Sys1 - Amphi 9

Protocoles de Transport

Guillaume DIDIER - 17/11/2023

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, notion de Lien	Projet (binôme): Crawler HTTP & Dice Roll	Kurose Ch 3, 4.1, 4.3, 6.1
28/11/2023	Forwarding + Routage	TP (seul): Reliable Data Transport	Kurose 4.1, 4.3, 5.1, 5.2
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
Protocoles de transport

Intro aux protocoles de transport
- Le rôle de la couche transport
- Service fourni par la couche réseau IP sous-jacente

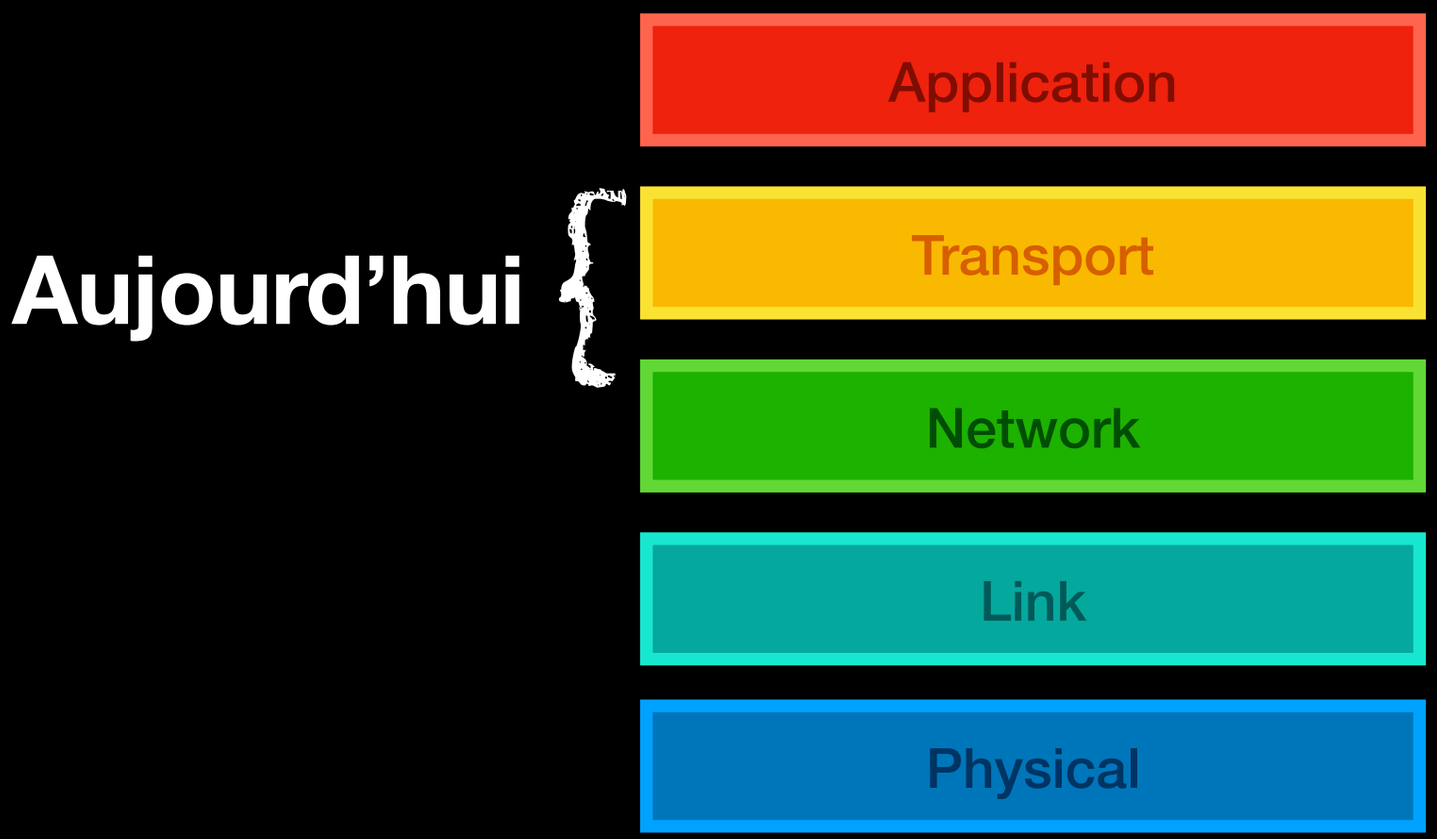
Transport UDP

Protocole de transport fiable:
- Le protocole à 1 bit
- Les protocole à fenêtre

TCP
- De RDT à TCP
- Ouvrir et Fermer une connection

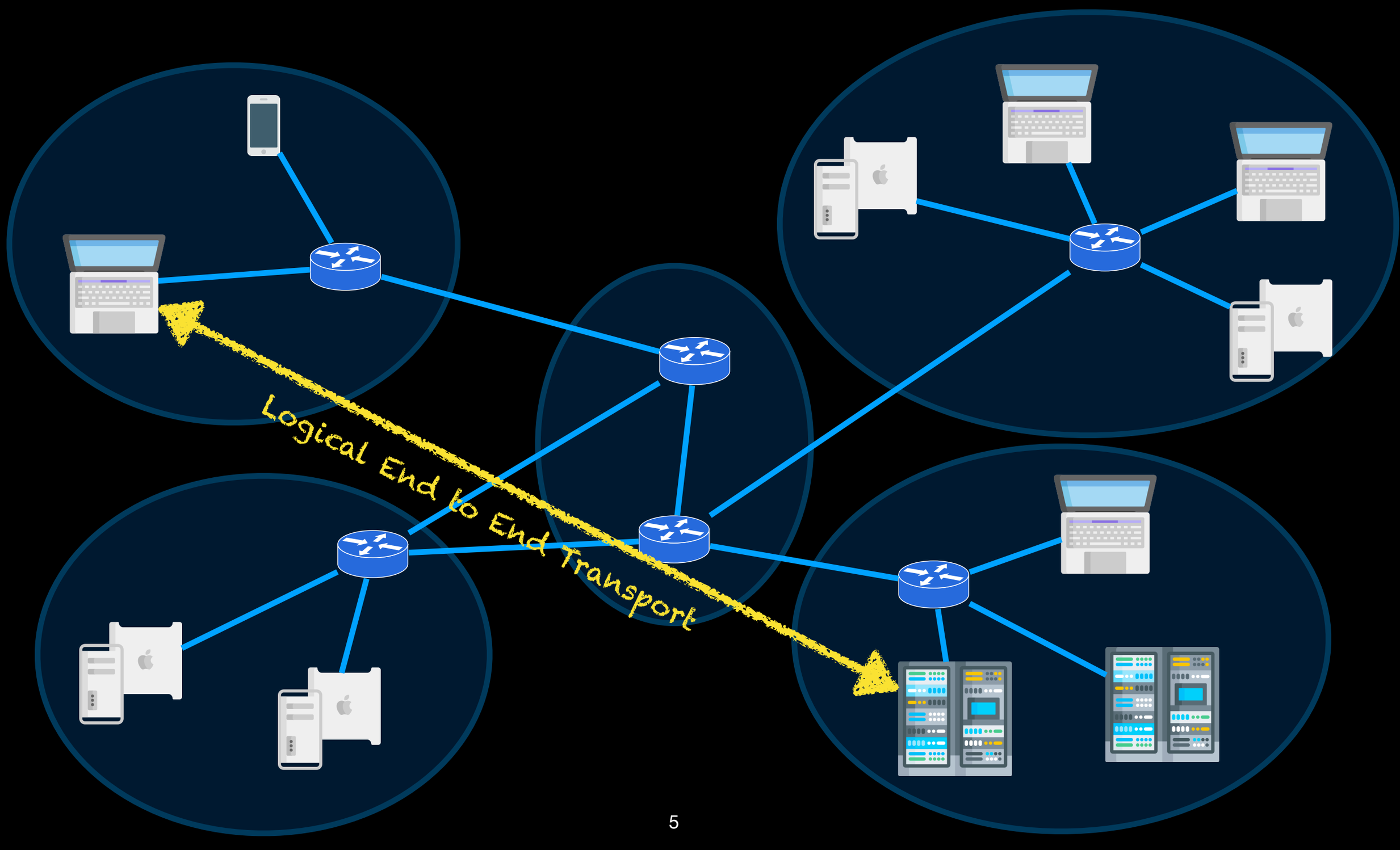
Rappel: Les couches réseau

Ou sommes nous aujourd'hui ?



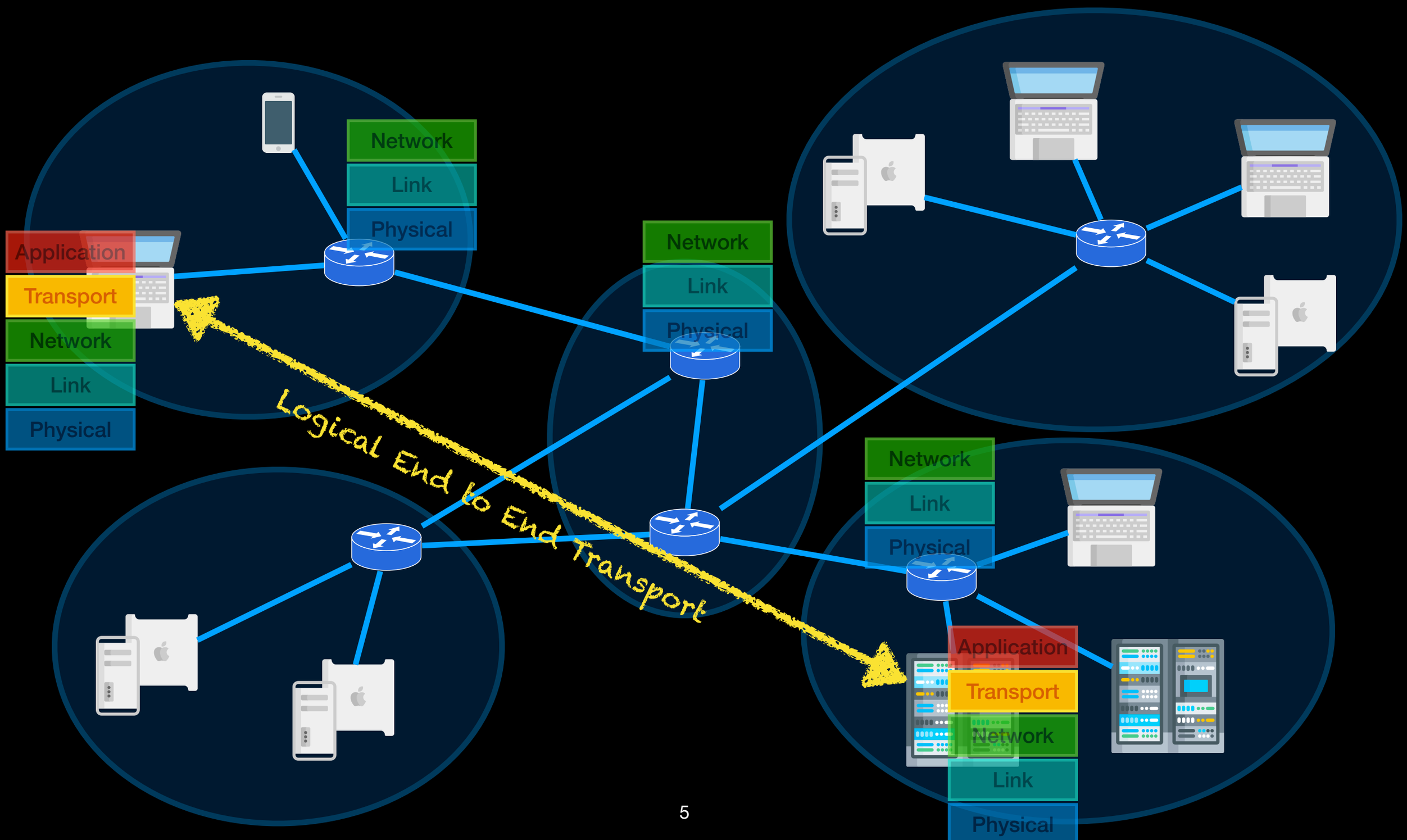
Couche transport

Quelle est le rôle d’un protocole de transport ?



Couche transport

Quelle est le rôle d’un protocole de transport ?



Rappel TCP vs UDP

Que fournissent ces deux transports

UDP

- Démultiplexage
- Détection d’erreur
- Pas d’autres garanties

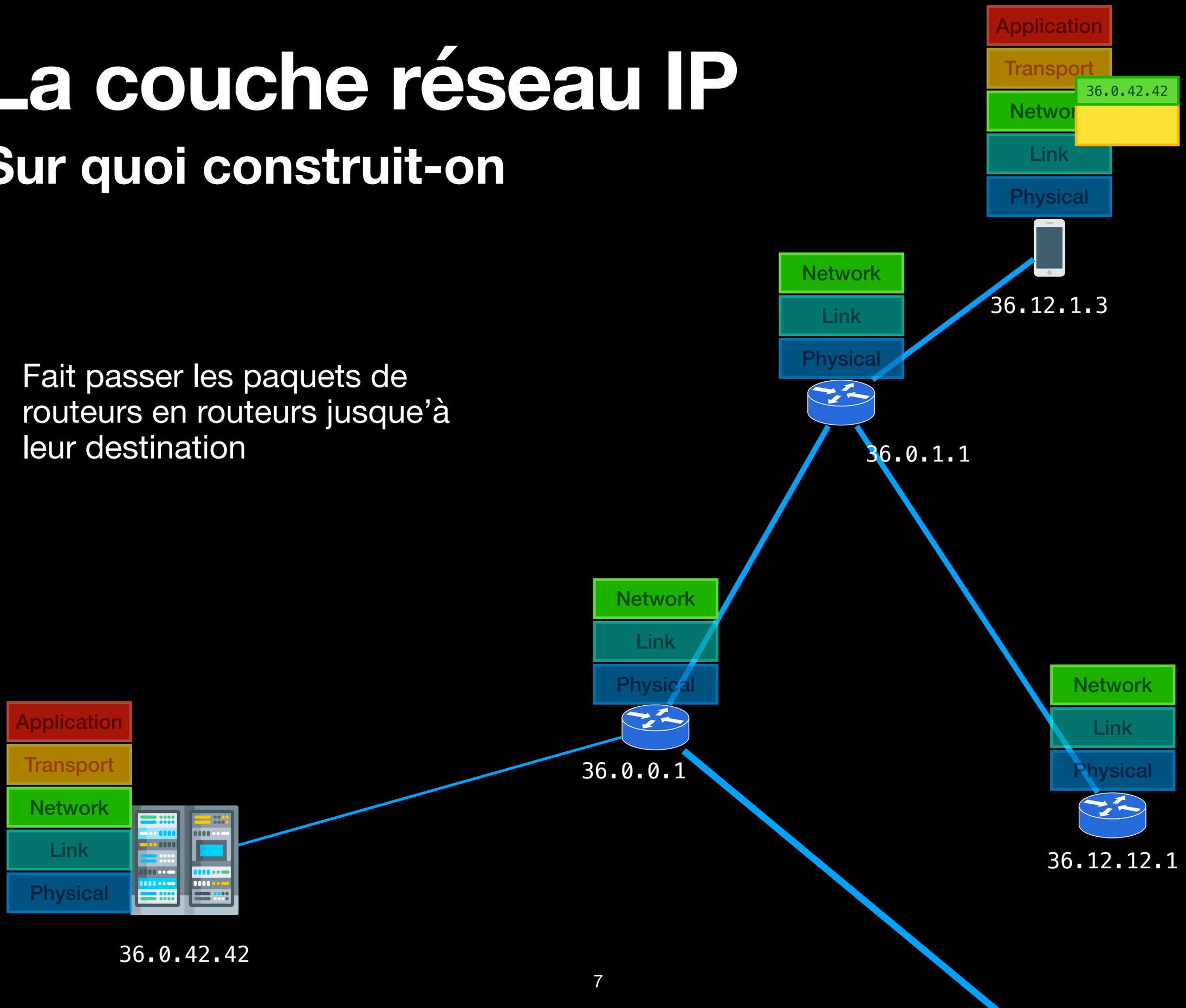
TCP

- Démultiplexage
- Détection d’erreur
- Transport Fiable des données:
  - Arrivées dans l’ordre
  - 100% d’arrivées
- Aucune garantie de Latence ou Bande Passante

