CHAPITRE 3

LA COUCHE TRANSPORT

Introduction (1)

Plan :

- **Intro**.
- UDP
- Transfert fiable
- TCP

- La couche transport
  - Permet à **deux applications** de dialoguer via le réseau
    - Il est nécessaire de pouvoir **identifier** les applications
    - Il faut identifier l'application émettrice et l'application destinataire de l'information
  - Propose deux types de transport différents
    - fiable: la couche transport assure que les informations arriveront à destination, sans erreurs
    - non fiable: aucune garantie n'est donnée.

Introduction (2)

Plan :

- **Intro**.
- UDP
- Transfert fiable
- TCP

– Elle utilise la couche réseau pour **acheminer l'information sur la machine destination**

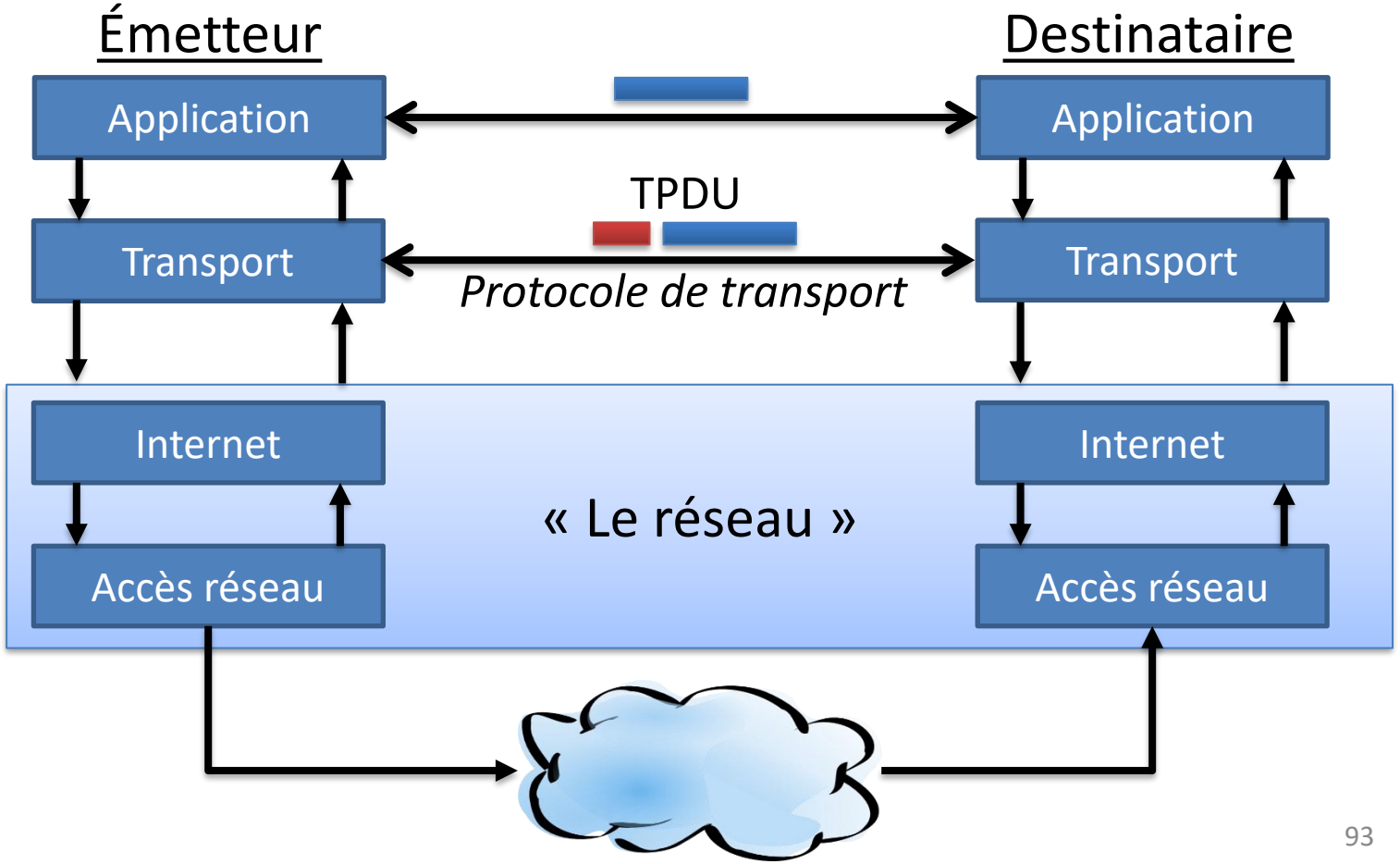
- La couche réseau ne fournit aucune garantie de livraison. Les erreurs suivantes peuvent survenir :
  - L'information peut être altérée
  - L'information peut être perdue
  - L'information peut arriver de manière désordonnée
  - L'information peut arriver dupplіquée
- La couche transport doit parvenir à fournir un service fiable en utilisant une couche réseau non-fiable.

Introduction (3)

Plan :

- **Intro**.
- UDP
- Transfert fiable
- TCP

- Découpe en couche



SWILABUS

Introduction (4)

- Comment identifier une application ?
  - *En utilisant un numéro de port*
- Comment associer un numéro de port à une application ?
  - Les numéros de port « bien connus »
    - Ces numéros de port biens connus sont *réservés* à des applications déterminées (ex: 25=SMTP, …)
    - Il font l'objet d'une standardisation lorsque la description du protocole est publiée.
    - Il « suffit » de retenir ces numéros
      - fichier `services`

Plan :

- **Intro**.
- UDP
- Transfert fiable
- TCP

Introduction (5)

– Si le numéro de port n'est pas connu ou est variable ? Comment faire ?
  -
  -
  -
  -

Plan :

- **Intro**.
- UDP
- Transfert fiable
- TCP

UDP (1)

Plan :

- Intro.
- **UDP**
- Transfert fiable
- TCP

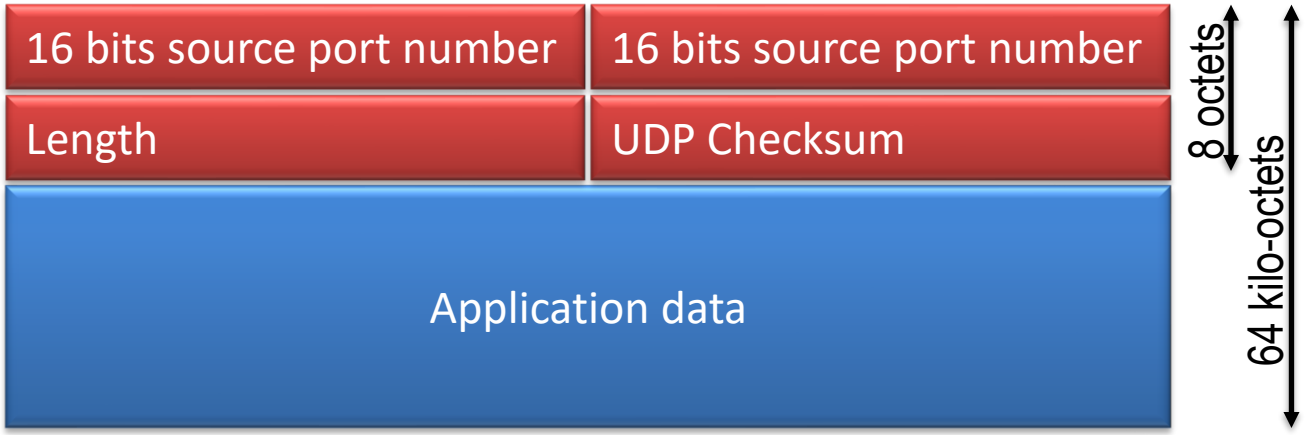
- UDP – User Datagram Protocol
  - Protocole de transport minimaliste
  - Permet l'échange de données sans fournir de garantie
  - Il s'agit d'une couche *presque transparente* au dessus de la couche réseau
    - Ne fournit presque pas de service en plus
    - Identifie les applications émettrices / destinataires
  - Quels sont les élément que **doit** contenir, au minimum, le protocole UDP ?

UDP (2)

Plan :

- Intro.
- **UDP**
- Transfert fiable
- TCP

– Description d'un TPDU UDP



- La taille maximale est définie par la couche réseau : 65 536 octets (64 Ko)
- Le champ checksum peut être rempli et permet de vérifier les données transmises (altération)
  - Si la valeur du champ vaut 0, alors les données ne sont pas vérifiées.

SWILABUS

UDP (3)

- Utilité d'un tel protocole ?
  - Utilisé pour des applications temps réels
    - Ces applications fonctionnent correctement si les délais ne sont pas trop grands
    - Si les informations sont en retards, elles sont considérées comme perdues.
    - Exemple:
      - » Les jeux massivement multi-joueurs
      - » Les diffusions en direct de contenu vidéo/audio
  - Utilisé dans des moments où le réseau n'est pas fiable
    - Certains protocole de signalisation de problèmes ou d'erreur (ex. : SNMP) préfère UDP

Plan :

- Intro.
- **UDP**
- Transfert fiable
- TCP

Transfert fiable (1)

- La couche réseau ne fournit aucune garantie
  - Comment faire face à :
    - Une information perdue ?
    - Une information qui arrive en désordre ?
    - Une information altérée ?
    - Une information dupliquée ?

Plan :

- Intro.
- UDP
- **Transfert fiable**
- TCP

Dans la suite, nous allons étudier comment, malgré une couche réseau non fiable, il est possible de **construire un protocole de transport qui garantit l'arrivée des informations**.

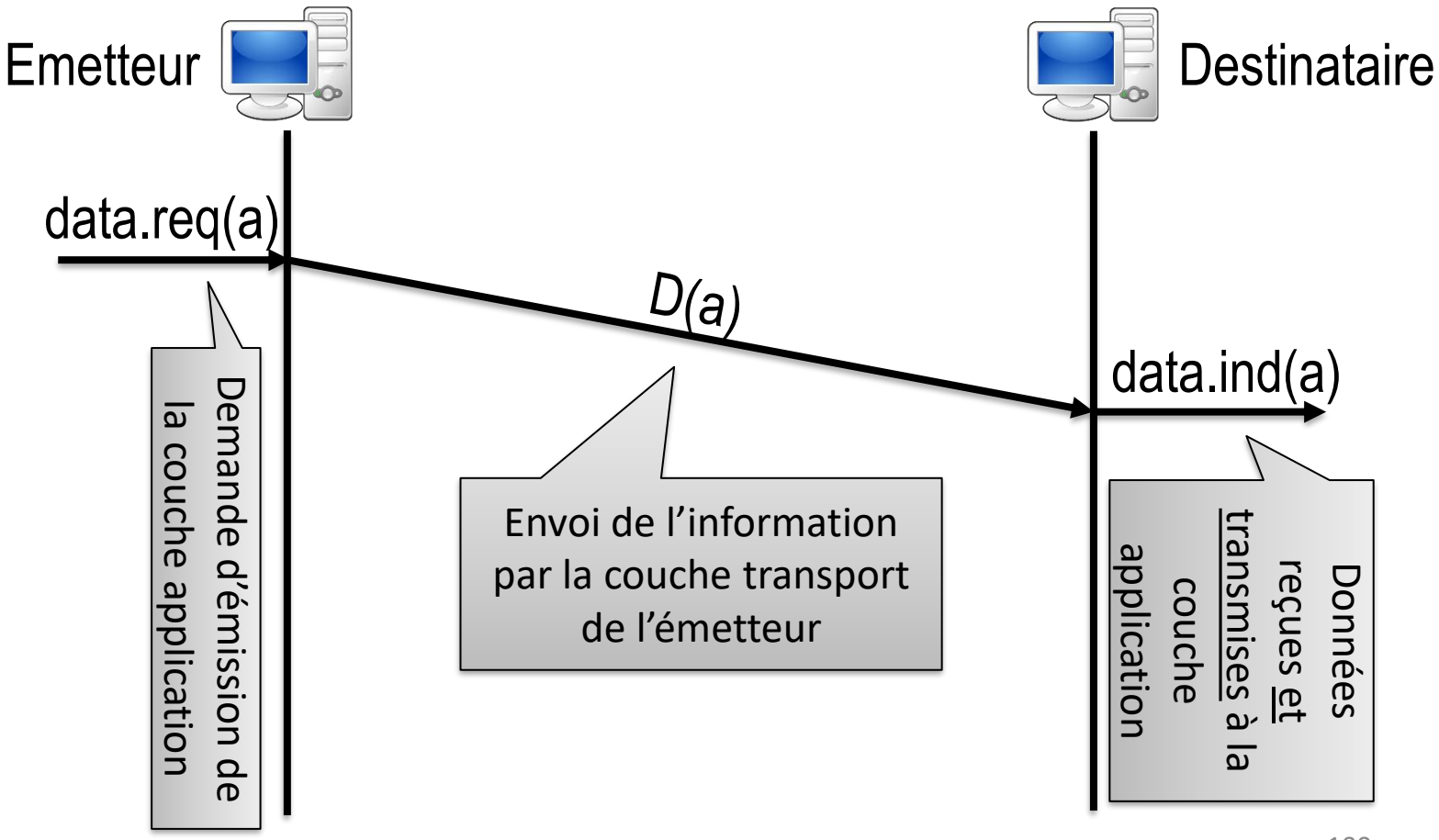
Les choix peuvent différer de TCP, le protocole de transport fiable standardisé, que nous étudierons par la suite.

Transfert fiable (2)

– Conventions

Plan :

- Intro.
- UDP
- **Transfert fiable**
- TCP



SWILABUS

Transfert fiable (3)

Plan :

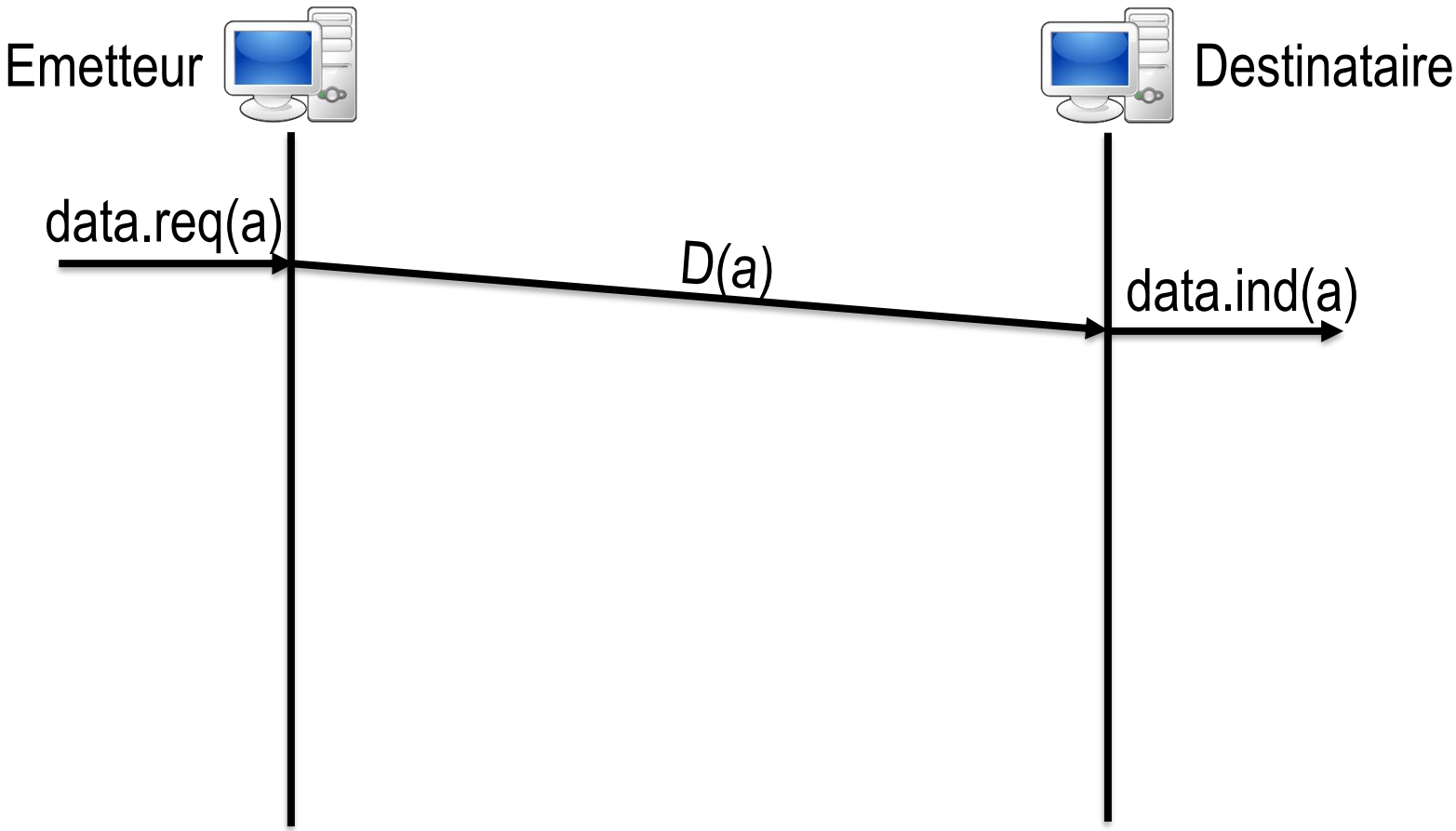
- Intro.
- UDP
- **Transfert fiable**
- TCP

– Problème 1

- Comment être sûr que l'information est arrivée à destination ?
- Comment être sûr que l'on ne submerge pas le récepteur en lui envoyant trop d'information ?

