREUR
  - CONTRÔLE DE FLUX
  - CONTRÔLE DE CONGESTION

COMPARAISON AVEC UNE LIAISON DE DONNÉES

1. Gestion des connexions
   - La liaison est bâtie sur un canal physique reliant toujours les 2 mêmes ETTD
   - TCP supporte des connexions entre 2 processus s'exécutant sur des hôtes quelconques de l'internet
   - ↳ Phase d'établissement plus complexe
2. Le RTT (*round trip time*) est
   - Pratiquement constant sur une liaison
   - Varie en fonction de l'heure de la connexion et de la "distance" séparant les 2 hôtes

COMPARAISON AVEC UNE LIAISON DE DONNÉES

3. LES UNITÉS DE DONNÉES
   - PEUVENT SE DOUBLER ET PEUVENT ÊTRE RETARDÉS DE FAÇON IMPRÉVISIBLE DANS LE RÉSEAU
   - ⇘ TCP DOIT PRÉVOIR LE CAS DE (TRÈS) VIEUX PAQUETS RÉAPPARAISSENT
4. LES BUFFERS DE RÉCEPTION
   - SONT PROPRES À *LA* LIAISON
   - SONT PARTAGÉS ENTRE TOUTES LES CONNEXIONS OUVERTES
   - ⇘ TCP DOIT ADAPTER LE MÉCANISME DE CONTRÔLE DE FLUX

COMPARAISON AVEC UNE LIAISON DE DONNÉES

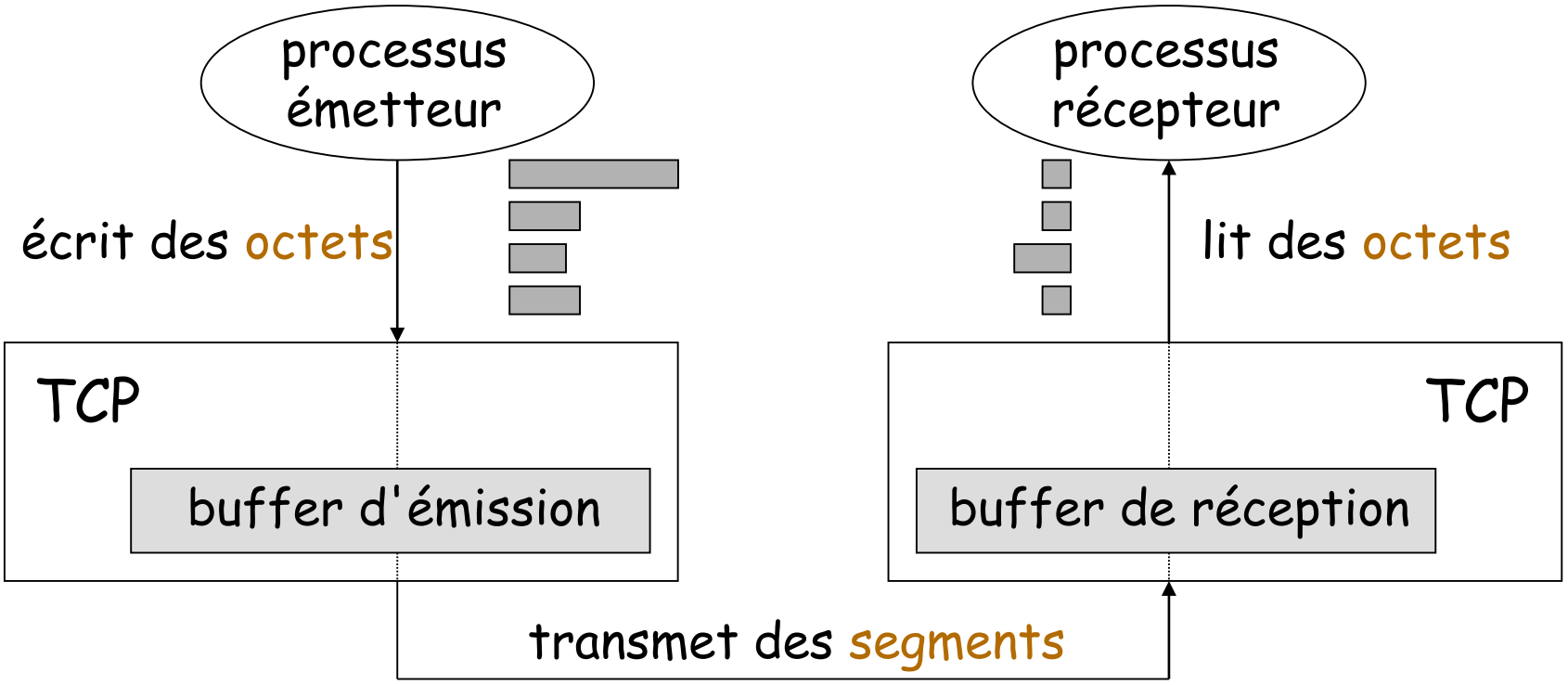
5. UNE CONGESTION
   - DU LIEN SUR UNE LIAISON DE DONNÉES NE PEUT PAS SE PRODUIRE SANS QUE L'ÉMETTEUR NE S'EN RENDE COMPTE
   - DU RÉSEAU PEUT SE PRODUIRE