La couche réseau IP

Sur quoi construit-on

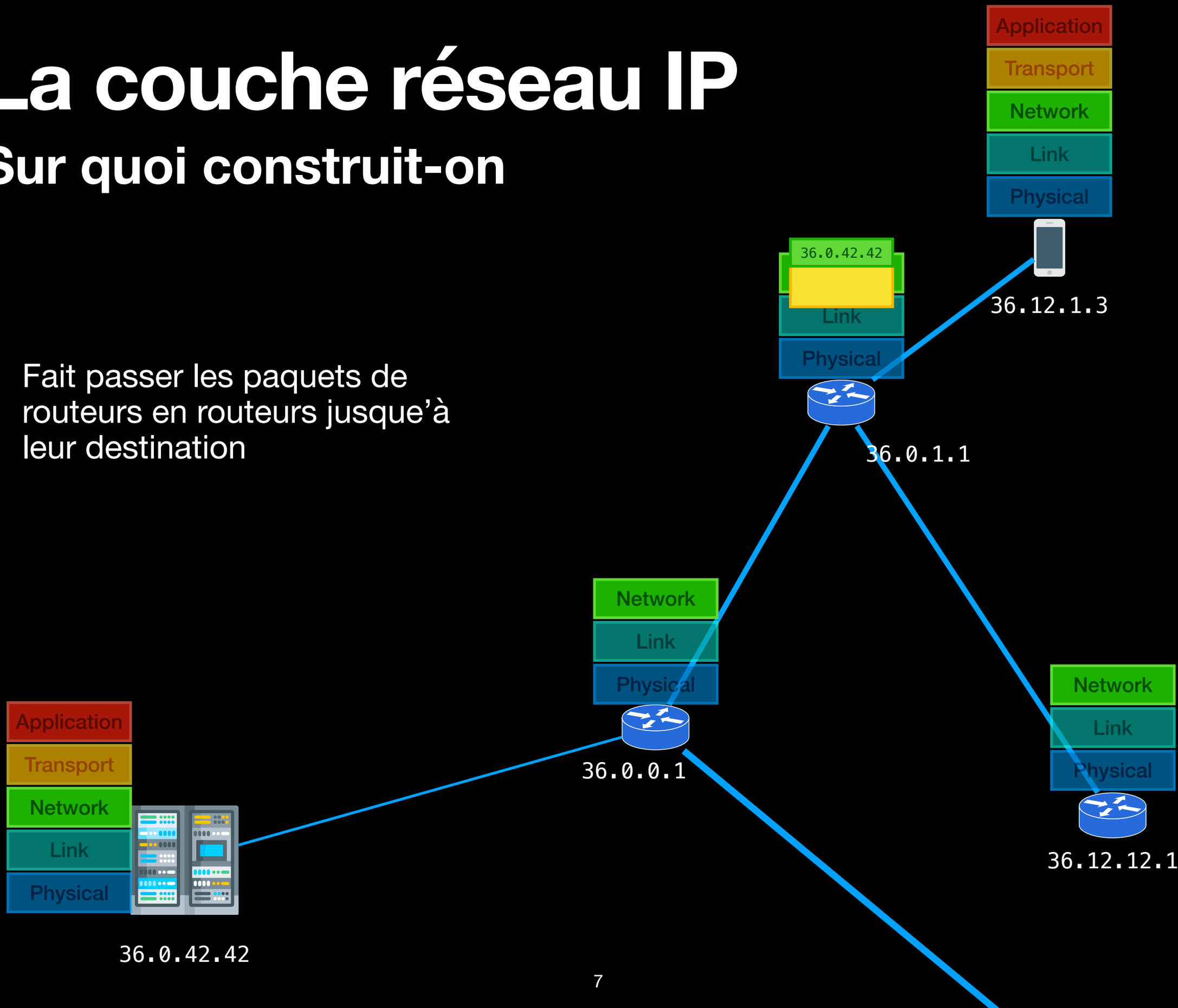
- Fait passer les paquets de routeurs en routeurs jusque'à leur destination



La couche réseau IP

Sur quoi construit-on

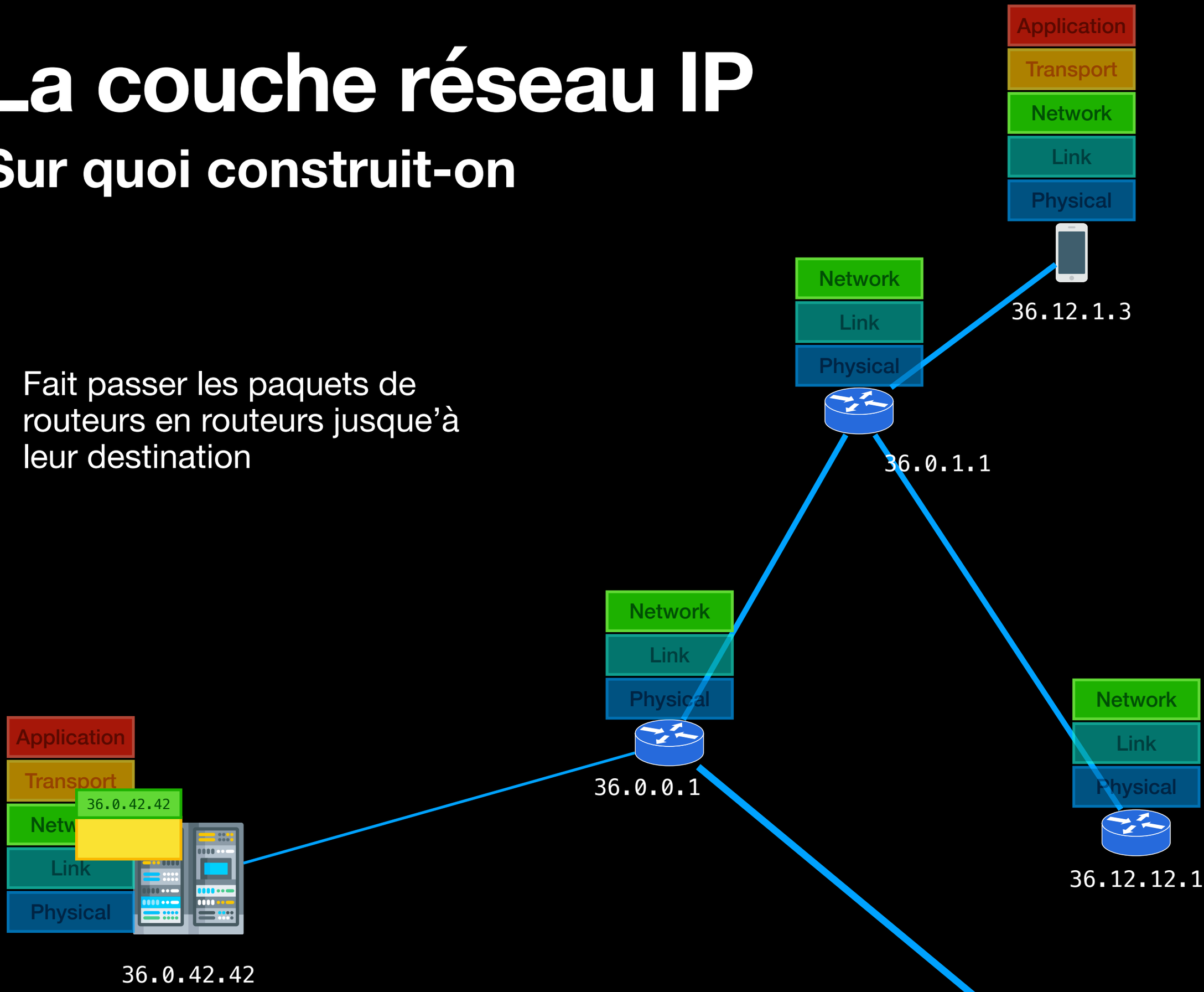
- Fait passer les paquets de routeurs en routeurs jusque'à leur destination



La couche réseau IP

Sur quoi construit-on

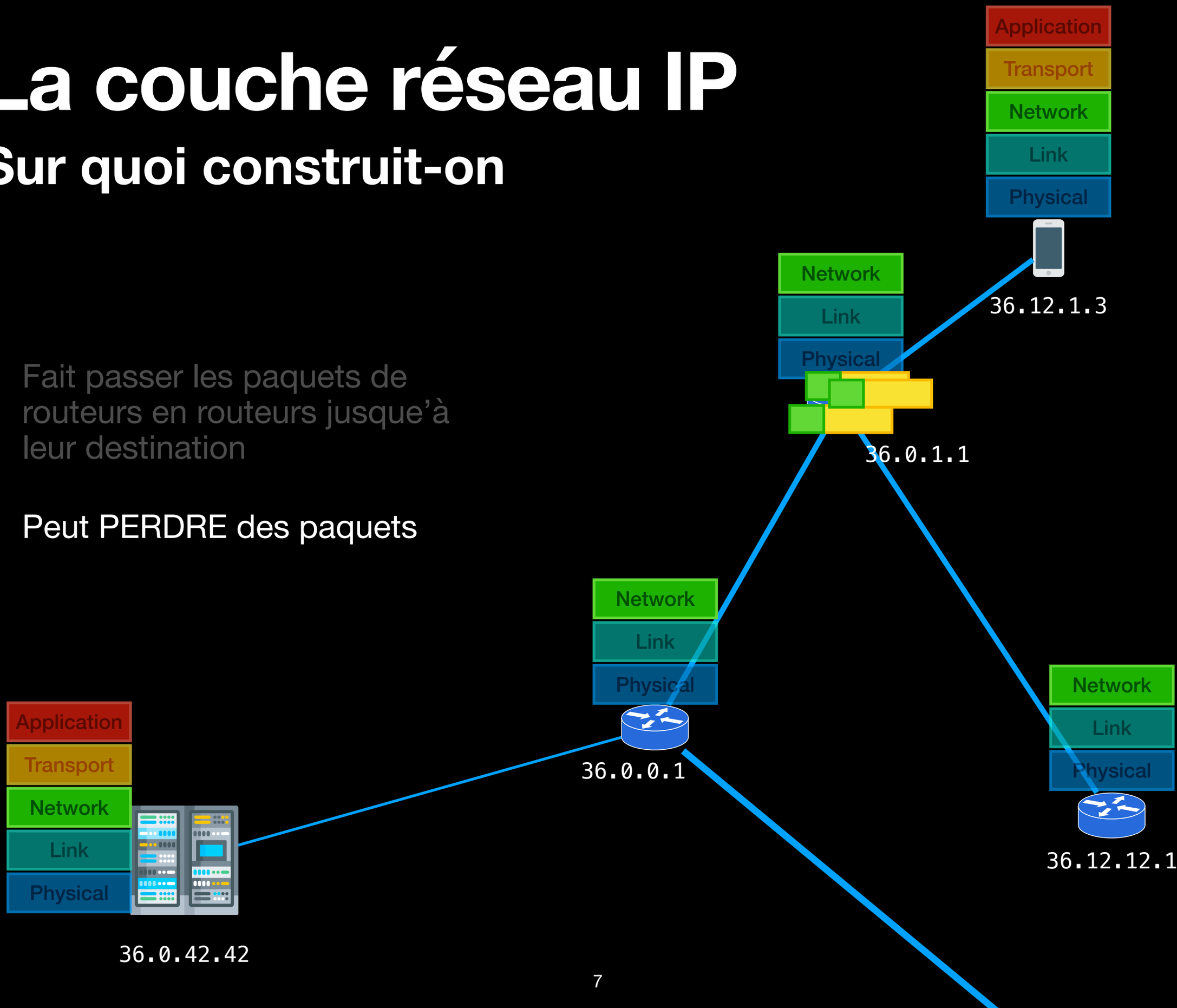
- Fait passer les paquets de routeurs en routeurs jusque'à leur destination



La couche réseau IP

Sur quoi construit-on

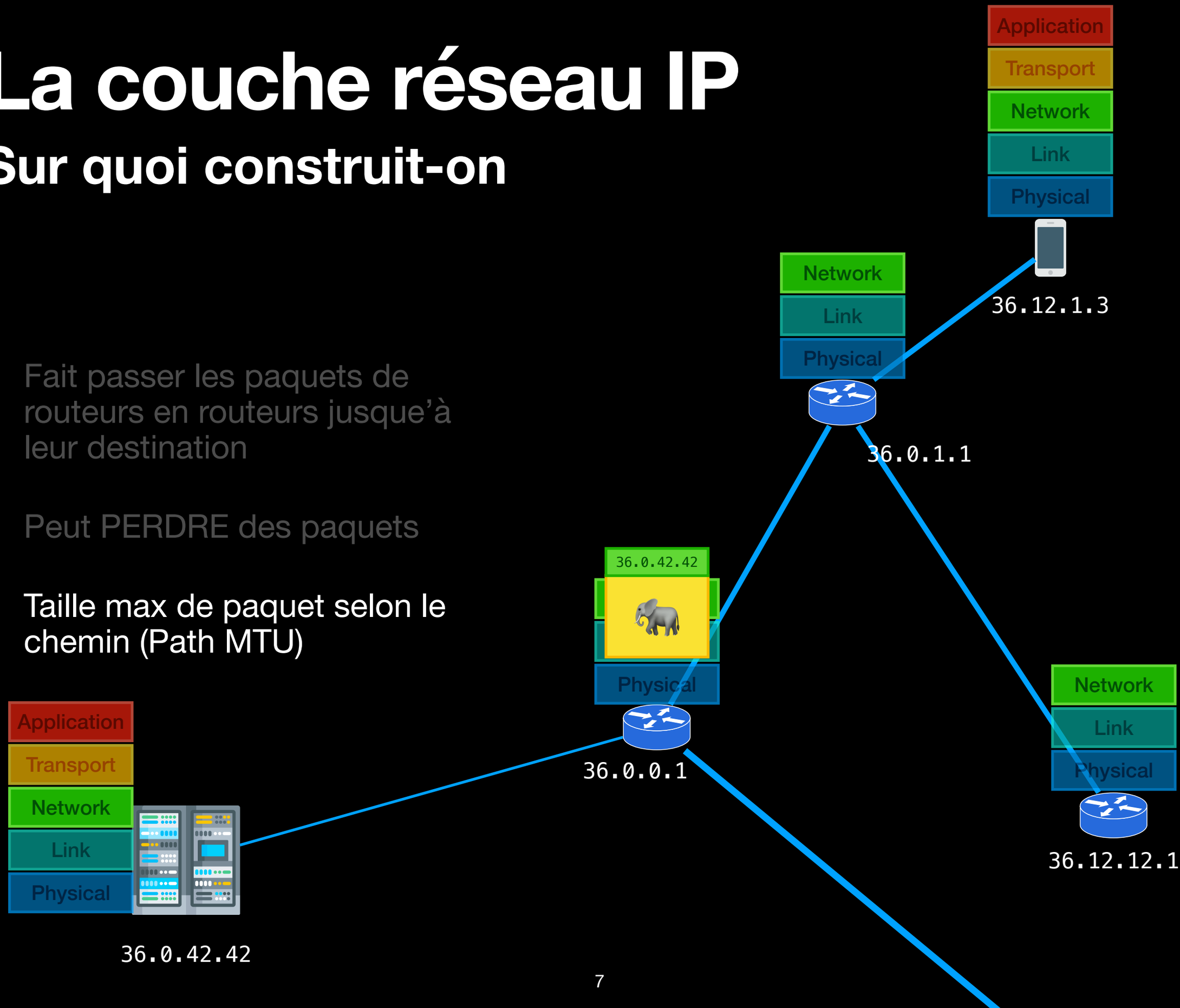
- Fait passer les paquets de routeurs en routeurs jusque'à leur destination
- Peut PERDRE des paquets



