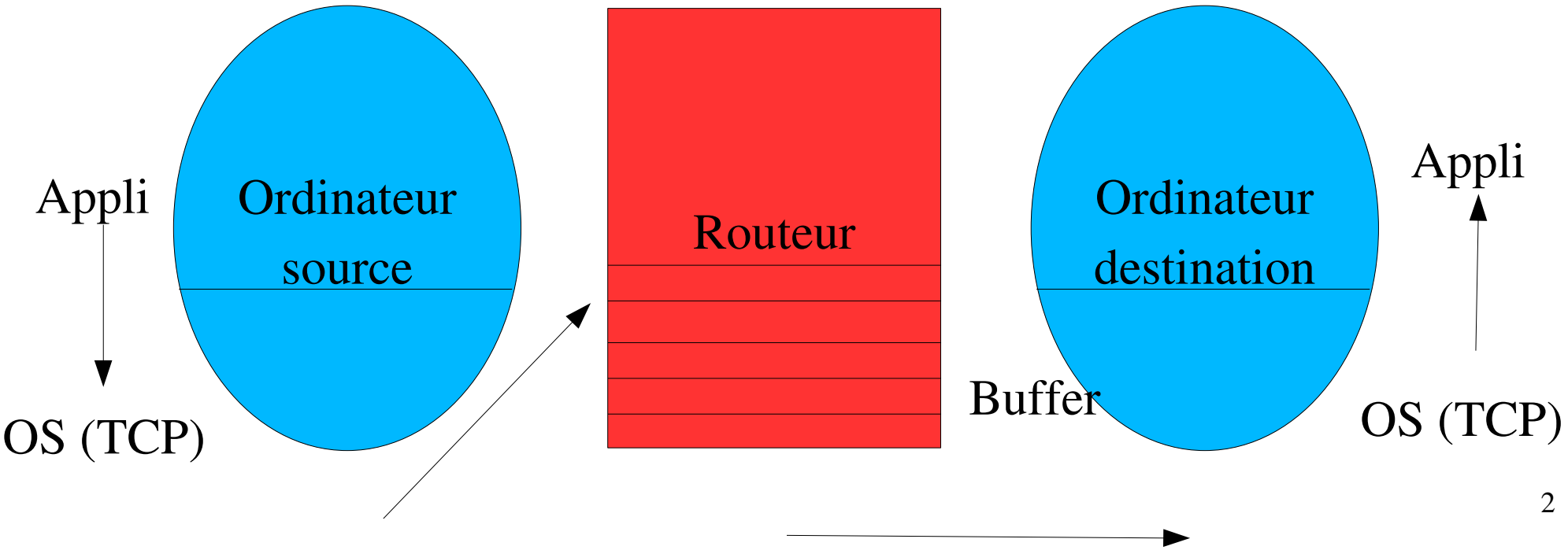
Contrôle de congestion dans le protocole TCP

Eugen Dedu
(utilise des transparents de S. Linck et des figures de Sanadidi)
UFC, UFR STGI, master SRM, Montbéliard
septembre 2007
Eugen.Dedu@pu-pm.univ-fcomte.fr

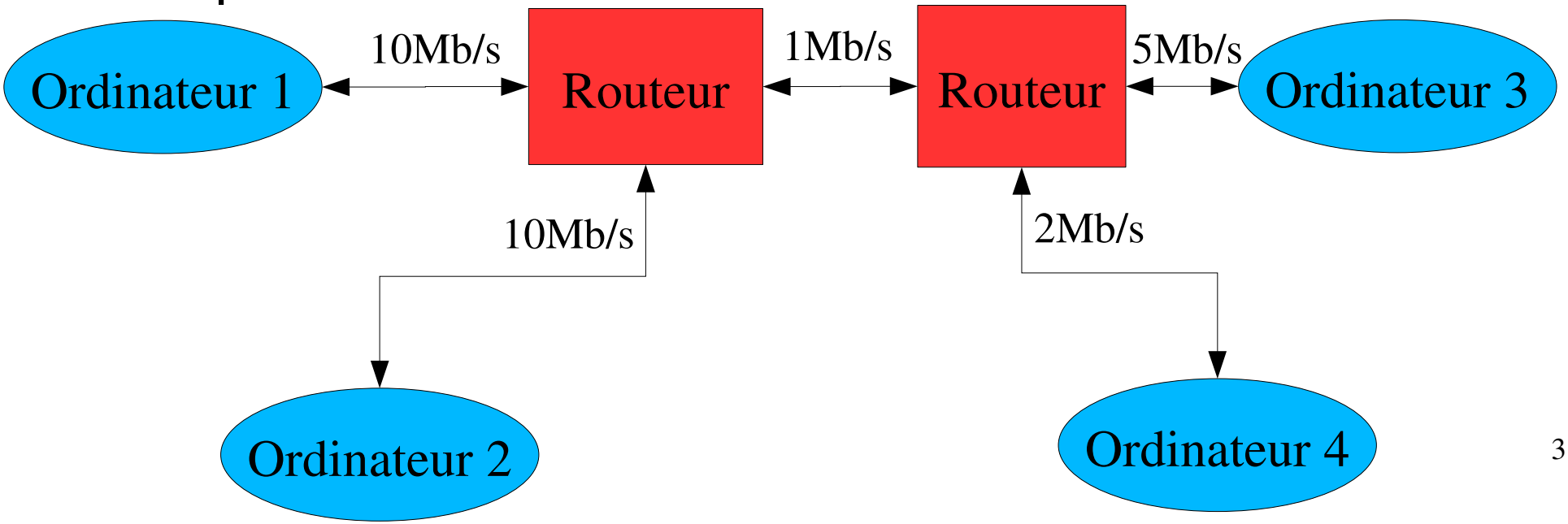
Introduction aux réseaux

- Transfert d'une donnée, par ex. un fichier par FTP
- Division en paquets par TCP de l'ordinateur source
- File d'attente (buffers) des routeurs



Introduction à la congestion

- Lien (bande passante, latence), analogie avec le train
- Ex. : taux de transfert idéals entre différents ordinateurs
- CC = adaptation à la bande passante disponible à chaque instant

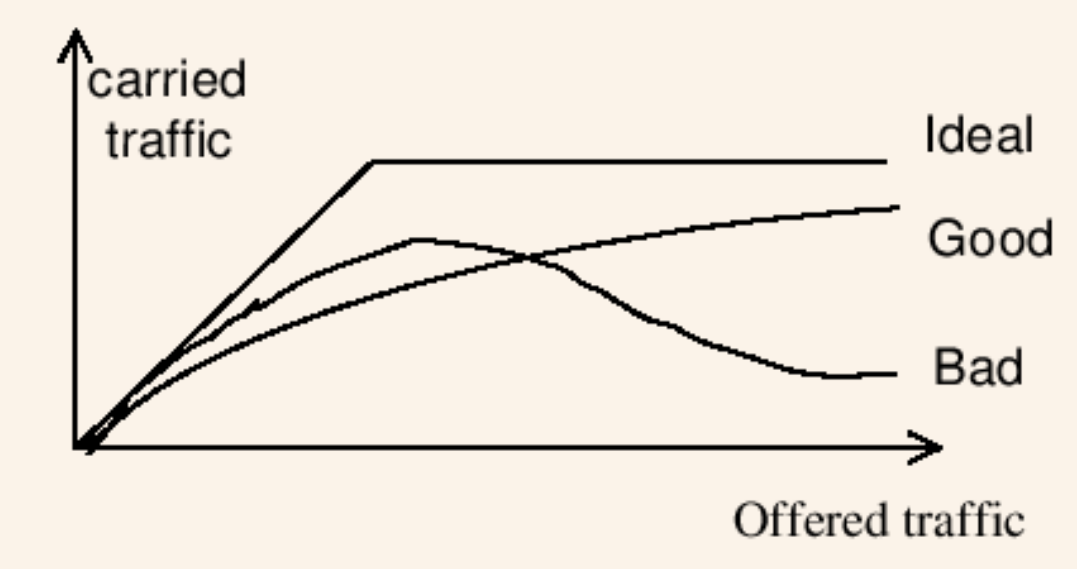


Inconvénients de la congestion

- Congestion = routeur avec file d'attente pleine
- Si routeur avec file d'attente (quasi-)pleine :
  - rejet/perte de paquet (car débordement mémoire routeur)
  - délais importants de transfert (car attente dans les files des routeurs)
- Causes de la perte d'un paquet :
  - problème matériel
  - problème d'environnement (souvent dans les réseaux sans fil)
  - mais surtout congestion d'un routeur
- En filaire, perte d'un paquet ~= congestion d'un routeur