   ↳ TCP VA METTRE EN ŒUVRE UN CONTRÔLE DE CONGESTION

BYTE-ORIENTED

- TCP EST ORIENTÉ FLUX D'OCTETS
  - LE PROCESSUS ÉMETTEUR "ÉCRIT" DES OCTETS SUR LA CONNEXION TCP
  - LE PROCESSUS RÉCEPTEUR "LIT" DES OCTETS SUR LA CONNEXION TCP
- TCP NE TRANSMET PAS D'OCTETS INDIVIDUELS
  - EN ÉMISSION
    - TCP BUFFERISE LES OCTETS JUSQU'À EN AVOIR UN NOMBRE RAISONNABLE
    - TCP FABRIQUE UN SEGMENT ET L'ENVOIE
  - EN RÉCEPTION
    - TCP VIDE LE CONTENU DU SEGMENT REÇU SANS UN BUFFER DE RÉCEPTION
    - LE PROCESSUS DESTINATAIRE VIENT Y LIRE LES OCTETS À SA GUISE

BYTE-ORIENTED

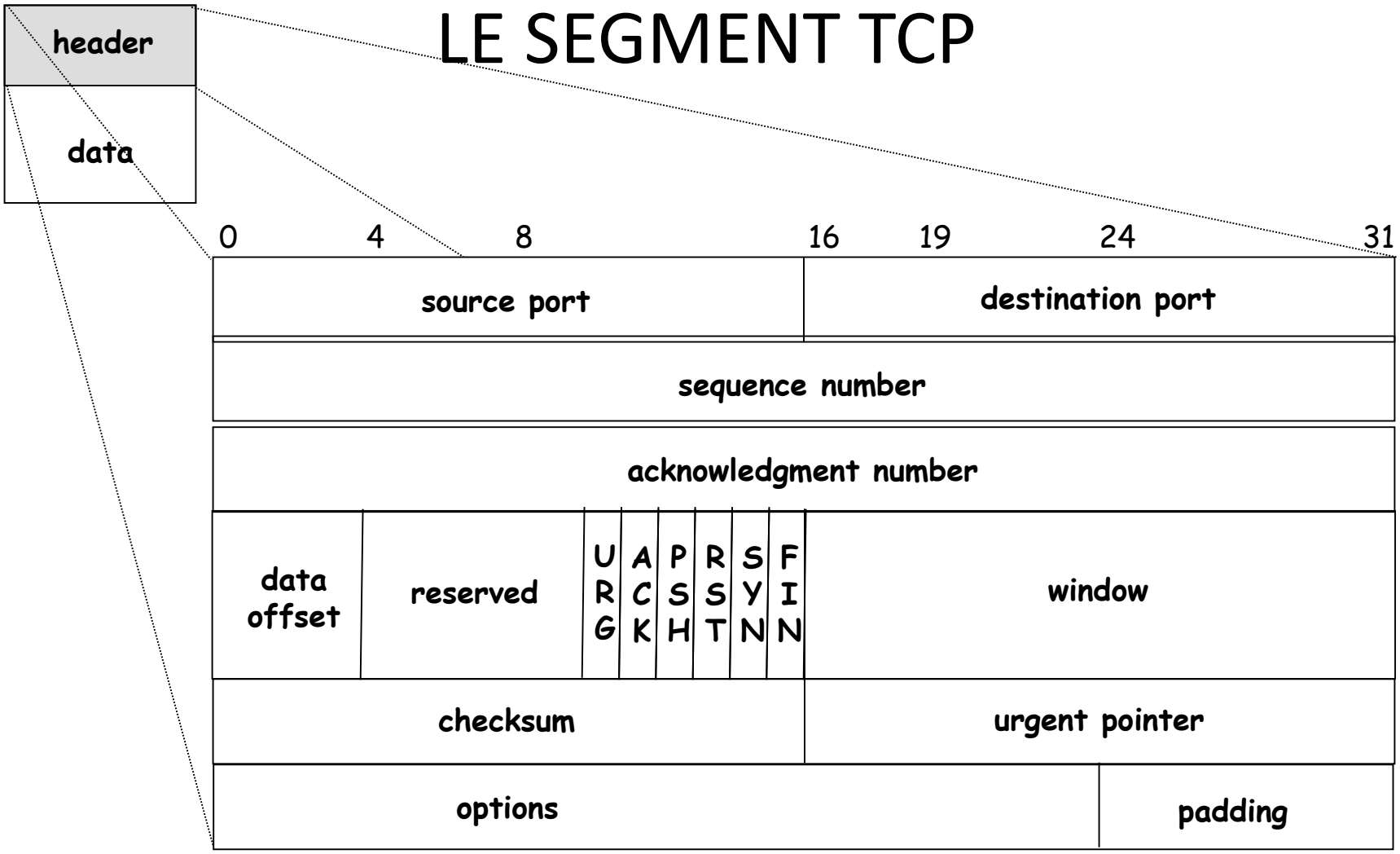


CONSTRUCTION D'UN SEGMENT

- QUAND EST-CE QUE TCP DÉCIDE D'ENVOYER UN SEGMENT ?