La couche réseau IP

Sur quoi construit-on

- 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)

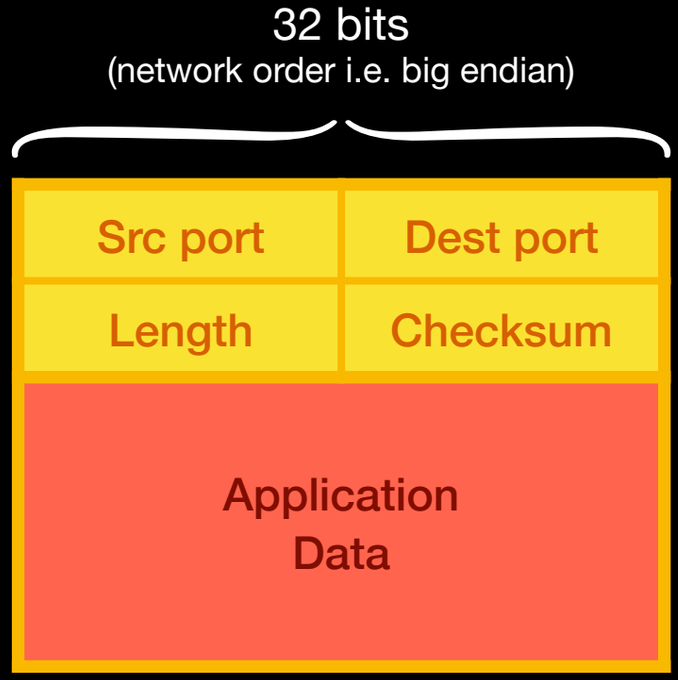


Comment marche UDP

The simplest transport

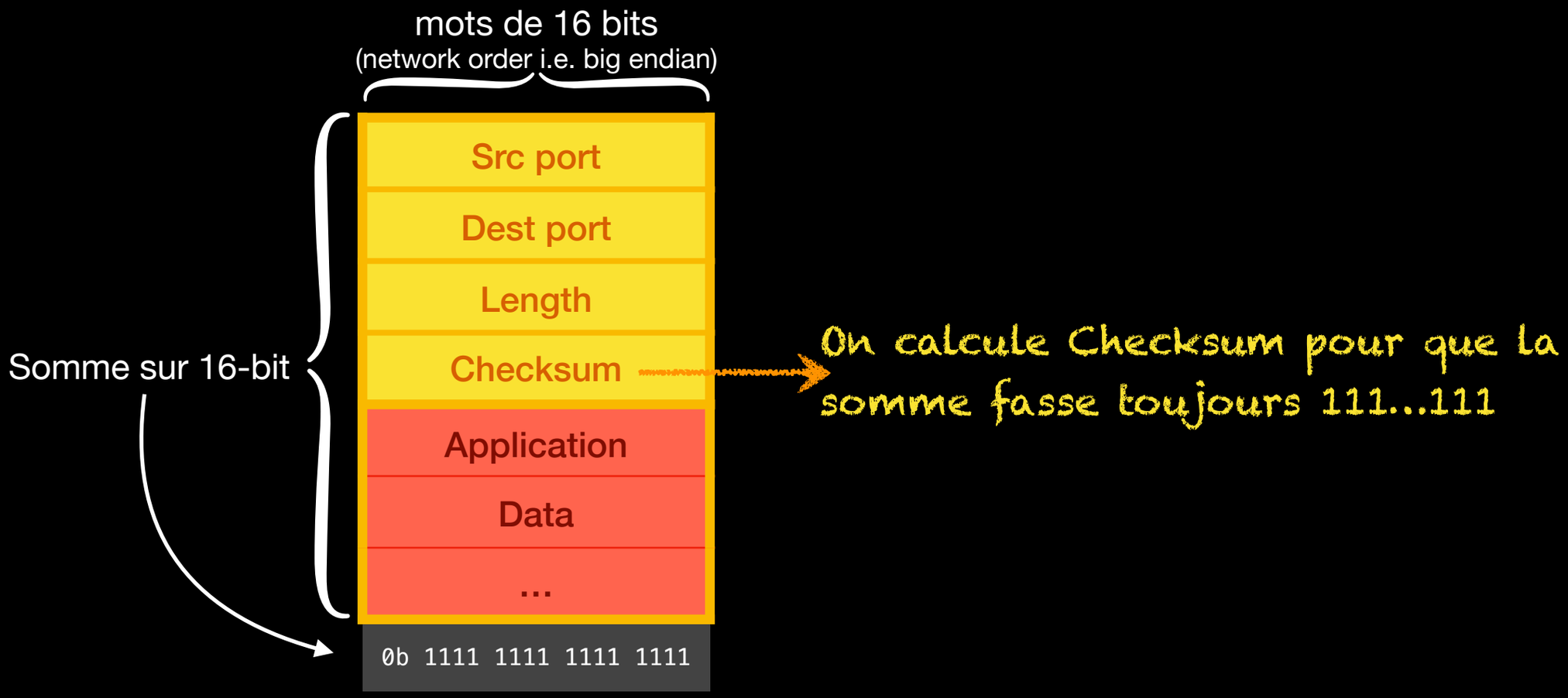
3 questions :

- Détection d’erreurs
- Démultipléxer
- À qui répondre



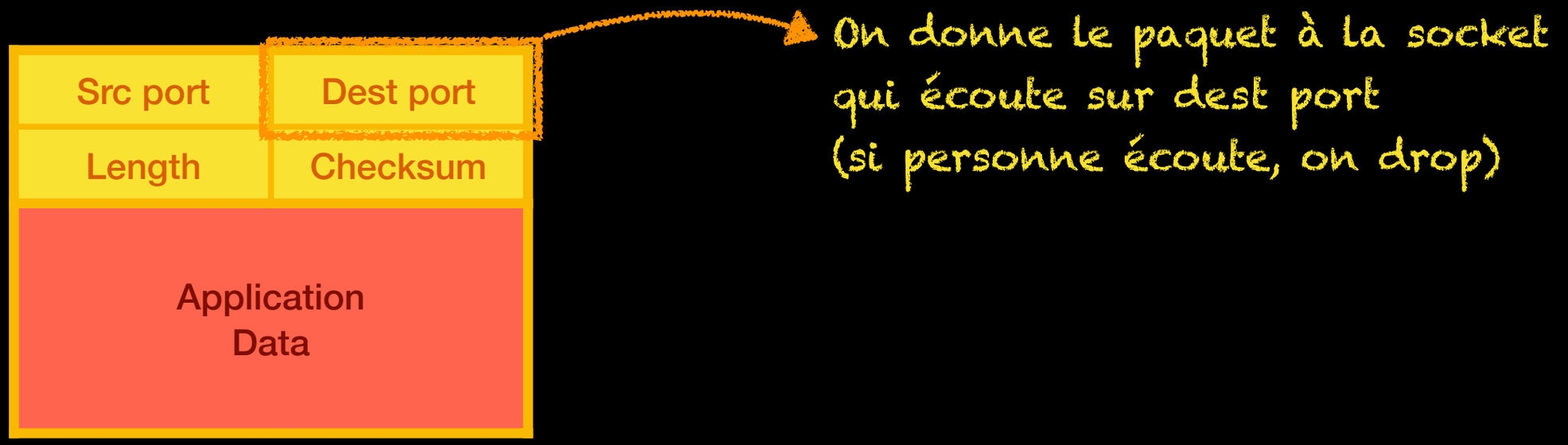
Détection d’erreur

La somme de contrôle IP (IP Checksum)



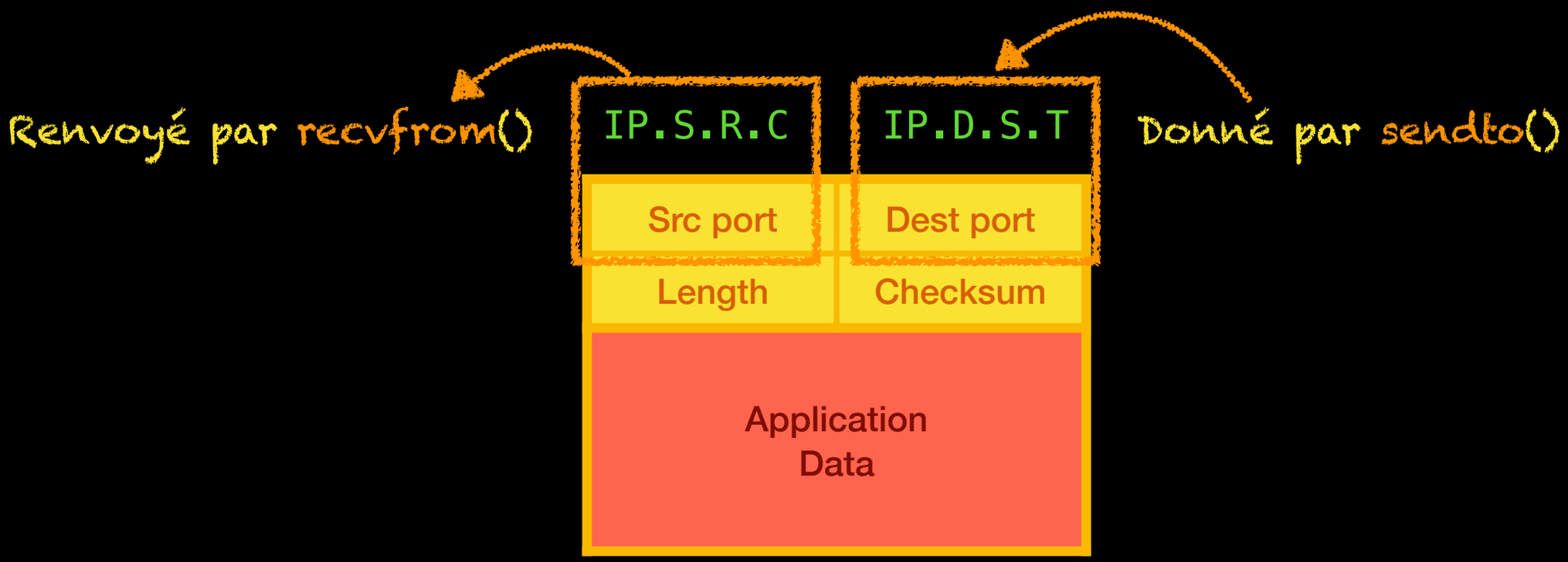
Le démultiplexage

Distribuer ~~le courrier~~ les paquets



Comment répondre ?

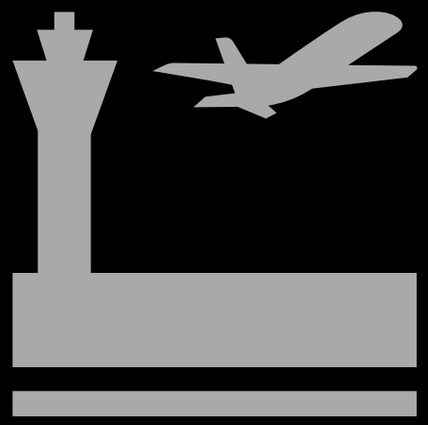
Pourquoi port source en plus du port destination ?



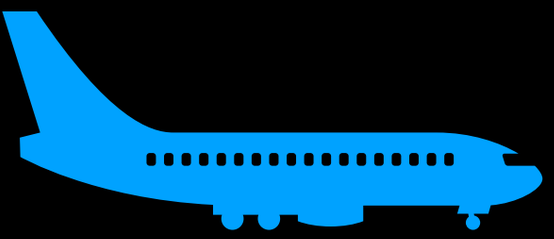
RDT

Reliable Data Transfer

Ground to AirFrans 001,
right turn on Alpha,
keep short of Mike-Alpha



(french accent)
AirFrans 001 to Ground,
turning right on Alpha,
keeping short of Mike-Alpha



