Sys1 - Amphi 10

Protocoles de Transport suite et Fin Protocole IP

Guillaume DIDIER - 21/11/2023

Where are we ?

Planning du cours

Date	Amphi	Pratique (TD/TP/Projet)	Références
17/10/2023	Introduction, Socket TCP	Pas de TP (Conf. Fields)	Kurose 1.1, 1.5, 1.7, 2.7.2 CS:APP 11.1, 11.2, 11.4
07/11/2023	Pas de CM (Parrainage)	Projet (binôme): Crawler HTTP & Dice Roll	
14/11/2023	Protocoles Applicatifs over TCP, UDP, e.g HTTP, DNS	Projet (binôme): Crawler HTTP & Dice Roll	Kurose 2.1, 2.2, 2.4, 2.7 CS:APP 11.3, 11.5, 11.6,
17/11/2023	Protocoles de Transport	Pas de TP (Rattrapage CM)	Kurose Ch 3, CS:APP 11.3
21/11/2023	**TCP suite et fin, protocole IP,**	**Projet (binôme): Crawler HTTP & Dice Roll**	**Kurose Ch 3, 4.1, 4.3**
28/11/2023	notion de Lien, Forwarding + Routage	TP (seul): Reliable Data Transport	Kurose 4.1, 4.3, 5.1, 5.2, 6.1
01/12/2023	Pas de CM (Rattrapage TP)	TP (seul): Reliable Data Transport	
05/12/2023	Routage (un peu plus de lien)	TD : IP + Routage	Kurose 5.1, 5.2, 5.4 (6)
12/12/2023	Conclusion : Les défis d'internet	TD : Révisions	Divers sections du Kurose

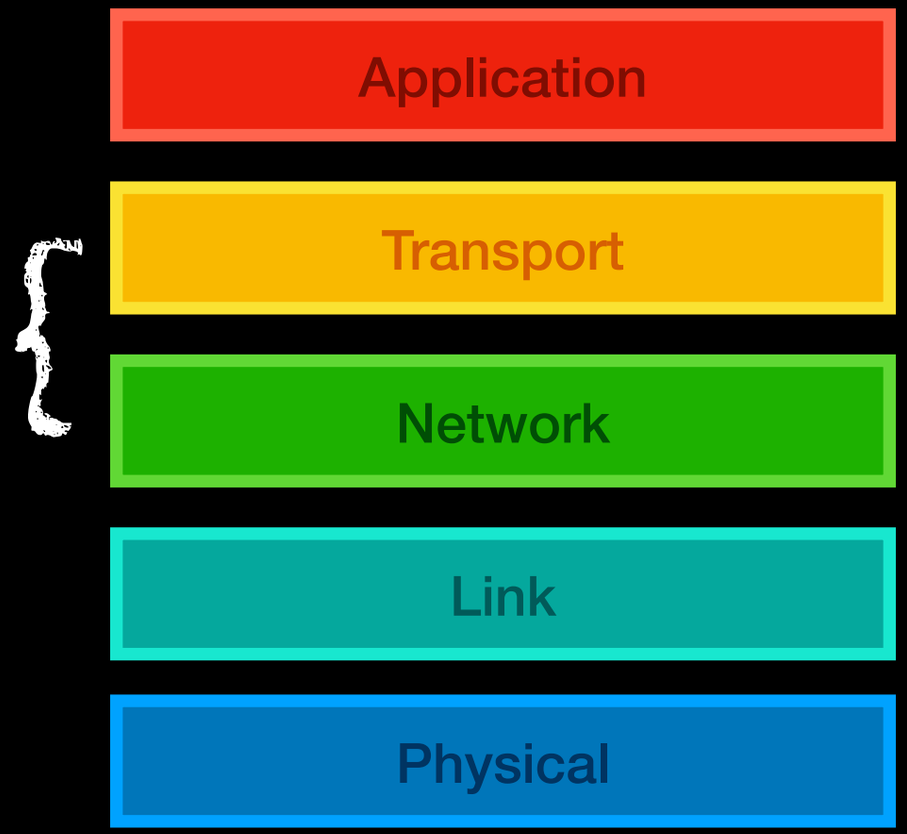
Programme du jour

- TCP Suite et Fin
  - Rappel de ou on s’était arrêté
  - Déterminer le bon timeout - estimer le RTT
  - La Congestion
  - Comment gérer la congestion
- La couche IP
  - Plan de donnée vs Plan de Contrôle
  - Fonctionnement de la transmission (Forwarding)

Rappel: Les couches réseau

Ou sommes nous aujourd’hui ?

Aujourd’hui



On en étions nous

TCP

✓ Handshake à 3 packets

✓ ACK avec les données dans l’autre sens

✓ Fast retransmit (triple duplicate ACK)

✓ Fermeture de connection

✓ Flow Control

✓ Taille max de paquet (notée **MSS**)

Questions encore ouverte :

➡ Valeur de timeout ?

➡ Ne pas saturer le réseau ?

Estimer le RTT

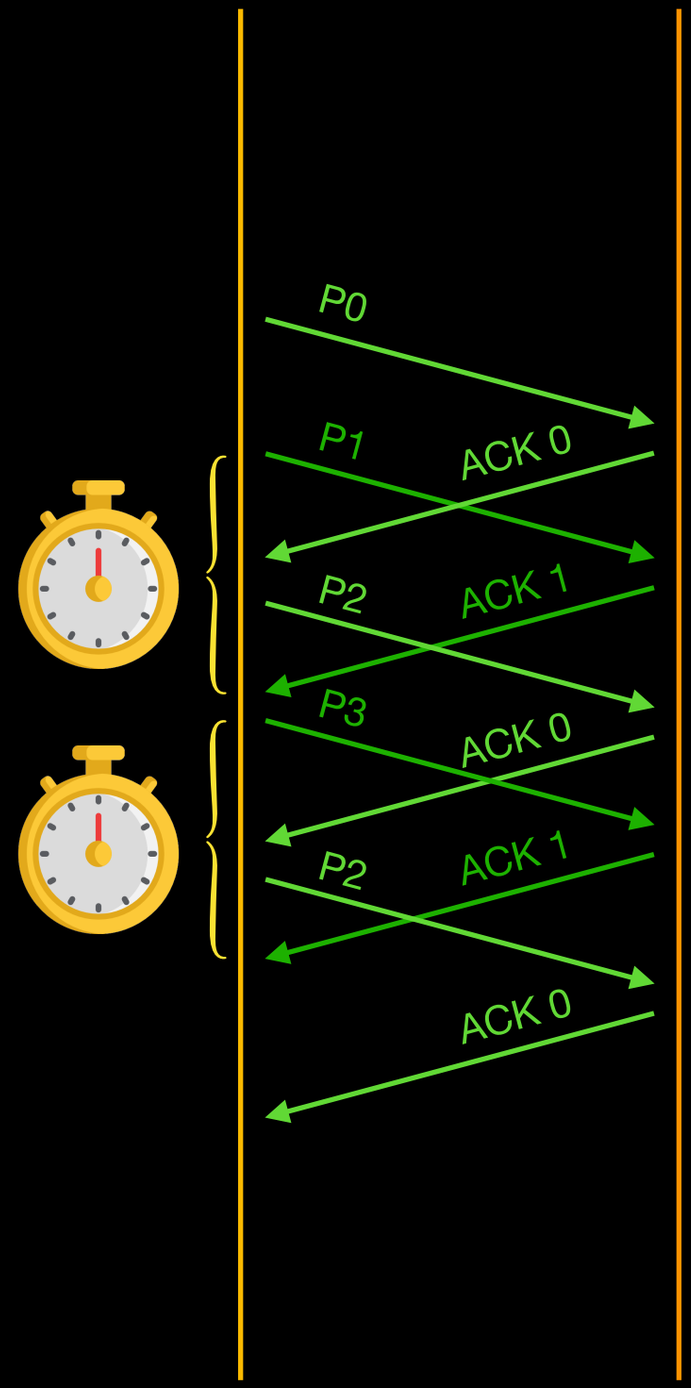
Comment bien dimensionner les timeouts ?

Mesurer le RTT des paquets

SampleRTT

On mesure le temps entre l’envoi d’un paquet et son ACK :

- Un sample à la fois (Pourquoi ?)
- Uniquement sur un paquet non retransmis (Pourquoi ?)



Statistiques glissantes

Moyenne et déviation standard approchée

$$\text{EstimatedRTT} = (1 - \alpha) \cdot \text{EstimatedRTT} + \alpha \cdot \text{SampleRTT}$$