Transfert fiable (4)

- Solution 1a



Plan :

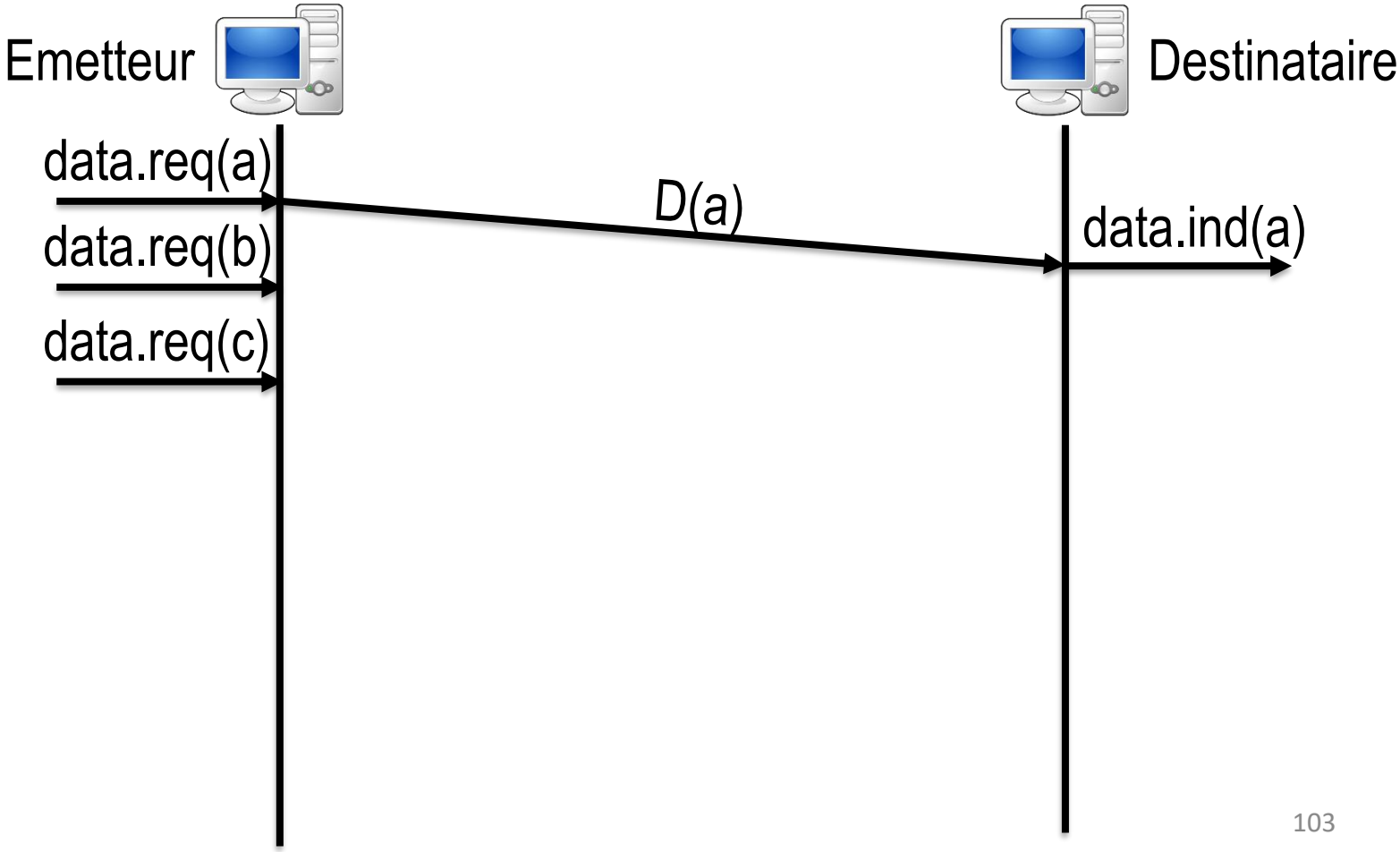
- Intro.
- UDP
- **Transfert fiable**
- TCP

Transfert fiable (5)

- Solution 1b

Plan :

- Intro.
- UDP
- **Transfert fiable**
- TCP



SWILABUS

Transfert fiable (6)

Plan :

- Intro.
- UDP
- **Transfert fiable**
- TCP

– Problème 2

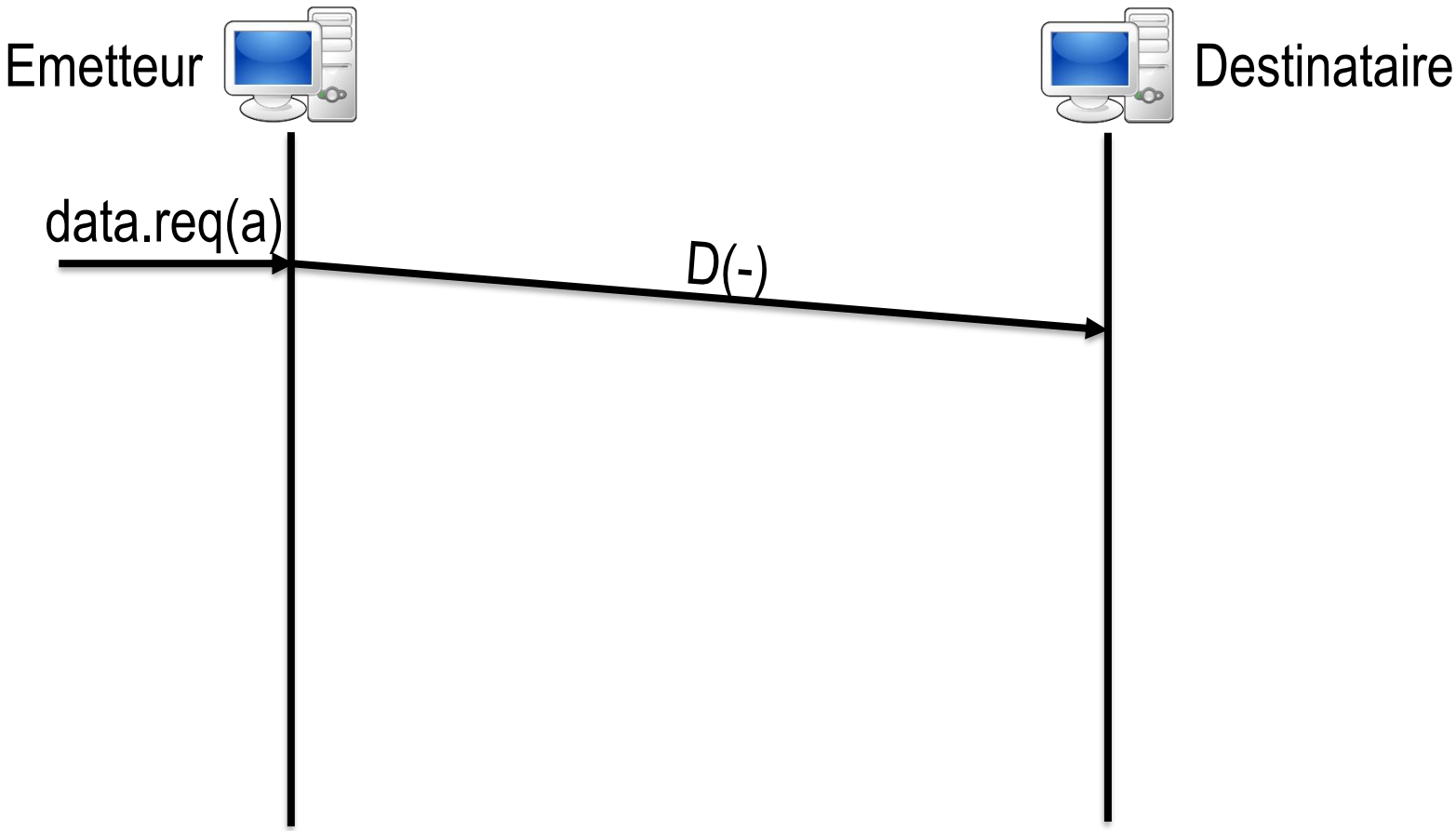
- Comment savoir si l'information est arrivée correctement à destination (non modifiée) ?
  - La couche réseau n'effectue aucune vérification sur le contenu des données émises
  - La couche physique peut être imparfaite et *produire* des erreurs de transmission
  - Comment les détecter ?
    - » Parité ?
    - » Checksum ?
      - Comptage de tous les octets envoyés. Cette valeur accompagne l'information émise et sert de vérification

Transfert fiable (7)

- Solution 2

Plan :

- Intro.
- UDP
- **Transfert fiable**
- TCP



Transfert fiable (8)

- Problème 3
  - Comment faire face aux pertes d'information ?
    - Rien n'assure que l'information « ne va pas se perdre en route ».
    - Comment faire puisque la destination n'est pas au courant qu'une information lui a été envoyée ?

Plan :

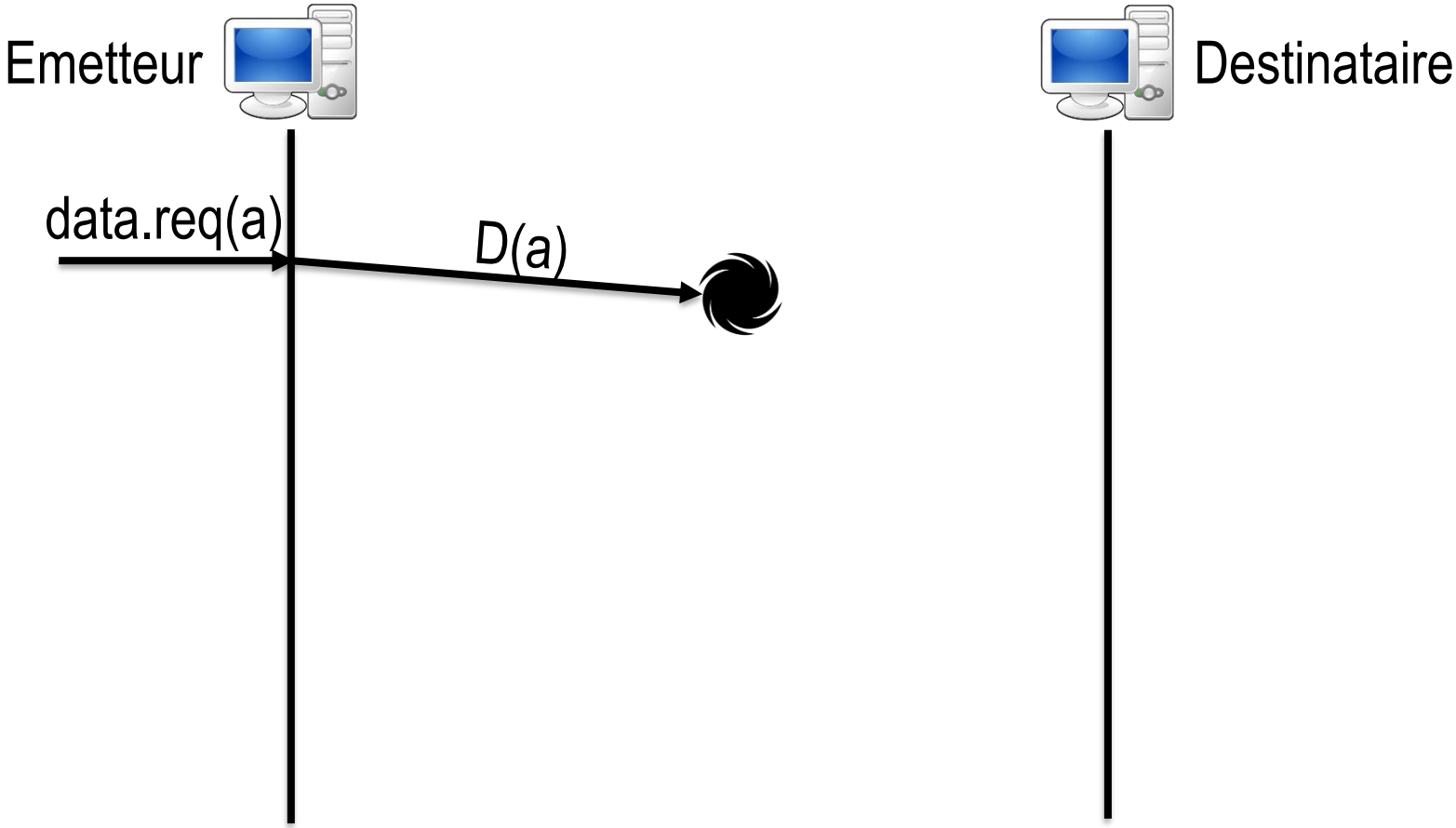
- Intro.
- UDP
- **Transfert fiable**
- TCP

Transfert fiable (9)

- Solution 3 (perte d'une donnée)

Plan :

- Intro.
- UDP
- **Transfert fiable**
- TCP

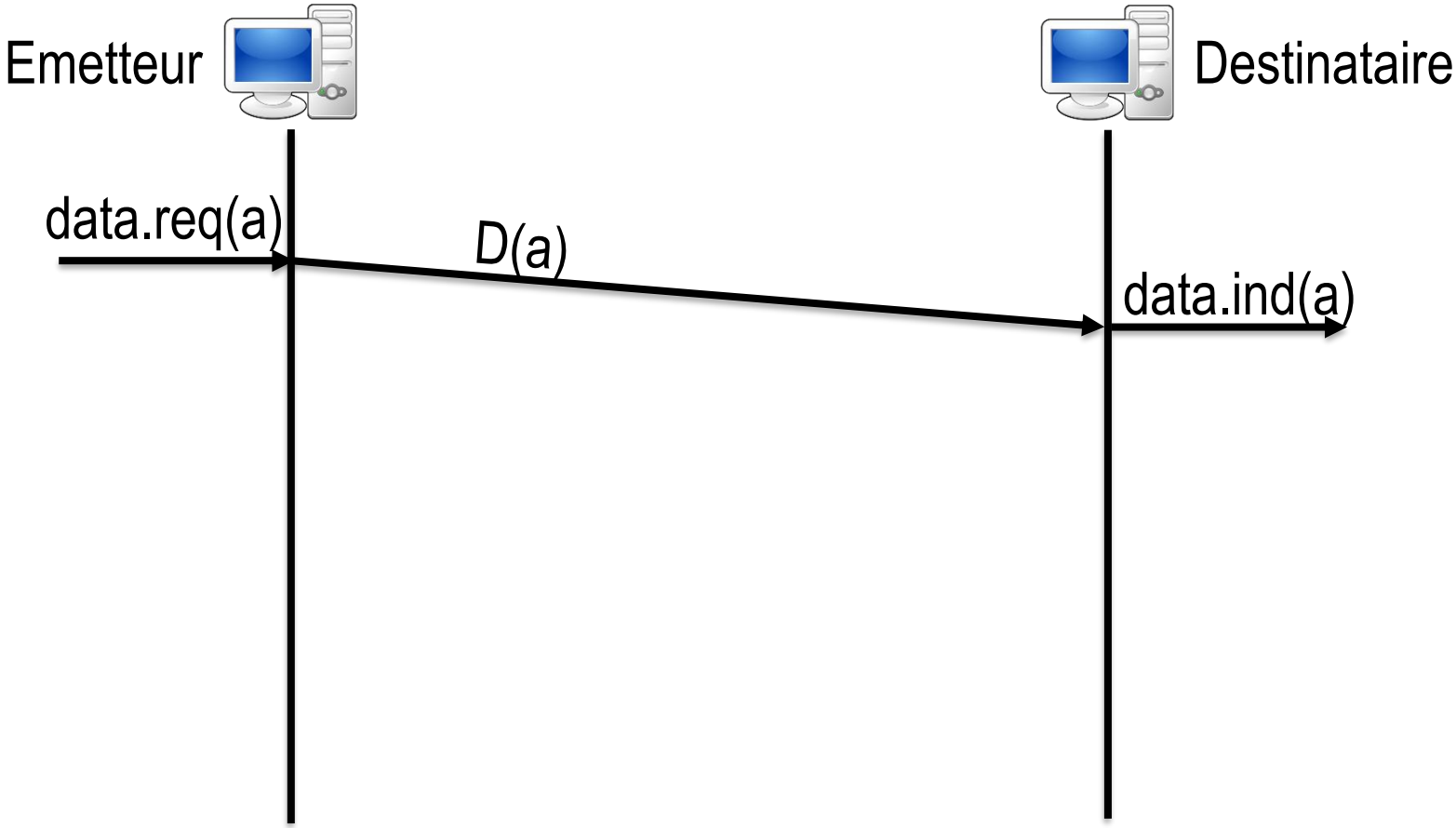


Transfert fiable (10)

– Problème 4 : perte de l'acquit

Plan :

- Intro.
- UDP
- **Transfert fiable**
- TCP

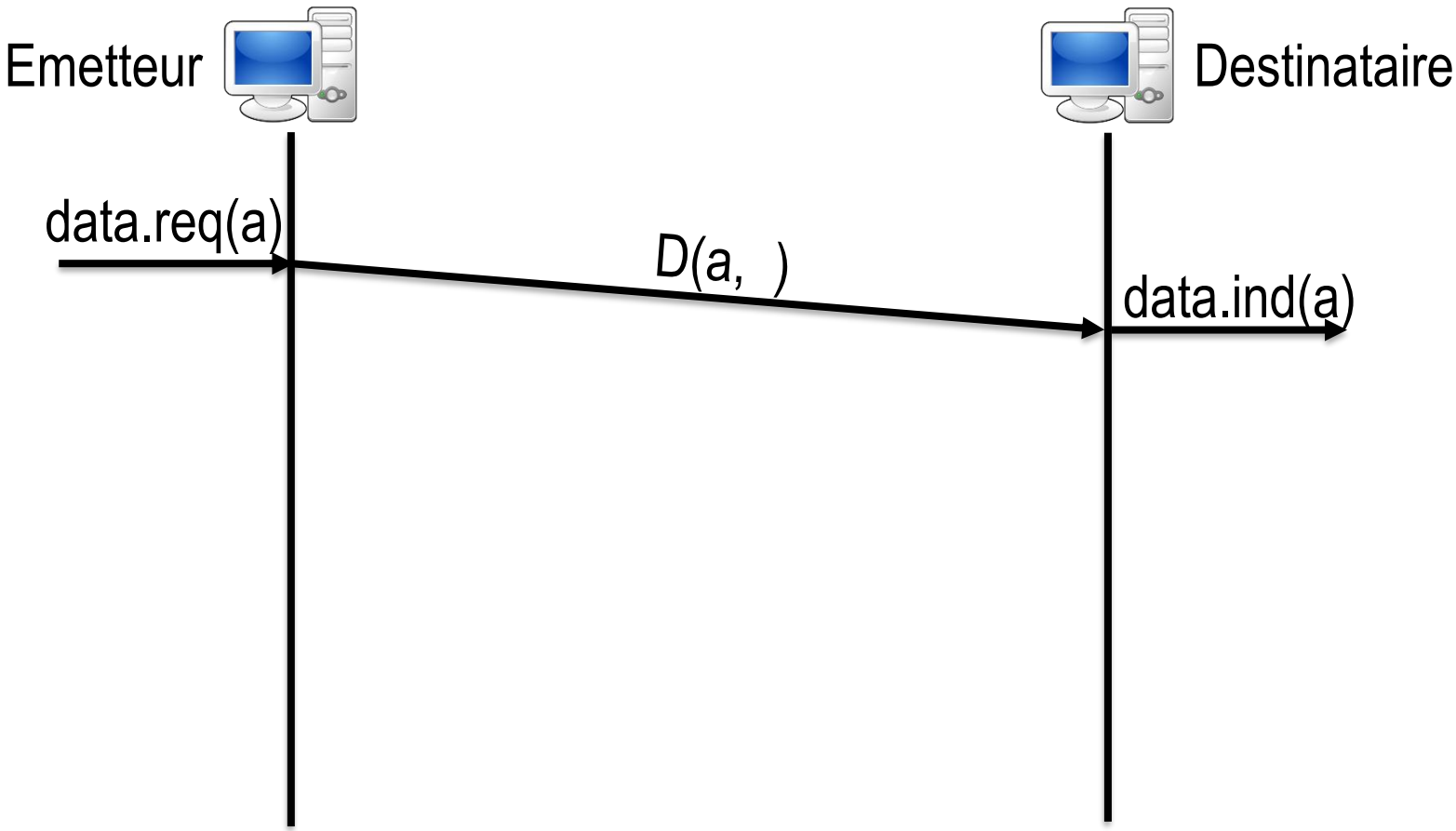


Transfert fiable (11)