Reliable Data Transfer

Les problèmes que TCP et tout transport fiable doivent résoudre

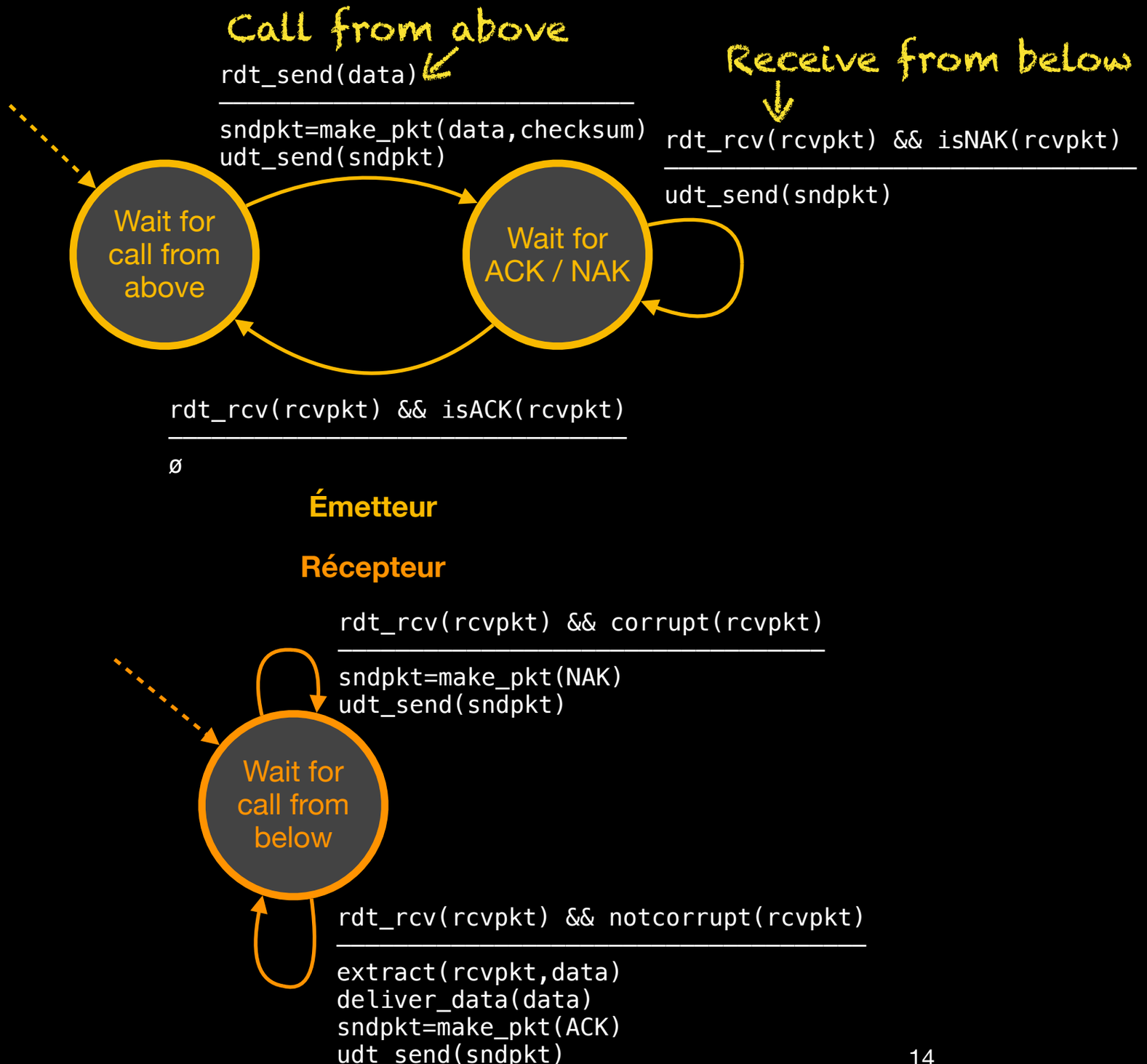
- Les paquets peuvent être corrompus

- Les paquets peuvent être perdus

- *Remarque: On peut ignorer un paquet corrompu*
  Ça fait juste un paquet perdu

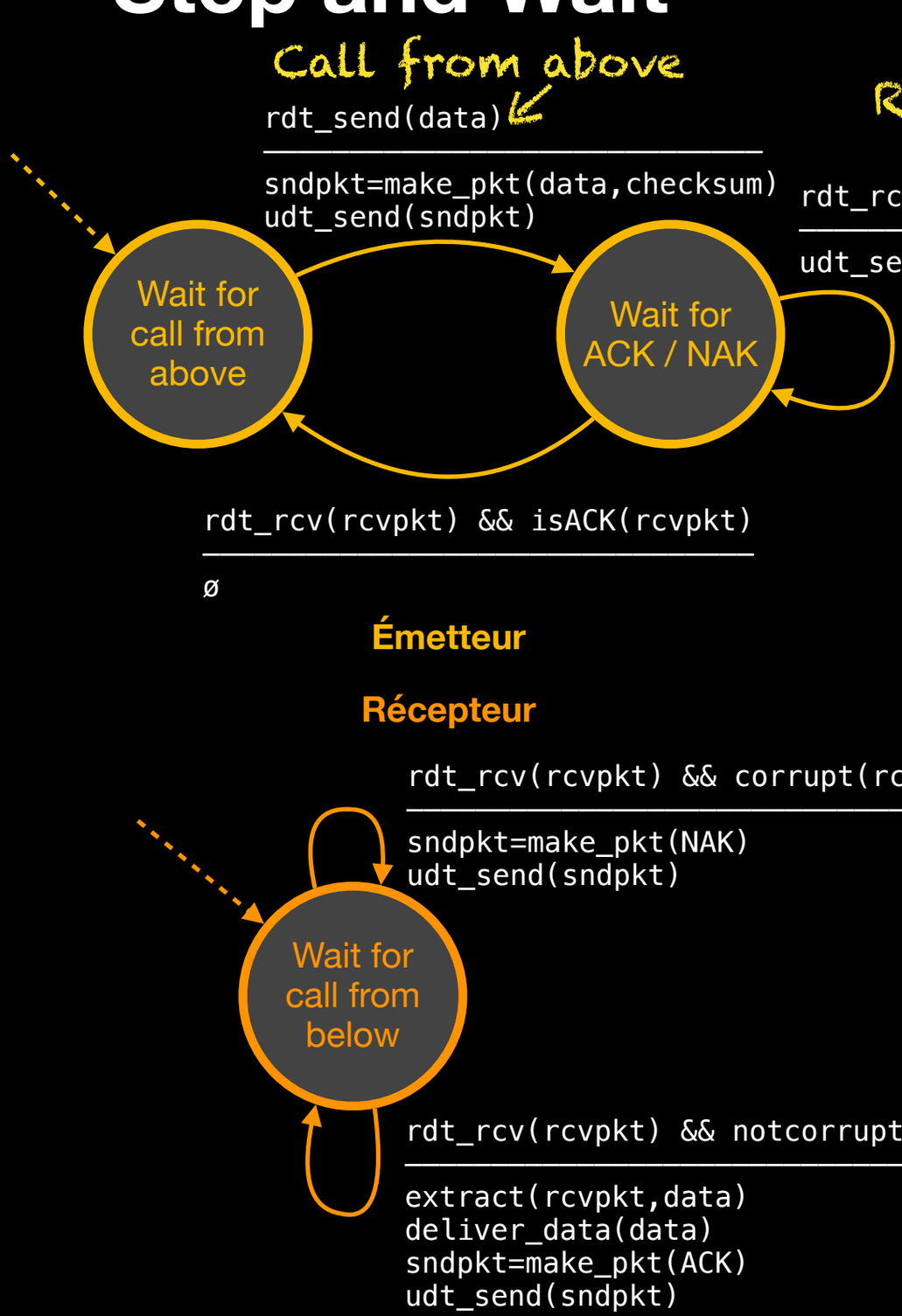
Première tentative

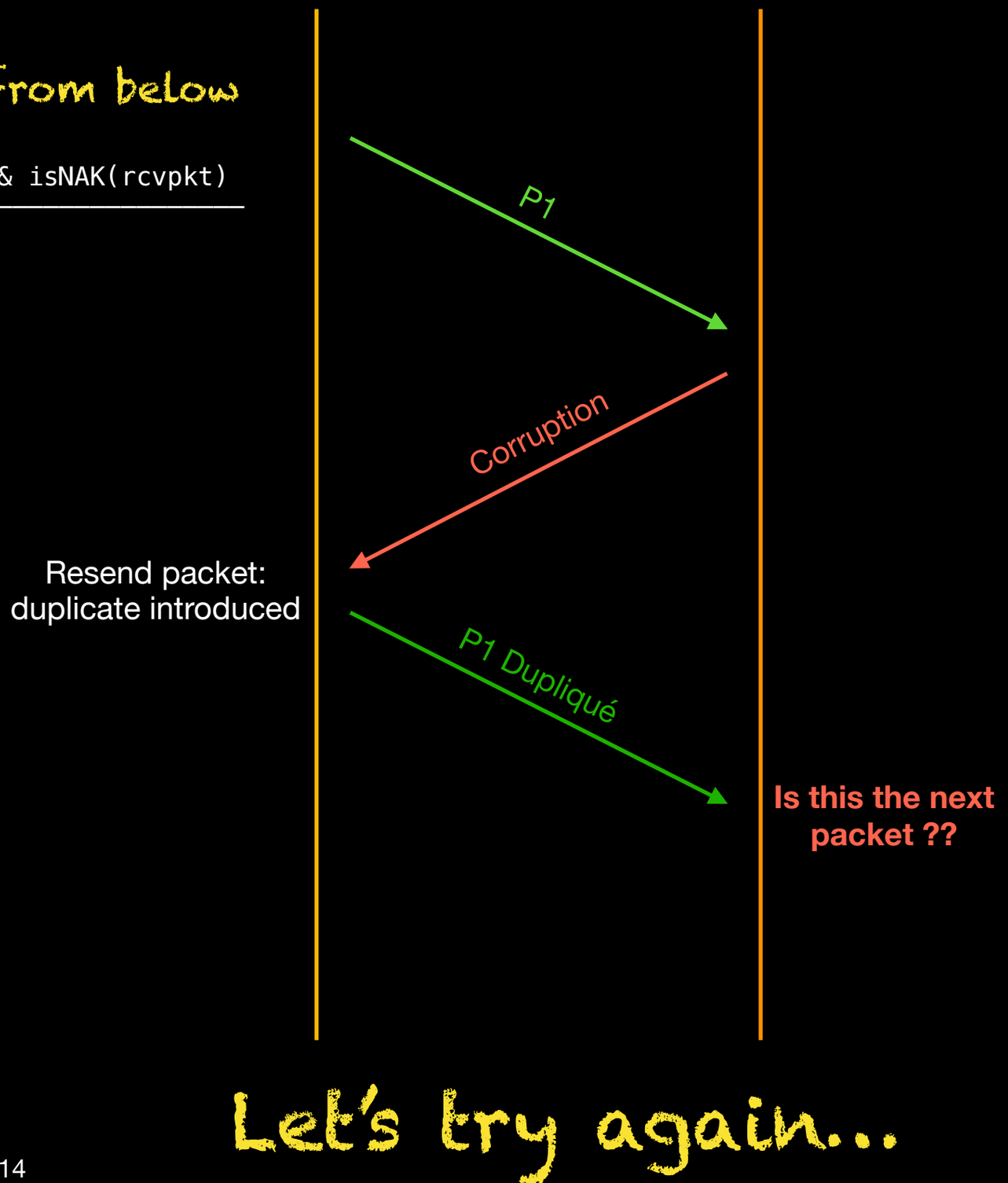
Stop and Wait



Première tentative

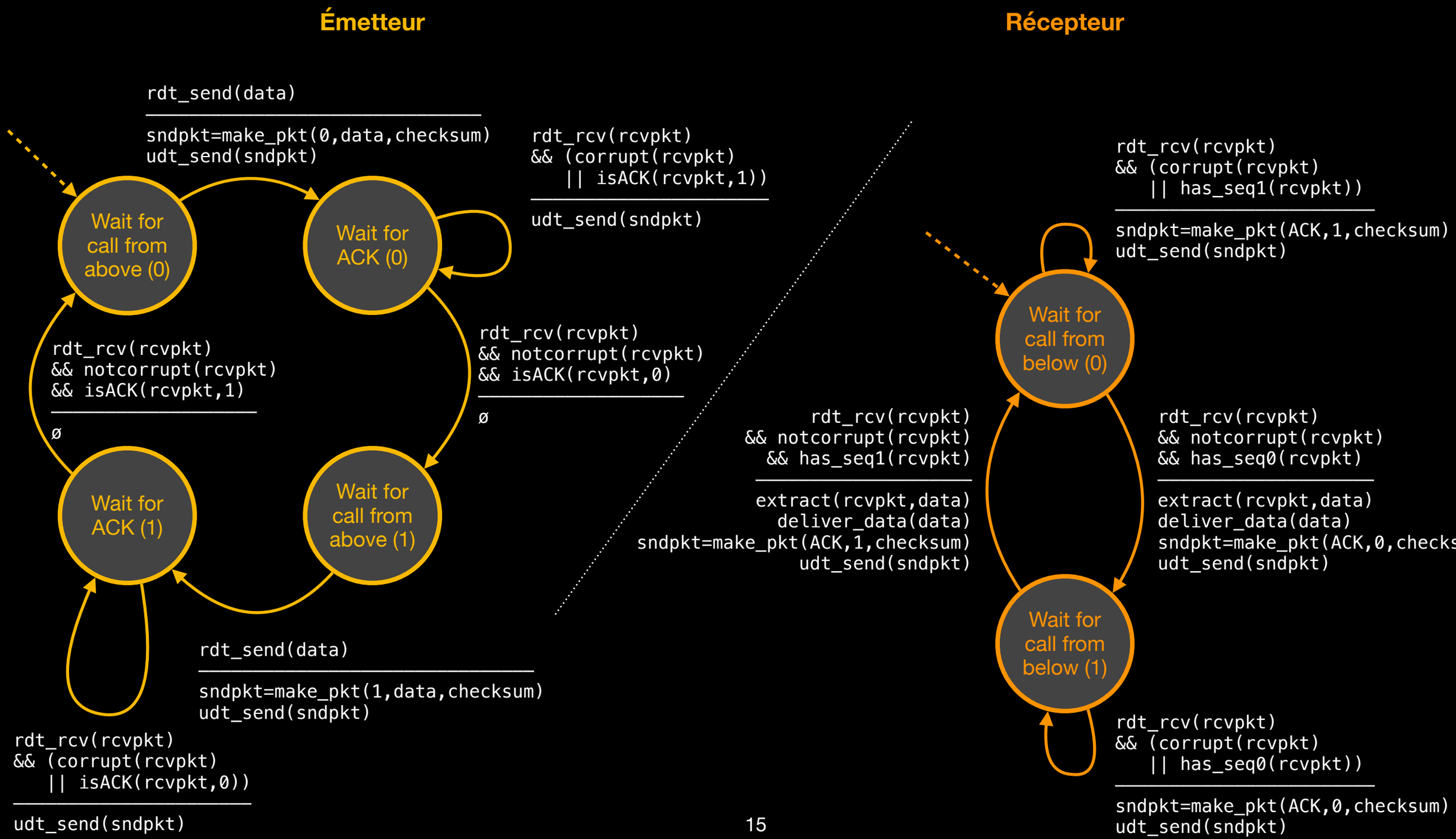
Stop and Wait





Deuxième tentative

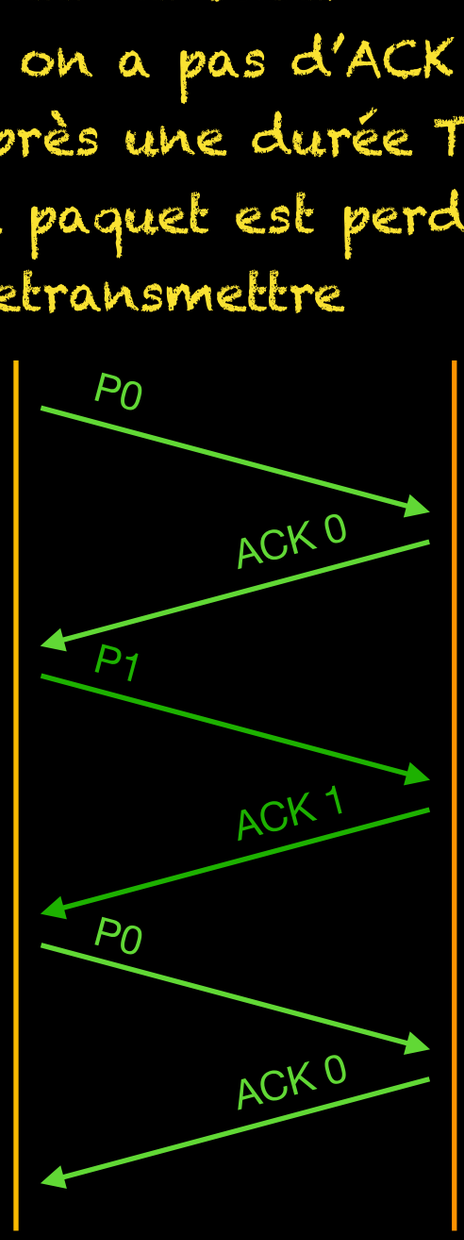
Le 1-bit protocol (sans perte) (rdt2.2 pour Kurose)



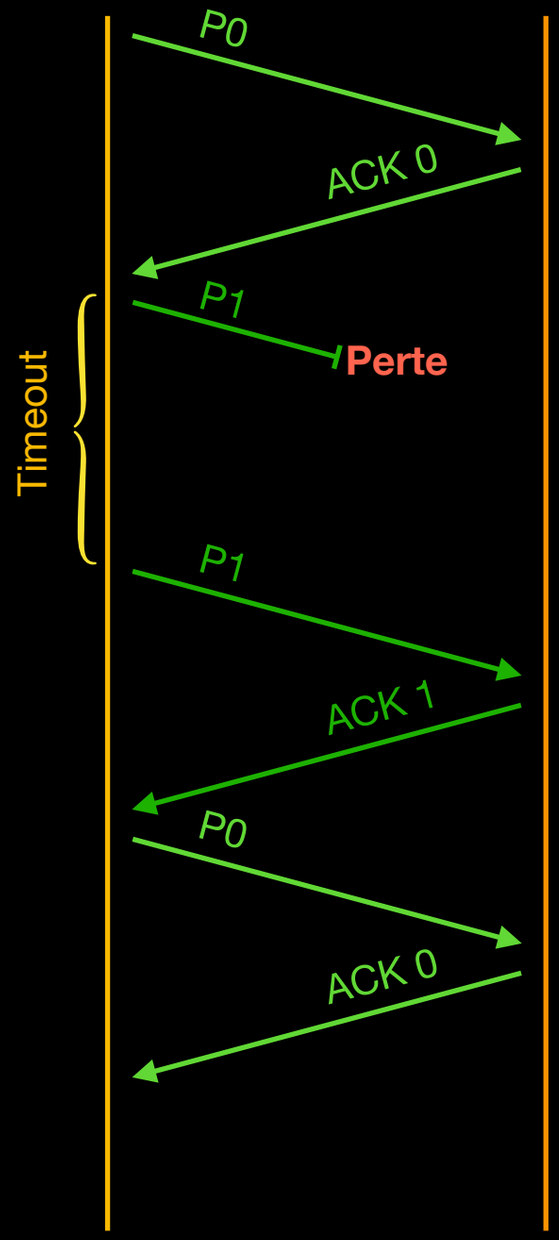
Gérer les packets perdus

Timers et retransmissions

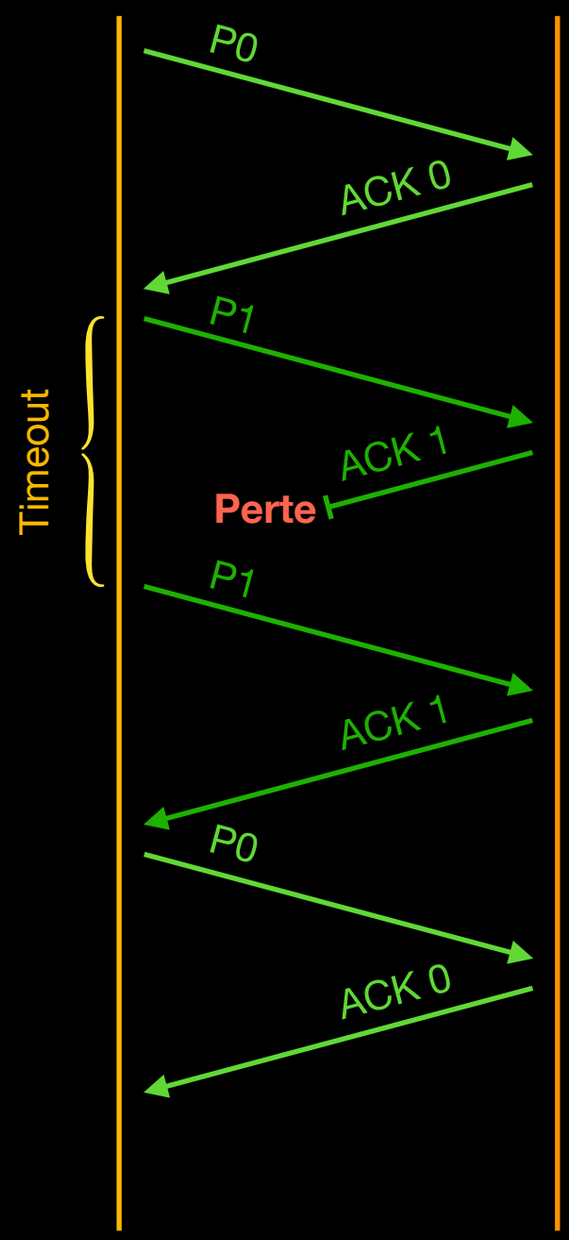
Idée de Base:
Si on a pas d'ACK
après une durée T,
le paquet est perdu.
Retransmettre