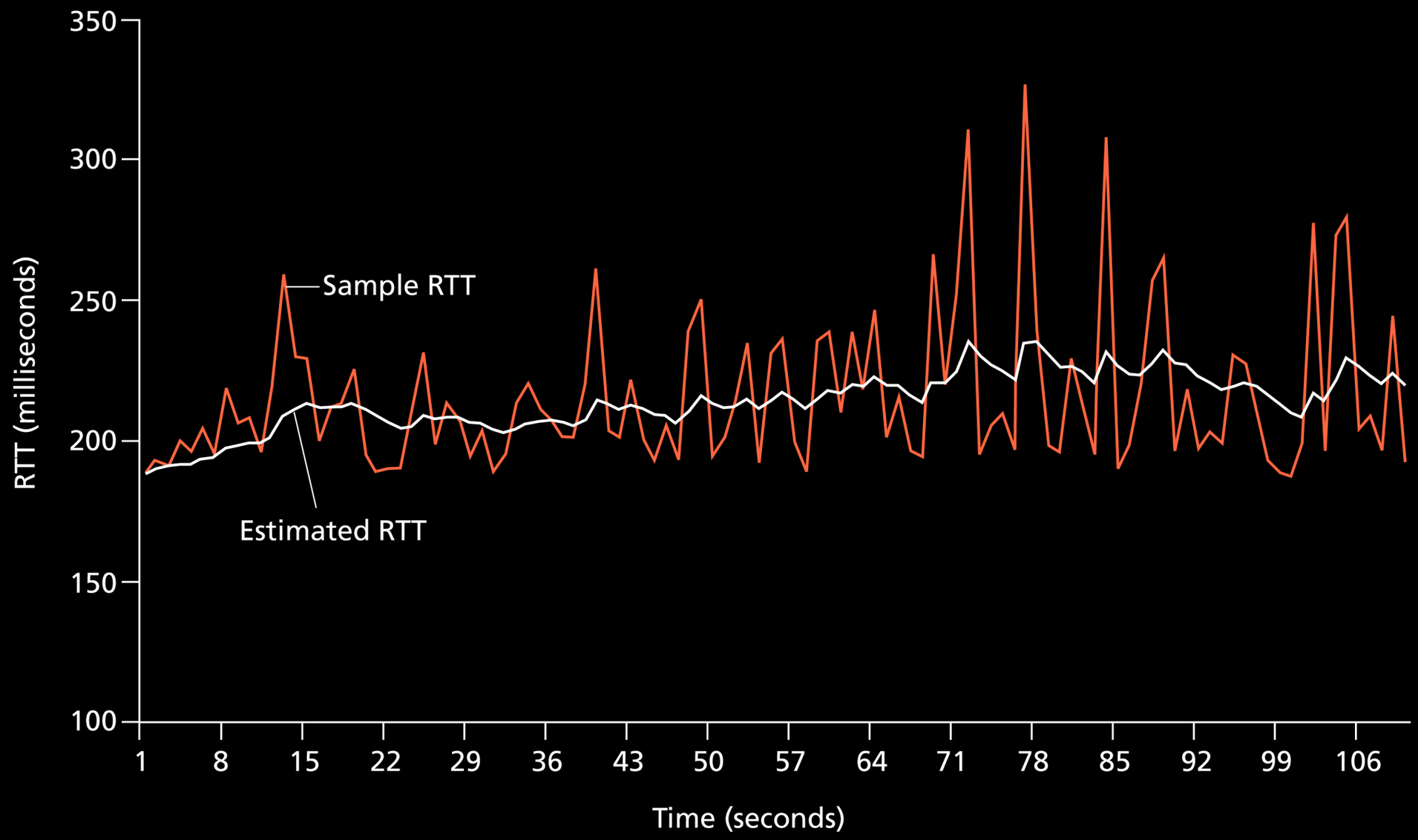
$$\text{Typically, } \alpha = \frac{1}{8}$$

$$\text{DevRTT} = (1 - \beta) \cdot \text{DevRTT} + \beta \cdot \left| \text{SampleRTT} - \text{EstimatedRTT} \right|$$

$$\text{Typically, } \beta = \frac{1}{4}$$

Aperçu du résultat

Lissage de l’aller-retour estimé



SampleRTT and EstimatedRTT between gaia.cs.umass.edu and fantasia.eurecom.fr

Source Kurose

Choix du timeout

La formule de TCP

$$\text{Timeout} = \text{EstimatedRTT} + 4 \cdot \text{DevRTT}$$

$$\text{avec} \begin{cases} \text{EstimatedRTT} = (1-\alpha) \cdot \text{EstimatedRTT} + \alpha \cdot \text{SampleRTT} & \alpha = \frac{1}{8} \\ \text{DevRTT} = (1-\beta) \cdot \text{DevRTT} + \beta \cdot \left|\text{SampleRTT} - \text{EstimatedRTT}\right| & \beta = \frac{1}{4} \end{cases}$$

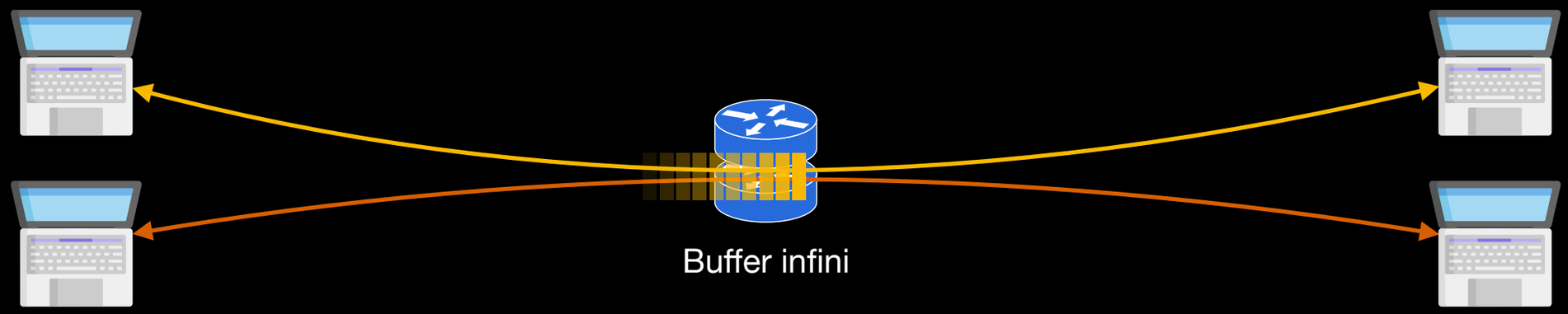
That's all!

