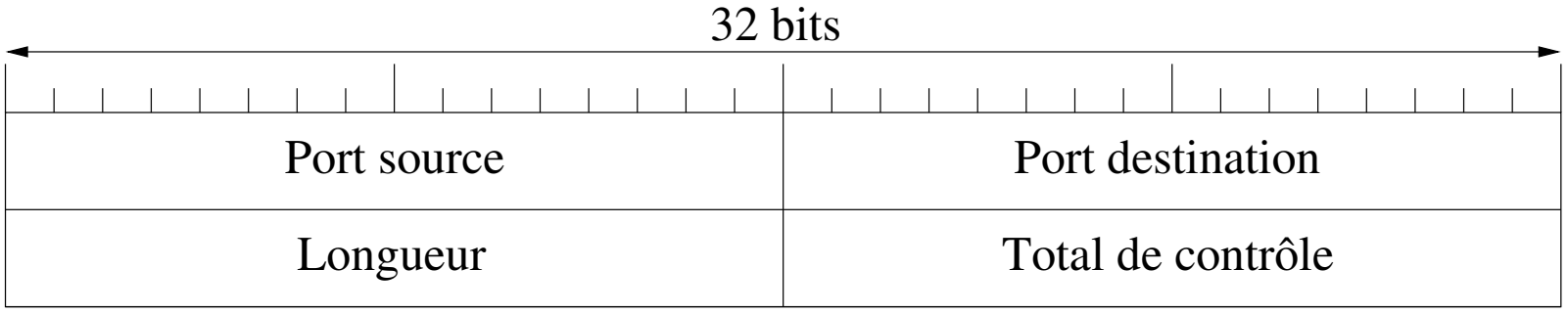
UDP — L'en-tête

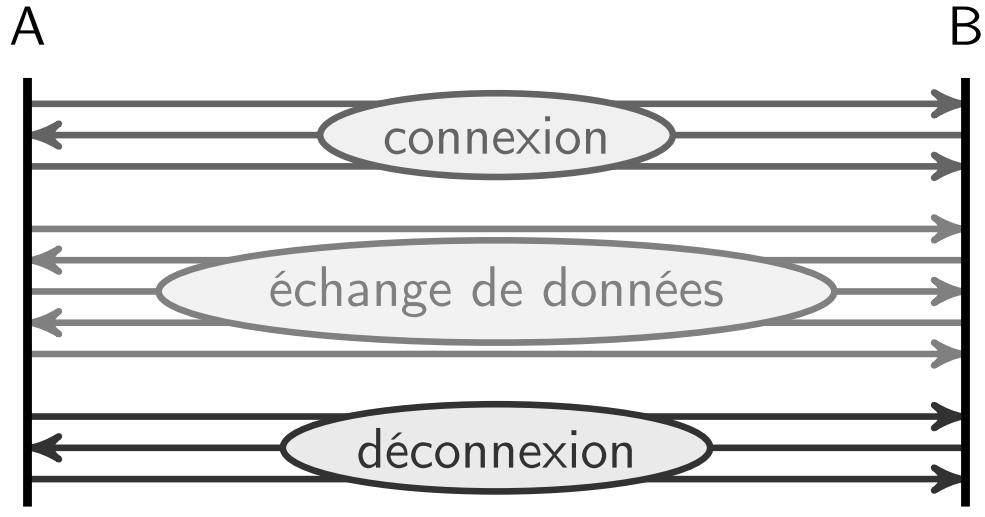
En-tête fixe de 8 octets :



- **Port source** (16 bits) = identifie le processus émetteur
- **Port destination** (16 bits) = identifie le processus destinataire
- **Longueur** (16 bits) = longueur totale du paquet (en-tête + données)
- **Total de contrôle** (16 bits) = code d'erreur calculé sur l'en-tête UDP + une partie de l'en-tête IP