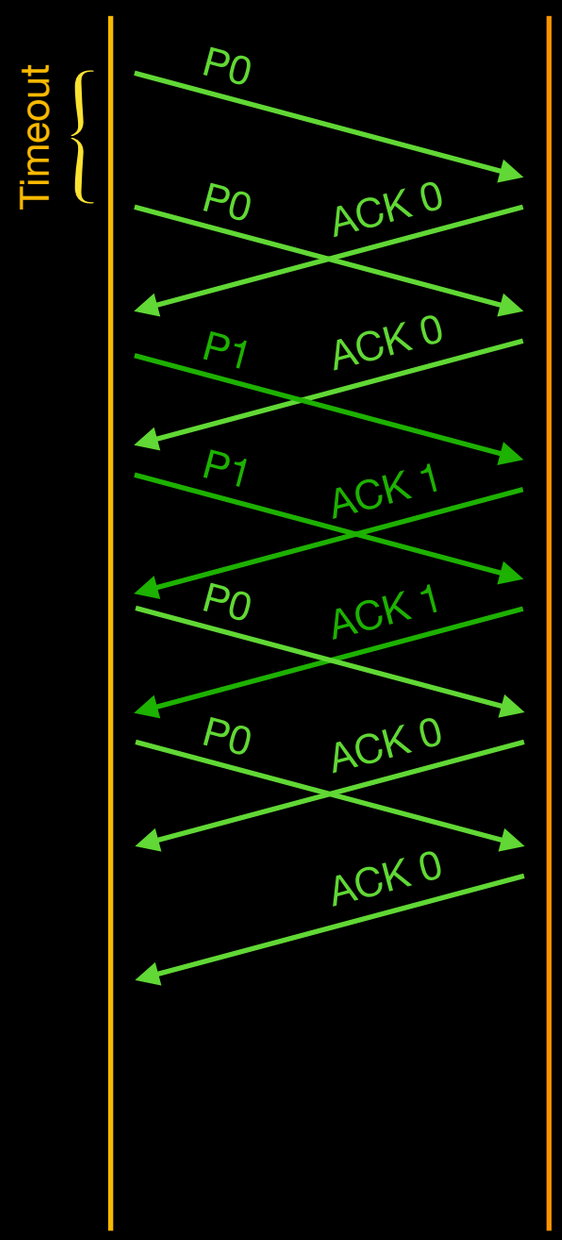
Nominal



Perte paquet



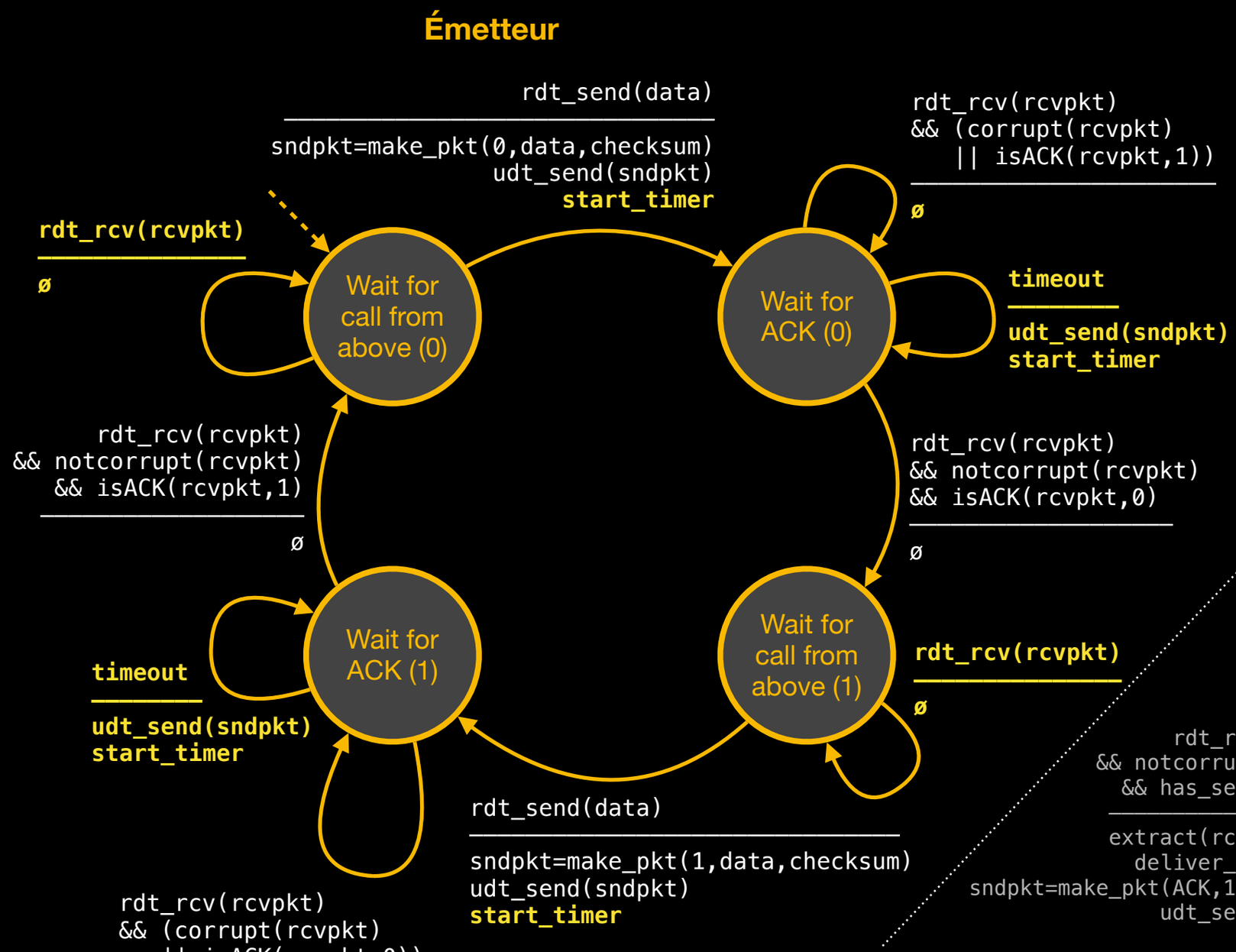
Perte ACK

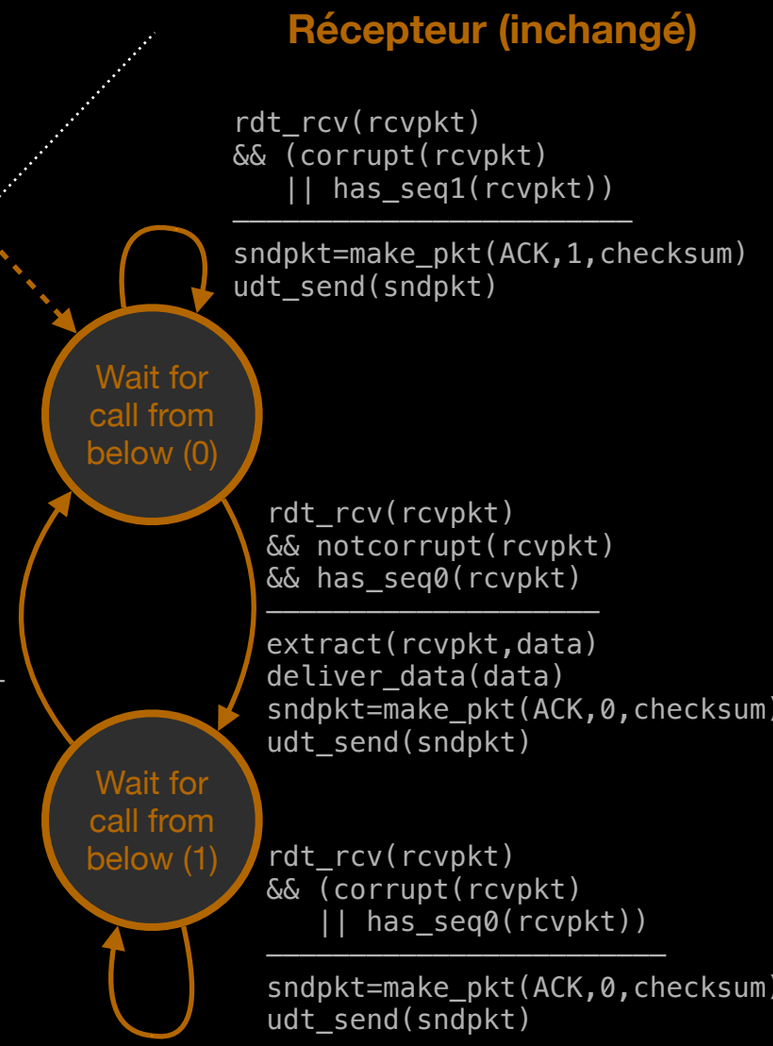


Timer trop court

Résumé du 1-bit protocol

Aka alternating-bit protocol

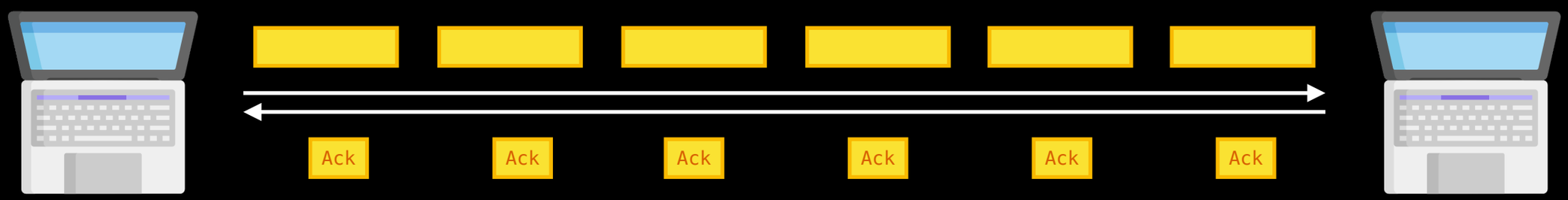




Limites du 1-bit protocol

Simple but not performant

Latence ? Bande Passante ?

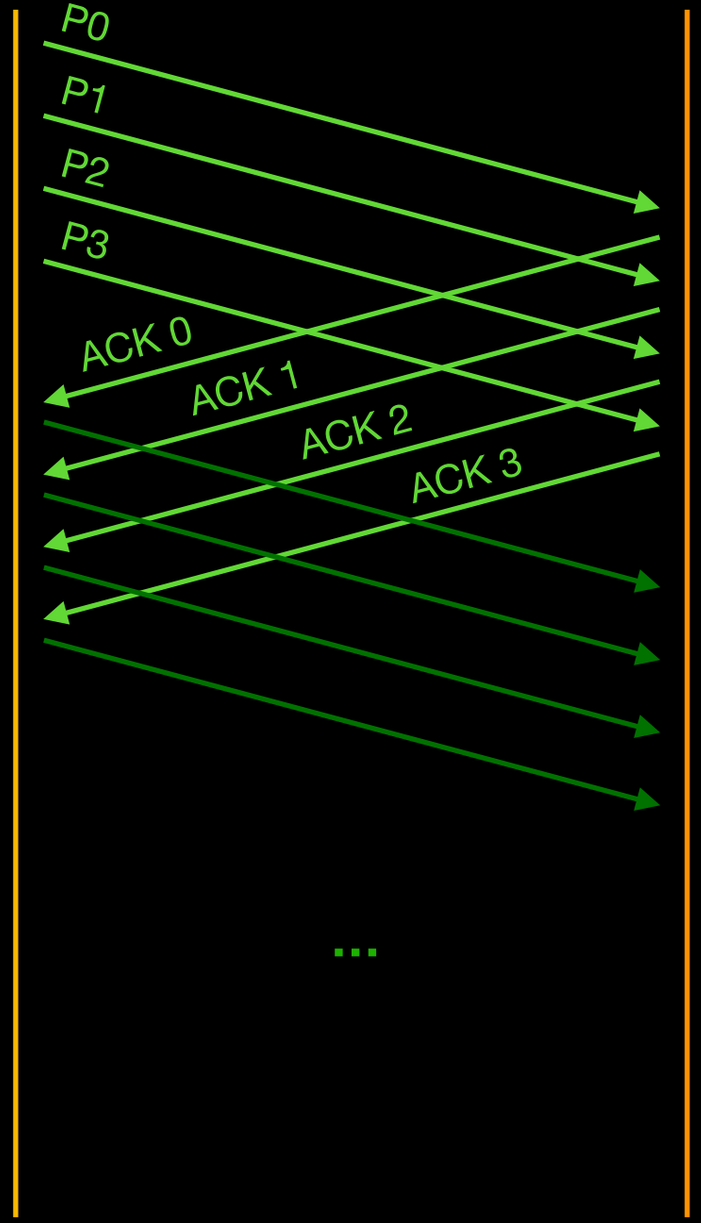


On suppose qu'il faut 100ms pour aller d'une machine à l'autre.
Par ailleurs, en bas, on suppose qu'un a 6 paquets et 6 ACKs en transit.