- Solution 4 (perte de l'aquit)

Plan :

- Intro.
- UDP
- **Transfert fiable**
- TCP



Transfert fiable (12)

Plan :

- Intro.
- UDP
- **Transfert fiable**
- TCP

– Problème 5

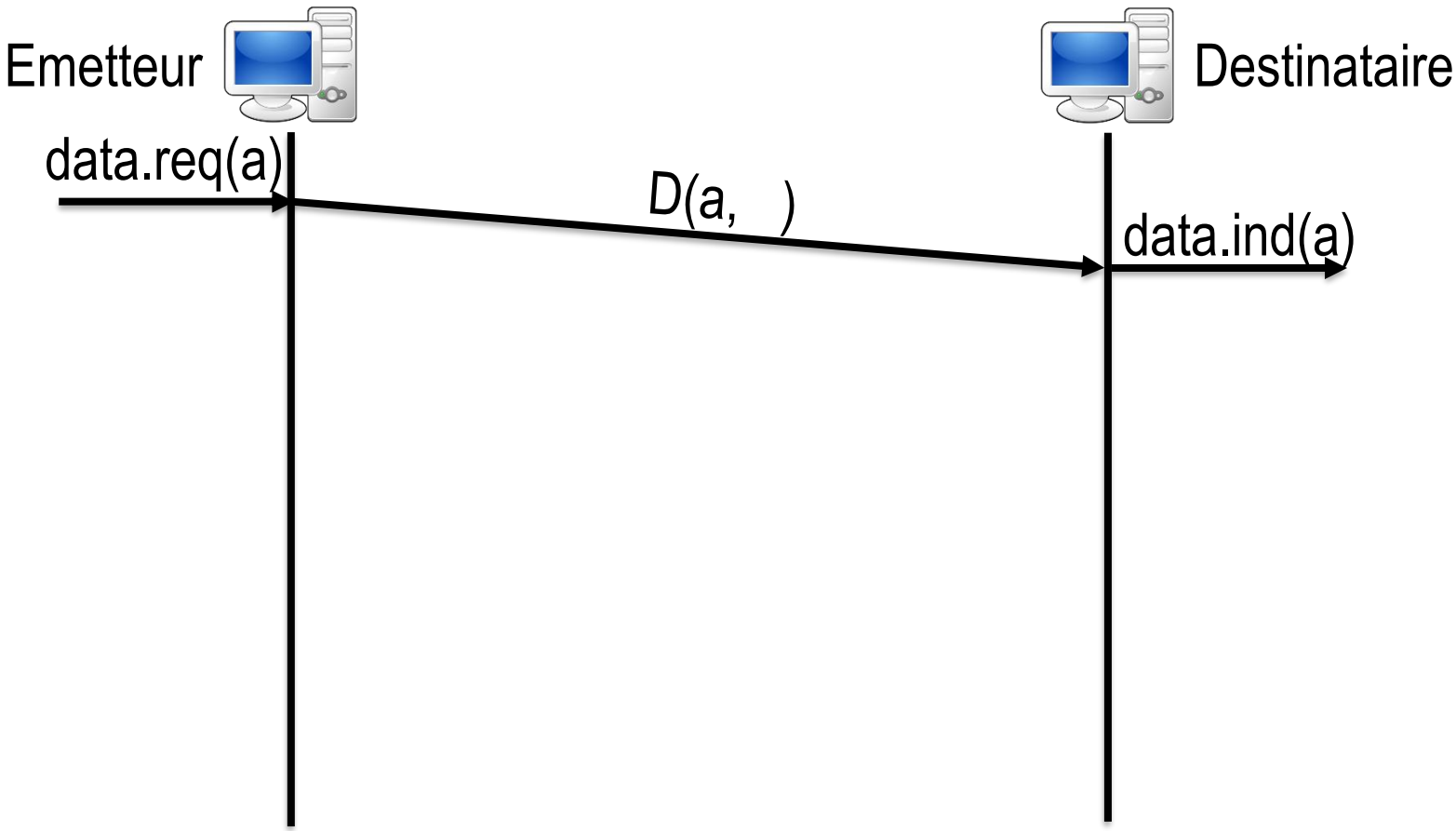
- Comment faire face aux duplications d'information ?
  - Il arrive que des informations arrivent « en double » sur le réseau
    - » Certains équippements doivent, parfois, recopier de l'information
    - » Durant un laps de temps très court, une information peut boucler, être mal orientée, …
  - Comment y faire face ?

Transfert fiable (13)

- Illustration problème 5

Plan :

- Intro.
- UDP
- **Transfert fiable**
- TCP



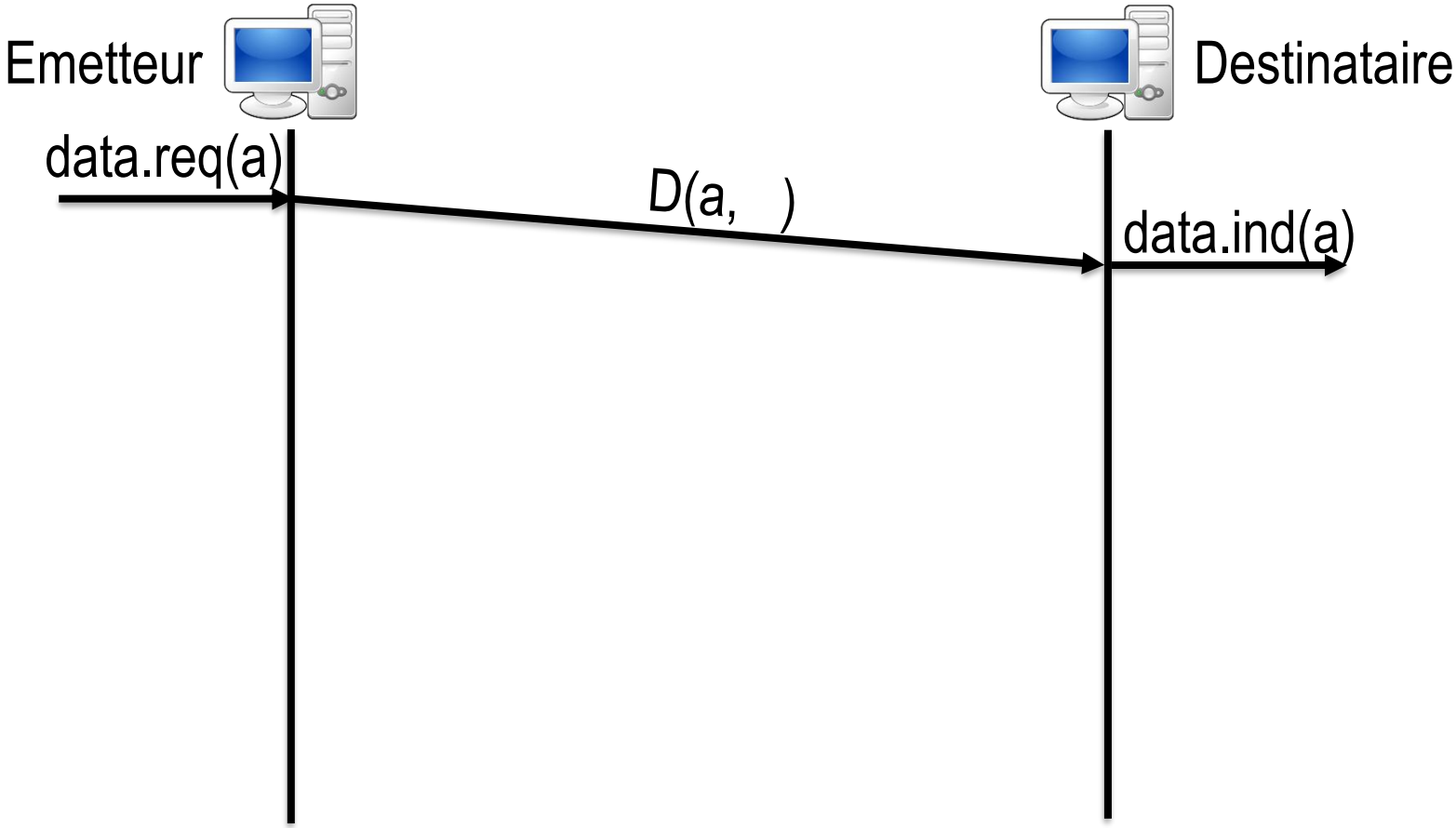
SWILABUS

Transfert fiable (14)

- Solution problème 5

Plan :

- Intro.
- UDP
- **Transfert fiable**
- TCP



Transfert fiable (15)

Plan :

- Intro.
- UDP
- **Transfert fiable**
- TCP

– Performance

- Quelles sont les performances d'un tel protocole ?
  - On est obligé d'attendre l'acquit d'un TPDU avant de pouvoir transmettre un autre
  - Les performances vont dépendre du *délai aller-retour* (= **round trip time** ou **RTT**) entre l'émetteur et le récepteur
    - » Peut parfois être élevé. Notamment en Wifi.
    - » La commande *ping* mesure celui-ci
  - Les performances vont également dépendre du **débit de la ligne** (# octets envoyés / temps nécessaire pour les envoyer), en bps, kbps ou Mbps.

Transfert fiable (16)

Plan :

- Intro.
- UDP
- **Transfert fiable**
- TCP

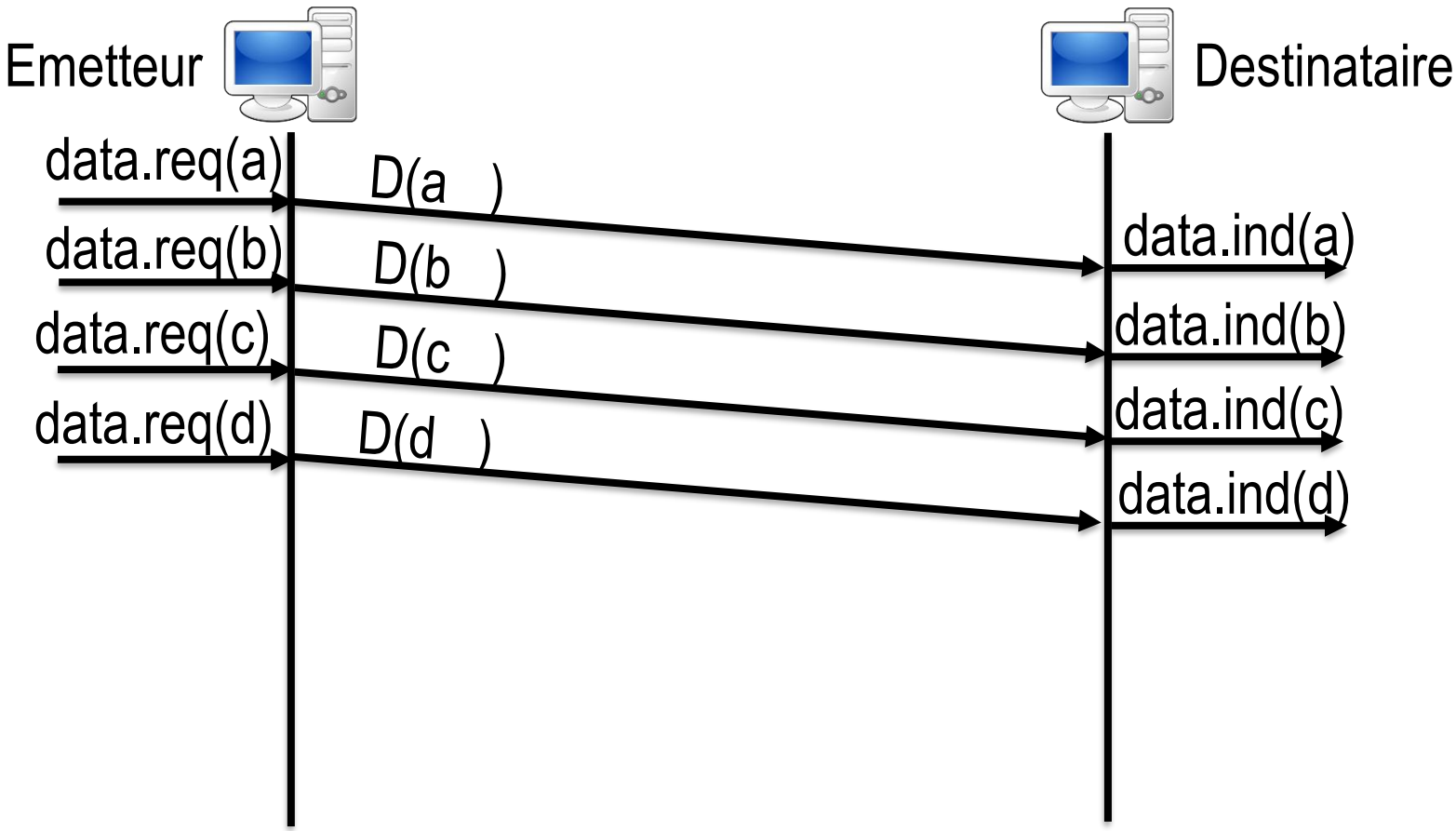
- Comment améliorer les choses ?
  - Permettre l'envoi de plusieurs données sans attendre d'acquit
    - » Comment être sûr que l'on ne va pas submerger le destinataire ?
    - » Comment être sûr que l'information arrive correctement à destination ?
    - » Que faire en cas de perte d'information ?

Transfert fiable (17)

– Solution

Plan :

- Intro.
- UDP
- **Transfert fiable**
- TCP



SWILABUS

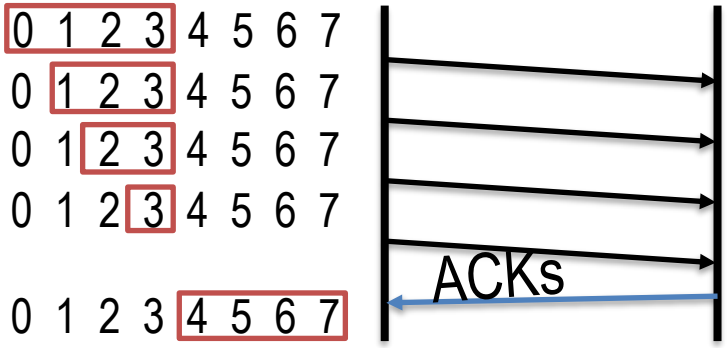
Transfert fiable (18)

Plan :

- Intro.
- UDP
- **Transfert fiable**
- TCP

- La **fenêtre glissante**
  - L'émetteur dispose d'une fenêtre contenant les numéros de séquence qu'il peut utiliser (=sending$_{window}$)
    - » Une fois tous les numéros utilisés, l'émetteur attend les acquits
  - Le destinataire dispose d'une fenêtre contenant les numéros de séquence qu'il accepte de recevoir (=receiving$_{window}$)

sending$_{window}$ <= receiving$_{window}$



SWILABUS

Transfert fiable (19)

Plan :

- Intro.
- UDP
- **Transfert fiable**
- TCP

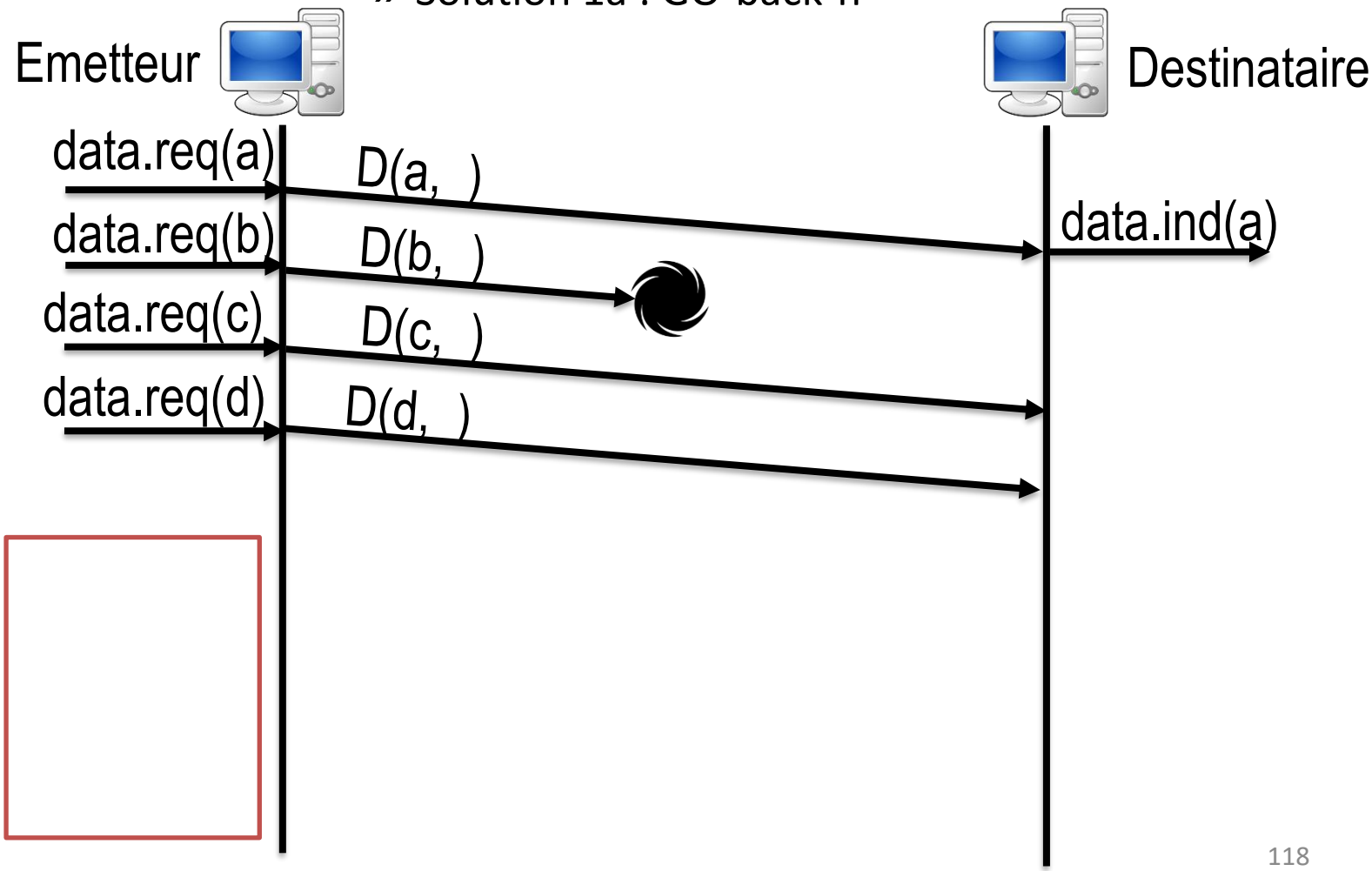
– Avantages
  » Comme plus d'information est envoyée ➔ amélioration des performances
  » Possibilité de **réduire le trafic de contrôle**
    • *Comment ?*
– **Que faire en cas de perte d'information ?**