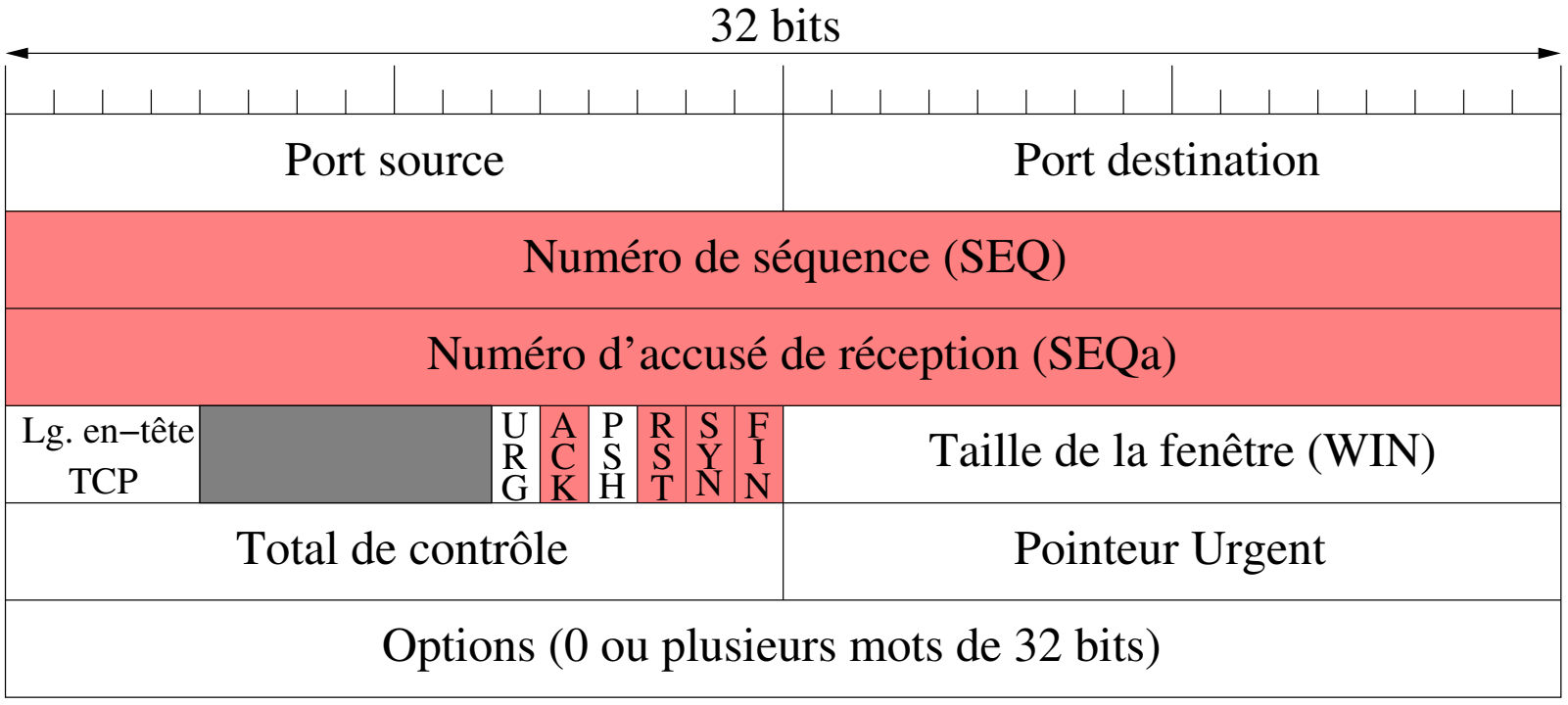
TCP — Déroulement d'une session

- TCP est un protocole en mode connecté :
  - phase de connexion avant tout échange de données
  - phase de déconnexion une fois l'échange terminé
- La phase de connexion sert :
  - à s'assurer que l'autre processus est prêt à communiquer ;
  - et à échanger des informations nécessaires à la suite de l'échange (dans le cas de TCP : des numéros de séquence).
- La phase de déconnexion sert à libérer des ressources (p.ex., de la mémoire allouée pour le contrôle de l'échange).