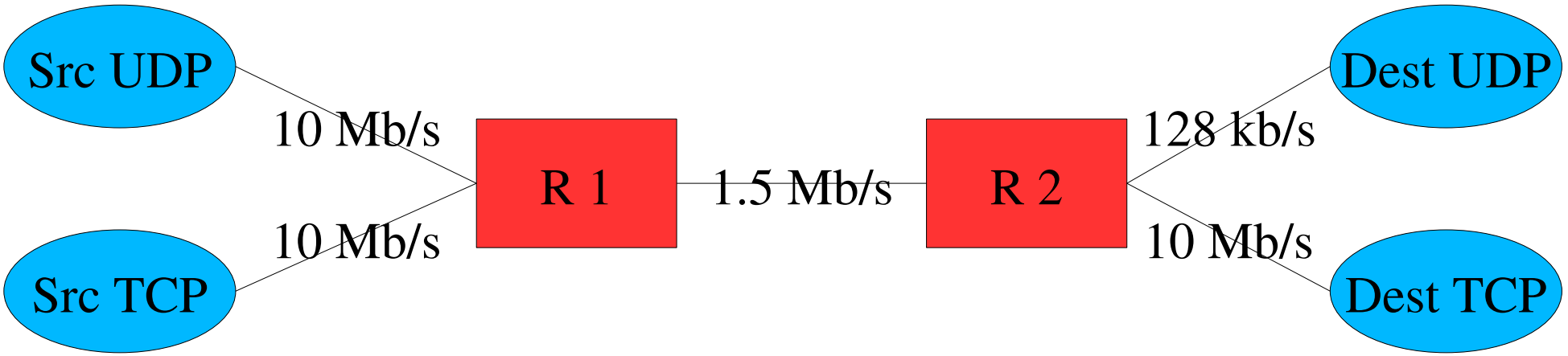
Exemple de congestion

- Ex. réel d'écroulement du réseau en 1986 [Jacobson]
  - liaison de 400 m avec 2 “routeurs”
  - débit des données : 32 kb/s -> 40 b/s
- Ex. d'un transfer avec CC



Ex. de problème réseau

- Un utilisateur se plaint d'un débit faible
- En regardant, une application UDP est très utilisé
- UDP ne s'adapte pas à la bande passante disponible
- TCP s'adapte => TCP est défavorisé



Netographie

- “Practical Analysis of TCP Implementations: Tahoe, I
- “Internetworking with TCP/IP”, Douglas Comer, 2000
- Wikipedia
- Page de Sally Floyd : ECN, RED, NS2, TFRC, DCCP

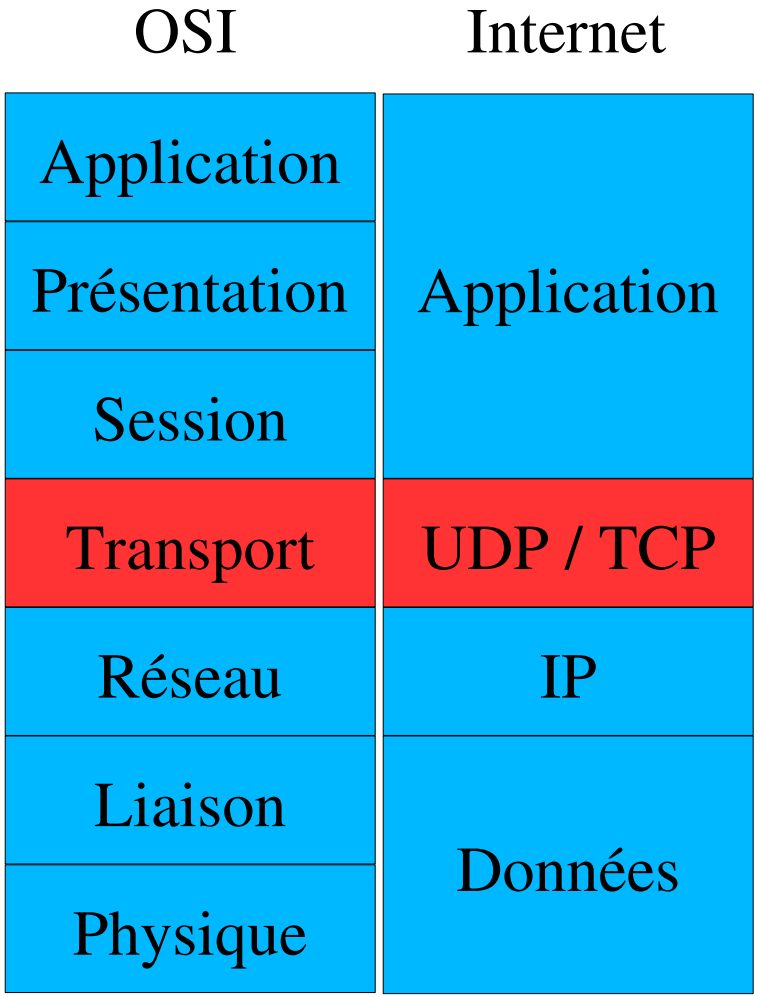
Plan

- Contexte
- TCP
  - présentation
  - définition des termes
  - contrôles de congestion
- Qualité de service : ECN
- Support des systèmes d'exploitation