Congestion Control

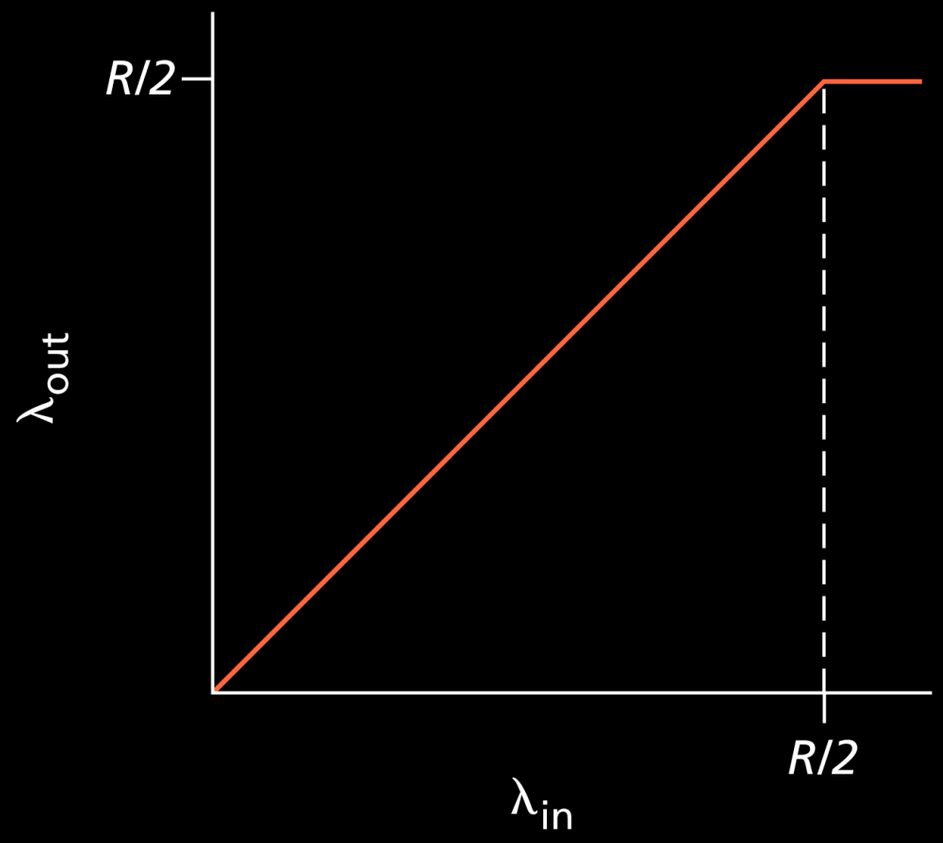
Concepts Généraux

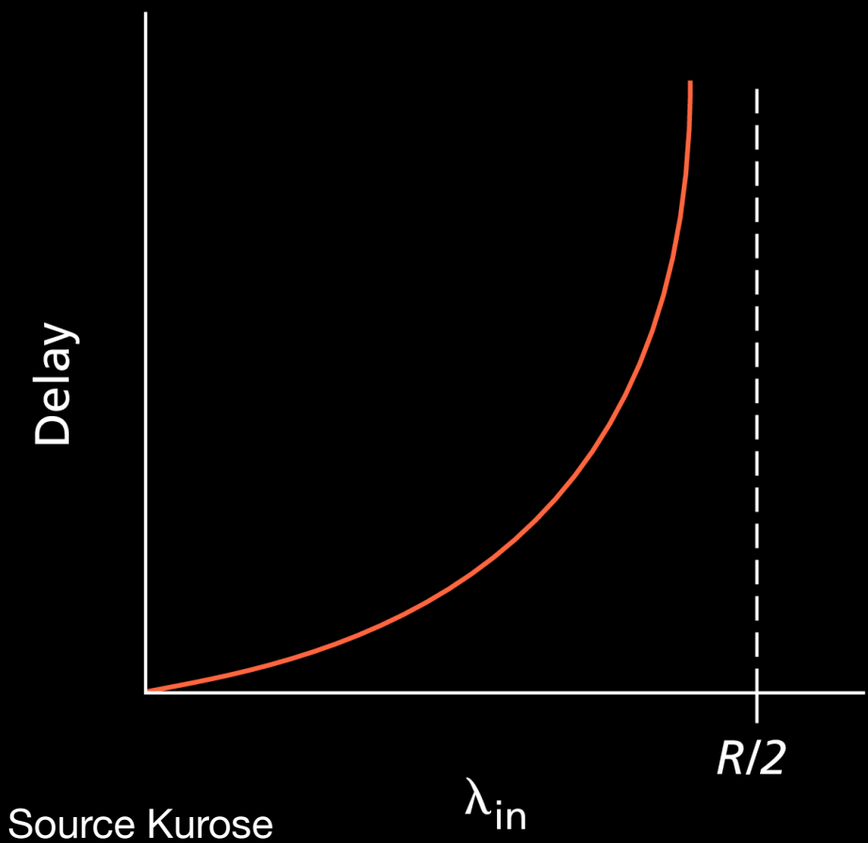
Un histoire d’horreur

Un routeur avec un buffer infini, mais un débit limité



Débit sortant et latence en fonction du débit entrant

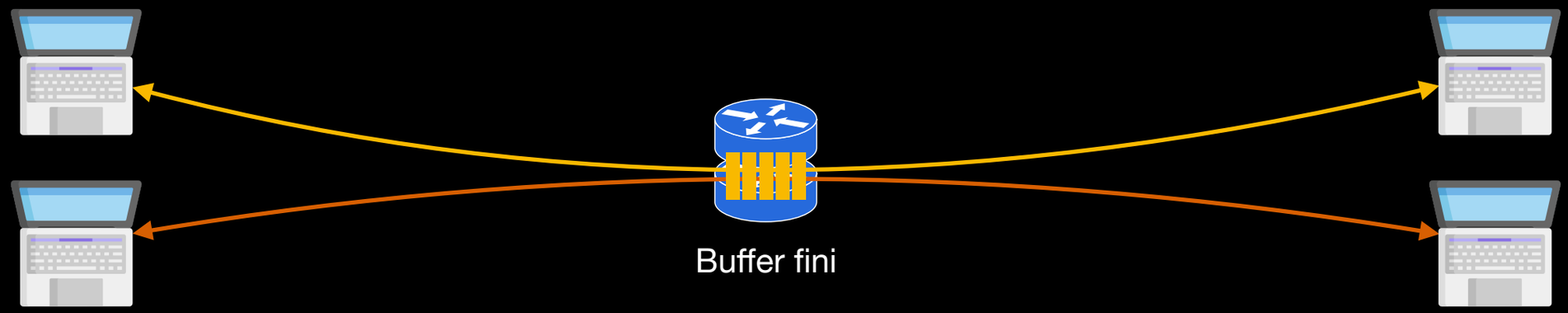




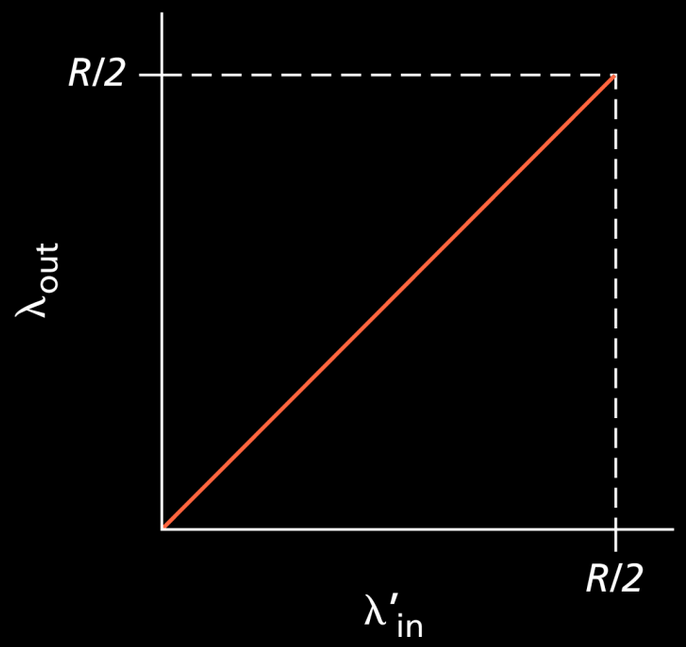
Source Kurose

Deuxième histoire d’horreur

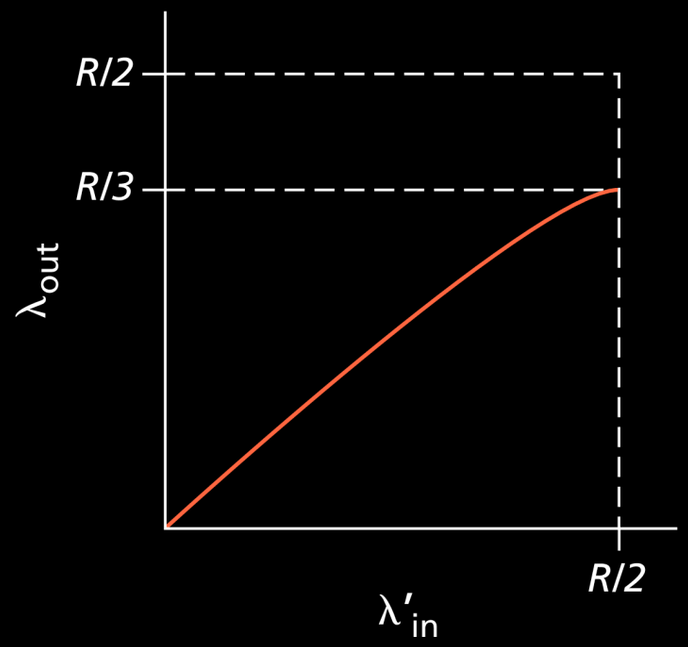
Un routeur avec un buffer fini



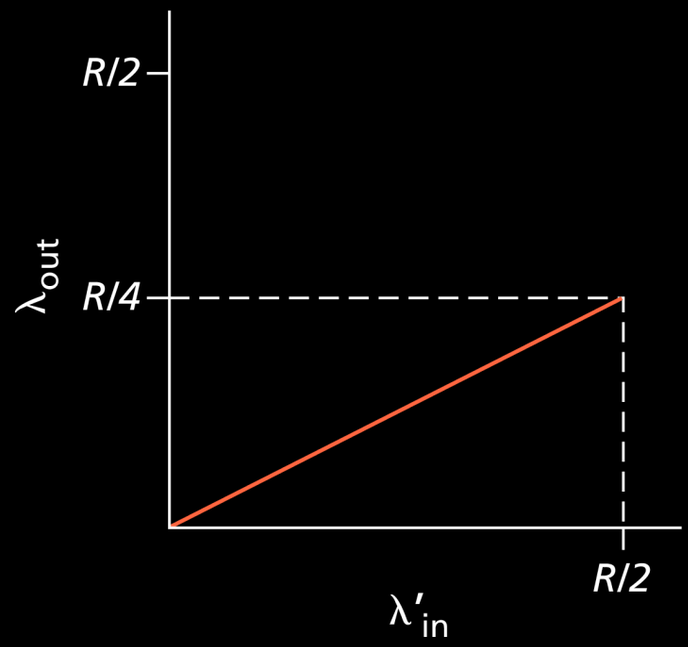
Débit application reçu en fonction du débit transport envoyé au routeur