TCP — L'en-tête

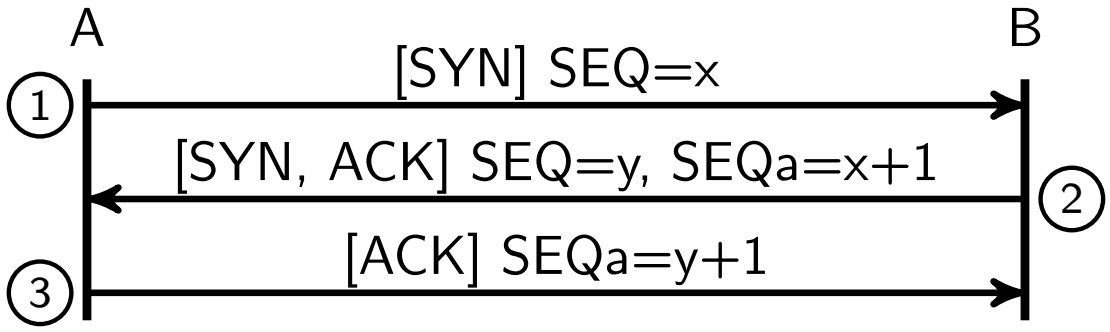
En-tête de 20 octets ou plus :



- **SEQ** (32 bits) = numéro du prochain octet de données envoyé
- **SEQa** (32 bits) = numéro du prochain octet de données attendu
- **ACK** (1 bit) = acquittement
- **RST** (1 bit, ReSeT) = refus de connexion ou déconnexion brutale
- **SYN** (1 bit, SYNchronisation) = demande de connexion
- **FIN** (1 bit, FINalisation) = demande de déconnexion

TCP — Connexion acceptée

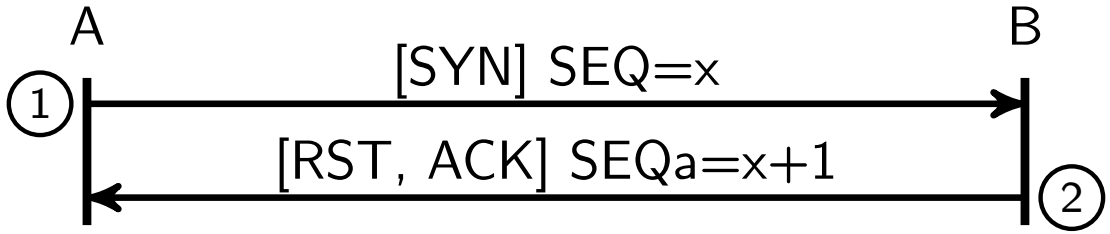
- connexion en 3 temps (three-way handshake)



1. A envoie un paquet
   - de demande de synchronisation (bit SYN=1) ;
   - et contenant un numéro de séquence initial choisi aléatoirement (SEQ=x).
2. B répond par un paquet
   - qui acquitte la demande de A (bit ACK=1 et SEQa=x+1) ;
   - et contient également une demande de synchronisation avec un numéro de séquence initial choisi aléatoirement (SEQ=y).
3. A répond par un paquet qui acquitte la demande de B.

TCP — Connexion refusée

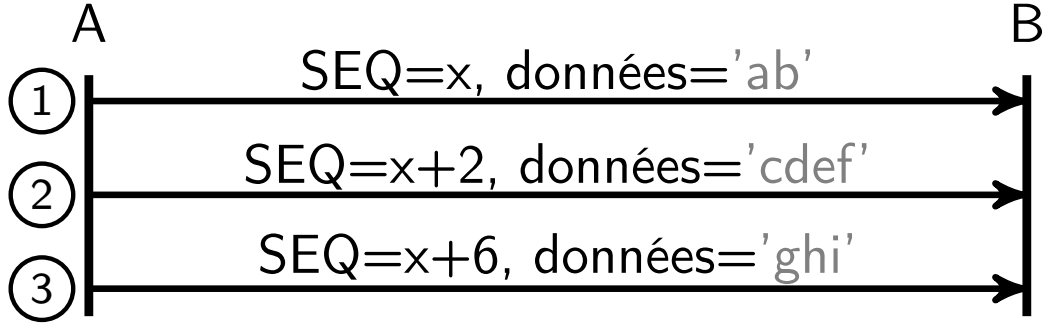
- aucun processus en écoute sur le port de réception du paquet SYN