1. IL A MSS (MAXIMUMSIZE SEGMENT) OCTETS DE DONNÉES À ENVOYER
   - MSS = MTU – EN-TÊTE IP – EN-TÊTE TCP
   - 1460
2. LE PROCESSUS LUI DEMANDE EXPLICITEMENT
   - FONCTION *PUSH*
3. LE TEMPORISATEUR EXPIRE
   - POUR ÉVITER D'ATTENDRE TROP LONGTEMPS MSS OCTETS

LE SEGMENT TCP



LES CHAMPS DE L'EN-TÊTE TCP

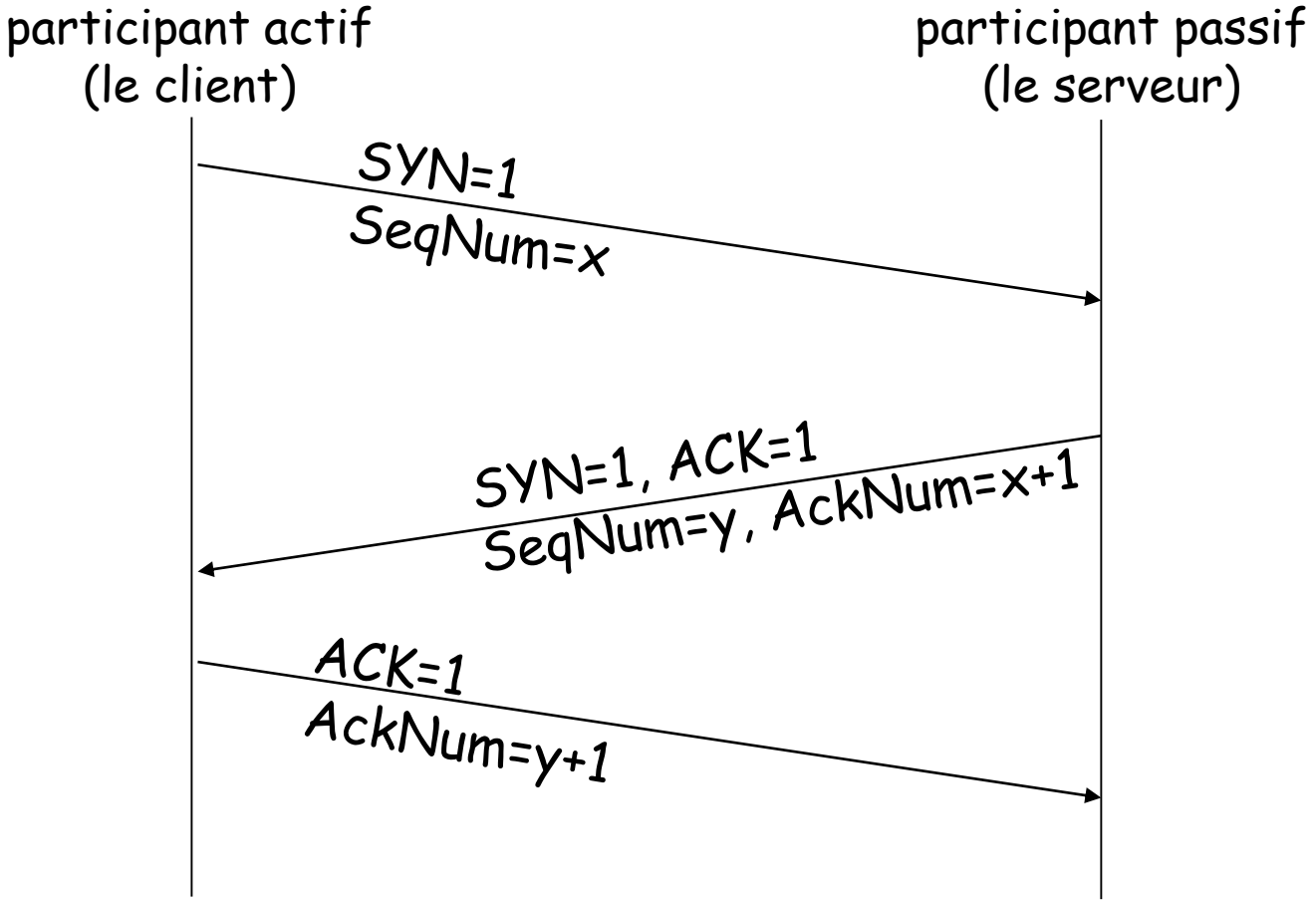
- SOURCE PORT : IDENTIFIE LE PROCESSUS SOURCE SUR LA MACHINE SOURCE
- DESTINATION PORT : IDENTIFIE LE PROCESSUS DESTINATAIRE SUR LA MACHINE DESTINATAIRE
- SEQUENCE NUMBER : N° DU 1ER OCTET DE DONNÉES DU SEGMENT
- ACKNOWLEDGMENT NUMBER : ACQUITTE TOUS LES OCTETS DE DONNÉES DE N° STRICTEMENT INFÉRIEUR
- DATA OFFSET : LG DE L'EN-TÊTE EN MOTS DE 32 BITS
- RESERVED : 6 BITS À 0