a. L’émetteur sait si le buffer est complet



b. L’émetteur ne retransmet que des paquets perdus

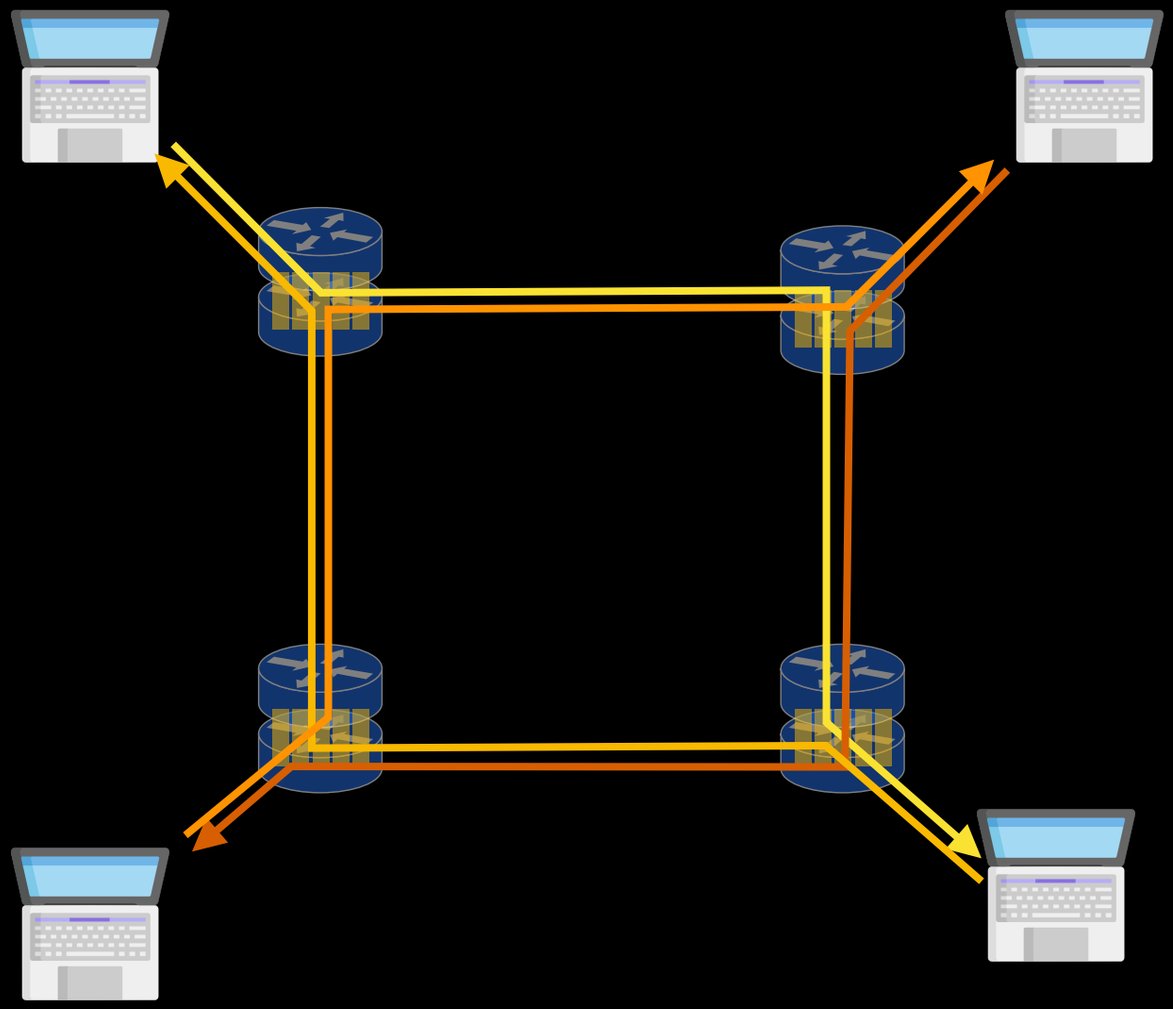


c. L’émetteur retransmet en moyenne 1 fois chaque paquet

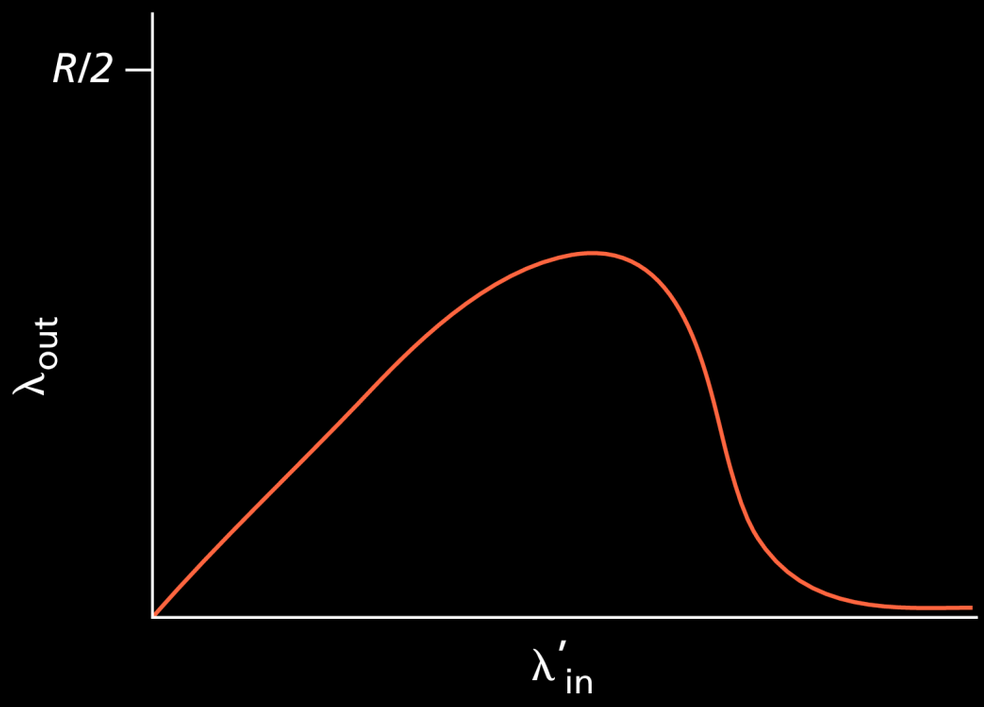
Source Kurose

Troisième histoire d’horreur

4 routeurs finis, avec 2 hops par connexion



Débit application reçu en fonction du débit transport envoyé par l’émetteur



Source Kurose

Comment éviter ça

Deux approches de gestion de la congestion

- Approche End-to-End : La couche transport détecte les problèmes seule.

Cas de TCP Classique

- Network Assisted : Le réseau informe la couche supérieur du problème.

Explicit Congestion Notification

TCP Congestion Control

Slow Start
Congestion Avoidance
Fast Recovery

Concepts de base

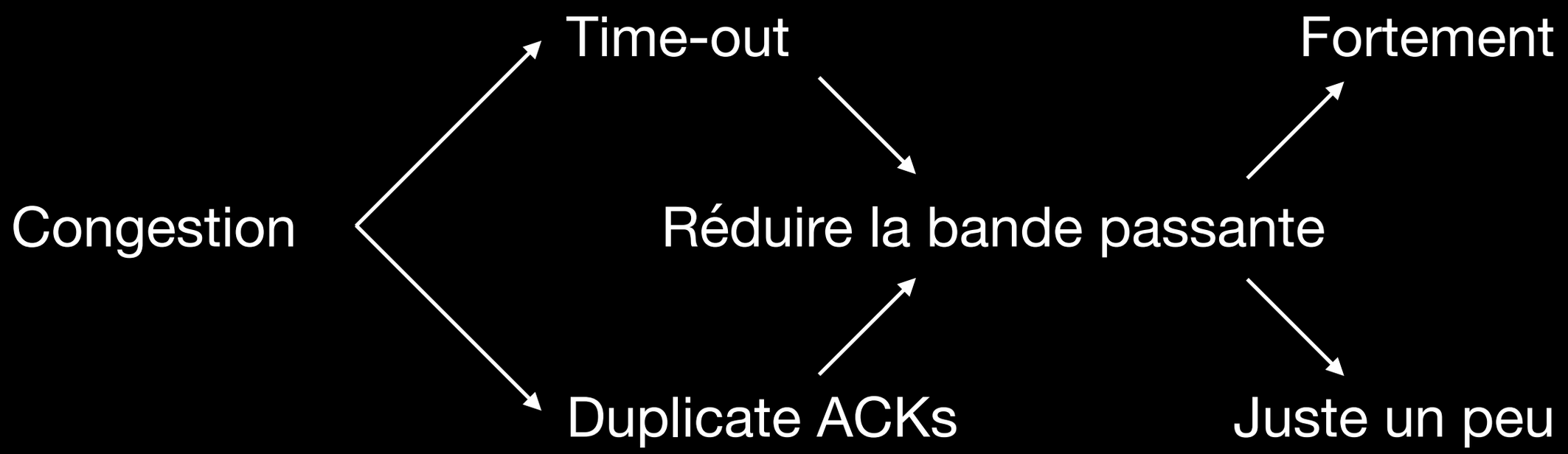
Fenêtre de Congestion

$$\text{LastByteSent} - \text{LastByteAcked} \leq \min\{\text{cwnd}, \text{rwnd}\}$$

$$\text{Débit sortant en moyenne : } \frac{\text{cwnd}}{\text{RTT}} B/s$$

Concepts de Base

Comment TCP détecte l'état de congestion du réseau

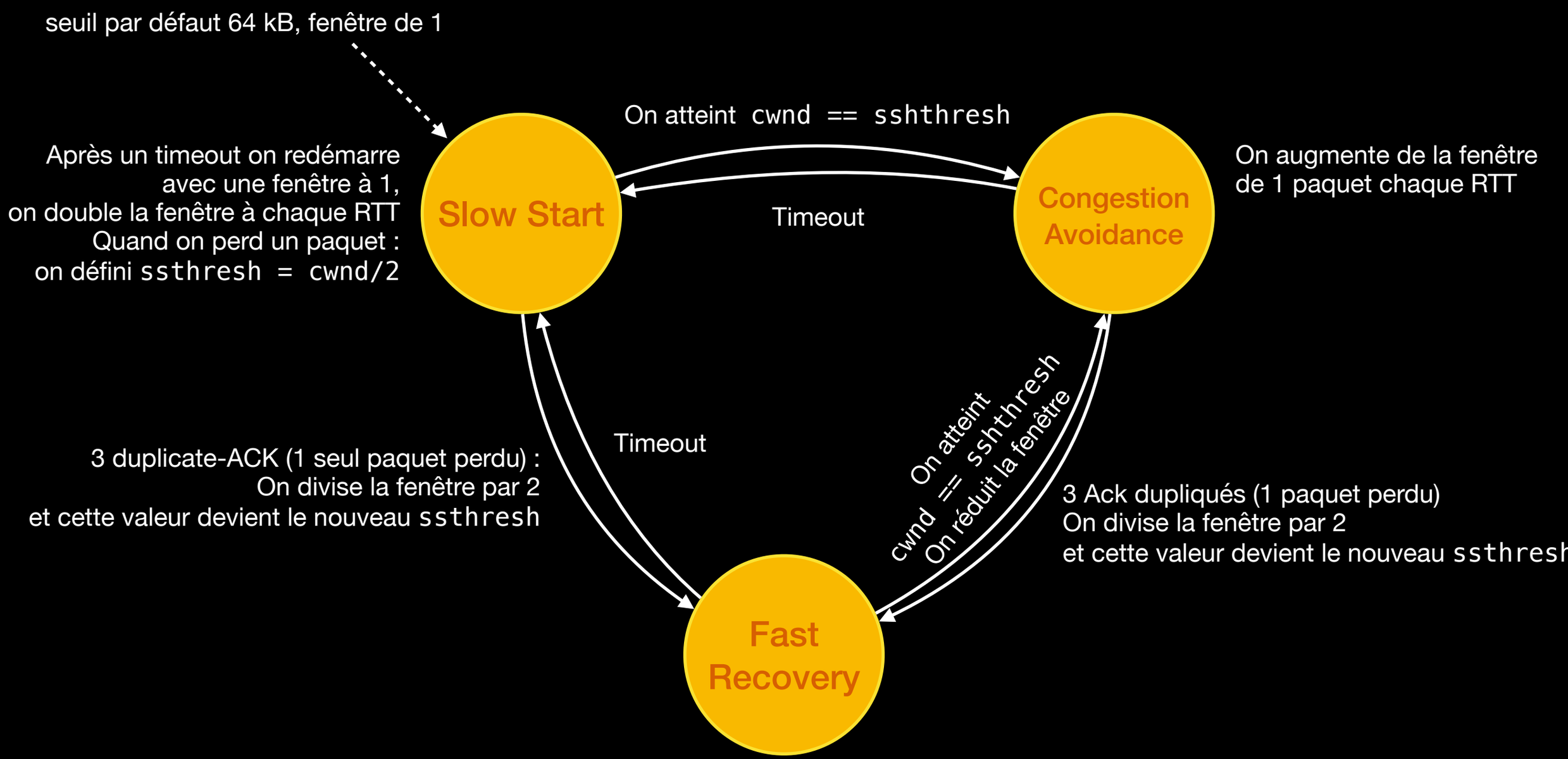


Réseau pas congestionné ⟶ Tous les paquets passent

Il y a peut-être la place d'augmenter le débit ? ⟶ Augmenter prudemment

FSM simplifiée de la congestion TCP

(As of TCP Reno)



Congestion TCP

Slow Start

- On n’a aucune idée de la bande passante dispo
- On teste en exponentielle (en pratique `cwnd += MSS` à chaque ACK reçu)
- La première fois qu’on fait ça, on a un seuil par défaut de 64 kB
- On fixe le seuil à 1/2 fenêtre lorsqu’on perd un paquet (`ssthresh = cwnd/2`)
- Si on atteint le seuil (après la première fois), on arrête et évite la congestion, Congestion Avoidance (`ssthresh == cwnd`)
- Si la perte de paquet est un timeout on recommence à `cwnd = 1`
- Si on perd juste 1 paquet (3 duplicate ACK), on fait une récupération rapide (Fast Recovery) (puis normalement on évite aussi la congestion)