Transfert fiable (20)

» Solution 1a : GO-back-n

Plan :

- Intro.
- UDP
- **Transfert fiable**
- TCP



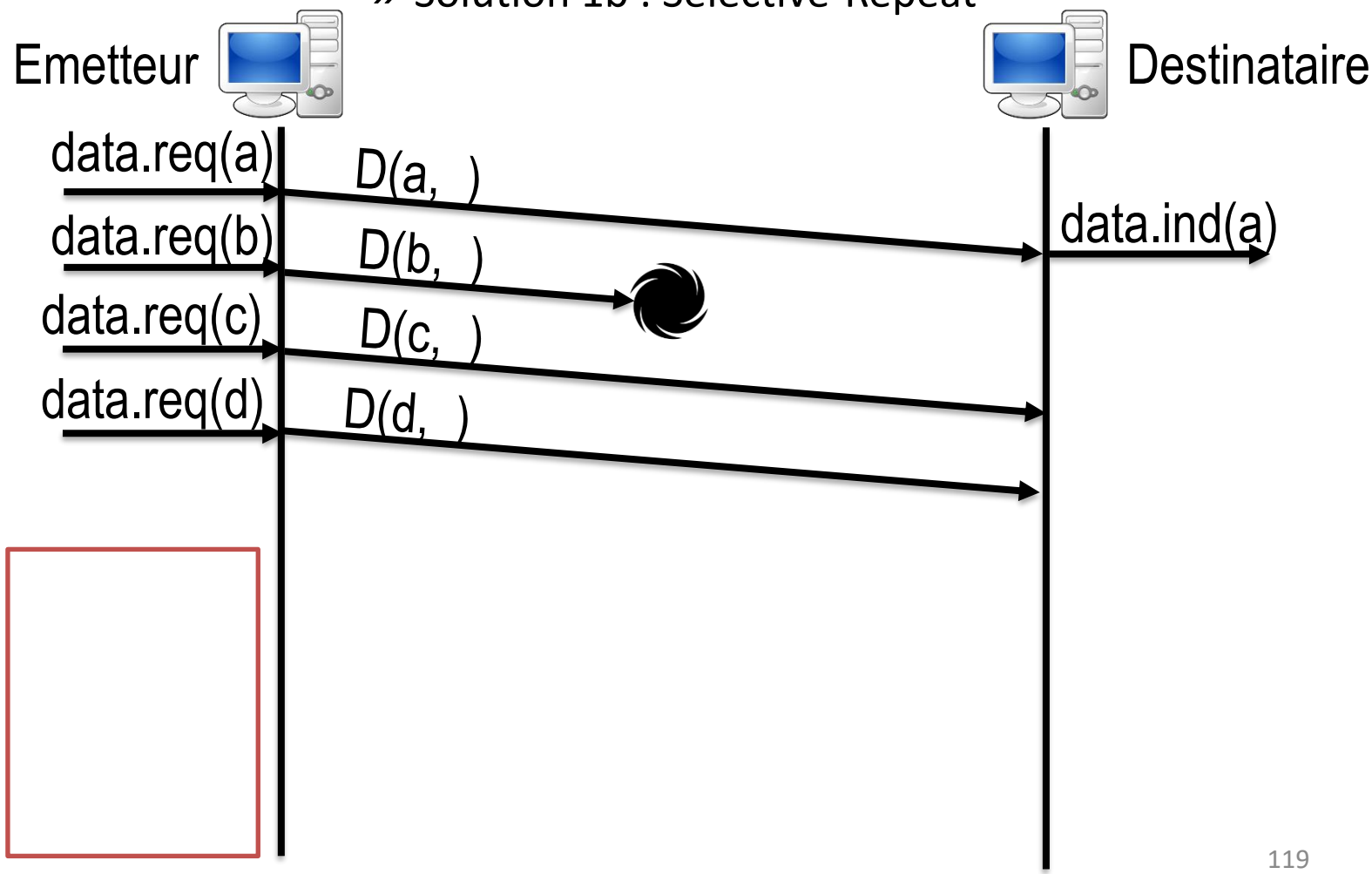
SWILABUS

Transfert fiable (21)

» Solution 1b : Selective-Repeat

Plan :

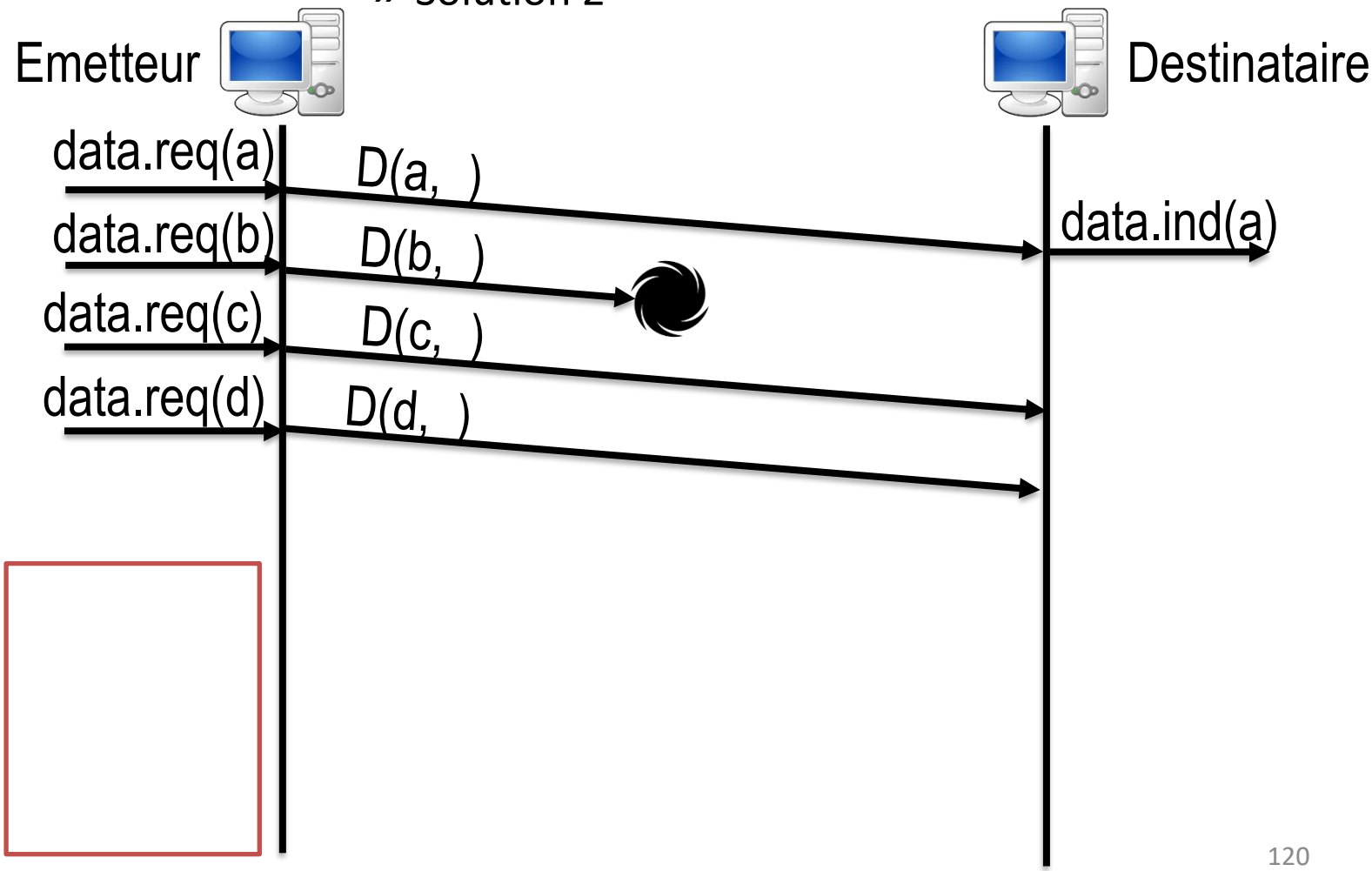
- Intro.
- UDP
- **Transfert fiable**
- TCP



SWILABUS

Transfert fiable (22)

» Solution 2



Plan :

- Intro.
- UDP
- **Transfert fiable**
- TCP

SWILABUS

Transfert fiable (23)

Plan :

- Intro.
- UDP
- **Transfert fiable**
- TCP

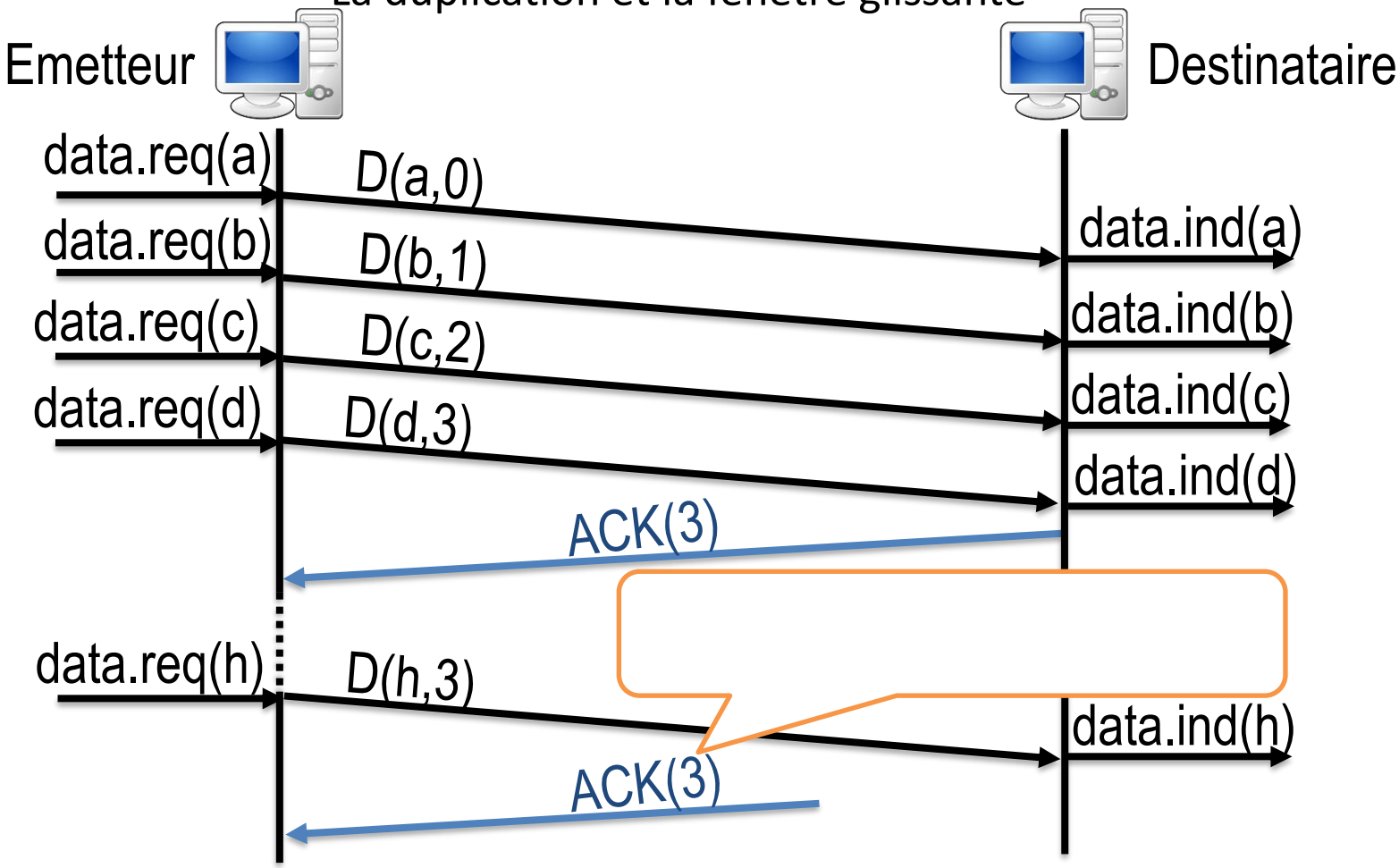
– La capacité de traitement du destinataire peut varier au fur et à mesure de ses activités
  - » Comment signaler un changement à l'émetteur?
  - » En ajoutant une information dans les acquits
    - L'acquit mentionne également la taille actuelle de la fenêtre de réception
    - La taille de cette fenêtre peut décroitre jusqu'à 0 pour indiquer que le destinataire est actuellement submergé
    - Ensuite la fenêtre peut croitre pour informer l'émetteur que le destinataire peut à nouveau traiter de l'information.

Transfert fiable (24)

– La duplication et la fenêtre glissante

Plan :

- Intro.
- UDP
- **Transfert fiable**
- TCP



SWILABUS

Transfert fiable (25)

» La couche réseau résout ce problème.
Comment ?

- Elle **assure** qu'un paquet ne peut survivre plus de MSL (Maximum Segment Lifetime) secondes dans le réseau
- Il faut être sûr qu'un numéro de séquence ne soit pas recyclé avant ce délai.

Plan :

- Intro.
- UDP
- **Transfert fiable**
- TCP

Transfert fiable (26)

Plan :

- Intro.
- UDP
- **Transfert fiable**
- TCP

– La connexion

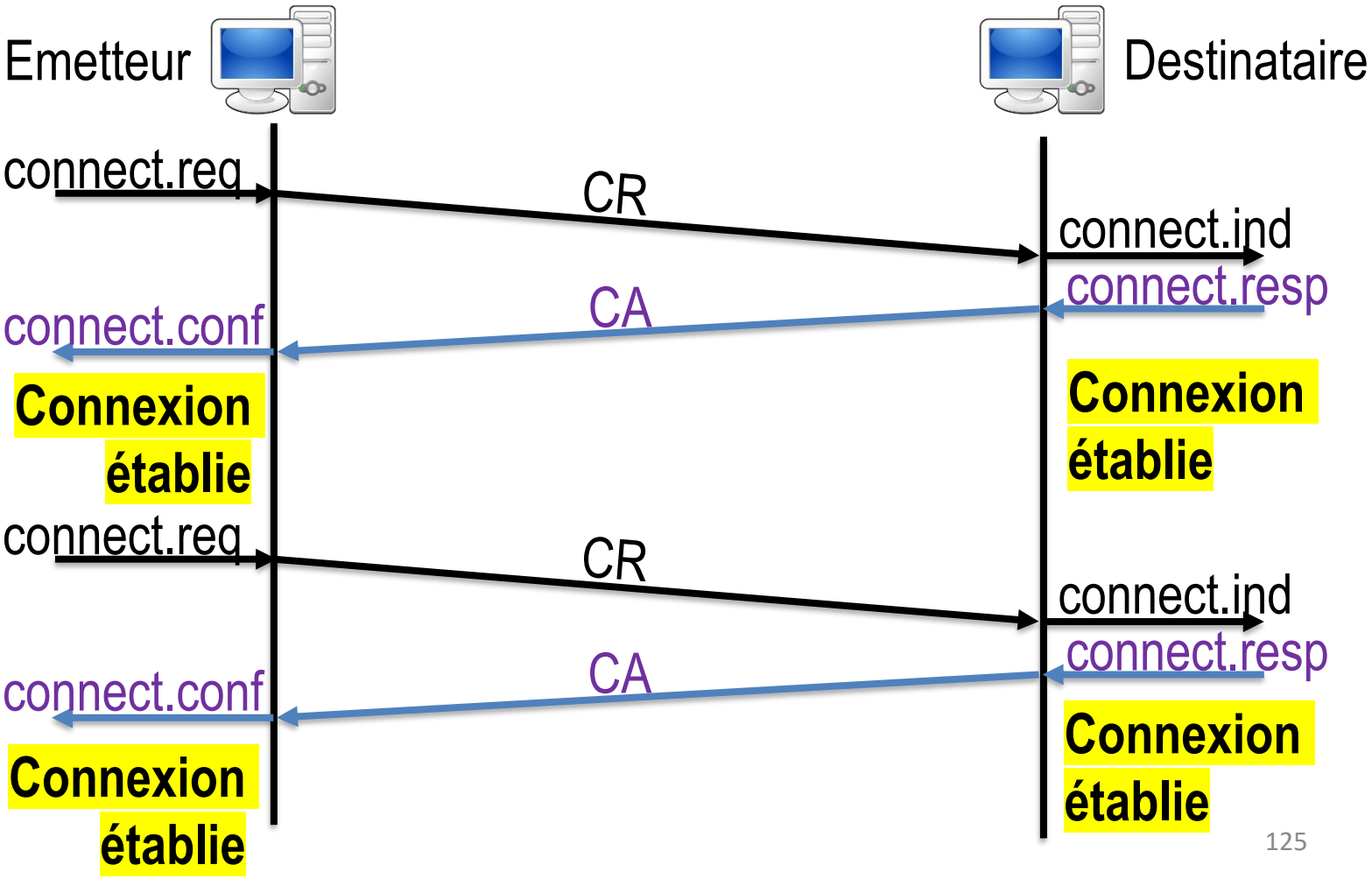
- Comment faire si l'on veut négocier certains paramètres avant l'échange de données ?
  - Profiter de la phase de connexion
    - » Comment faire pour établir la connexion ?
    - » Comment faire pour établir plusieurs connexions entre une même paire de machines?