- URG : MIS À 1 POUR SIGNALER LA PRÉSENCE DE DONNÉES URGENTES
- ACK : MIS À 1 POUR INDIQUER QUE LE CHAMP ACKNOWLEDMENT NUMBER EST SIGNIFICATIF
- PSH : MIS À 1 POUR SIGNALER LA FIN D'UN MESSAGE LOGIQUE (PUSH)
- RST : MIS À 1 POUR RÉINITIALISER LA CONNEXION (PANNE, INCIDENT, SEGMENT INVALIDE)
- SYN : MIS À 1 POUR L'ÉTABLISSEMENT DE LA CONNEXION
- FIN : MIS À 1 POUR FERMER LE FLUX DE DONNÉES DANS UN SENS
- WINDOW : # D'OCTETS DE DONNÉES QUE LE DESTINATAIRE DU SEGMENT POURRA ÉMETTRE
- CHECKSUM : OBLIGATOIRE, CALCULÉ SUR LA TOTALITÉ DU SEGMENT ET SUR LE PSEUDO EN-TÊTE
- URGENT POINTER : POINTE SUR LA FIN (COMPRISE) DES DONNÉES URGENTES
- OPTIONS
- PADDING : ALIGNEMENT DE L'EN-TÊTE SUR 32 BITS