Contexte : modèle OSI

- OSI : Open Systems Interconnection, 1982
- Divisé en couches

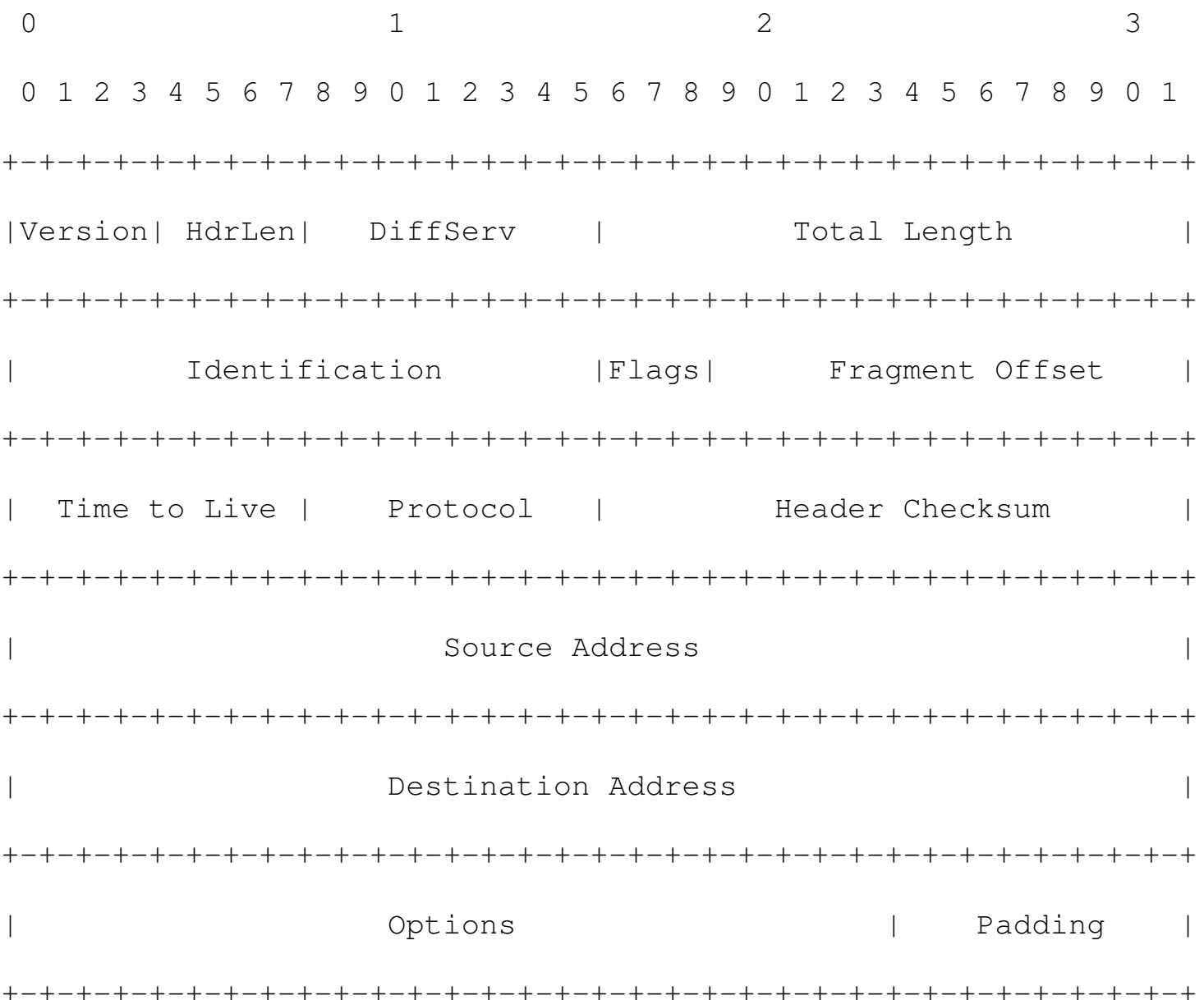
- Transfert de données avec CRC
  - end-to-end
- UDP : 0 % de fiabilité
- TCP : 100 % de fiabilité



Couche réseau : protocole IP

- Couche réseau
- Assure le transfert des données entre deux machines sur des réseaux différents
- Routage

En-tête IP


0 1 2 3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
Version
HdrLen
DiffServ
Total Length
Identification
Flags
Fragment Offset
Time to Live
Protocol
Header Checksum
Source Address
Destination Address
Options
Padding

Historique du CC : “passé”

- 1980 : UDP, sans CC
- 1988 : TCP Tahoe, 1er CC
- 1990 : TCP Reno, se remet plus rapidement lors d'une perte
- 1994 : TCP Vegas, basé sur l'historique
  - bcp de machines utilisent TCP Tahoe/Reno, et il est plus faible en concurrence avec TCP utilisé => pas trop utilisé
- 1994 : ECN, informations sur la congestion
- 1996 : TCP SACK, gère mieux une perte suivie de non-pertes
- 1999 : TCP NewReno, se remet mieux lors de plusieurs pertes consecutives

Historique du CC : “futur”

- 2000 : TFRC, basé équation
- 2001 : TCP Westwood+, presqu'égal avec TCP, mais adapté aux réseaux sans fil
- 2004 : TIBET, High-speed TCP, DCCP, ...

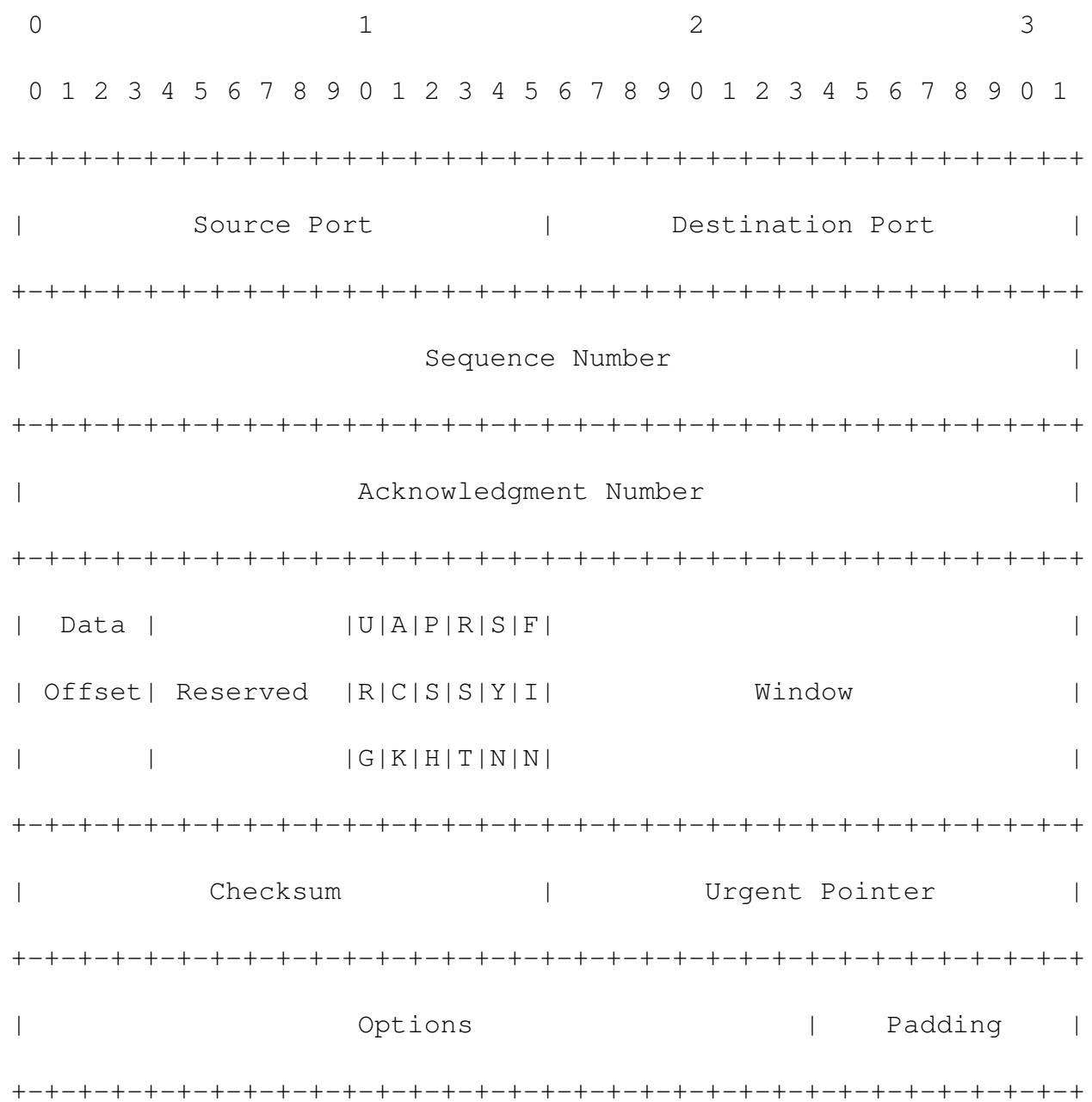
Contexte : UDP

- UDP = User Datagram Protocol
- Envoi de paquets :
  - à tout moment
  - sans confirmation de réception => fiabilité 0 %
- Utile pour des services simples ou des données vidéo etc.
  - NFS, ...
- CBR = Constant Bit Rate
  - envoi régulier de paquets

TCP

- TCP = Transmission Control Protocol
- Fiabilité 100 %
- Ordonnancement
- Contrôle de flux
- Contrôle de congestion basé fenêtre
- Orienté connexion, full-duplex
- Orienté bit : TCP divise les données en paquets
- ...