Congestion TCP

Congestion Avoidance

- Augmentation de fenêtre d'un paquet par RTT
  (en pratique `cwnd += MSS * (MSS/cwnd)` à chaque ACK)
- Si on time out, retour au Slow Start
- Si on a 3 duplicate ACKs, on récupère rapidement (Fast Recovery), avant de revenir ici en principe
- Dans les deux cas, on met à jour la fenêtre et le seuil de fin de slow start à 1/2 fenêtre (`ssthresh = cwnd / 2; cwnd = ssthresh`)

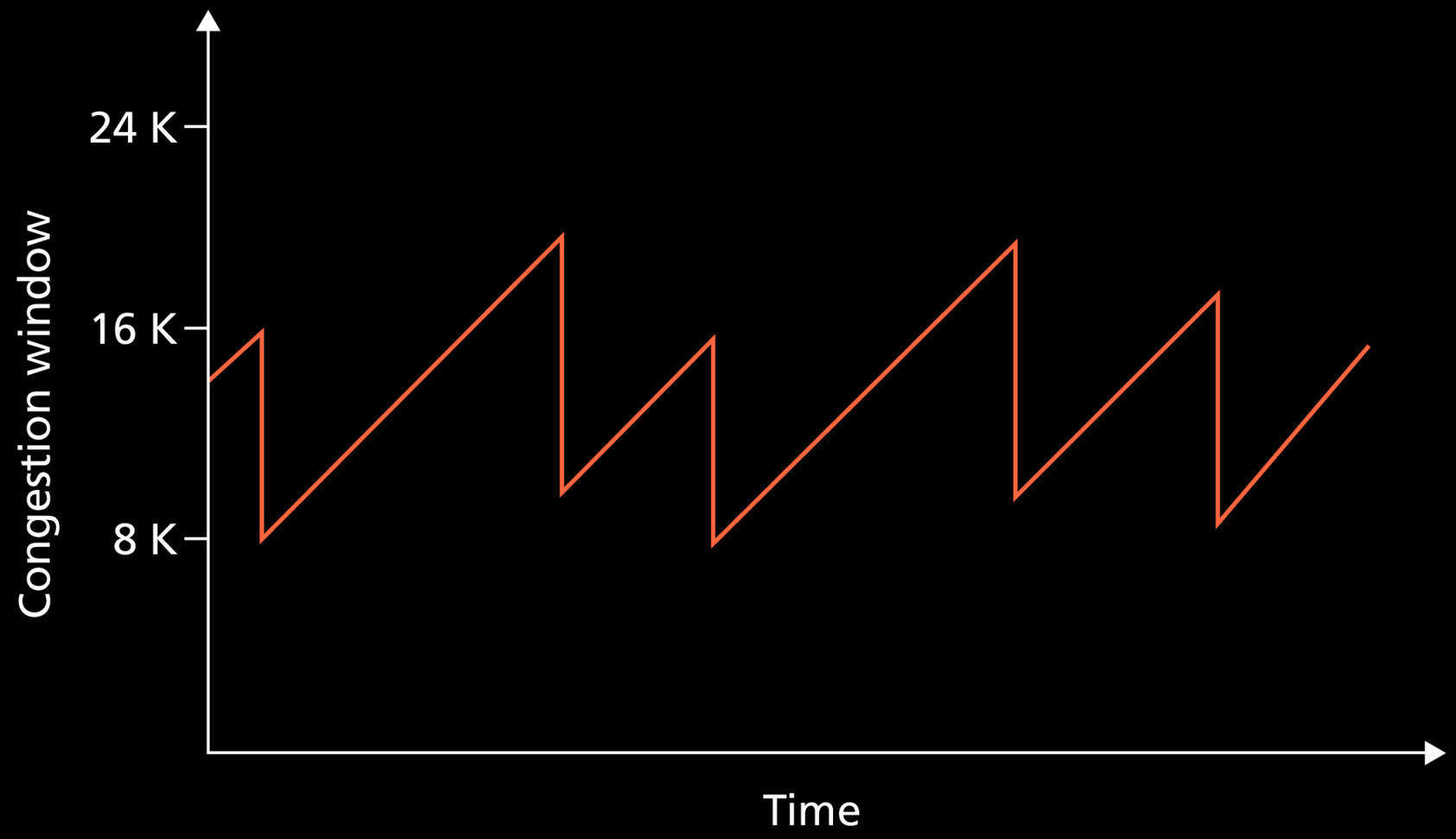
Congestion TCP

Fast Recovery

- Pour chaque ACK dupliqué (y compris les trois premier on augmente la fenêtre d'un paquet : `cwnd += MSS` à chaque ACK)
- On retransmet le paquet manquant
- Si on time out retour en Slow Start
- Dés qu'on a un ACK non dupliqué, on retourne en Congestion Avoidance, en réinitialisant la fenêtre au seuil (`cwnd = ssthresh`)

Synthèse

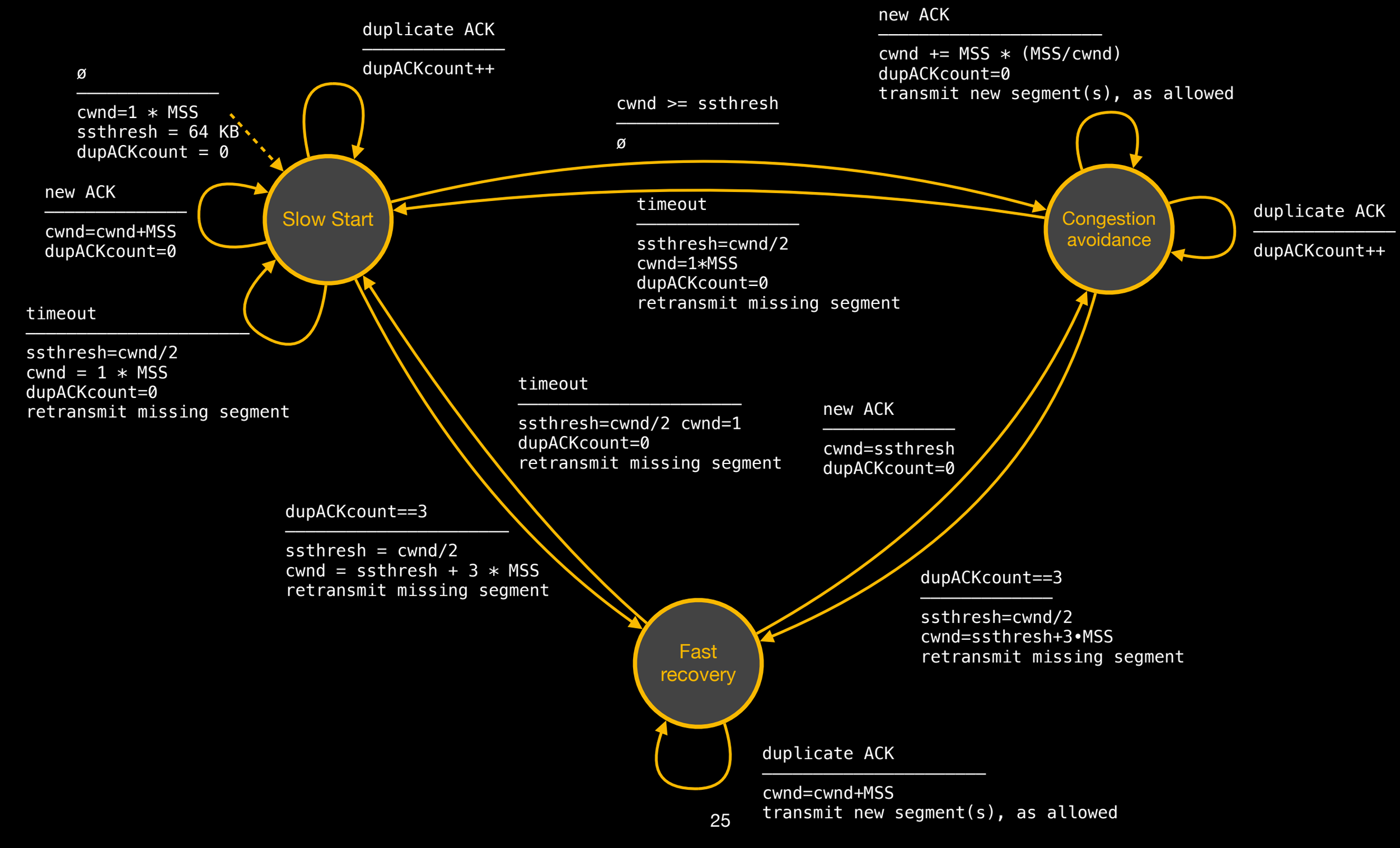
Additive increase - multiplicative decrease



Source Kurose

Retour sur la FSM

TCP Reno



Retour sur les retransmissions TCP

Que renvoie-t-on après une perte de paquet ?