LES PORTS *WELL-KNOWN*

NO PORT	MOT-CLÉ	DESCRIPTION
20	FTP-DATA	FILE TRANSFER [DEFAULT DATA]
21	FTP	FILE TRANSFER [CONTROL]
23	TELNET	TELNET
25	SMTP	SIMPLE MAIL TRANSFER
37	TIME	TIME
42	NAMESERVER	HOST NAME SERVER
43	NICNAME	WHO IS
53	DOMAIN	DOMAIN NAME SERVER
79	FINGER	FINGER HHTPS 443
80	HTTP	WWW
110	POP3	POST OFFICE PROTOCOL - VERSION 3
111	SUNRPC	SUN REMOTE PROCEDURE CALL

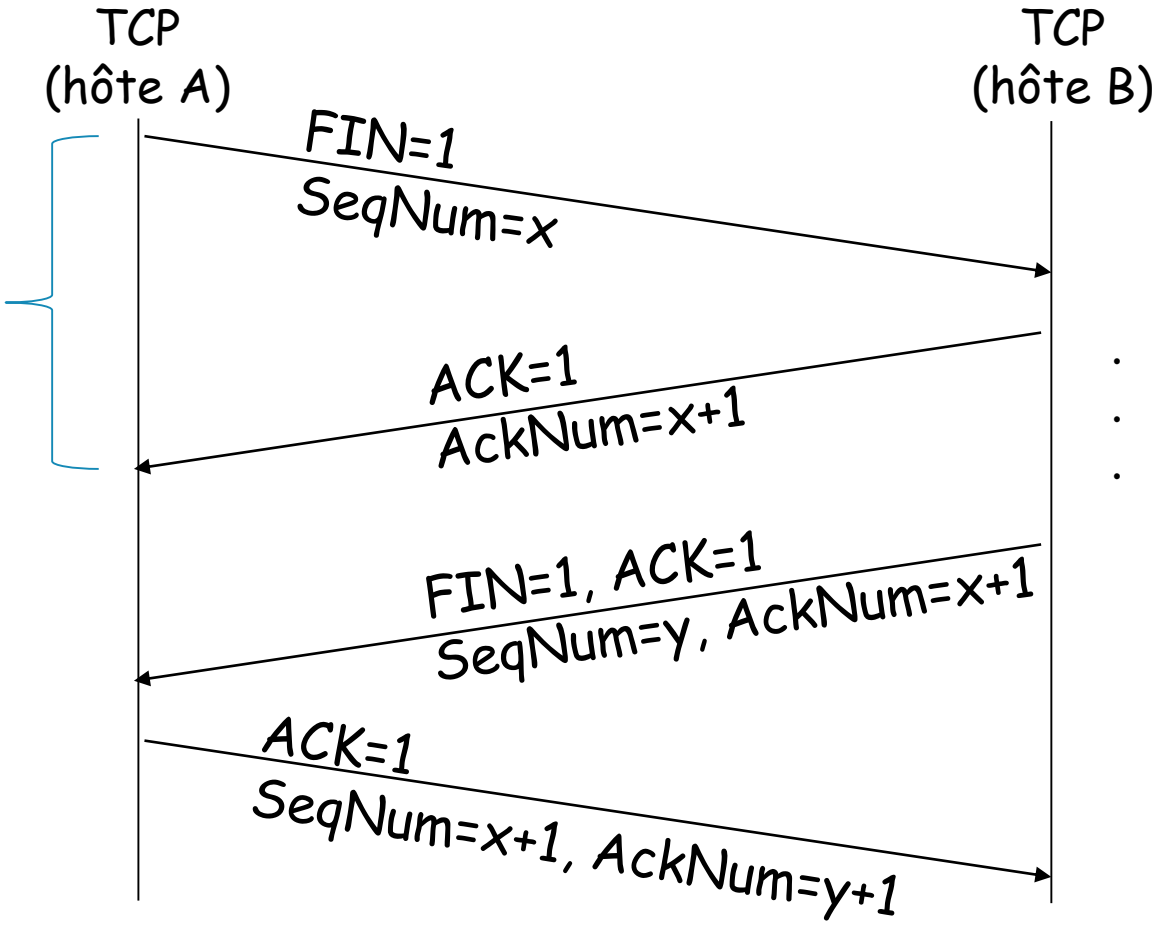
- LES 3 SEGMENTS ÉCHANGÉS

L'ÉTABLISSEMENT DE CONNEXION



- LES 2 SENS DE TRANSMISSION SONT FERMÉS SÉPARÉMENT

LA LIBÉRATION DE CONNEXION



TRANSFERT DE DONNÉES

- TCP assure un transfert de bout en bout fiable

↳Contrôle de flux :l'émetteur n'inonde pas le récepteur en émettant trop, et trop vite

- ↳ *Stop and wait*
- ↳ Fenêtre glissante

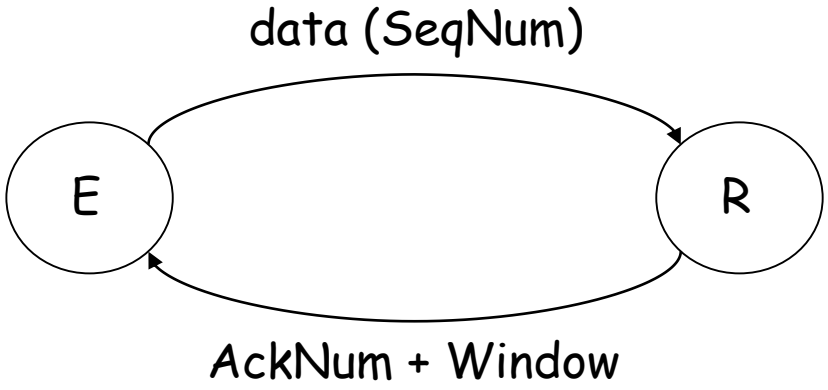
↳Contrôle d'erreur

LE CONTRÔLE DE FLUX

- Une entité réceptrice TCP dispose de ressources (buffers) en nombre limité
- La taille de la fenêtre reflète la disponibilité des buffers de réception
- Une entité TCP gère un nombre de connexions variable