Transfert fiable (27)

Plan :

- Intro.
- UDP
- **Transfert fiable**
- TCP

- Solution 1



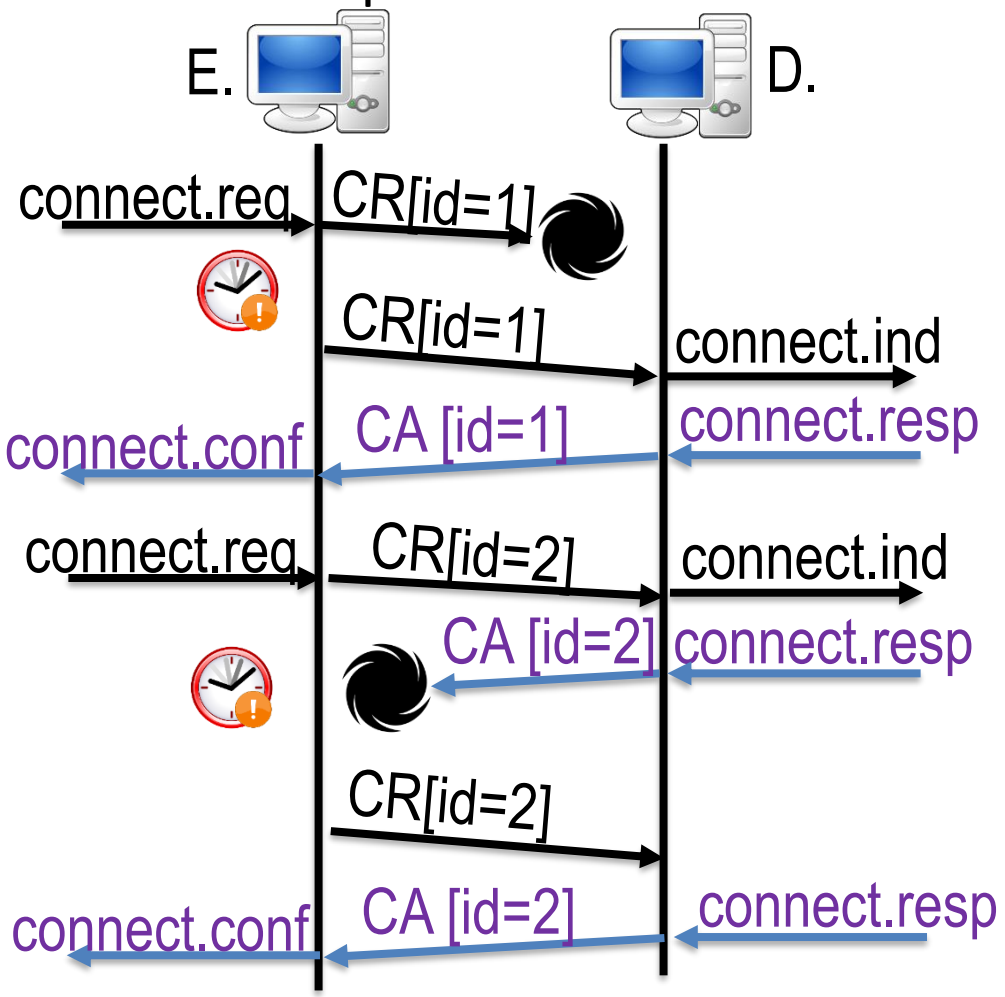
SWILABUS

Transfert fiable (28)

Plan :

- Intro.
- UDP
- **Transfert fiable**
- TCP

- Gestion des pertes



SWILABUS

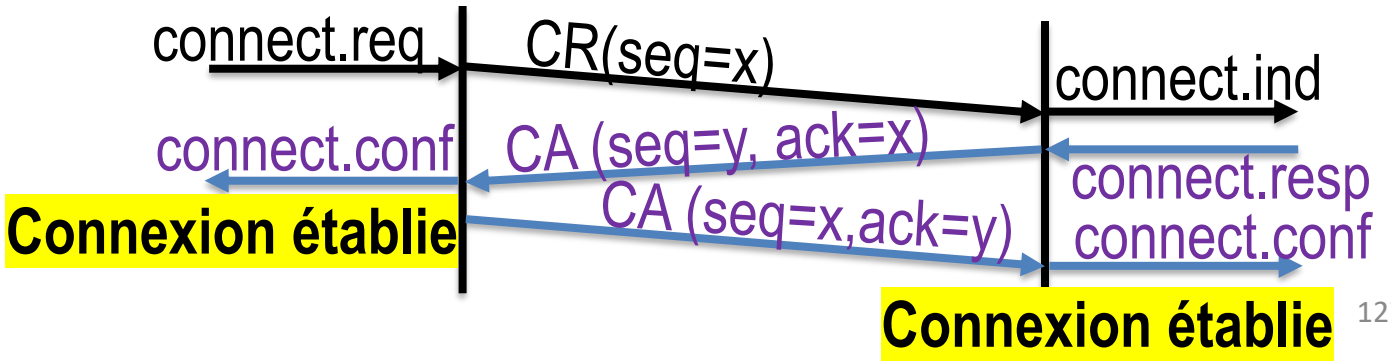
Transfert fiable (29)

Plan :

- Intro.
- UDP
- **Transfert fiable**
- TCP

- Gestion des doublons
  - But: éviter d'établir une connexion si cela n'est pas nécessaire (➔ détection des duplications)
  - Comment faire ?
    - » L'horloge fournit des numéros de séquence croissants
    - » Utiliser ces nuémros de séquence lors de la demande de connexion et lors de l'acquit de celle-ci
    - » Possibilité de détecter les CR & CA dupliqués

Le three-way handshake



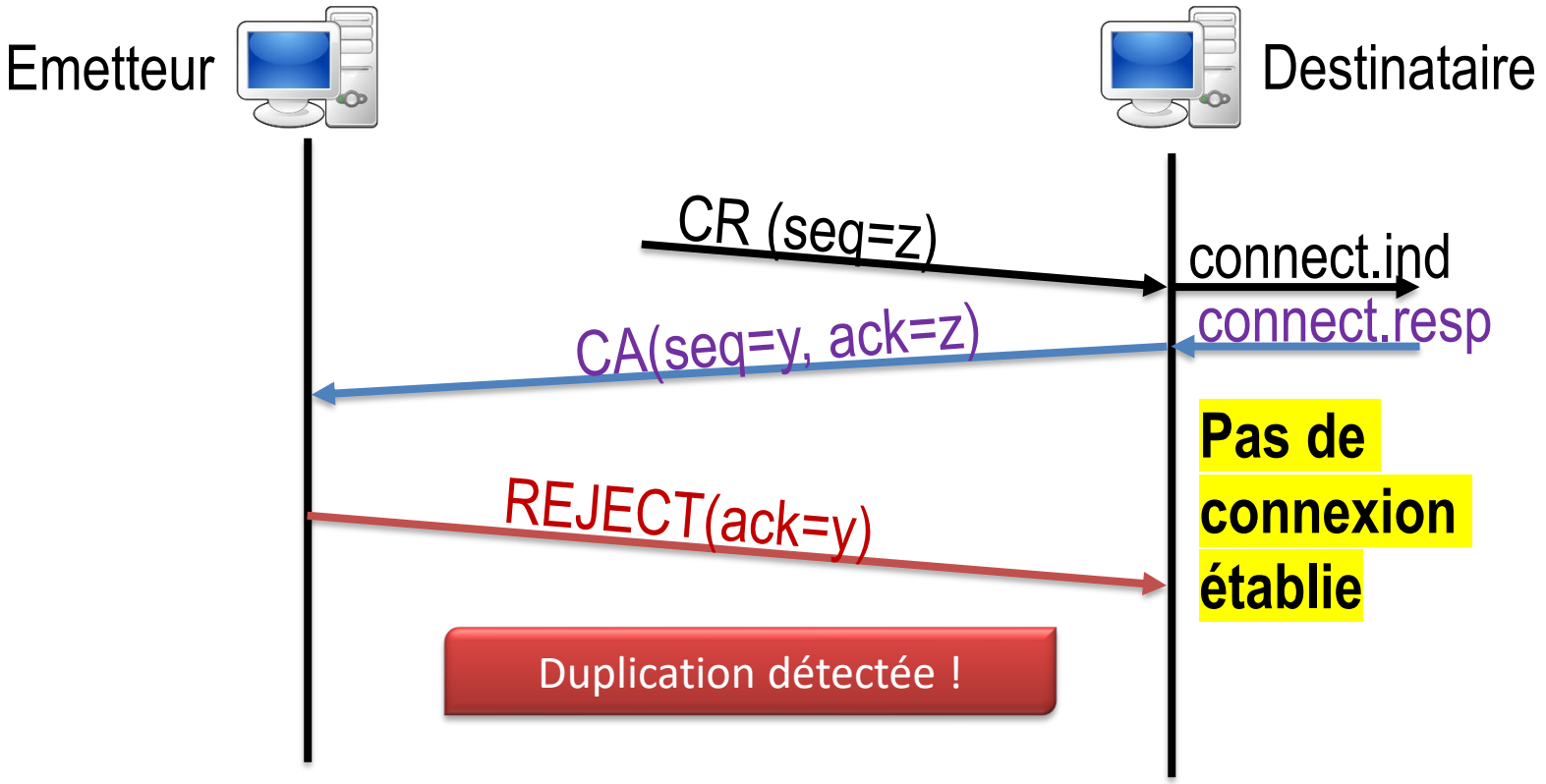
SWILABUS

Transfert fiable (30)

Plan :

- Intro.
- UDP
- **Transfert fiable**
- TCP

– Illustration (CR dupliqué)



SWILABUS

Transfert fiable (31)

Plan :

- Intro.
- UDP
- **Transfert fiable**
- TCP

– Gestion de la déconnexion

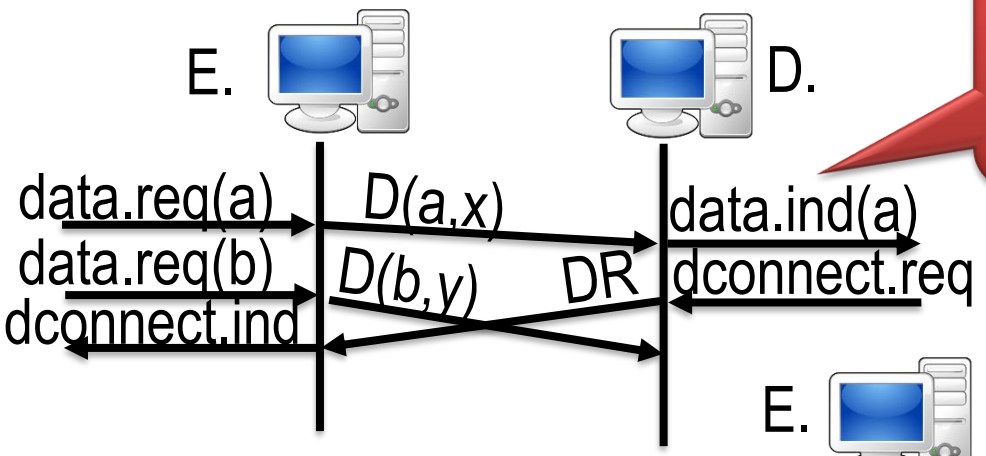
- La fermeture de la connexion peut être **abrupte**
  - Une des deux entités indique à l'autre qu'elle souhaite cesser la connexion et coupe cette dernière directement après.
  - Il est possible que des données ne soient jamais reçues.
- La fermeture de la connexion peut être **symétrique**.
  - Fonctionnement analogue au three-way handhake
  - La déconnexion est *plus sûre* de cette manière.

Transfert fiable (32)

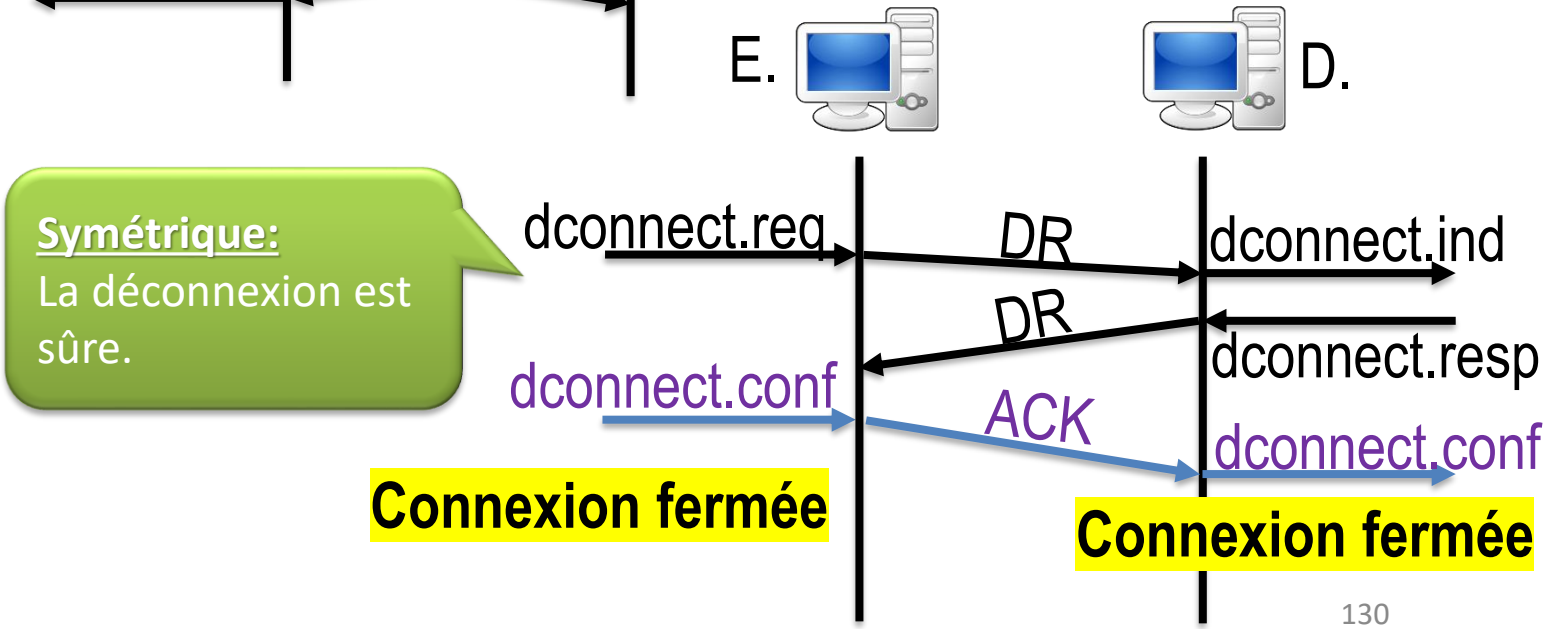
Plan :

- Intro.
- UDP
- **Transfert fiable**
- TCP

- Illustration



Abrupte:
L'émetteur n'est pas averti que la donnée ***b*** n'a pas été reçue.

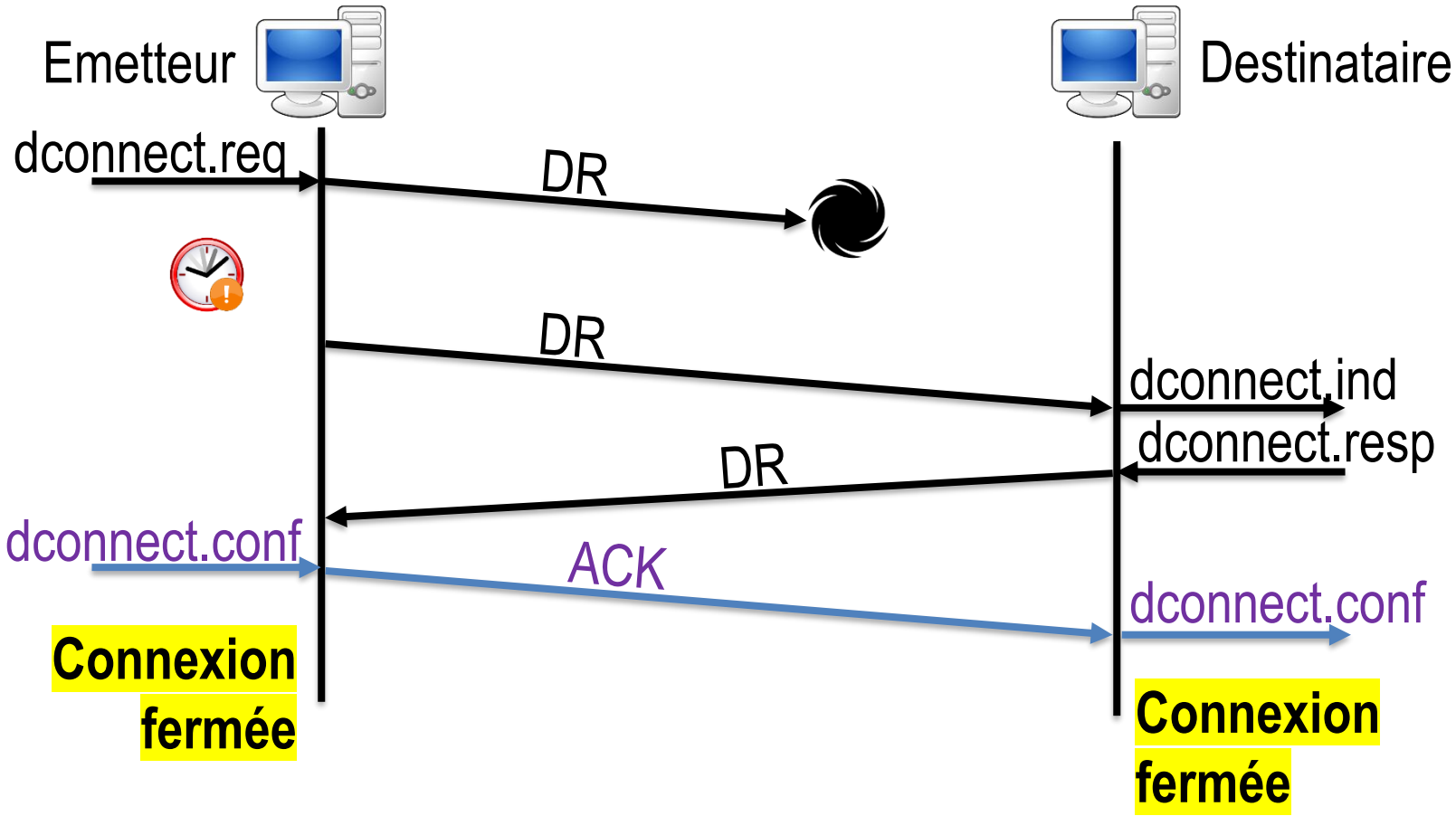


Symétrique:
La déconnexion est sûre.