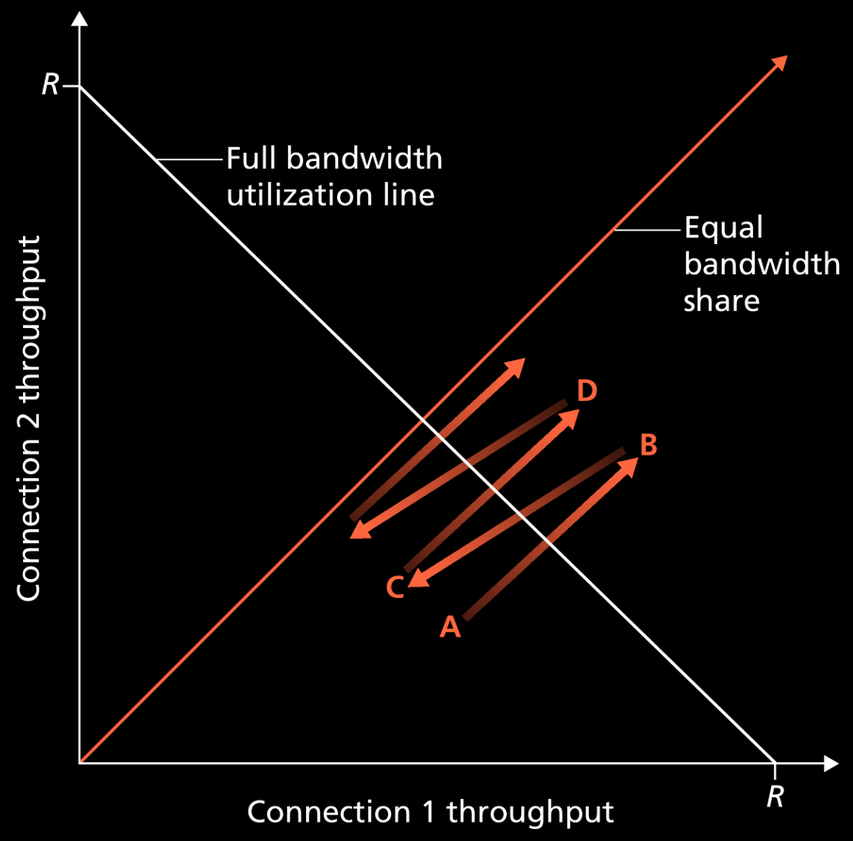
- Pour revenir sur le cours précédent, au vu des mécanismes de gestion de la congestion, sur le fait de renvoyer un paquet après une perte.

- Lorsqu'on a un timeout, la fenêtre revient à 1, donc effectivement on renvoie un unique paquet

- Lorsqu'on détecte une perte par ACK dupliqué, on divise la fenêtre par deux. Si celle-ci était remplie, on n'a pas de marge pour renvoyer de nouveaux paquets et on renvoie exactement le paquet perdu.

Équité

Partage de la bande passante



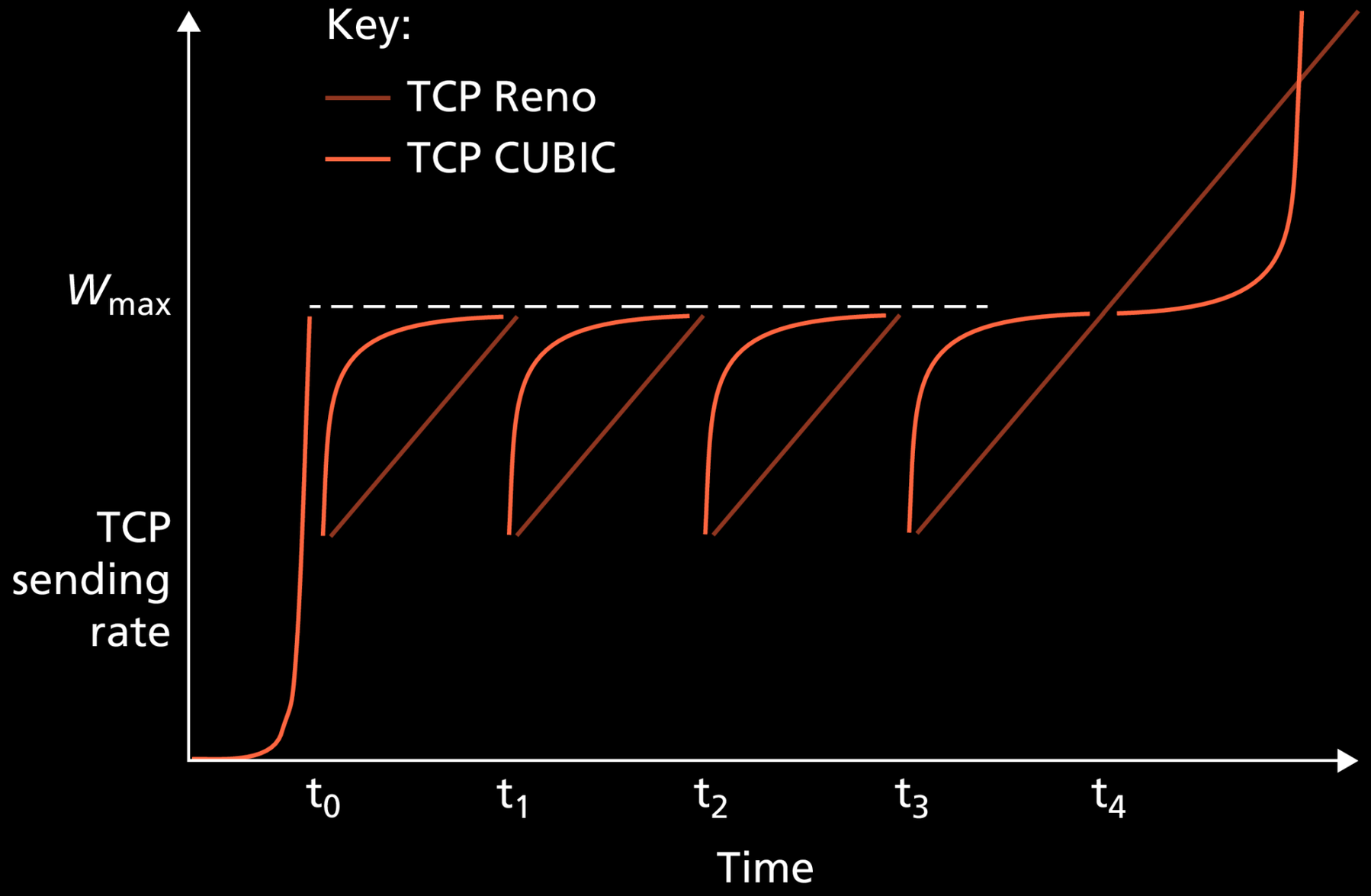
- Marche bien entre 2 connections TCP
- Mais ouvrir plusieurs connections TCP en parallèle permet d'accroitre sa part de débit
- Marche moins bien lorsque'il y a aussi de l'UDP

Domaine de recherche actif

Source Kurose

Améliorer la stratégie

TCP Cubic - Aperçu



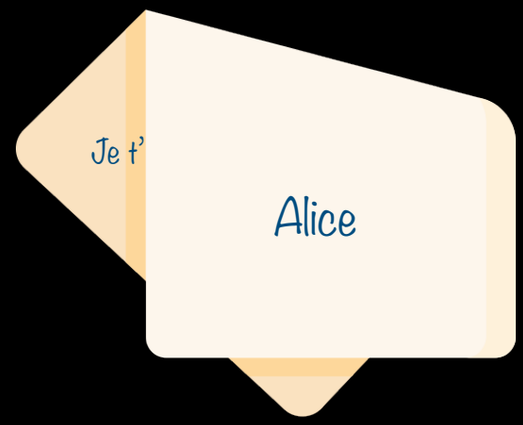
Source Kurose

Questions ?

TCP est un vaste sujet, cf. Kurose, RFCs, etc.

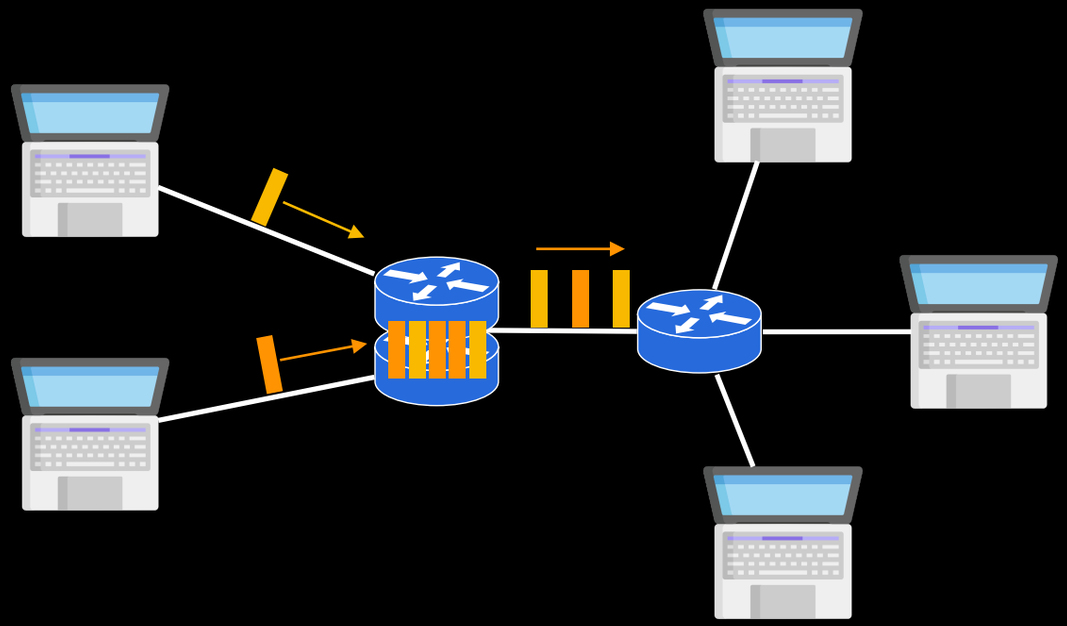
La couche réseau IP : Le plan de données