En-tête TCP


0 1 2 3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
Source Port
Destination Port
Sequence Number
Acknowledgment Number
Data Offset
Reserved
U|A|P|R|S|F|
R|C|S|S|Y|I|
G|K|H|T|N|N|
Window
Checksum
Urgent Pointer
Options
Padding

Principe “end-to-end”

- Un hôte est impliqué dans 1 transfert, un routeur dans beaucoup de transferts
- Les deux hôtes d'extrémité sont reponsables du débit de transfert des données :
  - à quel vitesse transmettre
  - quand transmettre
  - quand accélérer et décélérer le débit
- Le réseau ne leur fournit pas des informations explicites

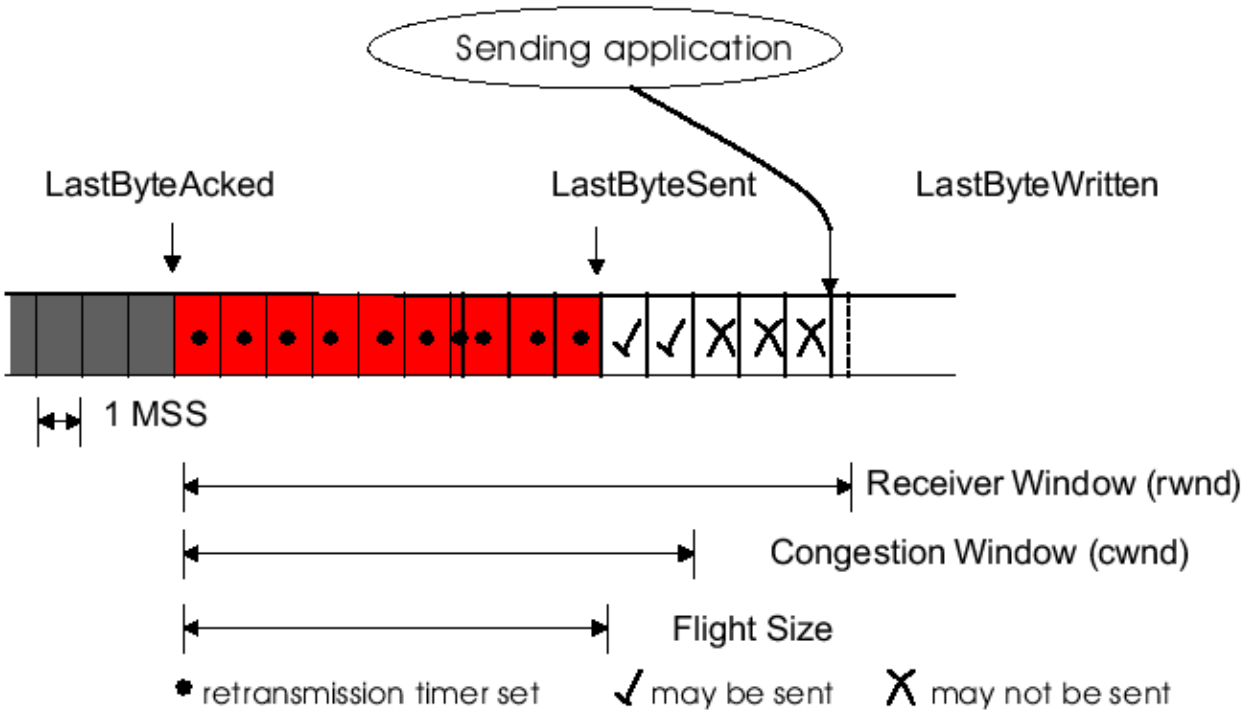
TCP : définitions

- Contrôle de flux : par rapport au récepteur
  - l'émetteur adapte le nombre de paquets envoyés à la taille du buffer de réception
- Contrôle de congestion : par rapport au réseau
  - l'émetteur adapte le débit des données envoyées à la bande passante instantanée du réseau
  - NB : ce n'est pas la taille des paquets, mais leur débit d'envoi qui change

TCP : définitions : fenêtres

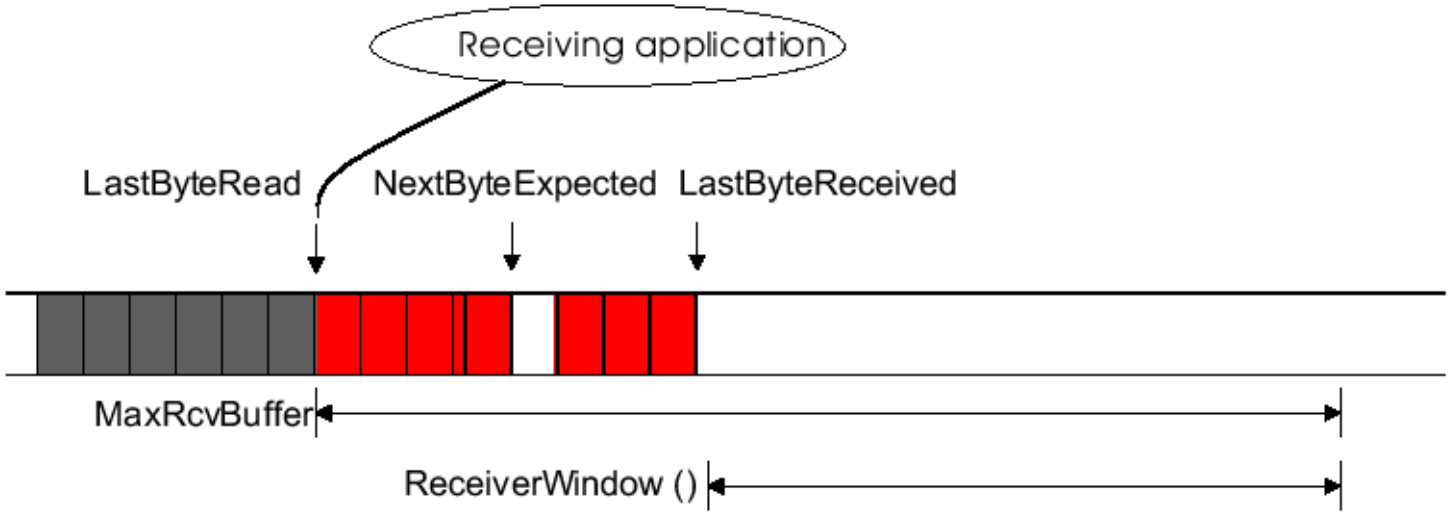
- Buffer de réception : espace de stockage des données (reçues ou non)
- Fenêtre de réception
  - nombre max de paquets que le récepteur est capable de recevoir à un certain moment (l'espace libre au récepteur)
- Fenêtre d’émission : les données de l'application
- Fenêtre de congestion (cwnd)
  - sous-fenêtre mobile de la fenêtre d'émission
  - nombre max de paquets que l'émetteur peut envoyer sans recevoir aucun accusé
- Seuil de démarrage lent (ssthresh)
  - estimation de la bande passante disponible

Fenêtres de l'émetteur



source

Fenêtres du récepteur



source

TCP : définitions : accusés de réception

- Accusé de réception
  - récepteur -> émetteur
  - numéro du 1er octet attendu par le récepteur
- TCP classique : les accusés sont cumulatifs
- DupACK : un accusé identique au précédent
  - si paquet N arrive au récepteur avant N-1, son accusé est identique à l'accusé de N-2
- Delayed ACK : retarder les accusés
  - après min (le 2ème paquet reçu, un temps fixe, eg 500 ms) (sauf cas particuliers, voir TP) [RFC 2581]