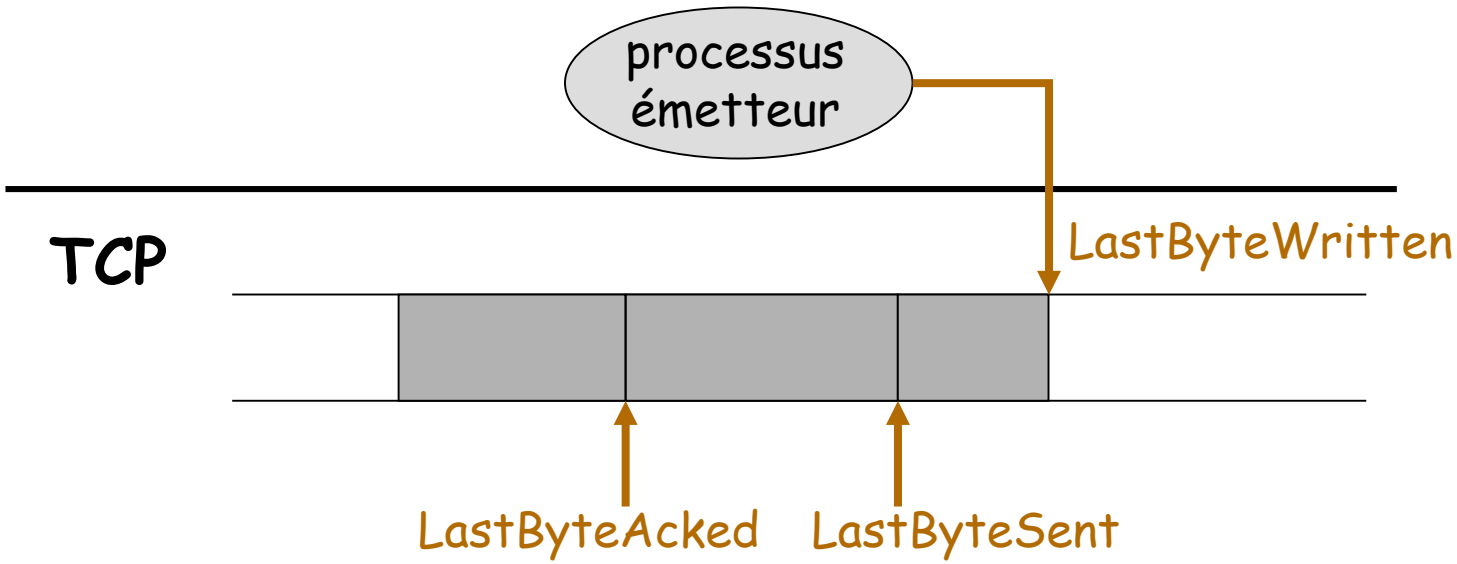
↳ fenêtre dynamique

- progression de la fenêtre par acquittement et crédit
- champ *window*
- indique le # d'octets de données que l'entité est prête à recevoir



BUFFER D'ÉMISSION

- En émission, un buffer stocke
  - Les données envoyées et en attente d'acquittement
  - Les données passées par le processus émetteur mais non encore émises