SWILABUS

Transfert fiable (33)

- Perte du 1ier DR



Plan :

- Intro.
- UDP
- **Transfert fiable**
- TCP

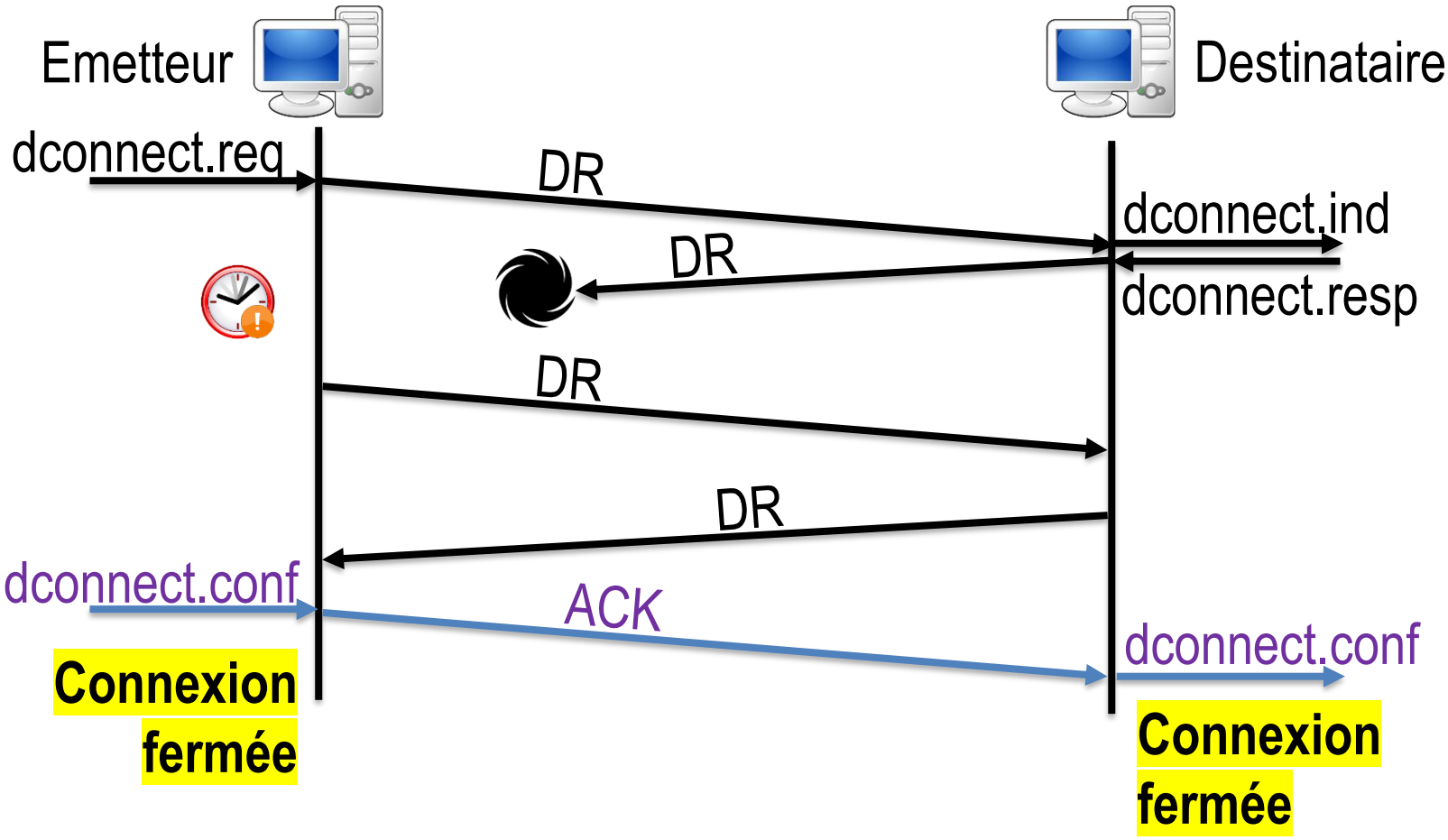
SWILABUS

Transfert fiable (34)

- Perte du 2ème DR

Plan :

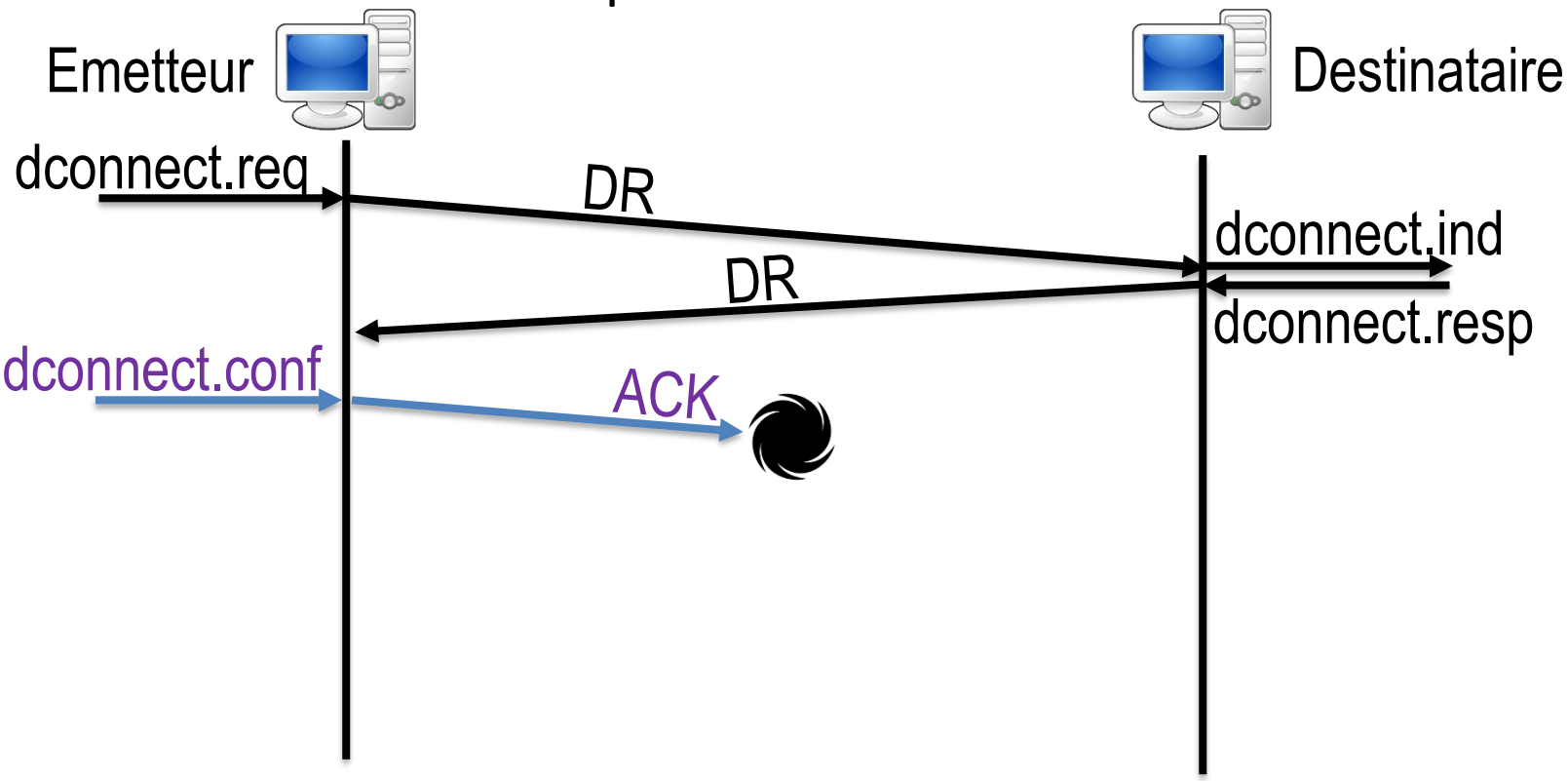
- Intro.
- UDP
- **Transfert fiable**
- TCP



SWILABUS

Transfert fiable (35)

- Perte de l'acquit



Plan :

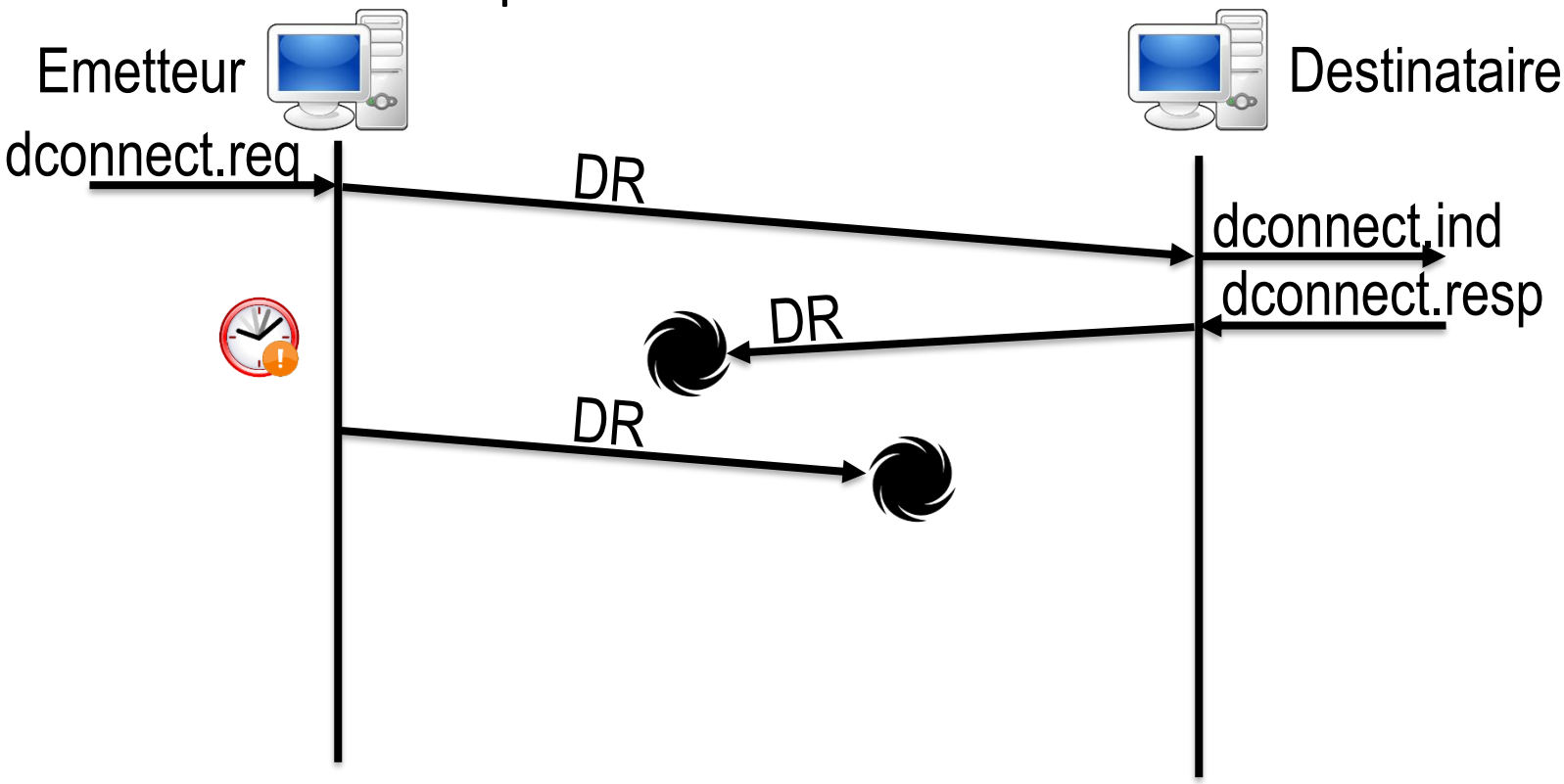
- Intro.
- UDP
- **Transfert fiable**
- TCP

Transfert fiable (36)

Plan :

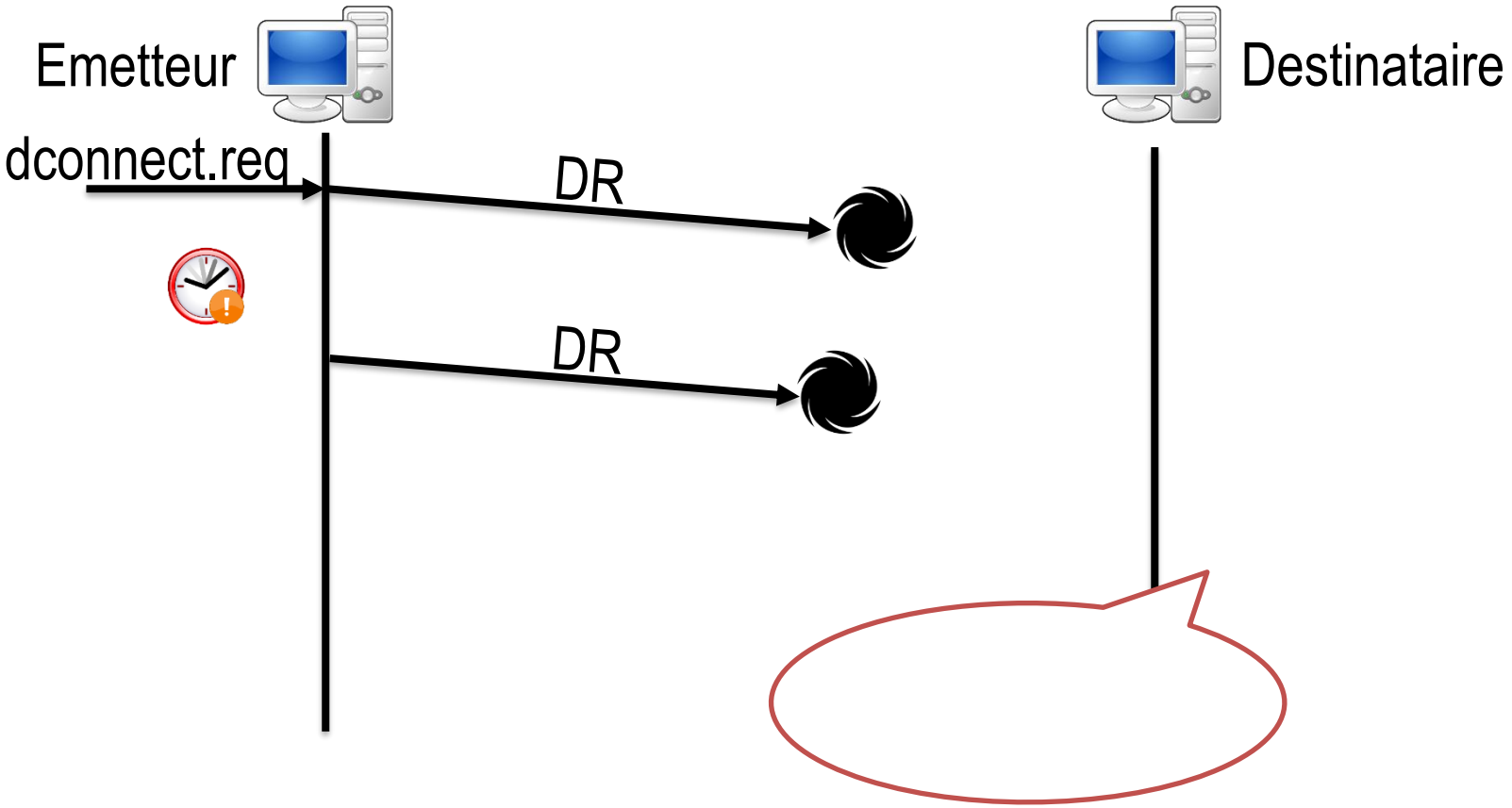
- Intro.
- UDP
- **Transfert fiable**
- TCP

- Perte de plusieurs DR



Transfert fiable (37)

- Perte de tous les DR



Plan :

- Intro.
- UDP
- **Transfert fiable**
- TCP

Transfert fiable (38)

Plan :

- Intro.
- UDP
- **Transfert fiable**
- TCP

– La communication bidirectionnelle
  - Nous avons considéré que seul l'émetteur envoyait des informations
  - Comment faire si les deux entités souhaitent échanger des informations ?
    – Ouvrir deux connexions différentes ?
    – Quid du trafic de contrôle ?
  - Solution plus intelligente ?
    –
    –

TCP (1)

- Implémentation du protocole TCP
  - Transmission Control Protocol
    - RFC 793, 1122, 1323, 2018, 2581, …
    - Orienté connexion (phase d'ouverture de connexion requise avant toute transmission) et fiable (transfert fiable d'information)
    - Il comprend:
      - Un mécanisme d'ouverture de connexion (utilisant le three-way handshake)
      - Un transfert fiable d'information, avec checksum, acquits et retransmission en cas de perte
      - Fermeture de connexion

Plan :

- Intro.
- UDP
- Transfert fiable
- **TCP**

TCP (2)

Plan :

- Intro.
- UDP
- Transfert fiable
- **TCP**

- Fonctionne en mode point-à-point, unicast
  - Un seul émetteur et un seul destinataire
- Identification d'une connexion TCP
  - Quadruplet formé par les adresses réseaux source et destination, les ports TCP source et destination
  - **Extension: Multipath TCP !**
- MSS: Maximum Segment Size
  - Taille maximale des données qui peuvent être placées dans le TPDU
  - Dépend souvent de la couche réseau et liaison de données sous-jacente
    - » Ex: 1500 bytes (ethernet)

TCP (3)

Plan :

- Intro.
- UDP
- Transfert fiable
- **TCP**

Focus sur MPTCP
RFC 6826

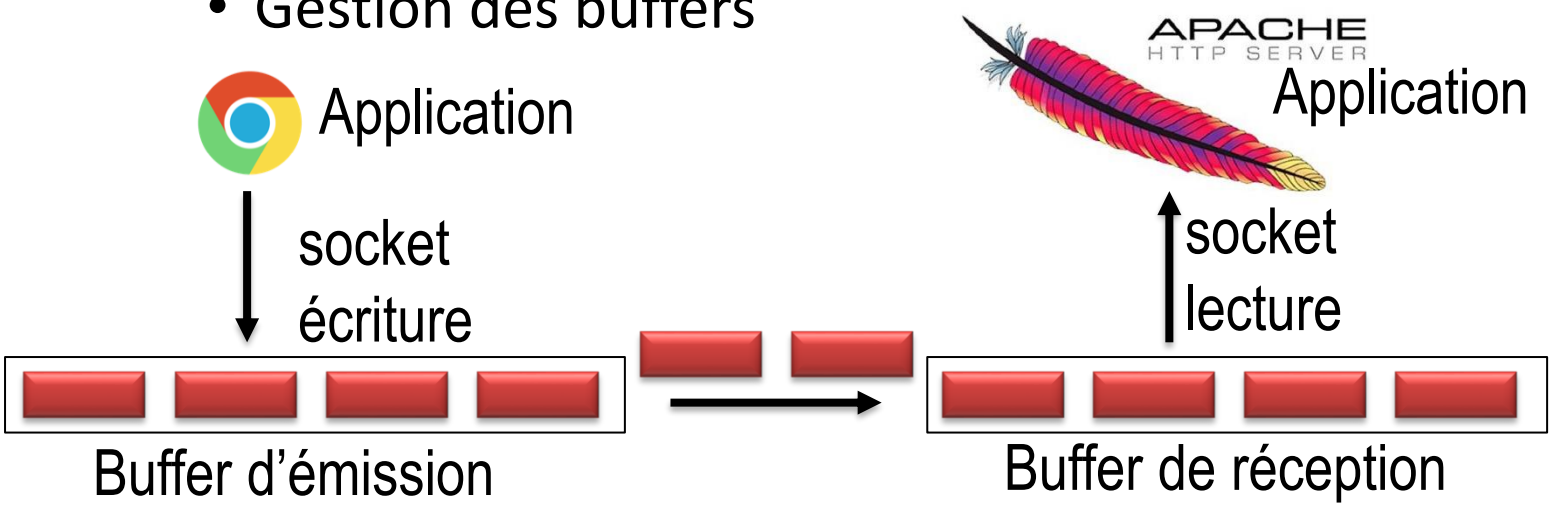
- Extension: le multipath TCP
  - Quand un périphérique possède des accès différents à des réseaux
    - » Smartphone: Wifi et 4G
  - Pourquoi ne pas utiliser **en même temps** ces deux connexions pour télécharger plus rapidement le contenu souhaité ?
    - » Cette possibilité est le mutlipath TCP
    - » Elle est disponible **sans modifier** l'équipement ou les applications
    - » Aujourd'hui, cette extension est présente dans la plupart des systèmes d'exploitation.