TCP : définitions : horloges

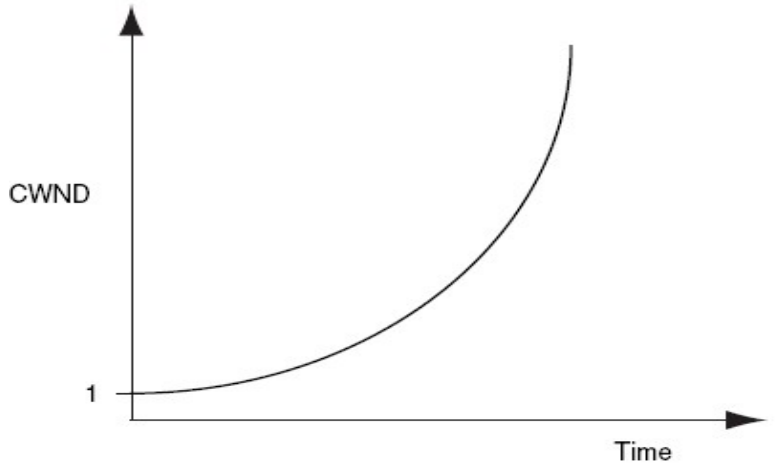
- RTT (Round Trip Time)
  - temps entre l’envoi d’un paquet et la réception de son accusé
- RTO (Retransmission Timeout)
  - à chaque envoi d'un paquet, une horloge propre est lancée
  - si l'horloge expire, le paquet est retransmis
    - le RTO est affecté dynamiquement, en fonction du RTT [RFC 2988]
    - exemple : RTO = 2*RTT

TCP : mécanismes de CC

- En gén., la perte d'un paquet est la seule information sur l'état du réseau
- Le CC est géré exclusivement par l'émetteur
  - le récepteur ne fait que renvoyer des accusés de réception
- Les algorithmes basiques de CC supportés par TCP sont [RFC 2581] :
  - slow start
  - congestion avoidance
  - fast retransmission
  - fast recovery
- cwnd++ (CA) ou /2 (si perte) => mécanisme AIMD (Additive Increase, Multiplicative Decrease)

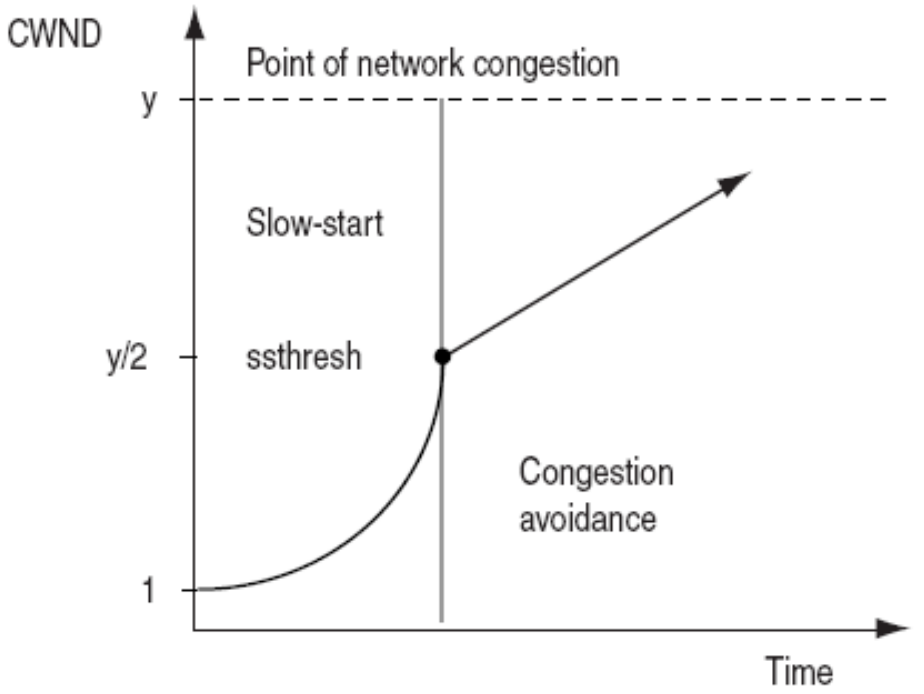
TCP : slow start

- But : retrouver rapidement la bande passante disponible
- cwnd = 1
- cwnd++ à chaque accusé reçu (cwnd *= 2 à chaque RTT)
  - (croissance exponentielle)
- ssthresh = valeur arbitraire
- Si atteinte ssthresh :
  - on entre en congestion avoidance
- Si perte :
  - ssthresh = cwnd / 2
  - cwnd = 1
  - on relance le slow start



TCP : congestion avoidance

- Utilisé quand cwnd >= ssthresh
  - quand cwnd < ssthresh, c'est le slow start qui est utilisé
- But : augmenter le débit en testant gentillement la bande passante disponible
- cwnd++ à chaque RTT
  - (croissance linéaire)
- Si perte :
  - ssthresh = cwnd / 2
  - cwnd = 1
  - retour au mode slow start

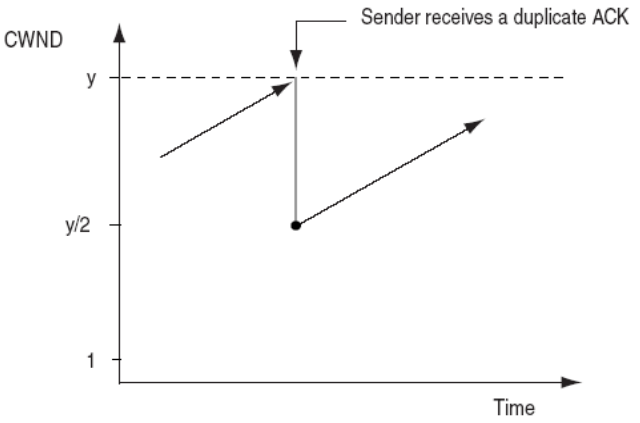


TCP : fast retransmission

- But : détecter plus rapidement la perte d'un paquet (et le retransmettre)
- Un paquet est considéré par l'émetteur comme perdu si :
  - pas d'accusé au timeout (=> pertes successives), déjà traité
  - ou bien réception de N dupacks, N = 3 en gén. (=> perte isolée)
- Fast retransmission : si N dupacks, on n'attend plus le timeout, mais :
  - on retransmet le paquet
  - on entre en slow start (Tahoe) ou fast recovery (les autres)

TCP : fast recovery

- ssthresh = cwnd / 2
- cwnd = ssthresh + 3 (“gonflement” de cwnd)
  - => envoi éventuel de nouveaux paquets
  - 3, car 3 paquets accusés
- Pour chaque dupack, cwnd++
  - => envoi éventuel d'un nouveau paquet
- Réception d'un non dupack (“dégonflement” de cwnd) :
  - cwnd = ssthresh
  - retour au congestion avoidance



TCP : algorithmes de CC

- Tahoe : slow start + congestion avoidance + fast retrans
- Reno : Tahoe + fast recovery
- Newreno : Reno + adaptation aux pertes successives
- Vegas : basé sur l'historique du RTT (état des routeurs)
- SACK, DSACK : spécifie exactement les paquets reçus
- Westwood+ : basé sur l'historique du RTT, meilleure utilisation si pertes aléatoires
- ABC : des octets à la place de paquets
- Beaucoup d'autres...