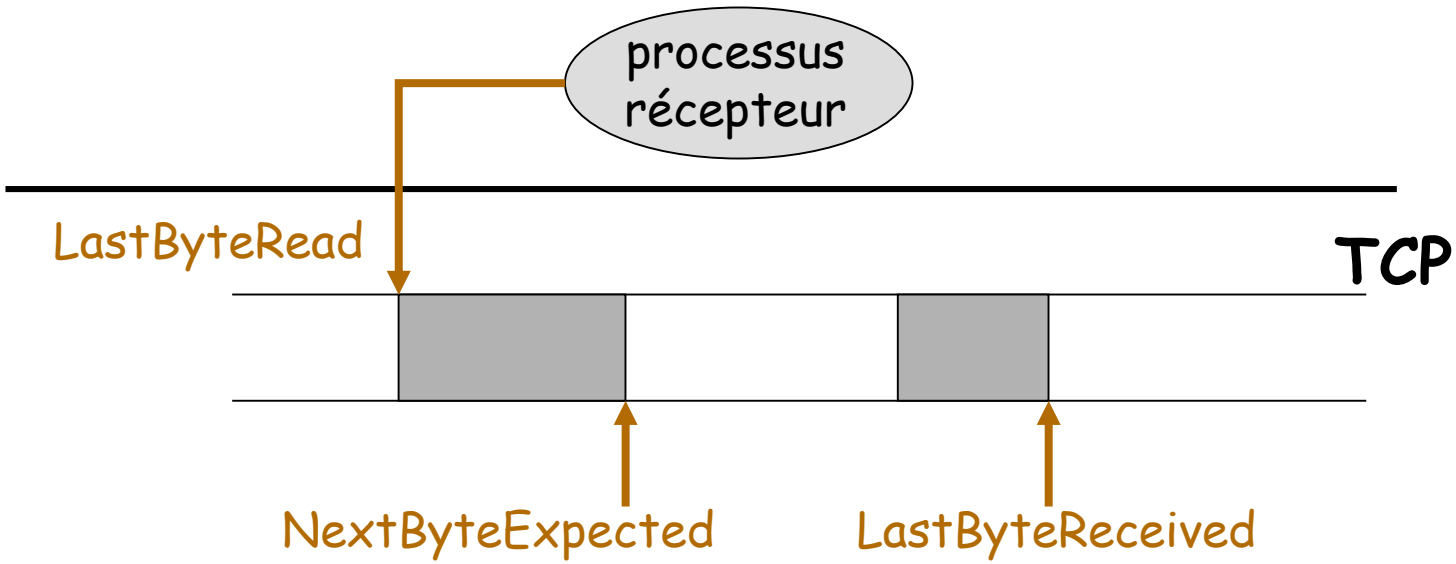


BUFFER D'ÉMISSION

- Remarque : on a toujours
  - Lastbyteacked <= lastbytesent <= lastbytewritten
- TCP vérifie à tout moment que
  - Lastbytesent – lastbyteacked <= advertisedwindow
- TCP calcule une fenêtre effective
  - Effectivewindow = advertisedwindow – (lastbytesend – lastbyteacked)
- En parallèle, TCP doit s'assurer que le processus émetteur ne sature pas on buffer
  - Si le processus veut écrire y octets et que (lastbytewritten – lastbyteacked) + y > maxsendbuffer TCP bloque l'écriture

BUFFER DE RÉCEPTION

- En réception, un buffer stocke
  - Les données dans l'ordre non encore lues par le processus récepteur
  - Les données déséquencées



BUFFER DE RÉCEPTION

- Remarque : on a toujours
  - Lastbyteread < nextbyteexpected <= lastbytereceived +1
- TCP vérifie à tout moment que
  - Maxrcvbuffer >= lastbytereceived – lastbyteread
- TCP communique une fenêtre mise à jour
  - Advertisedwindow = maxrecvbuffer – (lastbytereceived – lastbyteread)
- Au fur et à mesure que les données arrivent
  - TCP les acquitte si tous les octets précédents ont été reçus
  - Lastbytereceived glisse vers la droite ➔ rétrécissement possible de la fenêtre

RÉOUVERTURE DE FENÊTRE

- Problème
  - La fenêtre peut avoir été fermée par le récepteur
  - Un ACK ne peut être envoyé que sur réception de données (approche *smart sender/dumb receiver*)
  - ↳ Comment l'émetteur peut-il se rendre compte de la réouverture de la fenêtre ?
- Solution
  - Lorsque l'émetteur reçoit une advertisedwindow à 0, il envoie périodiquement un segment avec 1 octet de données pour provoquer l'envoi d'un ACK
  - Scaling

LE CONTRÔLE DE CONGESTION

- informellement : “trop de sources envoient trop de données, trop rapidement, plus que ce que le réseau peut absorber ”
- différent du contrôle de flux !

- perte de segments (overflow des buffers aux routeurs)
- long retards (temps d’attente élevés dans les buffers des routeurs)
- un des 10 problèmes majeurs en réseau !

- Reno vegas , cubic

LE CONTRÔLE DE CONGESTION

- " recherche" de la bande passante utilisable
- idéalement : transmettre aussi vite que possible sans perte (congwin aussi grand que possible)
- initialiser congwin à 1 MSS
- augmenter congwin jusqu 'à avoir une perte (congestion)
- perte : réinitialiser congwin, puis recommencer à augmenter

TCP calcul de la fenêtre de congestion "idéale »