TCP (4)

Plan :

- Intro.
- UDP
- Transfert fiable
- **TCP**

- Gestion des buffers



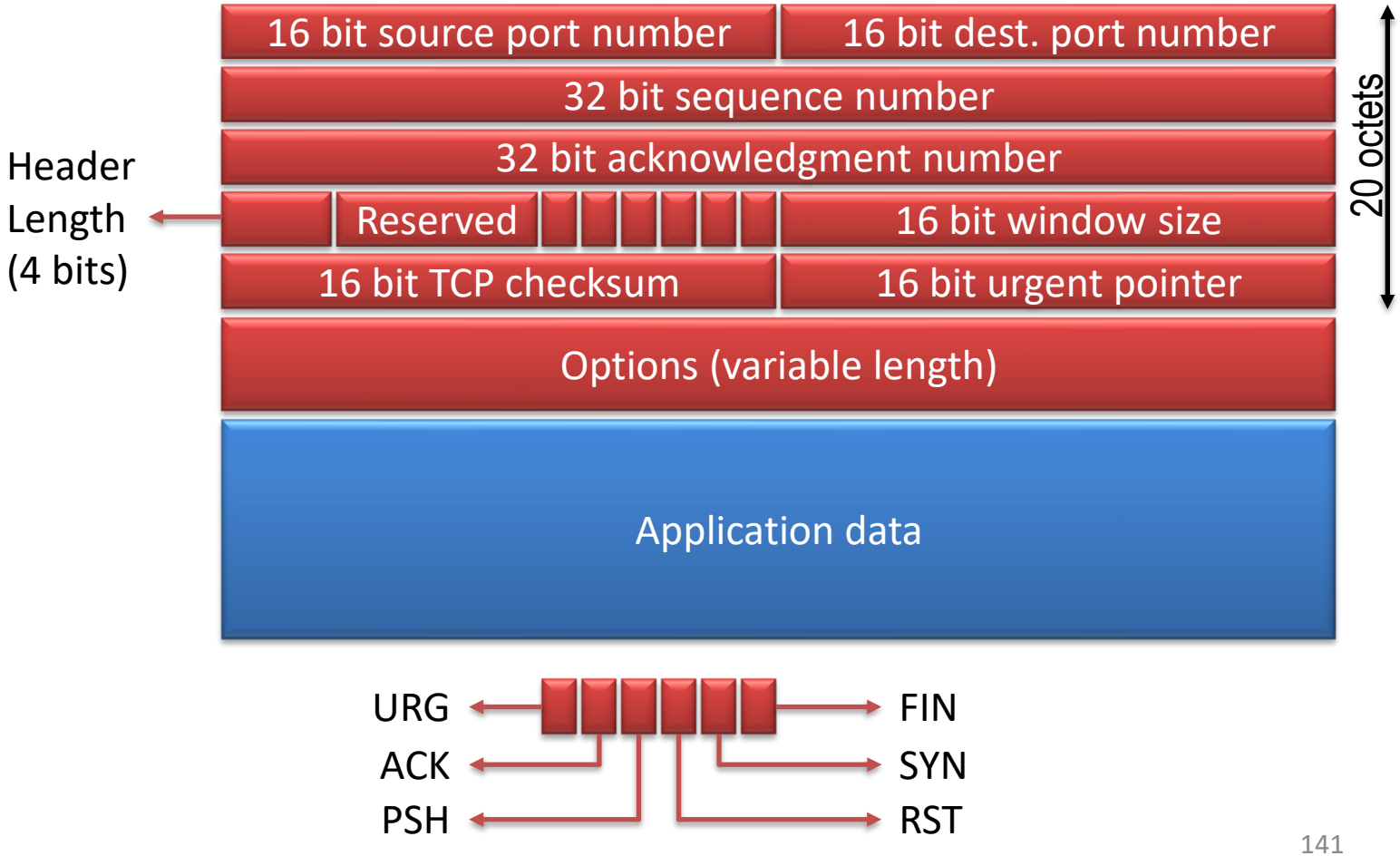
  - **L'émetteur** place les informations en attente de transmission dans le buffer d'émission.
    - » Dès qu'il le juge nécessaire, les informations sont envoyée
  - Le **destinataire** place les informations reçues dans le buffer de réception. Une fois toutes les informations reçues, elles sont transmises à l'application.

TCP (5)

Plan :

- Intro.
- UDP
- Transfert fiable
- **TCP**

- Contenu d'un TPDU TCP



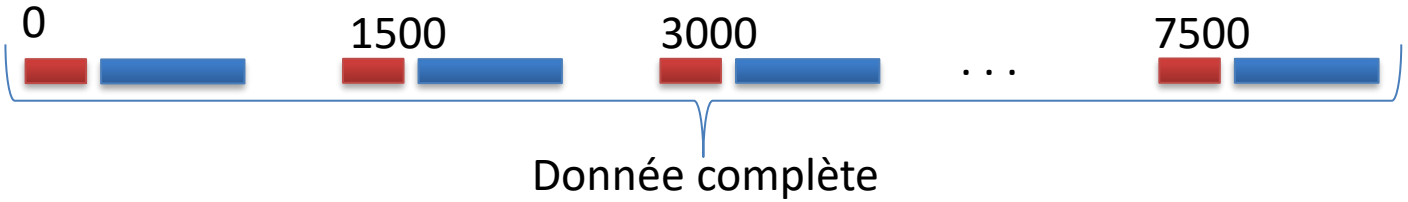
SWILABUS

TCP (6)

Plan :

- Intro.
- UDP
- Transfert fiable
- **TCP**

- Les numéros de séquence et d'acquit
  - TCP voit l'information comme des octets ordonnés
  - Les numéros de séquence sont attribués en respectant cette vue
  - Le numéro de séquence d'un TPDU est le numéro d'ordre du 1ier octet dans le TPDU (exemple avec un MSS de 1500 octets).

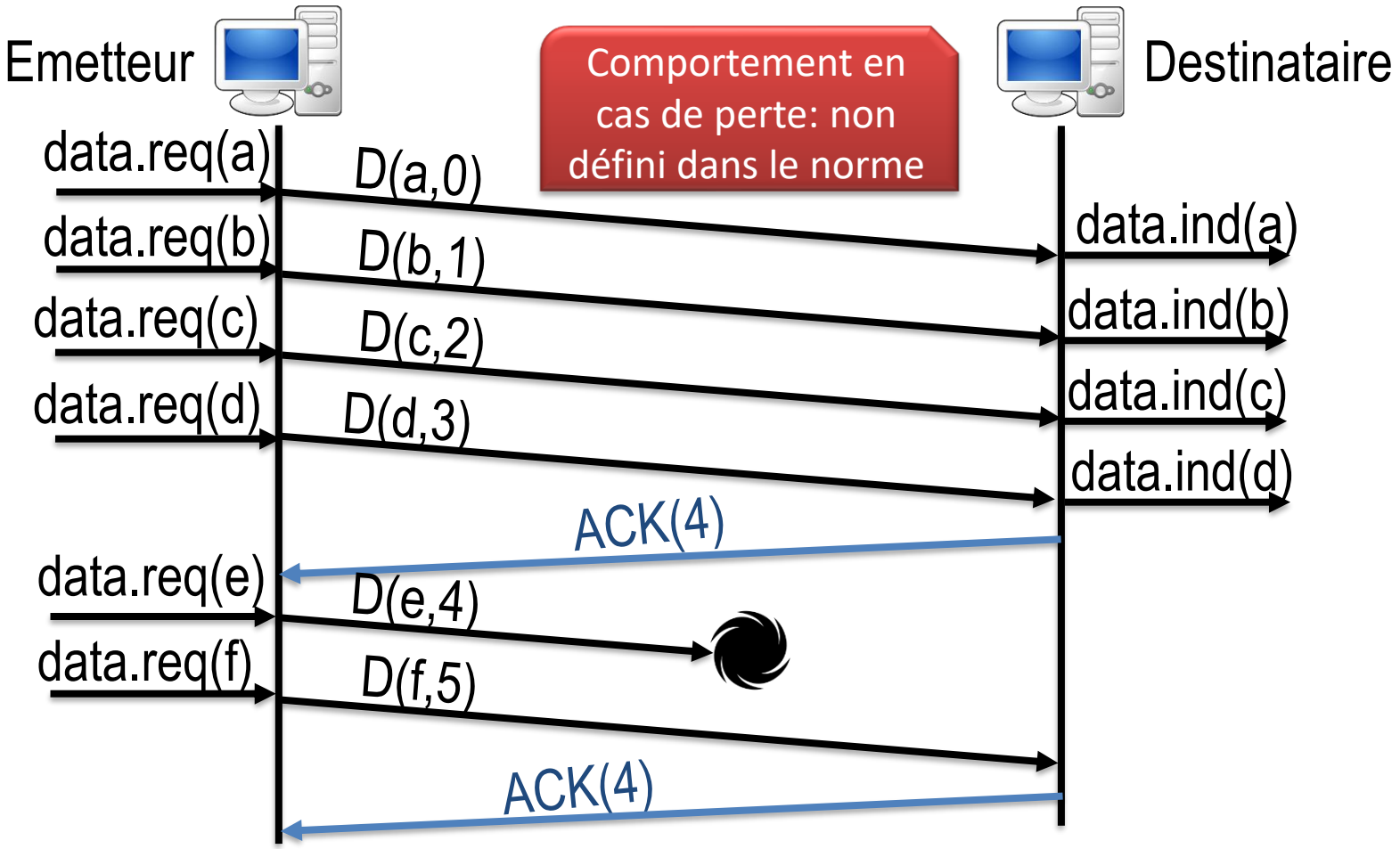


  - Les acquits peuvent être placés dans des TPDUs de données (**communication bidirectionnelle**)
  - En TCP, l'acquit précise **toujours** le **prochain numéro de séquence attendu**.

SWILABUS

TCP (7)

– Illustration (avec perte)



Plan :

- Intro.
- UDP
- Transfert fiable
- **TCP**

SWILABUS

Le 1ier n° de séquence n'est pas nécessairement 0

TCP (8)

Plan :

- Intro.
- UDP
- Transfert fiable
- **TCP**

- Valeur des temporisateurs (RFC 2988)
  - Des temporisateurs sont utilisés dans la couche transport pour déterminer quand les retransmissions doivent avoir lieu.
  - Quelle valeur fixer ?
    - » Il faut que ce temporisateur soit > RTT
    - » Cas d'un temporisateur trop petit
    - » Cas d'un temporisateur trop grand
  - De quoi doit dépendre cette valeur de temporisateur ?
    - » Du RTT
      - Instantané ? **Trop variable !**
      - Moyen ? **Comment l'évaluer ?**
    - » De la variabilité du RTT
      - Si très variable ➔ augmenter le temporisateur

TCP (9)

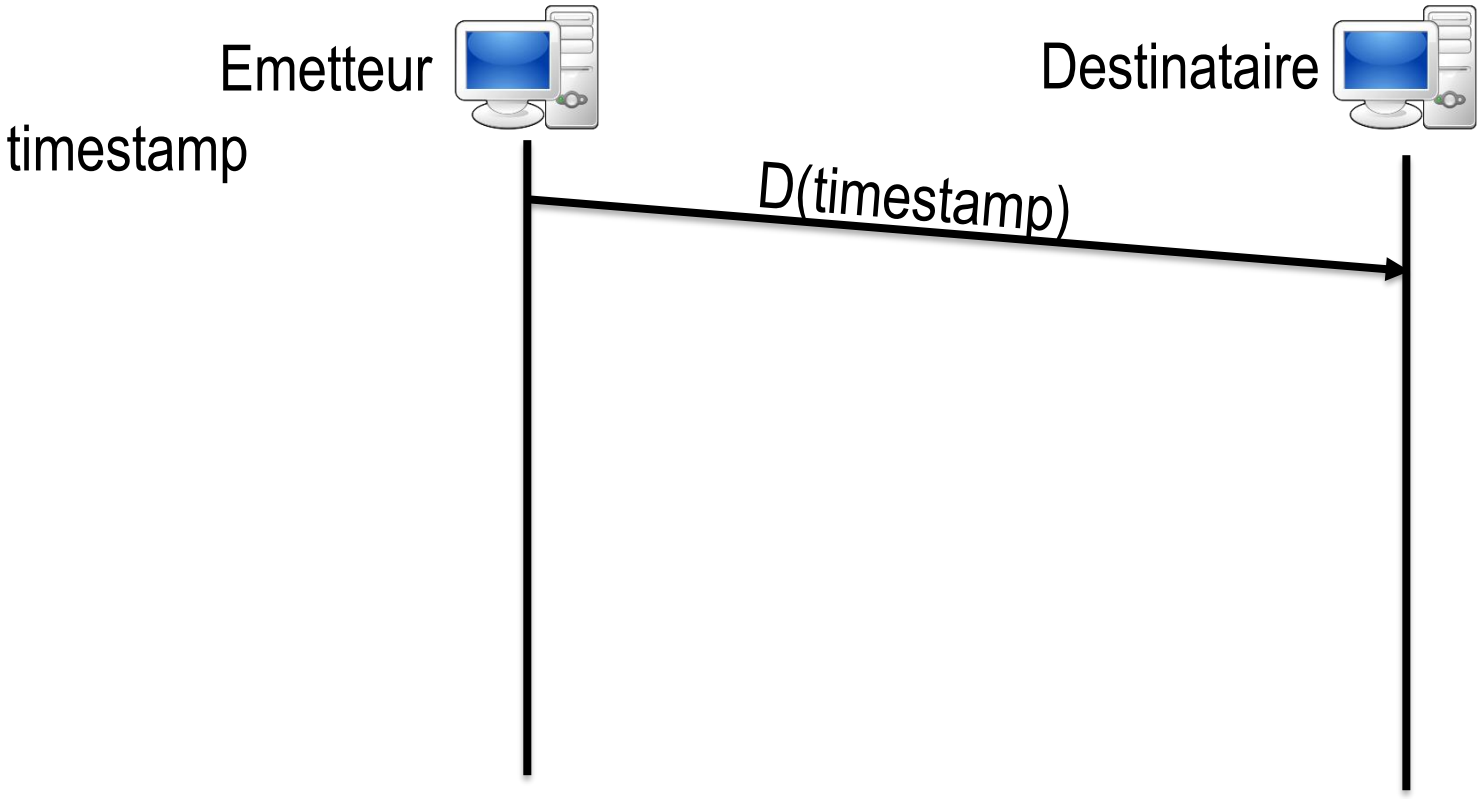
- Déterminer le RTT moyen
  - Approximation par Exponential Weighted Moving Average (EWMA)
    - » **rttMoyen = (1-a) rttMoyen + a rttInstant**
    - » La valeur conseillée pour **a** est 0.125
  - Déterminer la variabilité du RTT
    - » **rttVar= (1-b) rttVar + b |rttInstant – rttMoyen|**
    - » RttVar approxime la variabilité du RTT. Plus l'écart entre le RTT moyen et le RTT instantané est important, plus la variabilité augmente.
    - » La valeur conseillée pour **b** est 0.25
  - Fixer la valeur du temporisateur
    - » **timeoutVal = rttMoyen + 4 rttVar**
    - » Il doit être supérieur au RTT
    - » Il faut prendre en compte la variabilité

Plan :

- Intro.
- UDP
- Transfert fiable
- **TCP**

TCP (10)

– Comment mesurer le RTT instantané ?



Plan :

- Intro.
- UDP
- Transfert fiable
- **TCP**

TCP (11)

- Envoi des acquits

Condition	Action
1. Arrivée d'un TPDU en séquence avec le numéro de séquence attendu. Toutes les données ont été acquittées jusqu'à ce TPDU	Retarder l'acquit. Le destinataire peut attendre jusqu'à 500 ms l'arrivée d'un autre TPDU à aquitter. Si pas de TPDU, envoi de l'acquit.
2. Arrivée d'un TPDU en séquence avec le numéro de séquence attendu. Un autre TPDU n'est pas acquitté	Envoi immédiat d'un seul acquit cumulatif
3. Arrivée d'un TPDU **hors** séquence avec un numéro de séquence > à celui attendu	Envoi immédiat d'un **acquit dupliqué**
4. Arrivée d'un TPDU **hors** séquence avec un numéro de séquence qui couvre un trou	Envoi immédiat d'un acquit. Le numéro d'acquit envoyé est celui de la prochaine donnée attendue.

Tableau © Pearson education, 2013 [1]

Plan :

- Intro.
- UDP
- Transfert fiable
- **TCP**

TCP (12)

Plan :

- Intro.
- UDP
- Transfert fiable
- **TCP**

- Fast Retransmit
  - Amélioration de TCP qui permet de renvoyer une information non reçue sans attendre l'expiration du temporisateur
  - Fonctionnement
    - » Le récepteur acquitte tous les TPDUs (y compris ceux reçus hors séquences)
    - » Lorsque **3 acquits dupliqués** (=identiques) sont reçus, le TPDU indiqué est supposé perdu
      - Retransmission de ce TPDU avant l'expiration du temporisateur ➔ efficacité.
  - Fast-Retransmit est généralisé aujourd'hui (RFC 2581)

TCP (13)

Plan :

- Intro.
- UDP
- Transfert fiable
- **TCP**

- Gestion des pertes
  - TCP implémente *généralement*
    - » Les numéros de séquence retenus sont ceux qui arrivent en séquence
    - » Les TPDUs qui arrivent hors séquences sont sauvegardés
    - » Via Fast-Retransmit, TCP permet la retransmission rapide d'un TPDU perdu
- Emission des TPDUs (Algorithme de Nagle)
  - Dès qu'une information doit être transmise ?
  - Lorsque le TPDU est complètement rempli ?
    - » La première donnée est envoyée directement
    - » Les données suivantes sont accumulées dans le TPDU tant qu'il n'est pas rempli ou que le précédent TPDU n'est pas acquitté.