TCP : Tahoe

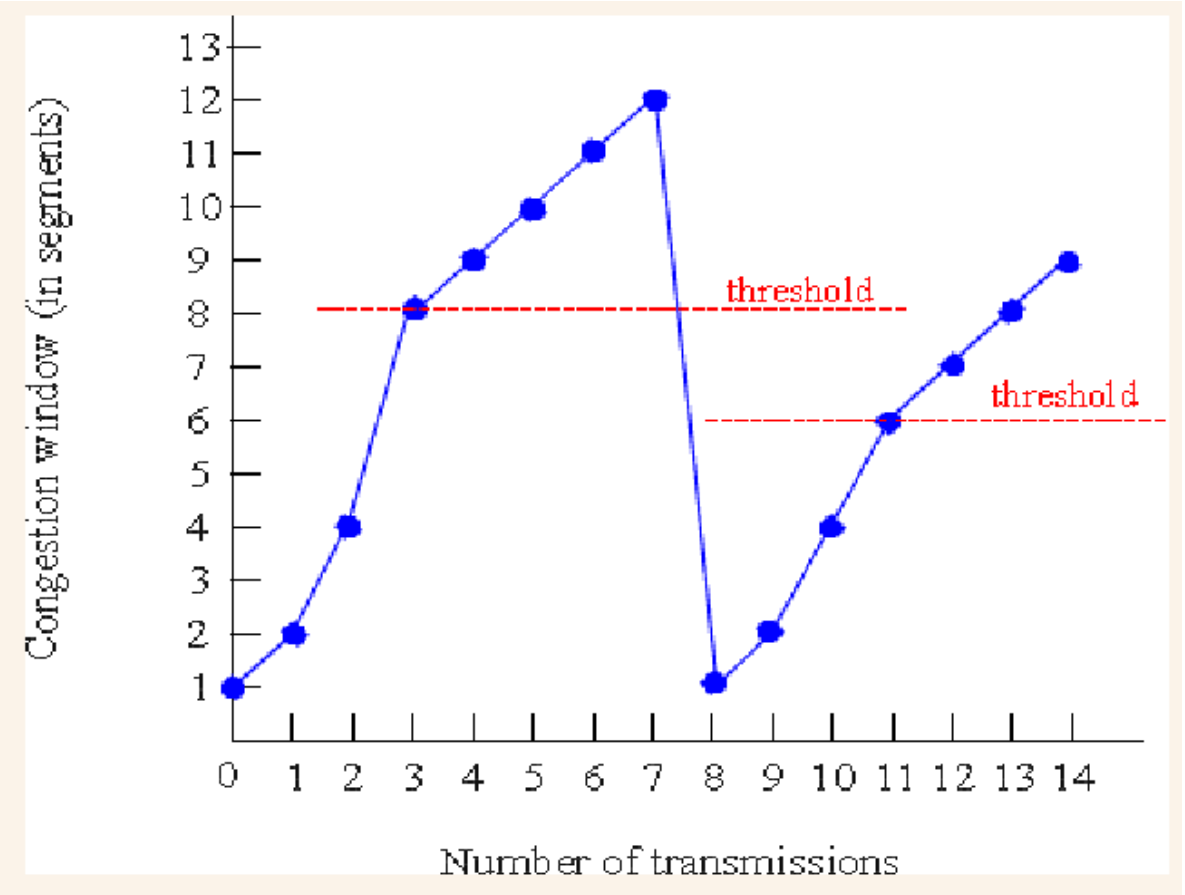
- Perte <=> timeout ou 3 dupacks
- Utilise :
  - slow start
  - congestion avoidance
  - fast retransmission

Initialisation :

cwnd <-- 1 (MSS);

ssthresh <-- Init_Ssthresh

State <-- Slow Start;



TCP : Tahoe : algorithme

ACK received in Slow Start:

    cwnd <-- cwnd +1 (MSS); /* "exponential" increase of cwnd */

    If cwnd > ssthresh Then

        State <-- Congestion Avoidance;

ACK received in Congestion Avoidance:

    If (#ack received = cwnd) Then

        cwnd <-- cwnd +1 (MSS); /* "linear" increase of cwnd */

Timeout expiration OR 3rd DupACK received :

    Retransmit (lost packet);

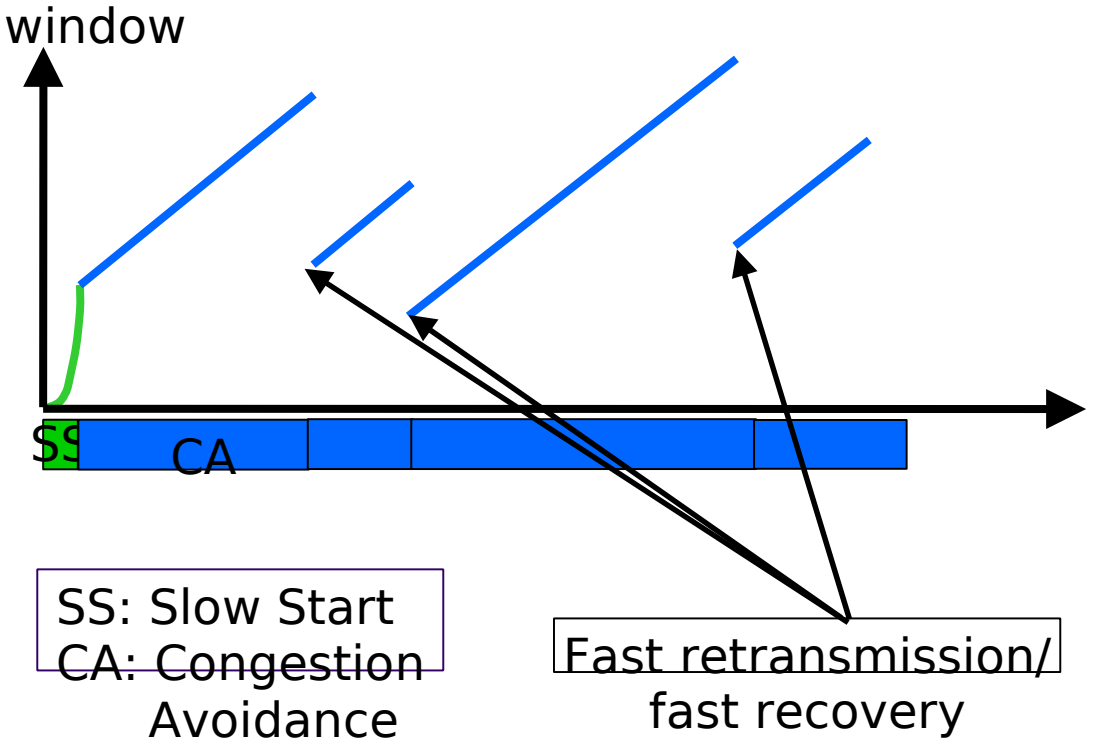
    ssthresh <-- cwnd/2; cwnd <-- 1(MSS);

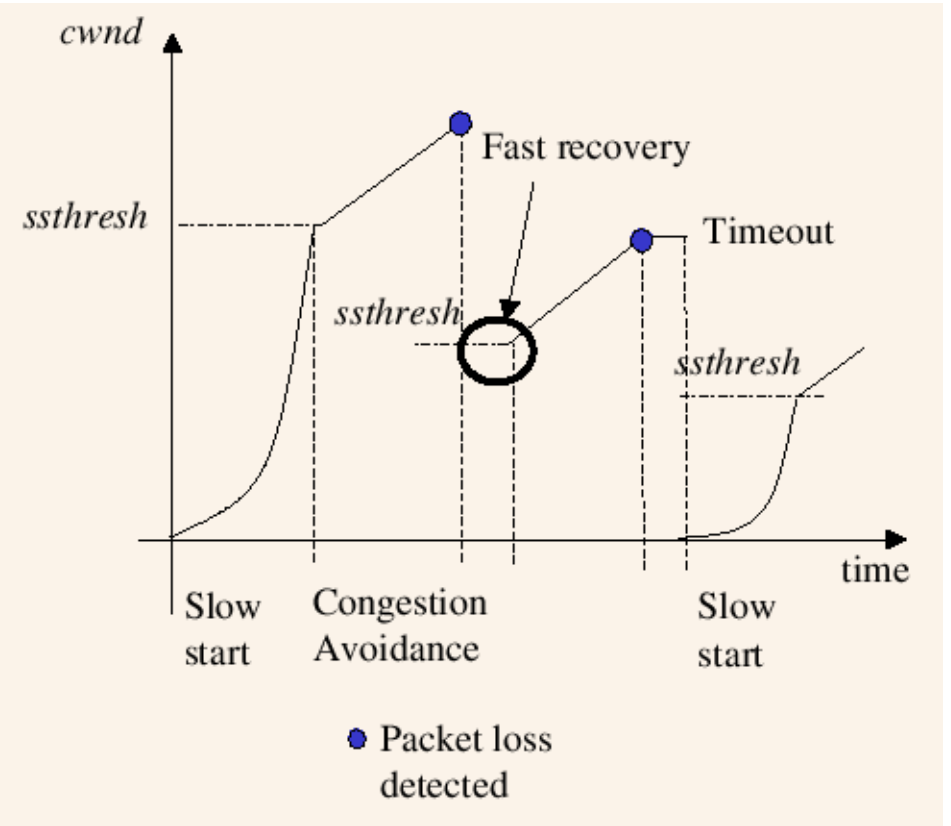
    State <-- Slow Start;