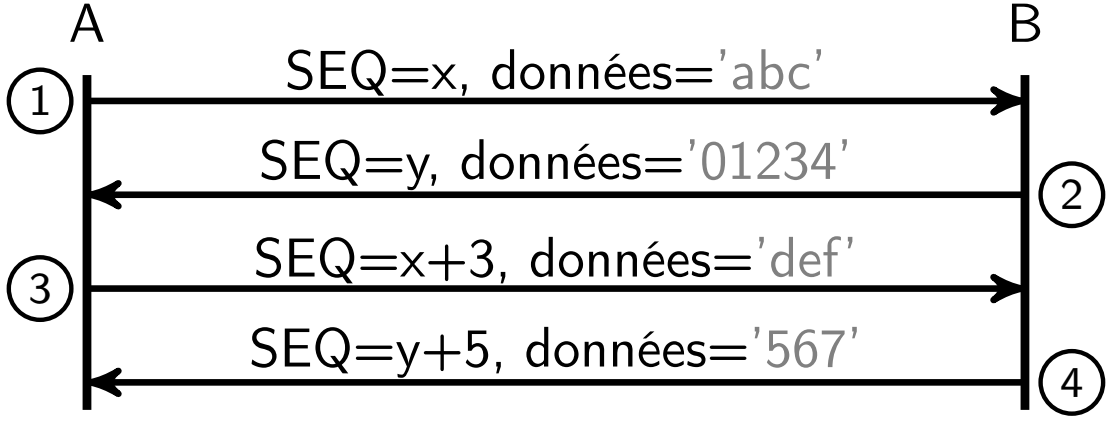
1. A envoie un paquet de demande de synchronisation.
2. B acquitte mais refuse la demande (bit RST=1).

TCP — Échange de données

- Les octets de données à envoyer sont numérotés.
- SEQ est le numéro de séquence du premier octet de données du paquet.
- Après l'envoi d'un paquet de données, le numéro de séquence est incrémenté du nombre d'octets de données envoyés.

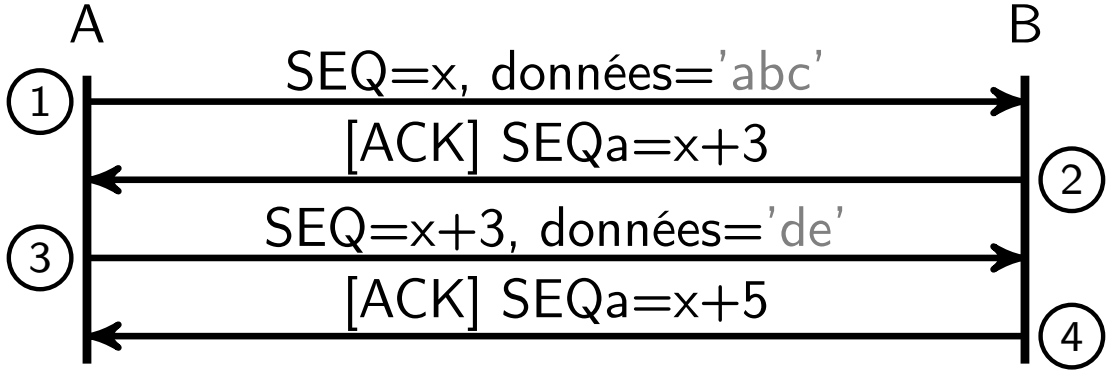


- communication bi-directionnelle ⇒ A et B ont des numéros de séquence indépendants.