Window protocols

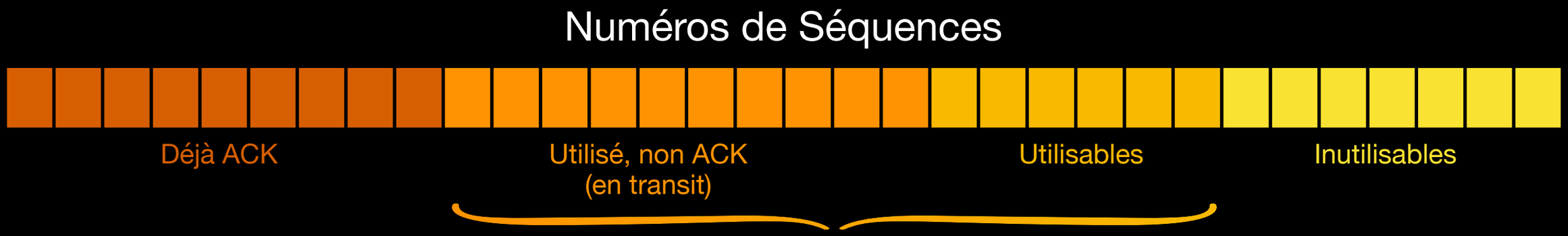
Augmenter le nombre de paquets en vol

- Il faut plus de numéro de séquence
  Identification unique des paquet distincts en transit
- Émetteur et récepteur peuvent avoir à mémoriser plusieurs paquets
  A minima l'émetteur doit garder les paquets dont la réception n'a pas été confirmée (unacknowledged)
- Deux approches générales possibles pour gérer les paquets perdus :
  - Go-Back-N
  - Selective retransmit



Go-Back-N

Revenir en arrière au premier paquet manquant



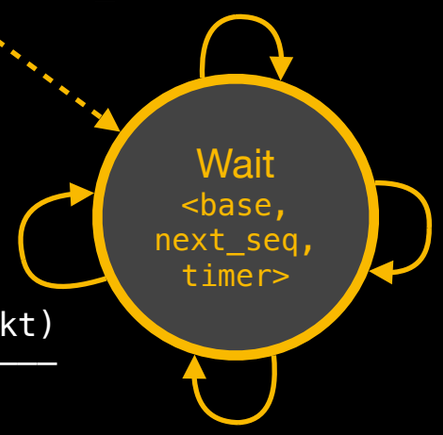

```
rdt_send(data)
_____________________________
if (next_seq < base + N) {
  sndpkt[next_seq] = make_pkt(next_seq, data, checksum)
  udt_send(sndpkt[next_seq])
  if (base == next_seq)
    start_timer
  next_seq++
} else
  refuse_data(data)
```

```
ø
_____________
base = 1
next_seq = 1
stop_timer
```

```
rdt_rcv(rcvpkt)
&& corrupt(rcvpkt)
_____________________
ø
```

```
timeout
_____________________________________
start_timer
udt_send(sndpkt[base])
udt_send(sndpkt[base+1])
...
udt_send(sndpkt[next_seq-1])
```

```
rdt_rcv(rcvpkt) && notcorrupt(rcvpkt)
_____________________________________
base = getacknum(rcvpkt) + 1
if (base == next_seq)
  stop_timer
else
  start_timer
```

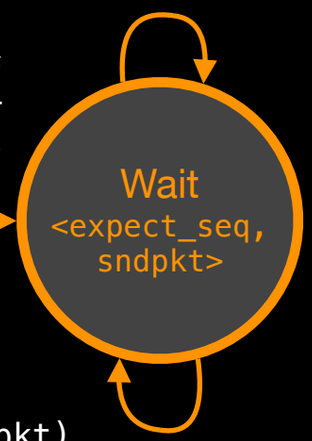


Émetteur (Gen FSM)


```
rdt_rcv(rcvpkt)
&& notcorrupt(rcvpkt)
&& hasseqnum(rcvpkt, expect_seq)
_____________________________________
extract(rcvpkt,data)
deliver_data(data)
sndpkt = make_pkt(expect_seq, ACK, checksum)
udt_send(sndpkt)
expect_seq++
```

```
                                  ø
                  _____________________
                          expect_seq=1
sndpkt=make_pkt(0,ACK,checksum)
```

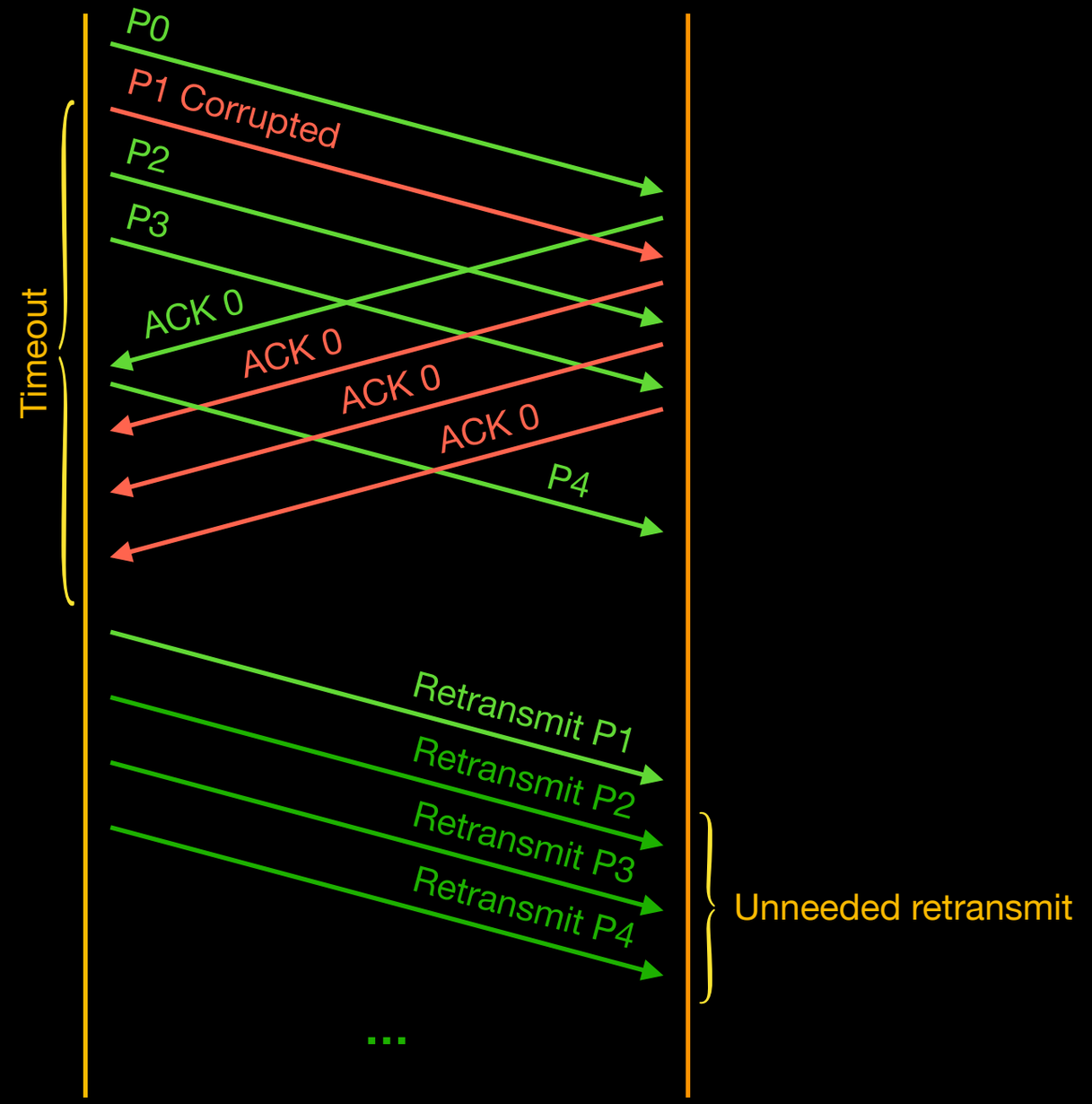
```
rdt_rcv(rcvpkt)
&& !(notcorrupt(rcvpkt)
     && hasseqnum(rcvpkt, expect_seq))
_____________________________________
udt_send(sndpkt)
```



Récepteur (GEN FSM)

Le problème avec GBN

With large windows and delays, costly retransmissions of N



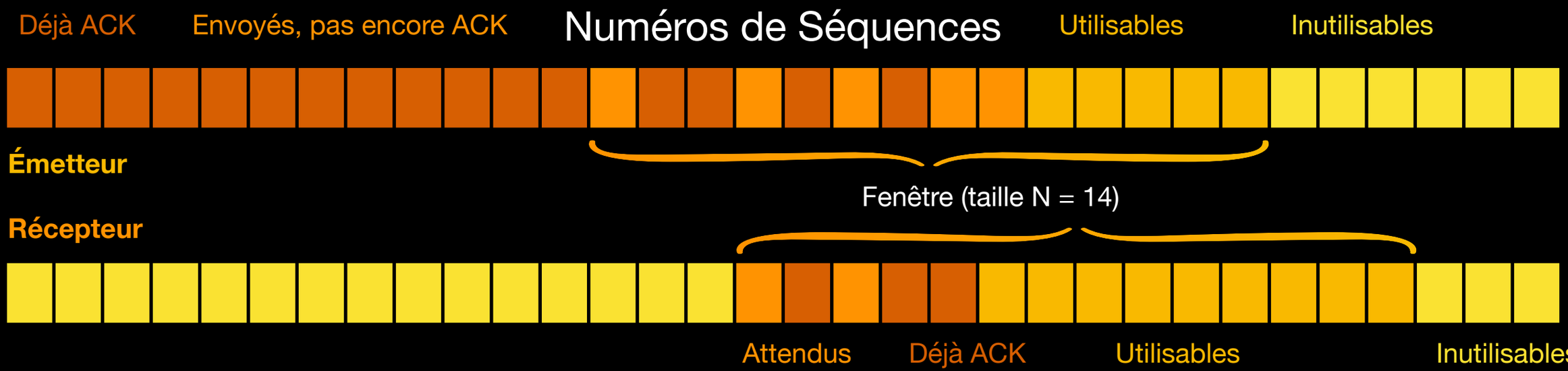
SR et la taille de la fenêtre

Idée : Envoyer des ACKs par paquets

- Timeout par Paquet

SR et la taille de la fenêtre

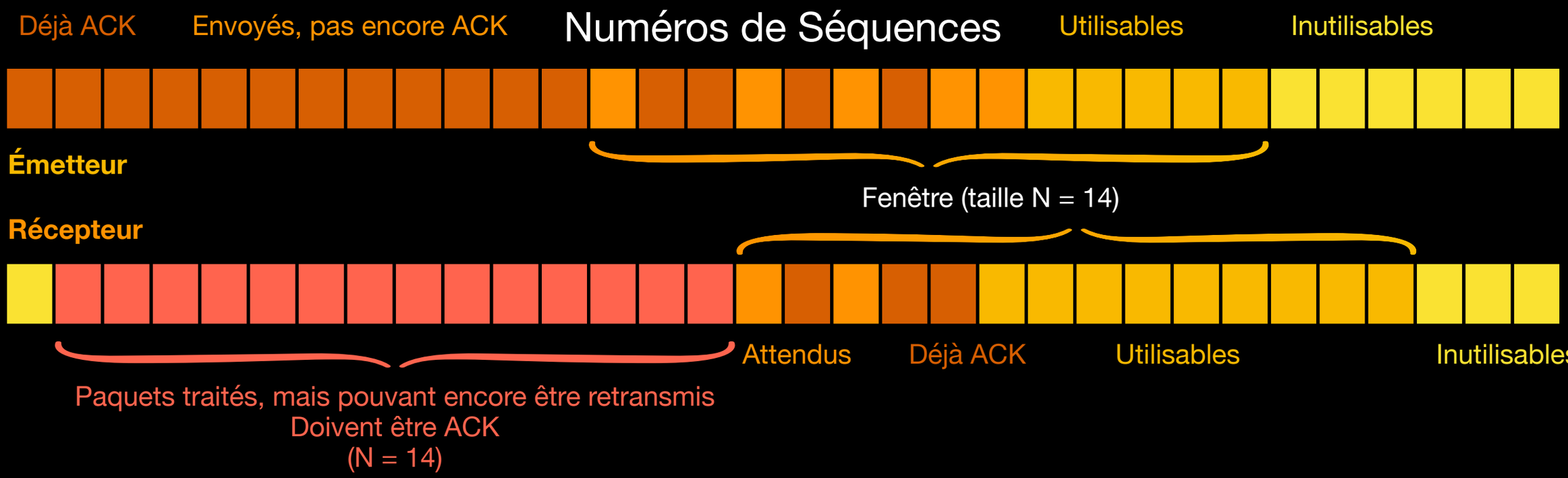
Idée : Envoyer des ACKs par paquets



- Timeout par Paquet
- Fenêtres émetteur et récepteur distinctes

SR et la taille de la fenêtre

Idée : Envoyer des ACKs par paquets

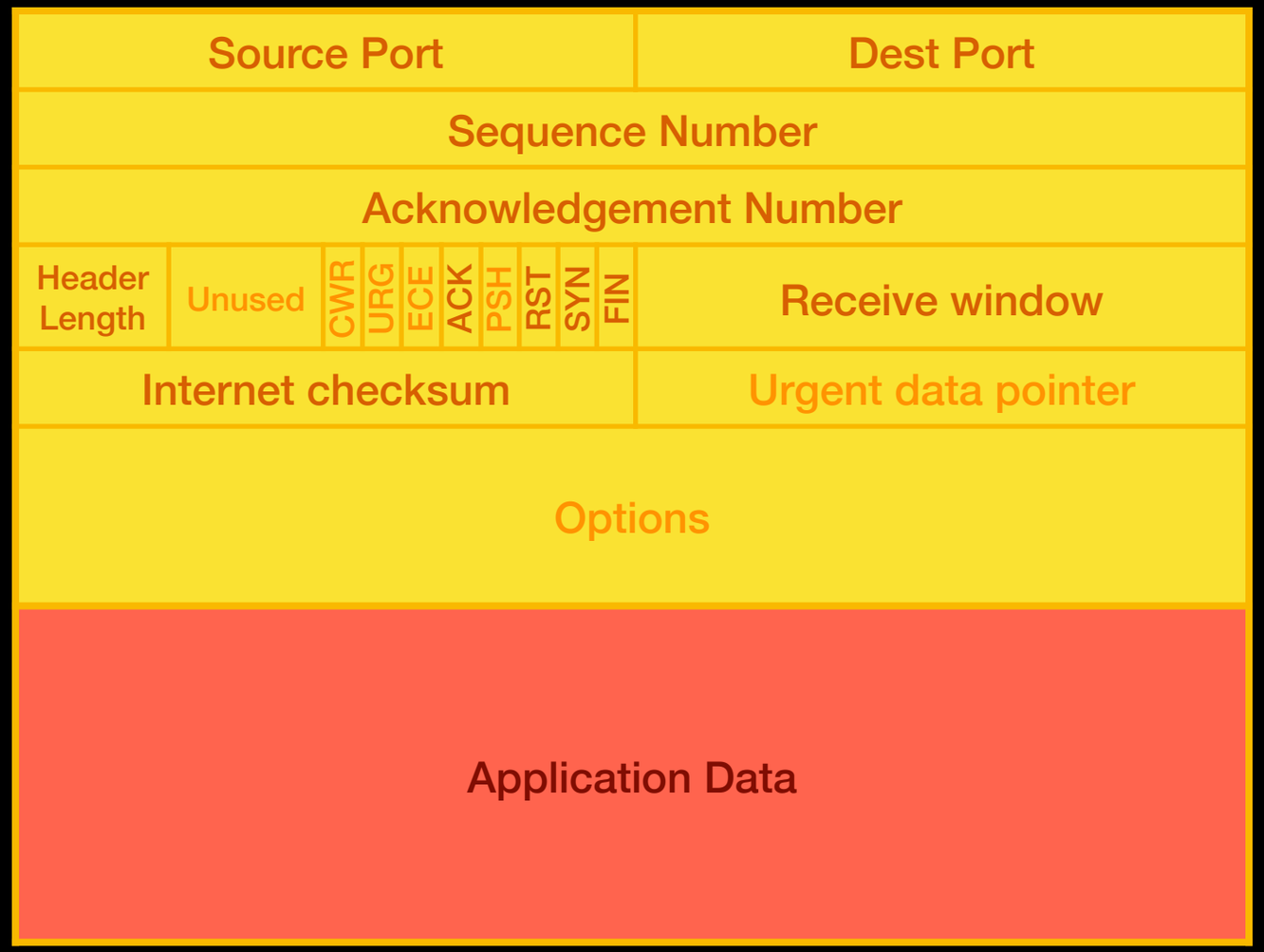


- Timeout par Paquet
- Fenêtres émetteur et récepteur distinctes
- Le récepteur doit continuer d'ACK des paquets déjà traités

N doit être <= 1/2 de l'espace de Seqnums

Même question pour Go-back-N ?