TCP : Reno

- Reno = Tahoe + fast recovery
- Plus rapide en récupération après N dupacks

source



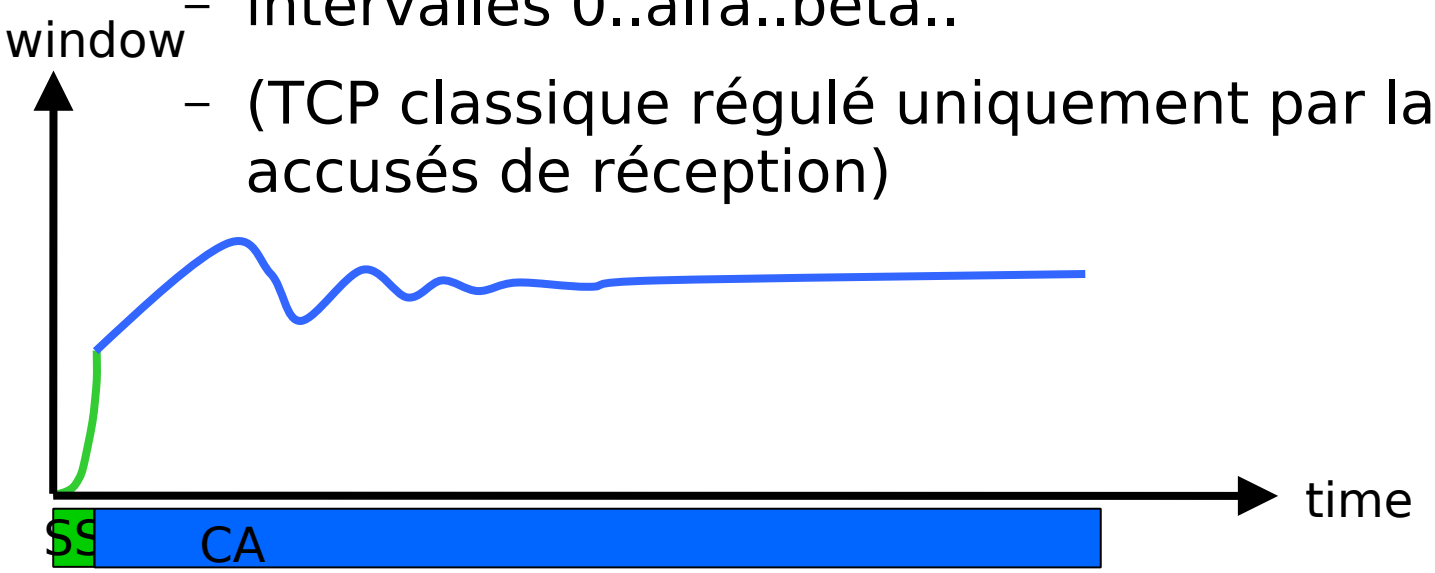


TCP : Newreno

- Reno + légère modification de fast recovery
- Meilleur si plusieurs pertes non contigus dans un même “vol” de paquets
  - Reno : cwnd peut décroître plusieurs fois dans un même RTT
- Lors de la réception d'accusés partiels (qui accusent une partie des paquets envoyés), envoie le paquet suivant perdu tout de suite (les paquets ?)
- Reste en fast recovery jusqu'à la réception des accusés de **tous** les paquets perdus dans le “vol” de paquets initial
- => Meilleur, car il coupe cwnd seulement après la 1ère perte

TCP : Vegas

- But : réduire le débit **avant** qu'une perte apparaisse
- L'historique du RTT donne des informations sur l'évaluation de la taille des buffers des routeurs
- cwnd régulé par le temps d'arrivée des accusés et par le RTT courant vs. moyen
  - intervalles 0..alfa..bêta..
  - (TCP classique régulé uniquement par la réception des accusés de réception)



TCP : SACK

- “Selective ACKs”
- Implémenté comme option TCP
  - négocié lors de l'initiation de la connexion
- Récepteur précise les paquets reçus :
  - début et fin de chaque bloc de segments contigus reçus
- Connaissance du numéro du ou des paquets à **ne pas** retransmettre
- Implémentable dans Reno par ex.
- DSACK (Duplicate SACK) : paquet reçu plusieurs fois

TCP : timestamps

- RFC 1323
- Permet, entre autres, d'obtenir avec précision le RTT d'un paquet
  - utile lors de retransmissions (à quel paquet se réfère un ack ?)
  - sur plusieurs paquets, permet de savoir si la congestion est sur le chemin ascendant ou descendant
- Option de TCP, 12 octets
- Émetteur et récepteur :
  - insèrent le timestamp dans chaque paquet au moment de l'envoi
  - renvoient le timestamp de l'autre extrémité

TCP : ABC

- ABC : Appropriate Byte Counting (1999, 2003, rfc 3465)
- En SS ou CA : augmentation à chaque accusé reçu
- Mais si un accusé est perdu et le prochain accusé accuse deux paquets, l'augmentation est faite une seule fois !!
- ABC : utiliser le nombre d'octets accusés au lieu du nombre d'accusés reçus

Concurrence des flux

- Les flux TCP sont équitables : chacun des N flux prend 1/N de la bande passante
  - en réalité, la fraction est proportionnelle au RTT du flux :
    - débit ~ MSS / (RTT * p), p = probabilité d'erreur
- Les flux UDP ne sont pas équitables
  - TCP est défavorisé par rapport à UDP
- Évolution de la vitesse de transfert en concurrence de flux

Réseaux sans fil : caractéristiques

- Filaire :
  - 99.9... % : perte due à une congestion d'un routeur
  - le reste : perte due à un problème matériel
- Non filaire :
  - beaucoup d'interférences, donc des pertes
  - lors d'une perte, TCP considère qu'il s'agit d'une congestion, donc il réduit le débit, le contraire de ce qu'il doit faire

TCP : Westwood+

- Mémorise les RTTs
  - estime la bande passante disponible
- Lors d'une perte, se remet à la valeur précédente de cwnd
  - (TCP classique réduisait par 2 le cwnd)
- Mieux adapté que les autres aux réseaux sans-fil

Réseaux à grand produit bande passante * délai : caractéristiques

- Ex. : 10 Gb/s et RTT de 100ms
- L'augmentation linéaire de cwnd est trop faible et la baisse 0.5 est trop importante
- Protocoles nouveaux : HighSpeed TCP, BIC, ...
- HighSpeed TCP, constante low_window :
  - si cwnd <= low_window, TCP classique
  - si cwnd > low_window, l'augmentation est plus grande et la baisse plus petite, en fonction du nb pertes, cwnd etc.
    - 1 => f(cwnd), 0.5 => f(cwnd)

TCP : BIC

- Adapté aux réseaux à grande bande passante * délai
- Fonction d'augmentation de cwnd modifiée
- Lors d'une perte, l'actuelle cwnd devient max et la nouvelle cwnd devient min
- Recherche binaire : cwnd suivante est (min+max)/2 jusqu'à perte d'un paquet
- Si max – min trop grand, augmentation linéaire jusqu'à un certaine différence
- Etc.