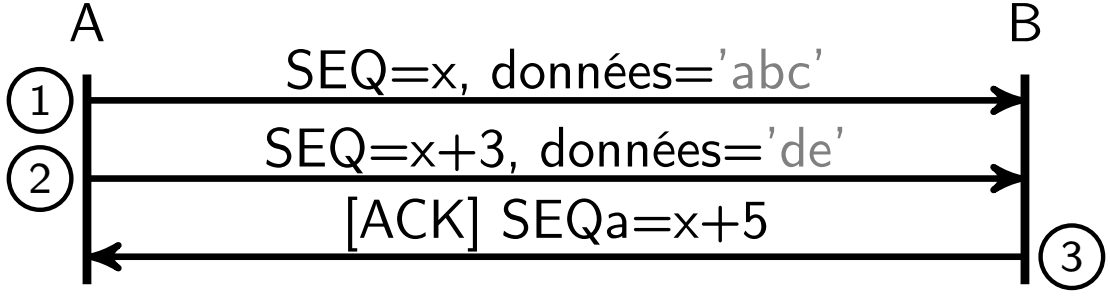
TCP — Acquittements

- Tout octet de données envoyé doit être acquitté.
- Cela se fait grâce au numéro de séquence acquitté (SEQa).



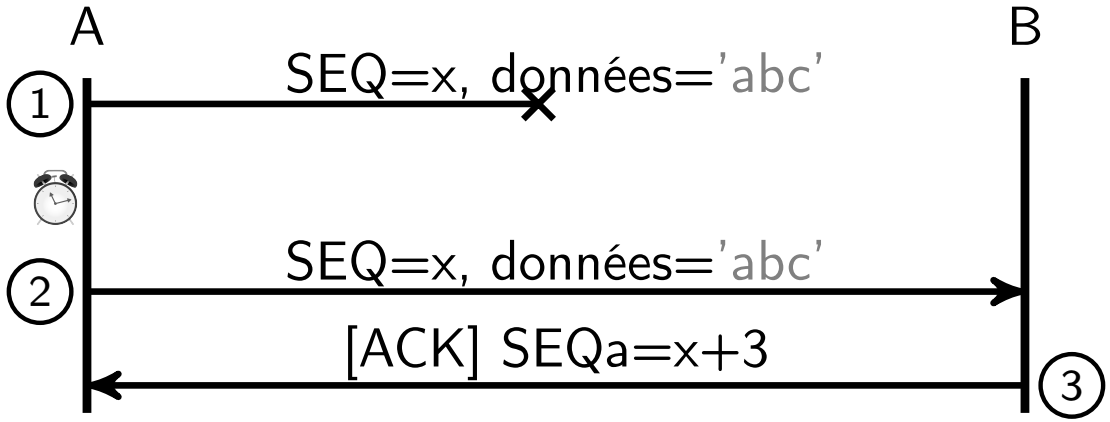
1. A envoie les octets x à $x + 2$.
2. B acquitte les octets reçus et indique que le prochain attendu est le $x + 3$.
3. A envoie les octets $x + 3$ à $x + 4$.
4. B acquitte les octets reçus et indique que le prochain attendu est le $x + 5$.

- On peut aussi acquitter plusieurs paquets avec un seul acquittement :