Psst, fais passer

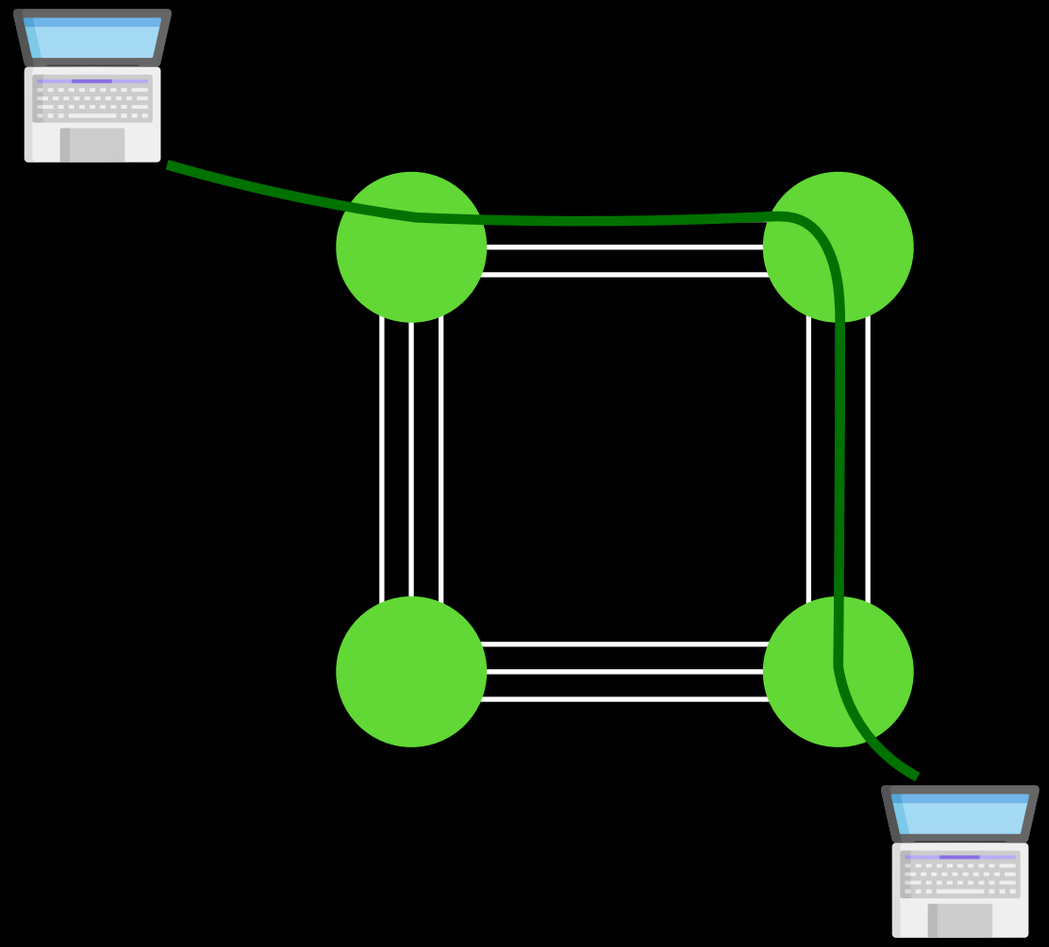


Pourquoi des paquets ?

Packet-Switching vs d’autres choix



Packet Switching



Circuit Switching

Voir Kurose 1.3 pour plus de détails

Plan de Donnée et Plan de Contrôle

Distinction importante

Plan de données

- Transmet les paquets utilisateurs
- Consulte une table de routage qui donne l'interface de sortie pour la destination souhaité
- Aujourd'hui (R4) et la semaine prochaine (R5)

Plan de contrôle

- Établi les tables de routage
- Exécute des algorithmes de routage
- La semaine prochaine (R5) et dans deux semaines (R6)

Services offerts par un réseau

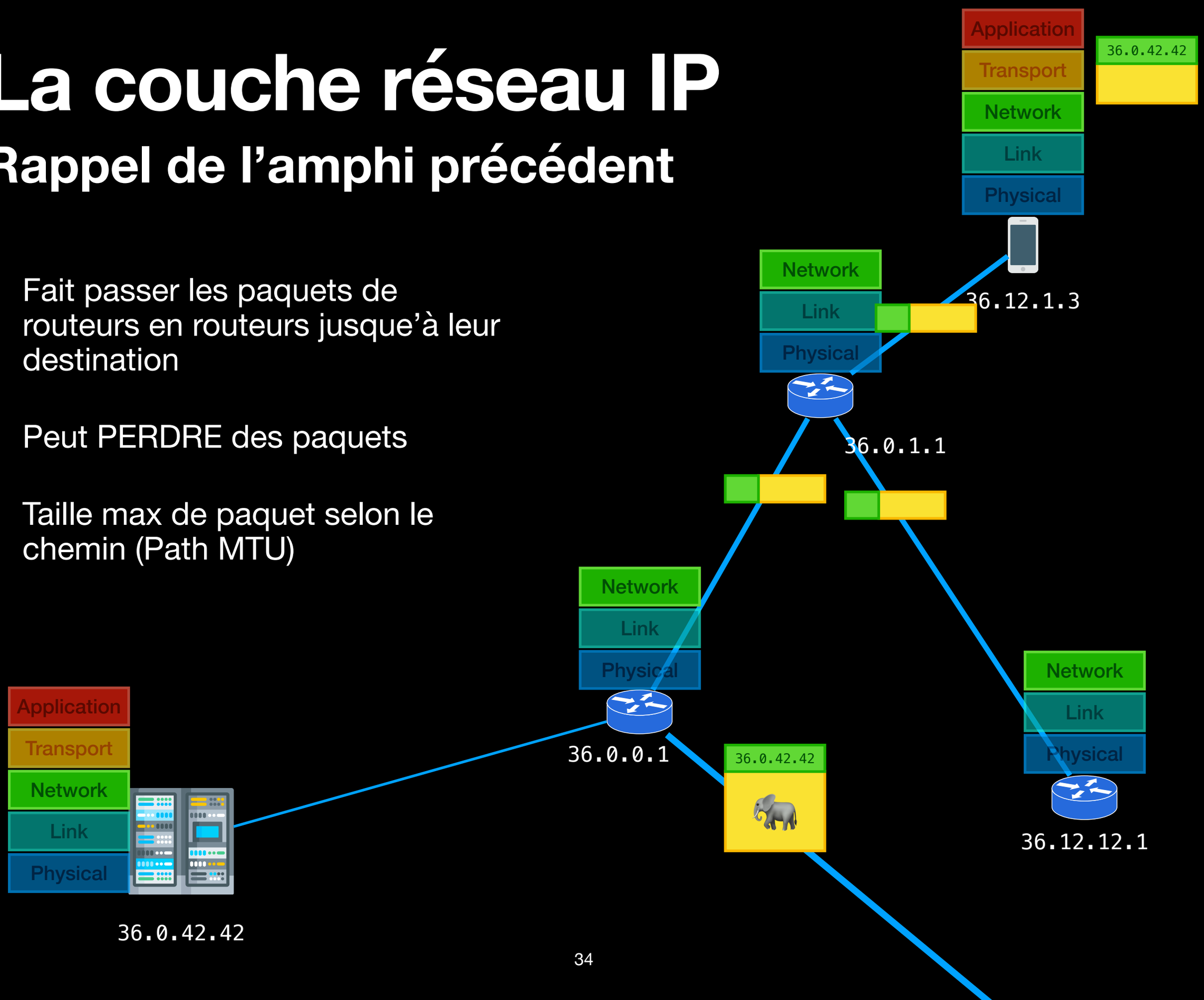
Et ce qu'IP veut bien fournir

- Guaranteed delivery
- Guaranteed delivery with bounded delay
- In-order packet delivery
- Guaranteed minimal bandwidth
- Security (e.g. chiffrement+authenticité)
- …
- Best-effort ← IP provides only this

La couche réseau IP

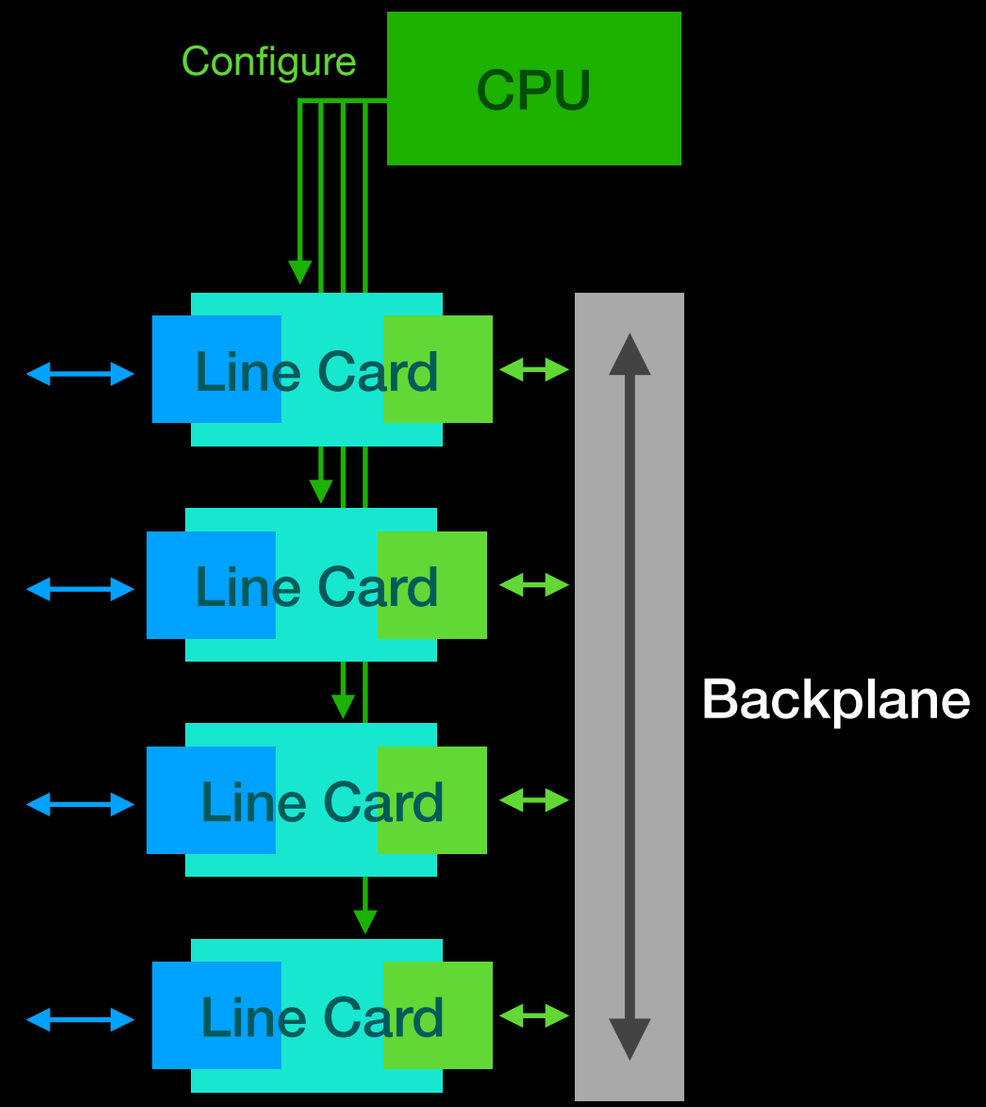
Rappel de l'amphi précédent

- Fait passer les paquets de routeurs en routeurs jusque'à leur destination
- Peut PERDRE des paquets
- Taille max de paquet selon le chemin (Path MTU)



Qu’est-ce qu’un routeur ?