TCP:
the Transmission Control Protocol



RDT naïf vs TCP

Ce qui distingue TCP d'un protocol naïf

- TCP est bi-directionnel
- Multiplexage
- ACK cumulatifs en octets, mais permet une retransmission selective
- Estimation dynamique du RTT
- Gestion dynamique de la taille de fenêtre
- … voir RFC 9293 (remplace la RFC 793 et ses compléments et errata)

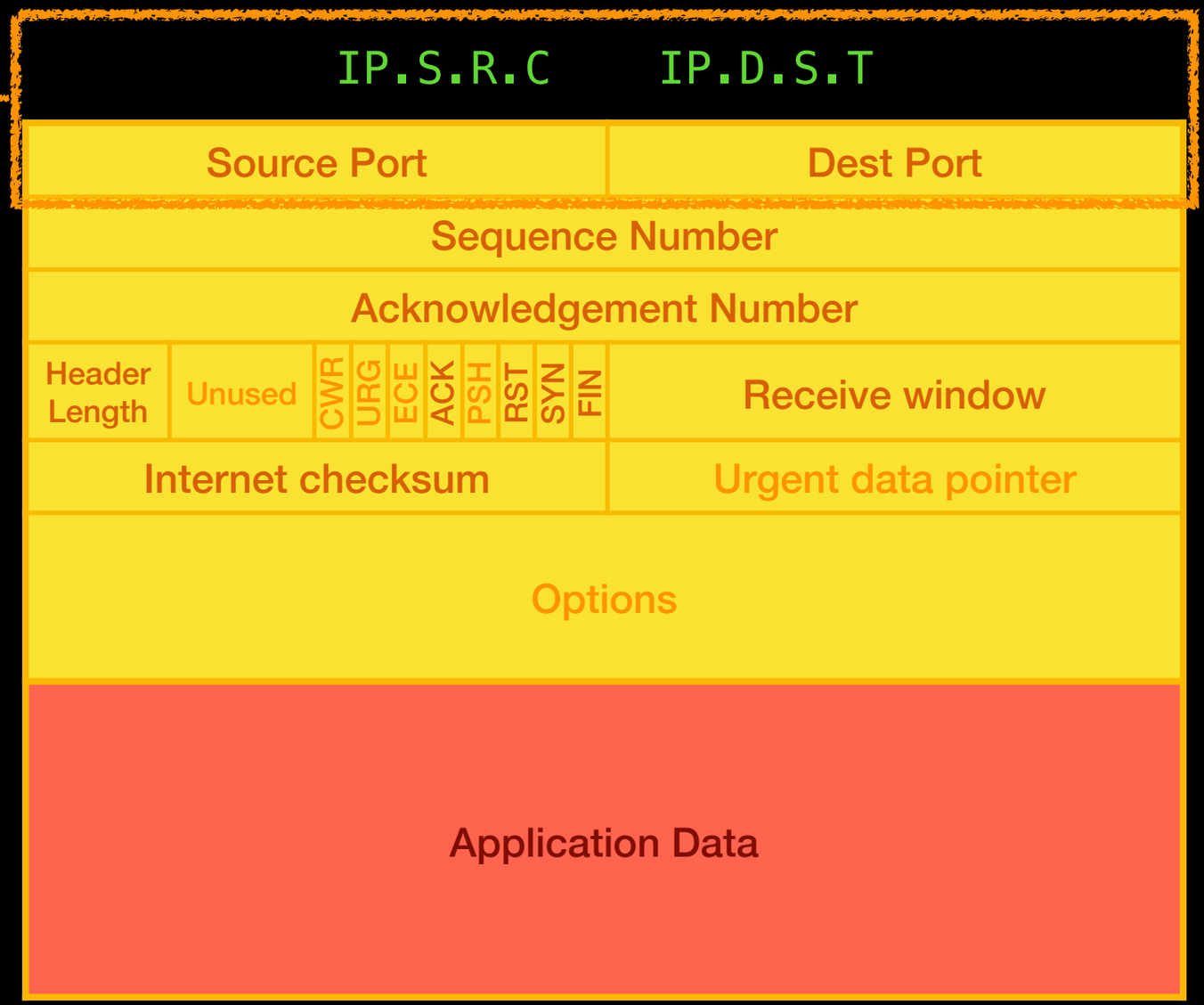
Les idées sont les mêmes,
mais les détails sont adaptés au réel

Le démultiplexage en TCP

Même idée, mais plus de champs

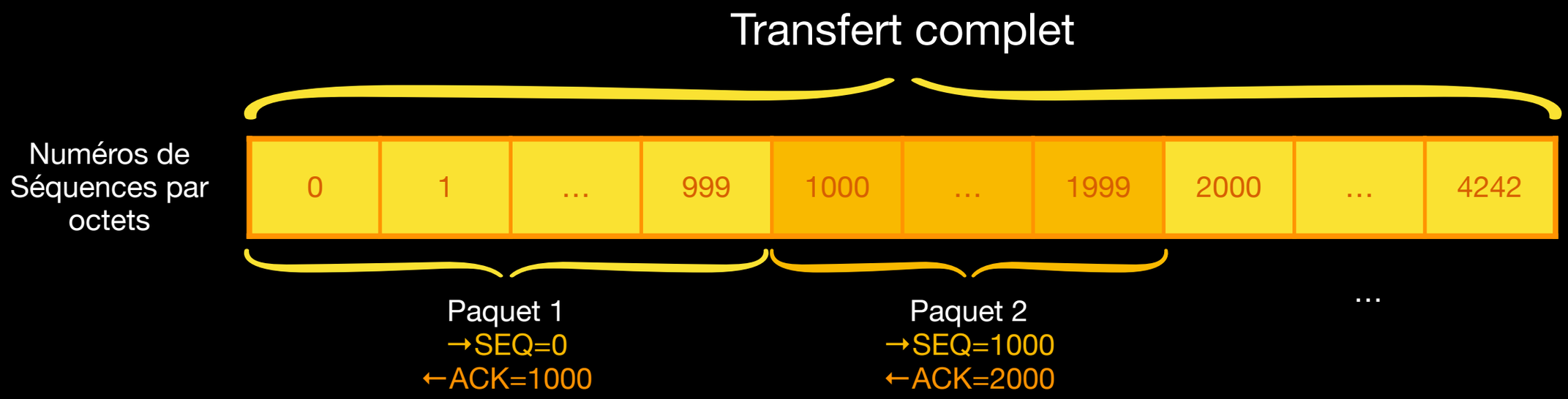
Identifie une socket connectée.

Une socket listen reçoit uniquement les demandes de connections (SYN), en filtrant sur le port destination et parfois l'IP destination.



ACK et retransmissions TCP

On raisonne en octet, dans chaque sens

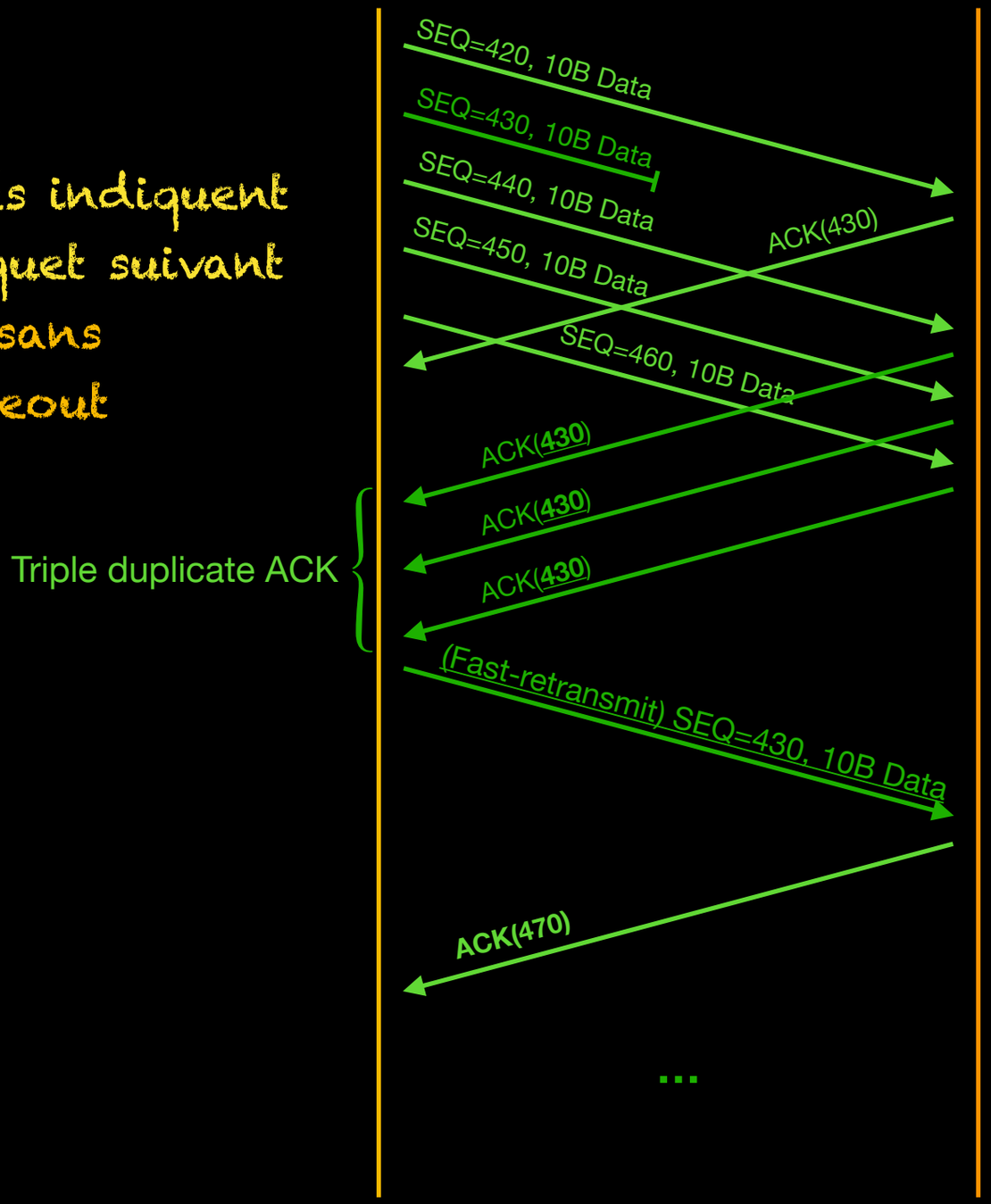


- Le Numéro de Séquence correspond au premier octet du paquet
- On ACK en indiquant le numéro du prochain octet attendu (ACK = SEQ+Length)
- Chaque paquet peut contenir données et ACK du sens opposé
- On retransmet le premier paquet non-ACK lors du timeout

Retransmission selective TCP

Fast Retransmit

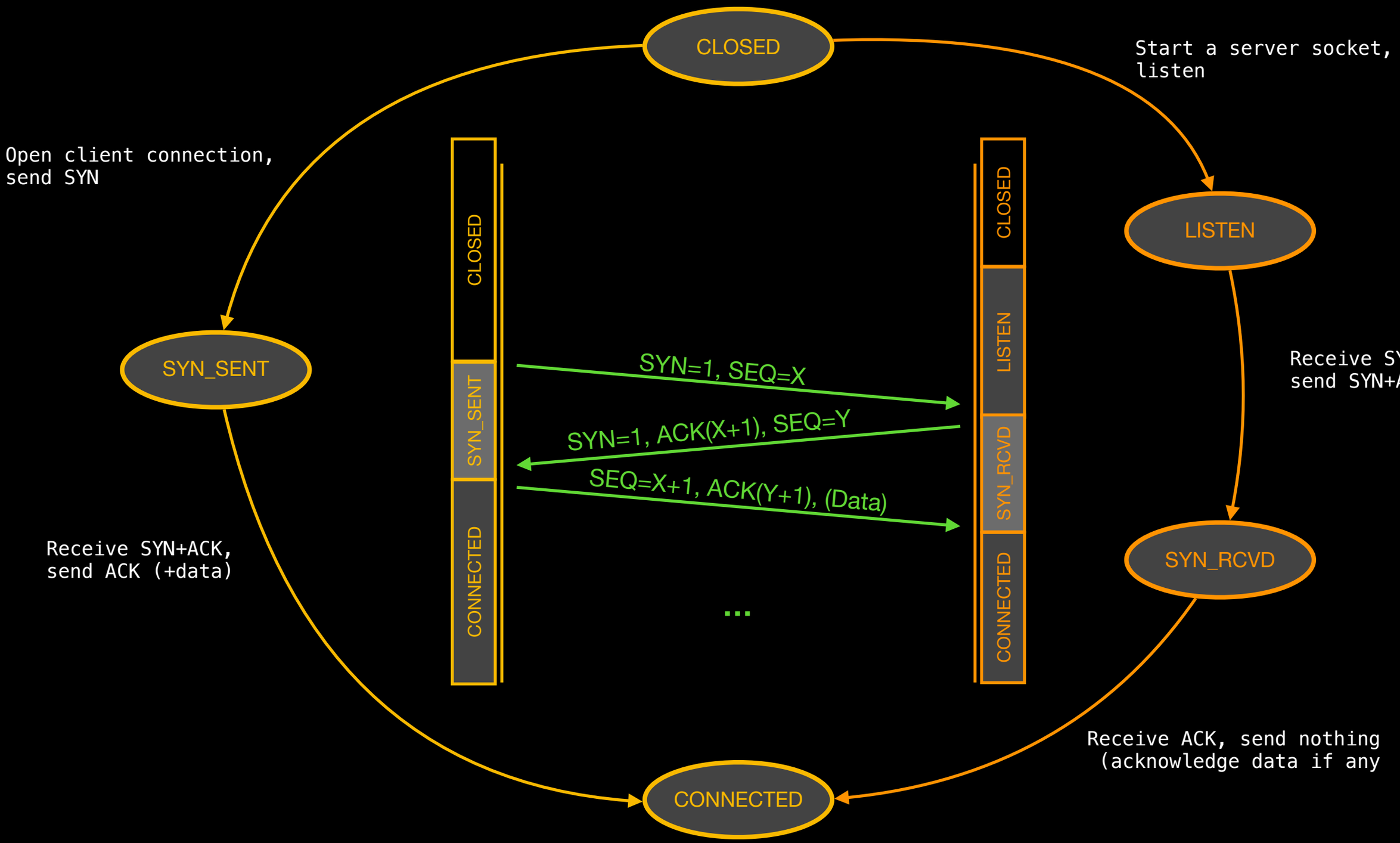
Idée de Base:
3 ACK dupliqués indiquent
la perte du paquet suivant
On retransmet sans
attendre le timeout