TCP (14)

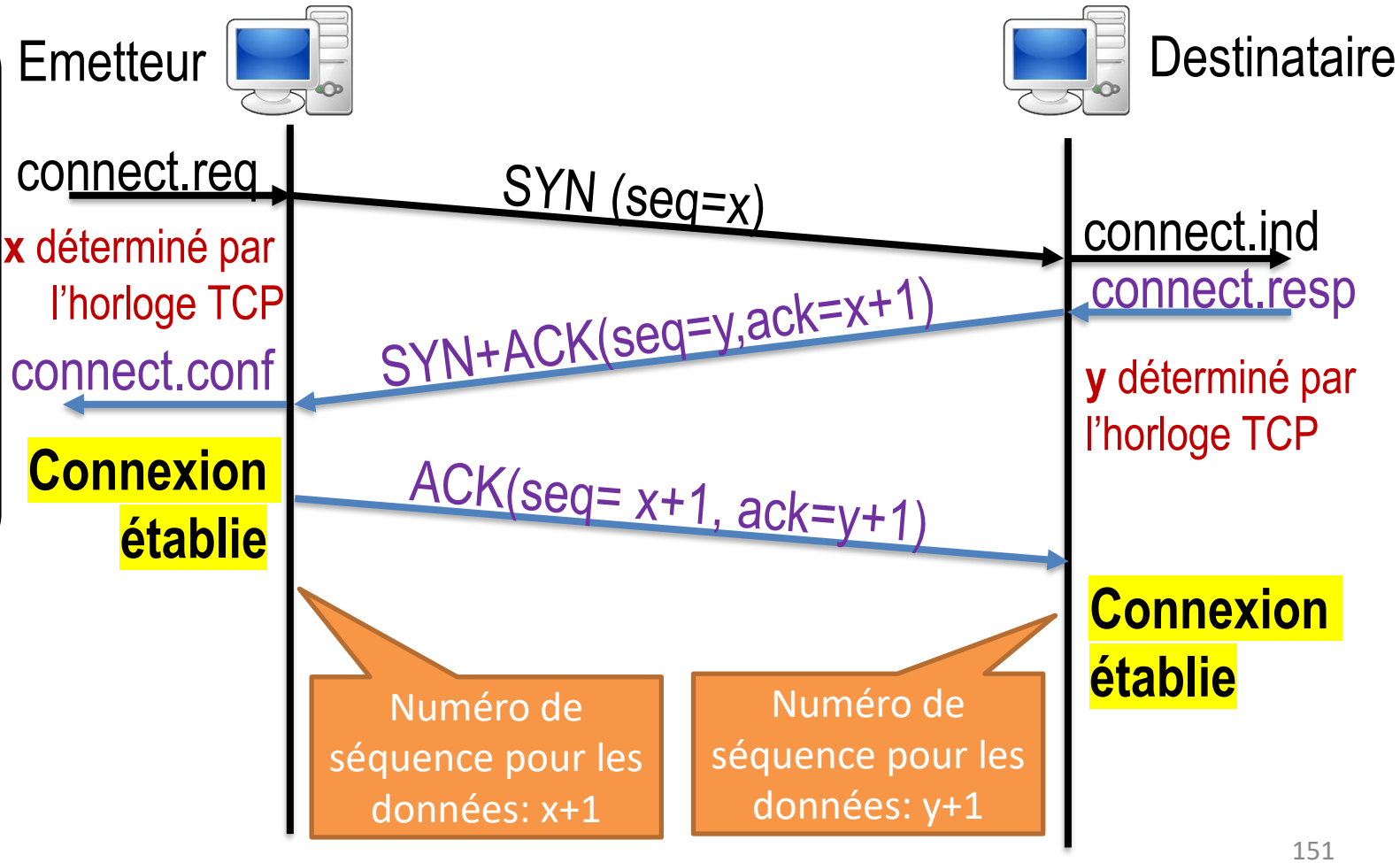
- Contrôle de flux
  - Comment être sûr que l'on ne submerge pas le destinataire ?
  - Ce dernier doit mentionner sa capacité de traitement au fur et à mesure qu'elle évolue
  - Le destinataire va mentionner une taille de fenêtre de réception
    - » Dès que la fenêtre est remplie, l'émetteur doit attendre un acquit
    - » La capacité de cette fenêtre est normalement codée sur 16 bits. Des extensions proposent un codage plus large (RFC 1323).

Plan :

- Intro.
- UDP
- Transfert fiable
- **TCP**

TCP (15)

- Etablissement de la connexion



Plan :

- Intro.
- UDP
- Transfert fiable
- **TCP**

SWILABUS

TCP (16)

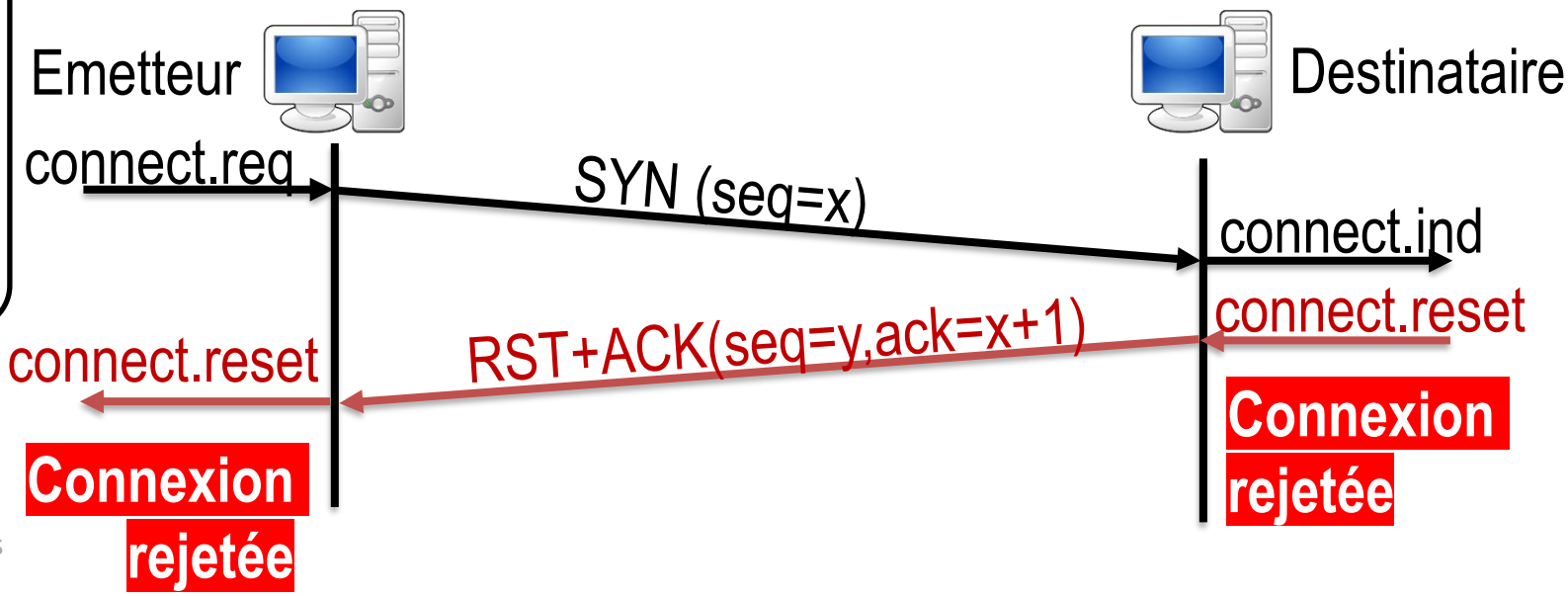
Plan :

- Intro.
- UDP
- Transfert fiable
- **TCP**

– Durant la connexion, des options peuvent être négociées (timestamp, MSS, …)

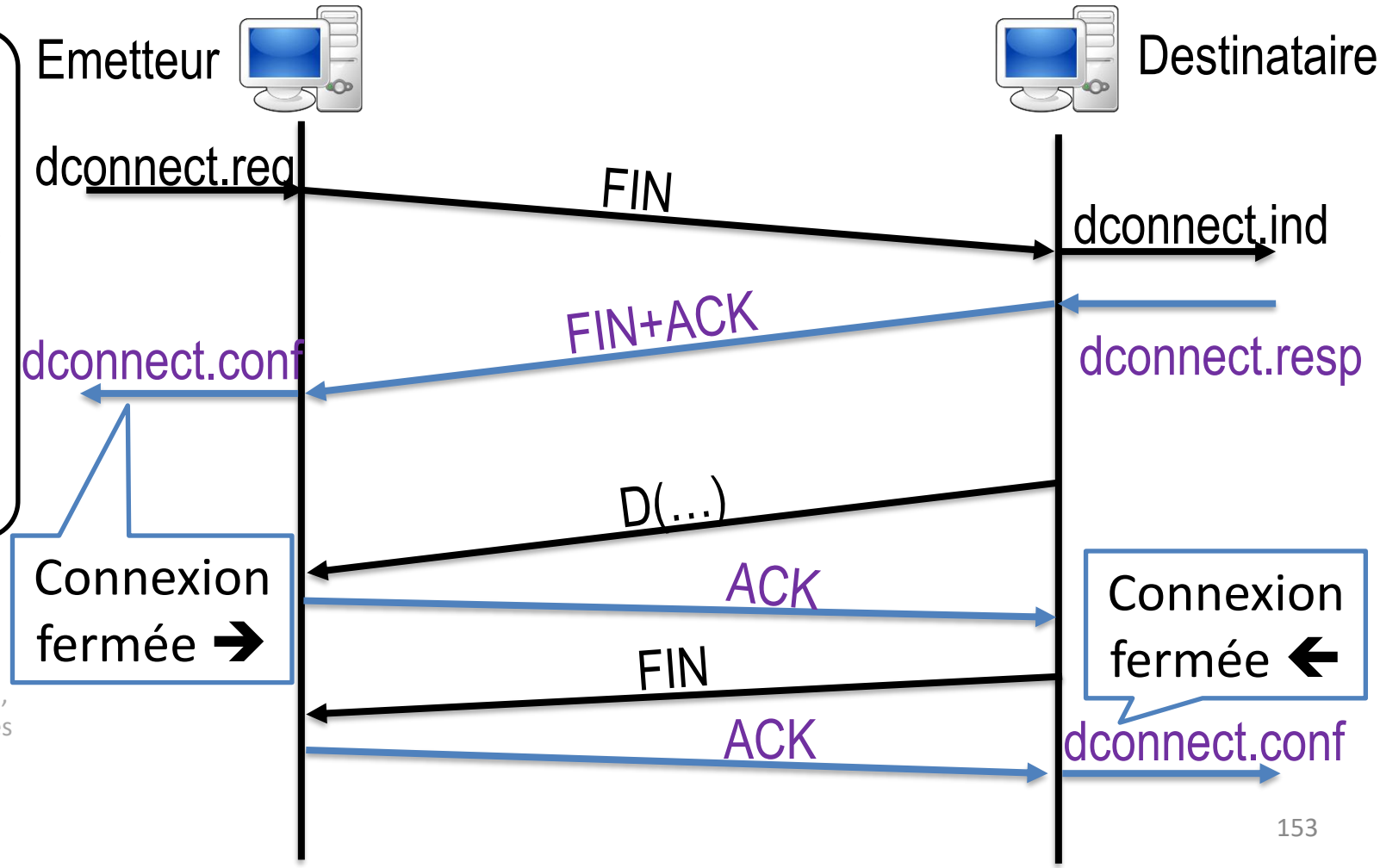
– Il est possible d'ouvrir une paire de connexion TCP entre 2 machines grâce aux n° de port source.

– Une connexion peut être refusée par le flag RST:



TCP (17)

- Fermeture de connexion



Plan :

- Intro.
- UDP
- Transfert fiable
- **TCP**

SWILABUS

TCP (18)

Plan :

- Intro.
- UDP
- Transfert fiable
- **TCP**

- Contrôle de congestion
  - La congestion
    - » C'est **une surcharge du réseau** qui entraîne des pertes de données
    - » Elle peut se produire sur un lien déterminé n'importe où sur Internet ou dans le réseau
      - Ex: une ligne plus lente
    - » Internet est un ensemble de réseaux inter-connectés ensembles, très hétérogènes
      - Avec des technologies différentes
      - Avec des débits (i.e. rapidité) différents
    - » Ces différences conduisent à certains endroits, à des surcharges ➔ **congestion !**
      - Les différences de vitesse conduisent à des accumulations et débordements

TCP (19)

Plan :

- Intro.
- UDP
- Transfert fiable
- **TCP**

– Que faire pour ne pas surcharger le réseau

– TCP utilise un mécanisme qui, lorsqu'une congestion est détectée, **diminue automatiquement le taux d'envoi des informations**

  » En cas de problème de débordement, si le taux d'émission est réduit, la congestion se résorbe

  » Mécanisme intelligent

– Si toutes les couches transports implémentent un mécanisme similaire, le partage de la ligne serait équitable en toute circonstance

– UDP n'implémente pas un tel mécanisme

  » L'utilisation d'UDP impose parfois de devoir implémenter un contrôle de congestion au niveau de la couche application.

TCP (20)

Plan :

- Intro.
- UDP
- Transfert fiable
- **TCP**

– Principes

» On définit une **fenêtre de congestion** qui limite le taux d'émission des données chez l'émetteur

» Cette fenêtre commence avec une taille de 1

» Cette fenêtre **croit de manière exponentielle** jusqu'à atteindre un ***seuil défini***

- Mode slow-start: la taille de la fenêtre double a chaque RTT (quand les données sont acquittées)
- La valeur du ***seuil*** est de 64 K au démarrage

» Une fois le ***seuil*** atteint, la fenêtre croit linéairement

- Mode additive increase multiplicative decrease (AIMD): le fenêtre croit d'1 MSS à chaque RTT

TCP (21)

Plan :

- Intro.
- UDP
- Transfert fiable
- **TCP**

– En cas de perte
  » Il faut diminuer le taux d'émission, donc la fenêtre de congestion
  » **Congestion légère** – réception de 3 acquits dupliqués
    • Retransmission par fast-restransmit
    • ***Seuil*** = fenêtre de congestion / 2
    • ***Fenêtre de congestion*** = seuil
    • Reprise en croissance linéaire (en AIMD)
  » Congestion forte – **expiration des temporisateurs**
    • Retransmission de l'information
    • ***Seuil*** = fenêtre de congestion / 2
    • ***Fenêtre de congestion*** = 1 MSS
    • Reprise en croissance exponentielle (slow-start) jusqu'à atteindre le ***seuil***

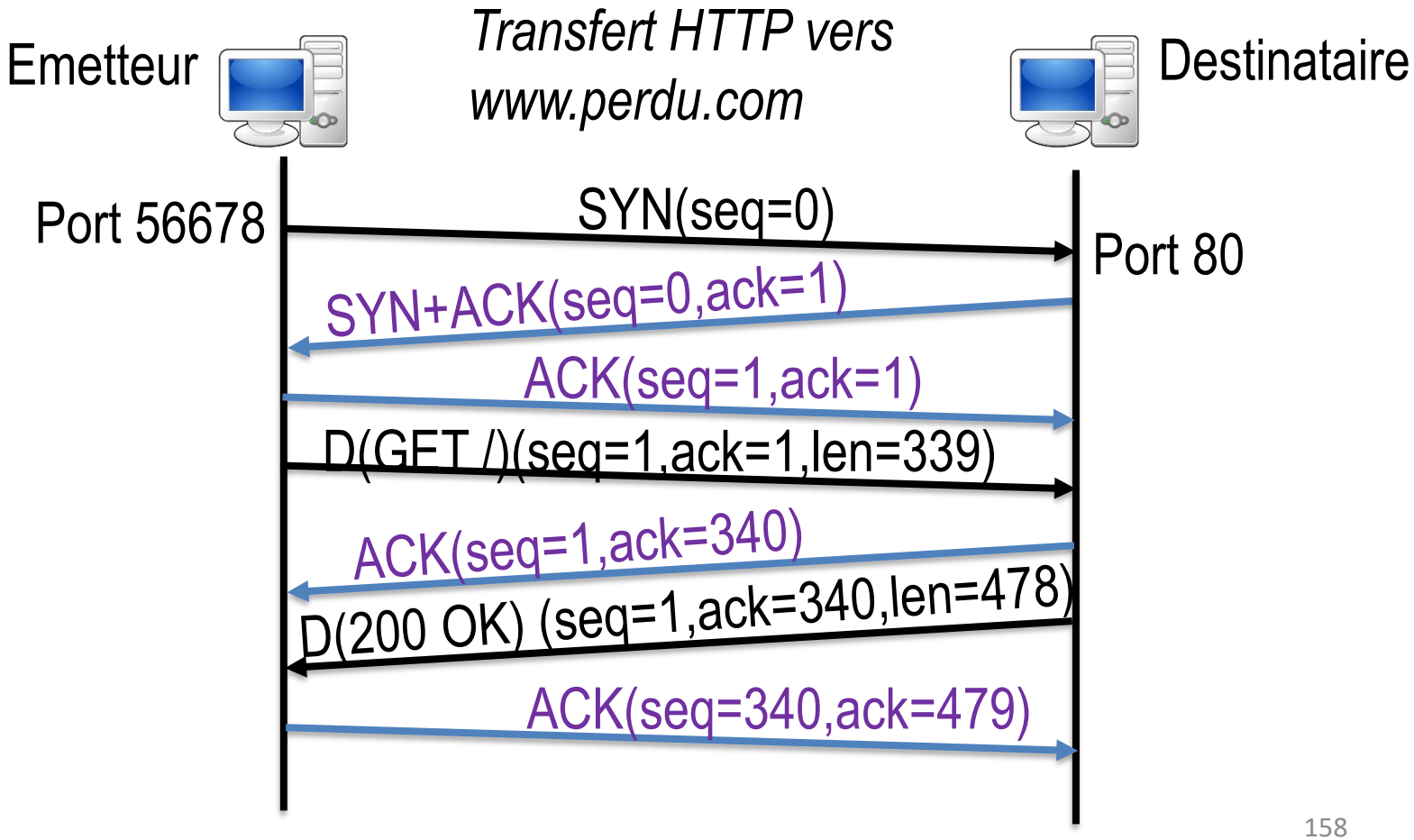
SWILABUS

TCP (22)

Plan :

- Intro.
- UDP
- Transfert fiable
- **TCP**

- Exemple



SWILABUS

TCP (23)

- Exemple (suite)