Un matériel qui transmet un paquet vers le prochain nœud en fonction de sa destination finale



Routage par préfix

Parce que 2^{32} (voir 2^{128}) adresses ça scale pas

- On attribue les adresses par blocs contigus
- On peut donc router des blocs contigus
- Qu'on peut identifier sous forme d'un préfix
- On sélectionne l'entrée la plus longue applicable
- Notation x.y.z.t/nn ou nn donne la longueur du préfix

Préfix (binaire)	Notation IPv4	Interface
11001000 00010111 00010	200.23.16.0/21	0
11001000 00010111 00011000	200.23.24.0/24	1
11001000 00010111 00011	200.23.24.0/21	2
–	0.0.0.0/0	3

Ce dont on ne parle pas

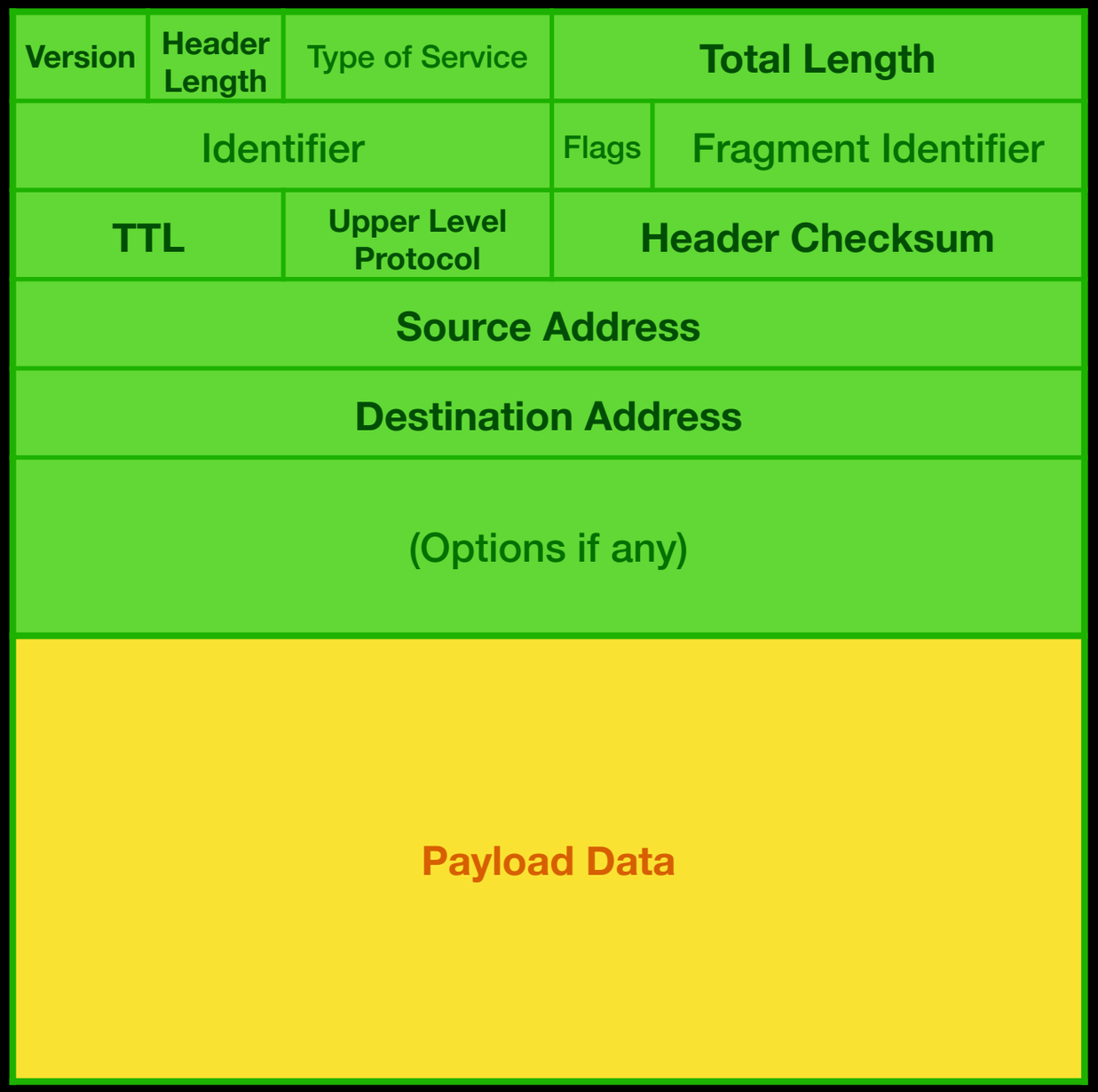
Fabrique de commutation (Backplane) - Lisez Kurose

Pour faire marcher un routeur,
il faut réussir à faire communiquer
toutes les entrées et sorties

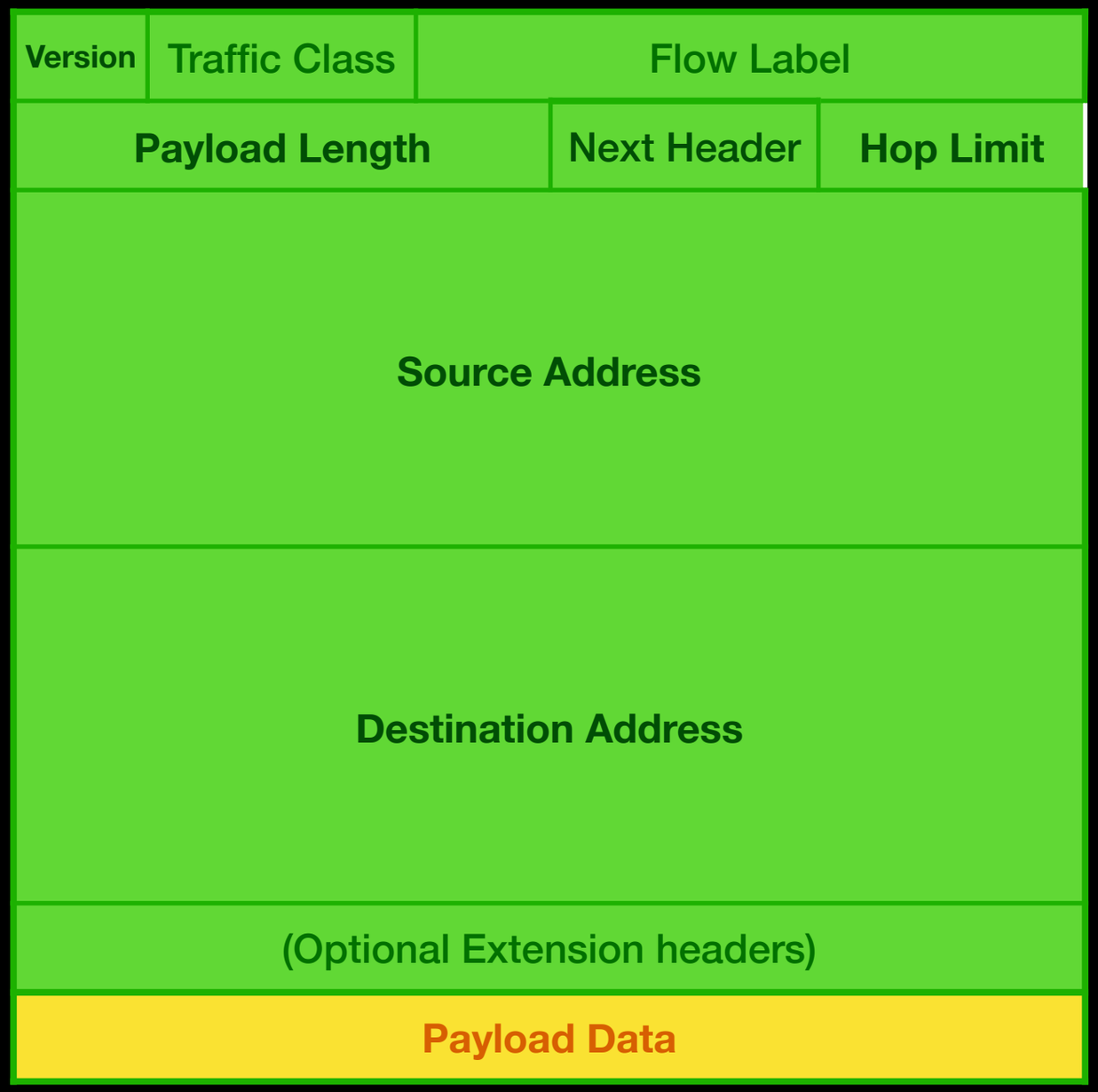
C'est un problème assez complexe

Anatomie d’un paquet IP

IPv4 vs IPv6



IPv4



IPv6

Résumé

Qu’avons nous vu aujourd’hui sur le plan de donnée

- Décision nœud par nœud
- Le plan de donnée lit des tables fournies par le plan de contrôle
- Ces tables utilisent le principe du plus long préfix correspondant
- Aperçu des structures des paquets

Questions ?

Résumé du jour

Congestion TCP et Plan de données IP

Est-ce que vos notes ont tout ça ?

- TCP
  - Fonctionnement de la gestion de congestion TCP dite Reno
    - Slow Start, Congestion Avoidance, Fast recovery
- Plan de donnée IP
  - Plan de donnée / Plan de contrôle
  - Fonctionnement général du plan de donnée
  - Plus long préfix correspondant

La prochaine fois

Suite du plan de données, début du plan de contrôle

- Plan de donnée
  - Comprendre comme on attribue les IP
  - Signification de quelques autres champs du paquet IP
  - Des histoires de files, de priorité et d’équité
  - Généralisations (d’autre protocoles que IP ; SDN)
- Plan de contrôle
  - Le problème à résoudre
  - Algorithmes fondamentaux, Dijkstra et Bellman-Ford