NB: We do not represent the data in the other direction

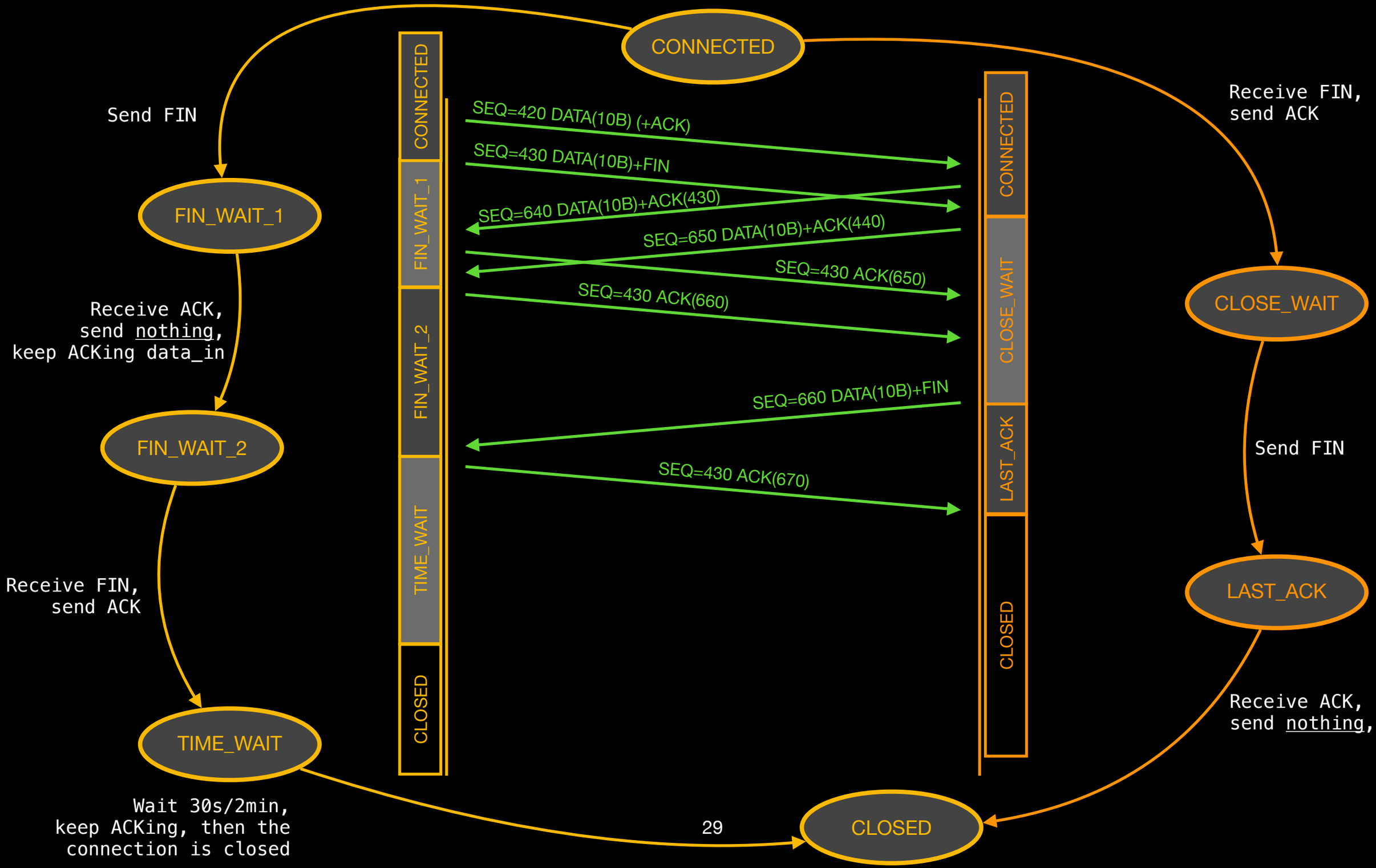
Ouvrir une connection TCP

SYN - SYN+ACK - ACK



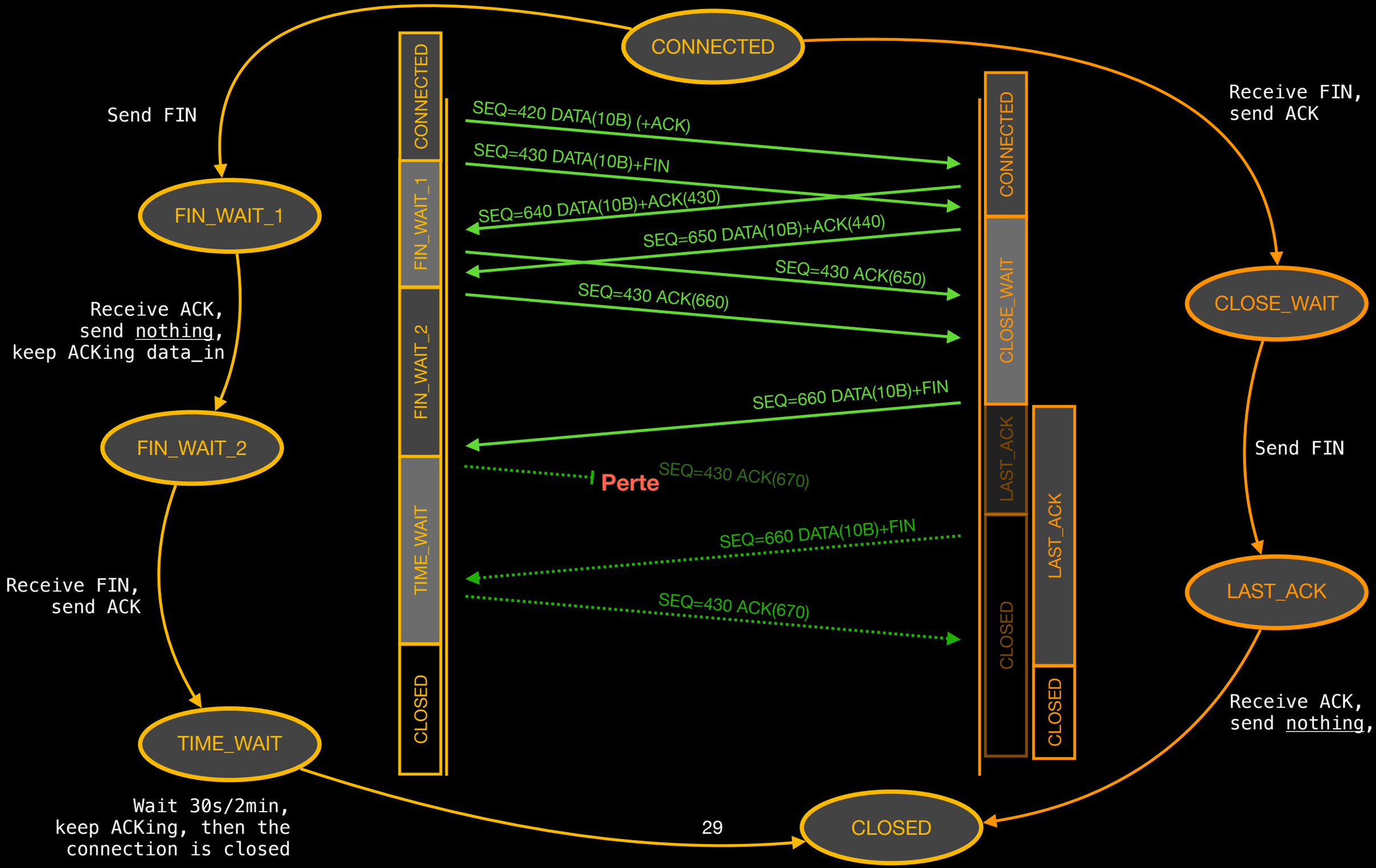
Fermer une connection TCP

Il y a 2 sens de communications



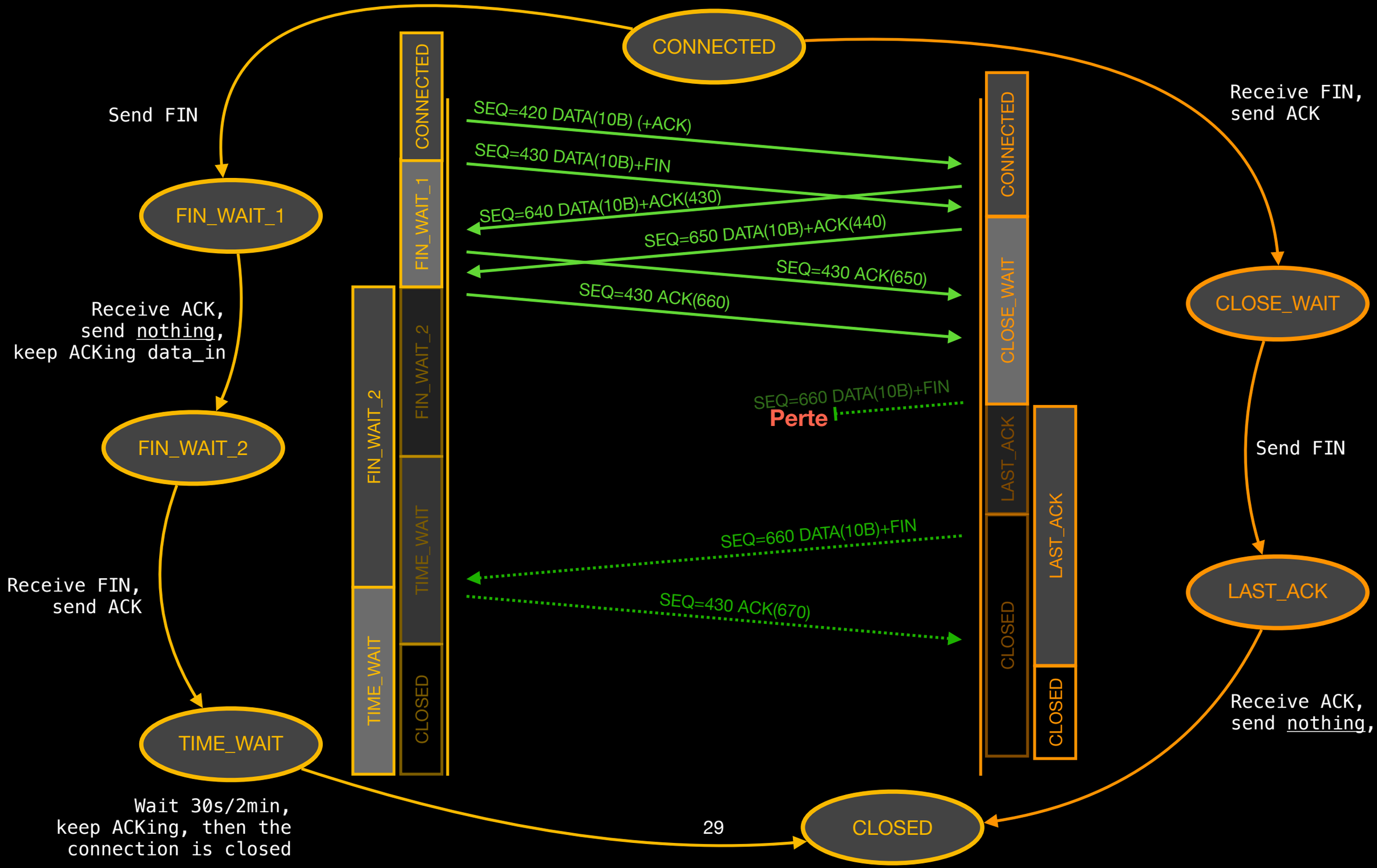
Fermer une connection TCP

Il y a 2 sens de communications



Fermer une connection TCP

Il y a 2 sens de communications



RST

Gestion d’erreurs

Ce numéro n’est pas attribué

Exemples :

- Pas de socket sur ce port
- SYN=1, mais listen queue pleine
- SYN=0, port valide mas Pas de connection ouverte
- SYN=1, Connection déjà ouverte pour ces paramètres
- Abort
- Paquet hors fenêtre en dehors d’un état synchronisé
  (autrement en dehors de ESTABLISHED, FIN-WAIT-1, FIN-WAIT-2, CLOSE-WAIT, CLOSING, LAST-ACK, TIME-WAIT)
  (En état synchronisé, envoyer un ACK vide avec les Sequence Numbers attendus)

RST illégitime

Il faut se protéger des petits malins

Méthode Linux :

- RST=1 et paquet hors fenêtre => DROP en silence
- RST=1 et les Sequence Numbers exacts => on obéit
- RST=1, Sequence Number acceptable mais inexacts
  (RCV.NXT < SEG.SEQ < RCV.NXT+RCV.WND)
  => Envoyer un “Challenge”ACK
  (SEQ=SND.NXT; ACK=RCV.NXT; ACK=1)

Flow Control

My buffer is full - stop sending please

Émetteur **Récepteur**

ACK=1, rwnd=<free space>

Buffer Réception

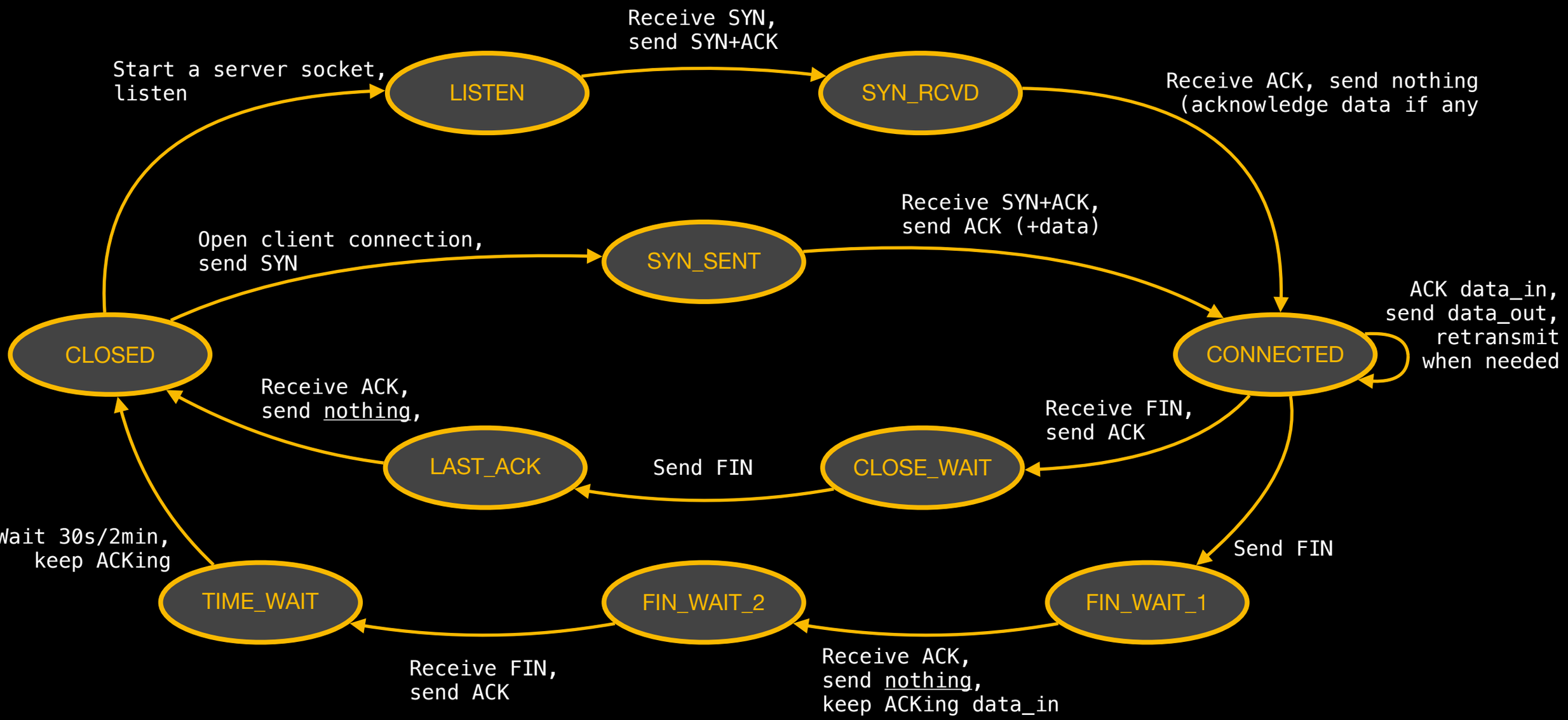
Is only allowed to send
rwnd un-acknoweldged bytes.

=> The in-flight packets
cannot overflow the buffer

The receiver manages the send-rate

TCP en bref

Machine à état de TCP (simplifiée)



Résumé du jour

Qu’avons nous vu aujourd’hui

Est-ce que vos notes ont tout ça ?

- Protocoles de Transport
  - Rôle
  - Rappelle des sémantiques de TCP et UDP
  - La couche IP au dessus de laquelle ils sont
- Implémentation de UDP
  - Demultiplexage
  - Détection d’erreur
- Transport Fiable de données
  - Construction du alternating bit protocol
  - Protocoles à Fenêtre
- TCP
  - Ouverture (3-way handshake), et fermeture.
  - ACK cumulative + rapid retransmit
  - Flow Control

Problèmes restant

On a pas fini TCP - la suite la semaine prochaine

S’adapter aux conditions du réseau :

- Comment on gère le délais de timeout si latence varie
- Comment on gère la taille de fenêtre pour partager la bande-passante
- Comment on s’adapte aux variations de congestion
- Comment être un bon citoyen et ne pas saturer les réseaux ?

Gestion de la Congestion

Et ensuite, parlons un peu de la couche réseau IP :

- Comment faire passer un message à l’autre bout du monde ?