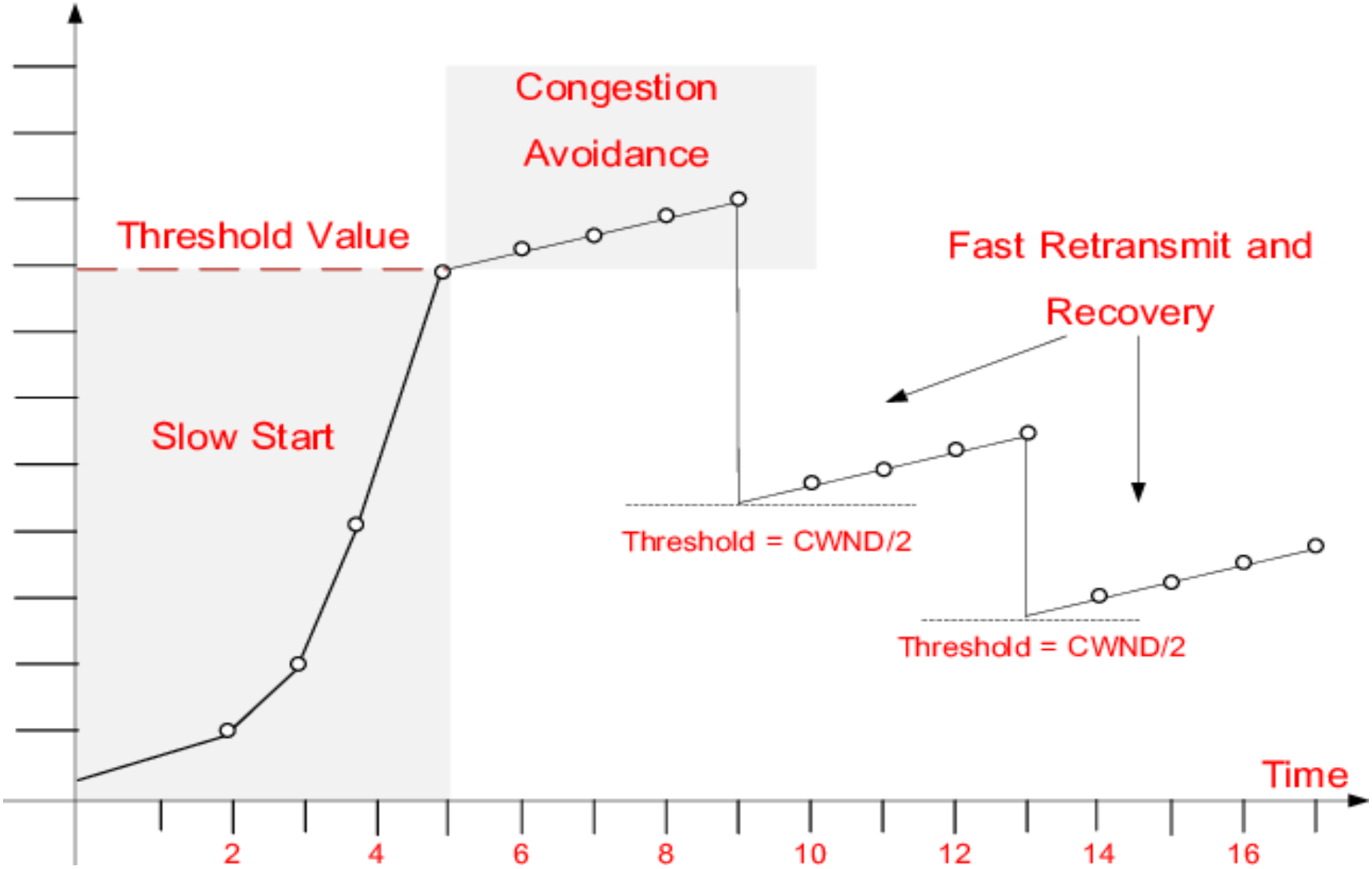
deux "phases"

- le slow start
- l'évitement de congestion

variables importantes

✓ congwin

✓ threshold: définit le seuil entre les deux phases (quand finit le slow start et commence l'évitement de congestion)


Congestion
Avoidance
Threshold Value
Fast Retransmit and
Recovery
Slow Start
Threshold = CWND/2
Threshold = CWND/2
Time
2
4
6
8
10
12
14
16

CONTRÔLE D'ERREUR

- Repose sur
  - Le champ checksum
  - Le champ sequencenumber
  - Des acquittements positifs (*dumb receiver* ➔ pas d'acquittements négatifs)
  - Un temporisateur de retransmission
  - Des retransmissions
- Rtt variable

↳ Dimensionnement dynamique du temporisateur

2 méthodes

- PROTOCOLE GO-BACK-N (GBN)
- SELECTIVE REPEAT

DIMENSIONNEMENT DU TEMPORISATEUR

- Algorithme initial

1. TCP calcule une estimation du RTT par une moyenne pondérée entre l'estimation précédemment calculée et le dernier échantillon mesuré du RTT
   - Estimatedrtt = α estimatedrtt + (1 – α) samplertt
   - Samplertt est obtenu en mesurant le délai séparant l'émission d'un segment de la réception de son acquittement
   - $0,8 < \alpha < 0,9$
2. TCP calcule la valeur du temporisateur
   - RTO Timeout = min (UBOUND, max (LBOUND, β estimatedrtt))
   - UBOUND est une limite supérieure sur le timer (p.E. 60 s)
   - LBOUND est une limite inférieure sur le timer (p.E. 1 s)
   - $1,3 <= \beta <= 2$