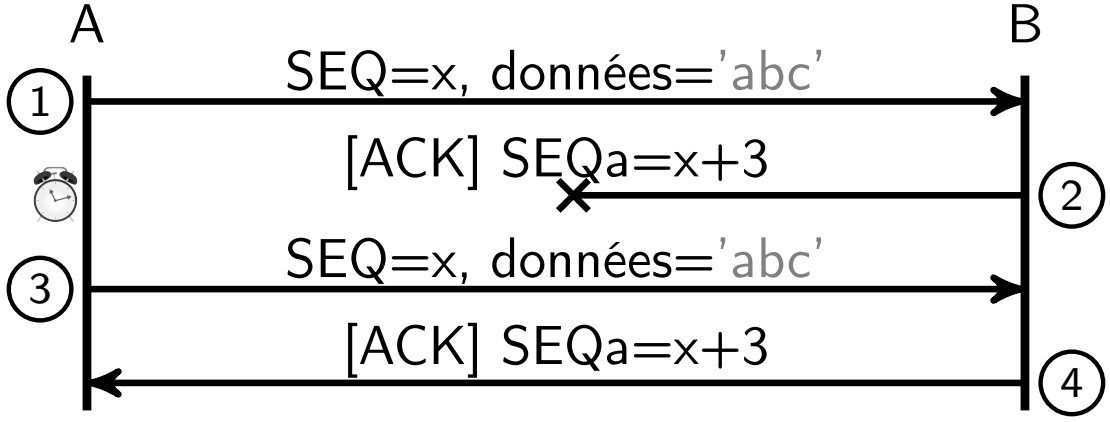
TCP — Temporisations

- utilisation de temporisation pour détecter les pertes de paquets
- Principe : passé un certain délai, si des données n'ont pas été acquittées, on les considère comme perdues et on retransmet.



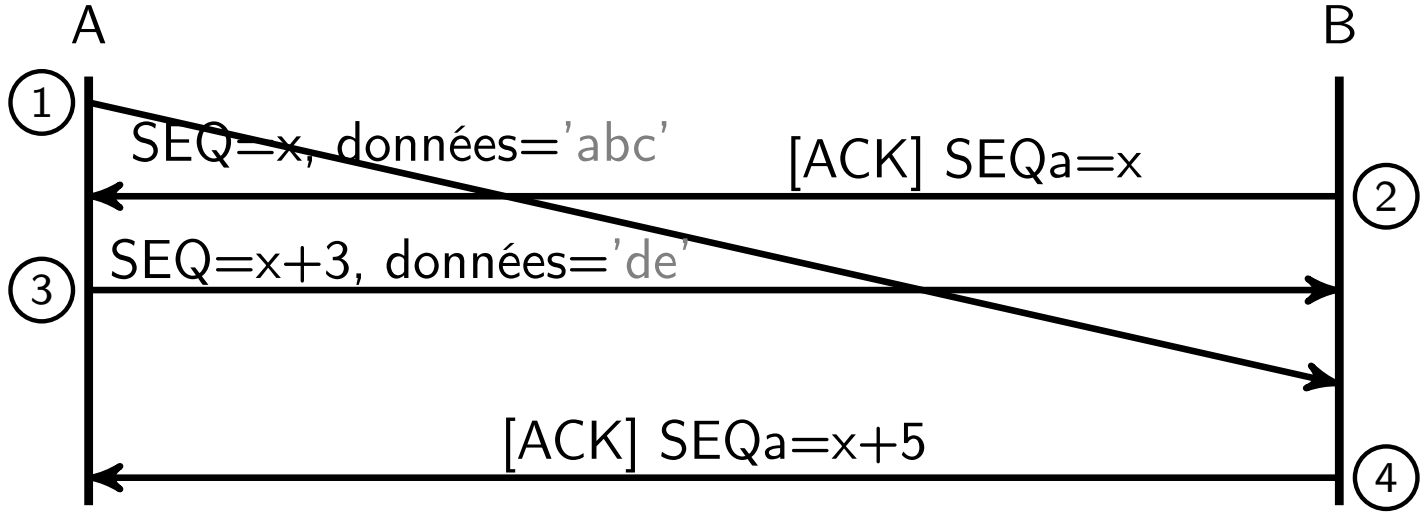
1. A envoie 3 octets de données.
2. Pas d'acquittement dans les temps ⇒ retransmission des 3 octets.

- Autre scénario avec une perte de l'acquittement :



TCP — Déséquencement

- Possibilité de mémoriser des octets de données non attendus.



1. A envoie 3 octets de données 'abc'.
2. B acquitte mais indique qu'il attend toujours l'octet x.
   Il mémorise les 3 octets.
3. A envoie 2 octets de données 'de'.
4. B acquitte les 5 octets reçus.

TCP — Déconnexion

- Tout octet de données doit avoir été acquitté avant la déconnexion.
- déconnexion en 3 temps