Réseaux à bande passante asymétrique

- Ex. : ADSL
- TCP n'est pas adapté

Qualité de service : routeurs

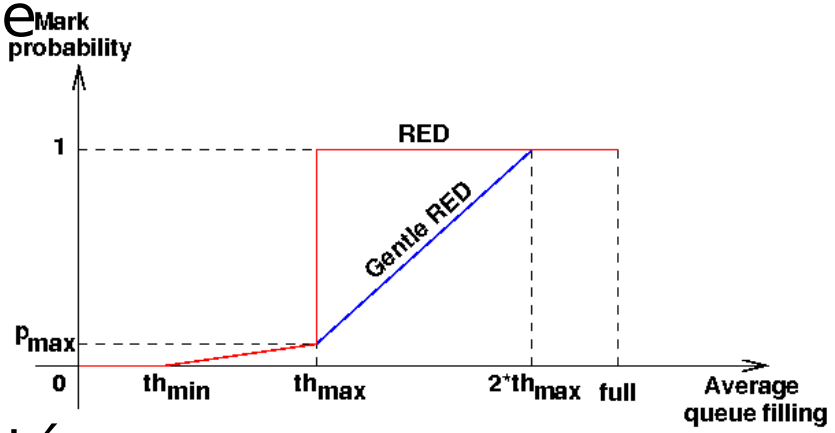
- Reçoit des paquets par toutes ses interfaces
- Envoie les paquets reçus sur la bonne interface
  - table de routage
- Pb. : la file d'attente peut se remplir

Qualité de service : gestion des files d'attente des routeurs

- “Statiques” :
  - DropTail : le paquet est rejeté ssi la file est pleine
    - très rapide
    - le plus utilisé
    - génère rafale de retransmissions
    - dès fois, le même flux est pénalisé
- AQM (Active Queue Management) : mesure régulièrement la taille de la file
  - RED : Random Early Detection

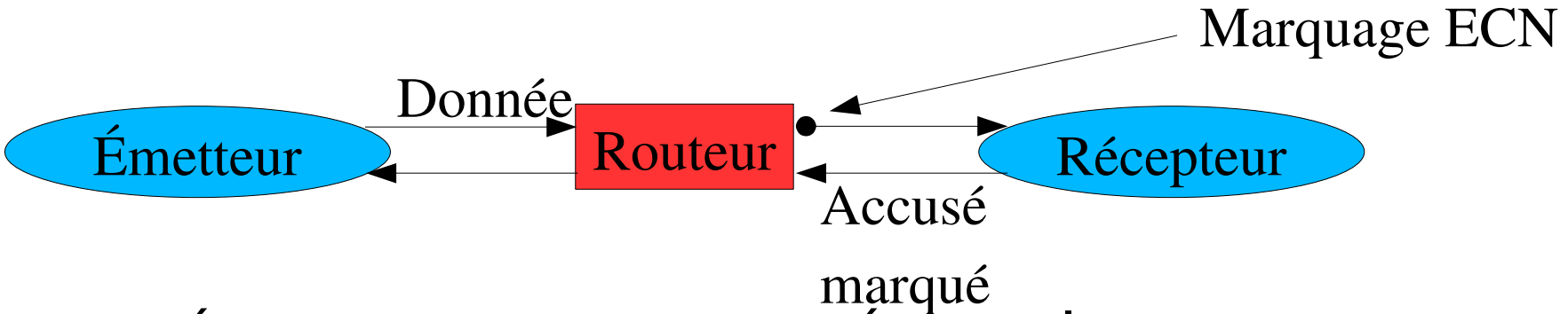
Qualité de service : algorithme RED

- But : avertir l'émetteur que la file est quasi-remplie
  - rejeter quelques paquets **avant** que la file soit pleine
  - les émetteurs réagissent en réduisant leur débit, comme si perte
- Deux seuils, th_{min} et th_{max}
  - si $0<=taille<th_{min}$, le paquet passe
  - si $th_{min}<=taille<th_{max}$, le paquet est rejeté avec une probabilité dépendante de taille
  - si $th_{max}<=taille$, le paquet est rejeté



Qualité de service : ECN

- ECN : Explicit Congestion Notification
- But : avertir l'émetteur **sans** perte de paquet



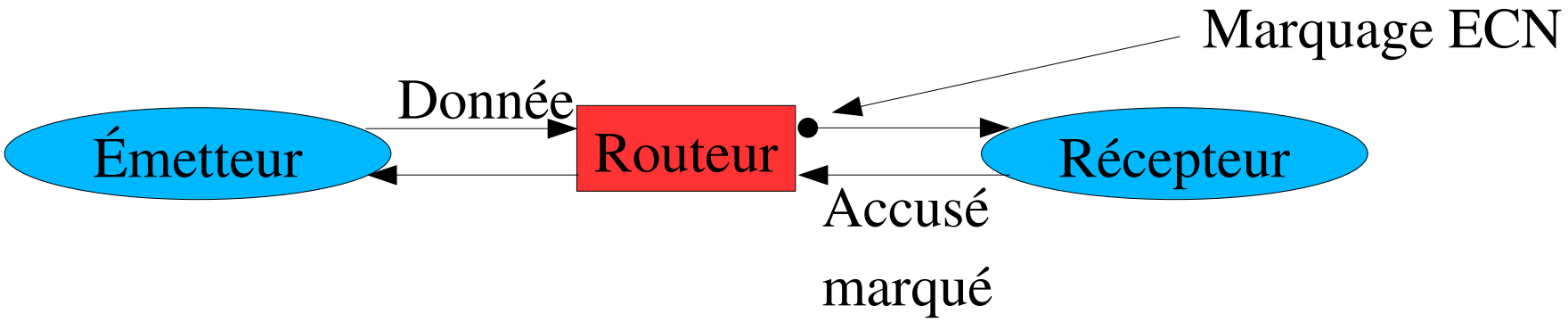
- Les accusés ne sont pas marqués par les routeurs
- Se met au-dessus de RED (par ex.), en changeant le rejet par le marquage (rfc 3168, page 9, par. 3 et 4)

Qualité de service : ECN : marquage

- À l'initiation du transfert : bit ECE (ECN Echo) du champ Code bits de TCP (avant le bit URG)
  - chaque machine spécifie si elle est capable ECN
- Pendant le transfert : le routeur peut modifier les 2 bits LSB du champ DiffServ d'IP
  - 00 : Non ECT (ECN Capable Transport), paquet non ECN
  - 01, 10 : ECT : paquet ECN, pas encore marqué
  - 11 : CE (Congestion Experienced), paquet ECN marqué

Qualité de service : ECN : marquage

- Pendant le transfert : 2 bits du champ Code bits (avant le bit URG) de TCP
  - le destinataire envoie ECE
    - envoie à chaque accusé jusqu'à réception de CWR
  - l'émetteur répond par CWR (Congestion Window Reduced)



Qualité de service : ECN

- Pour que ECN soit utilisable lors d’une transmission, il doit être utilisé et compris par :
  - les deux machines hôtes
  - les machines intermédiaires (routeurs, firewalls, …)
  - à ce jour, très peu de routeurs ne sont pas compatibles ECN
- ECN n'est pas utilisé en UDP

Support des systèmes d'exploitation : linux

- /proc/sys/net/ipv4/
  - tcp_ecn
  - tcp_congestion_control : bic, reno, highspeed, vegas, westwood etc.
  - tcp_timestamps
  - tcp_sack, tcp_dsack
  - plus d'info : *man tcp* et *man ip*, section *Sysctls*
- /etc/rc.local
  - echo 1 >/prc/sys/net/ipv4/tcp_ecn
- Code source linux de TCP : http://lxr.linux.no/source/net/ipv4/tcp_input.c?v=2.6.22, fonction tcp_acktcp_cong_avoid par exemple

Simulation

- ns2, opnet

Conclusions

- TCP adapte le débit des données envoyées à la bande passante disponible
- Les flux TCP sont équitables
- Plusieurs algorithmes de CC en TCP existent
  - Newreno avec SACK, les plus utilisés
- Quatre phases lors d'un transfert TCP :
  - slow start
  - congestion avoidance
  - fast recovery
  - fast retransmit

Conclusions

- TCP n'est pas adapté à tous les types de réseau
- Routeurs DropTail et RED
- ECN permet d'avertir l'émetteur du début d'une congestion sans perte de paquet