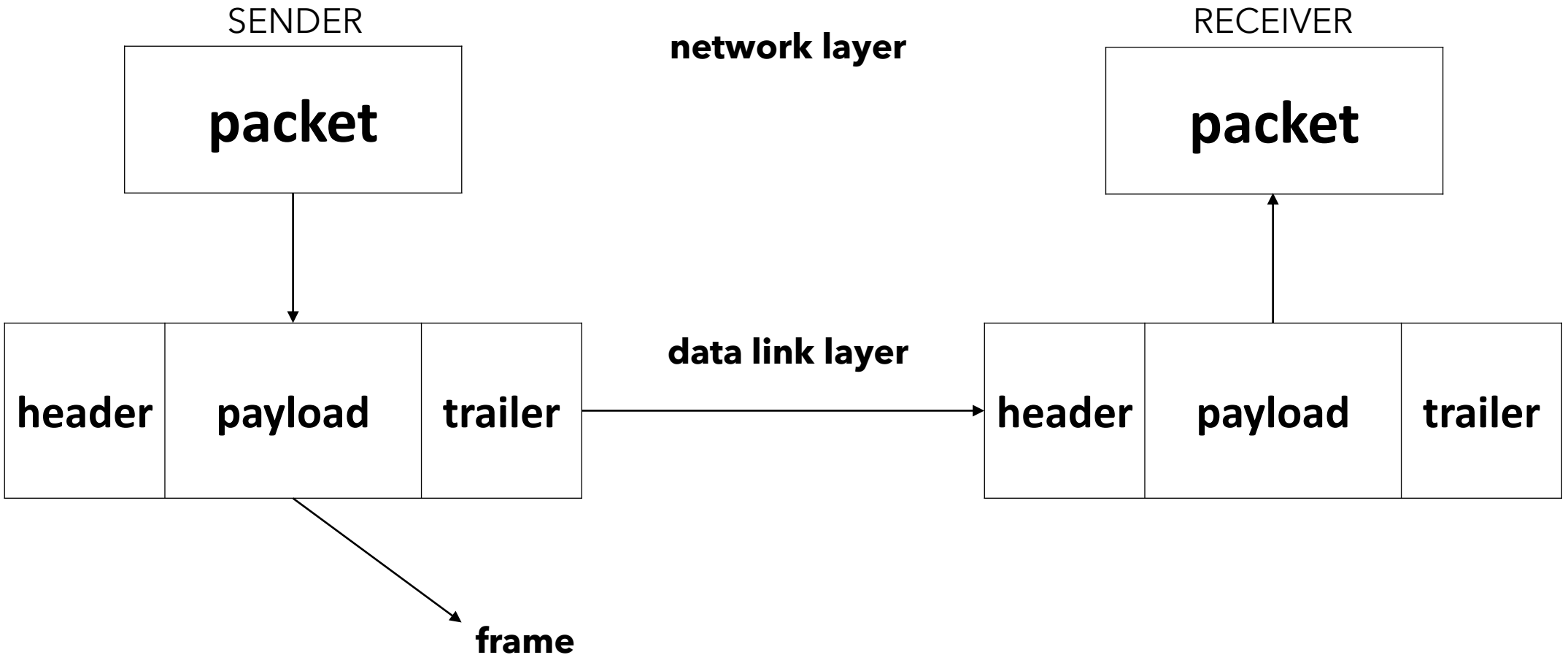
# Il data link layer (*layer 2*)
# Framing
# Controllo di flusso

**Liceo G.B. Brocchi – Bassano del Grappa (VI)**
**Liceo Scientifico – opzione scienze applicate**
Giovanni Mazzocchin

# Il livello data link

- Lo scopo del livello fisico è inviare singoli bit come segnali elettromagnetici
- Lo scopo del livello **data link** è trasmettere unità di informazione chiamate **frame** tra macchine adiacenti, ossia collegate tramite un mezzo trasmissivo (sia esso guidato o non guidato)
- I protocolli di questo livello sono implementati a livello di **NIC** (***N****etwork* ***I****nterface* ***C****ard* - hardware) o come driver del sistema operativo (software di sistema)
- I servizi principali offerti dal livello data link al livello superiore sono:
  1. framing
  2. controllo degli errori
  3. controllo di flusso

# Il livello data link

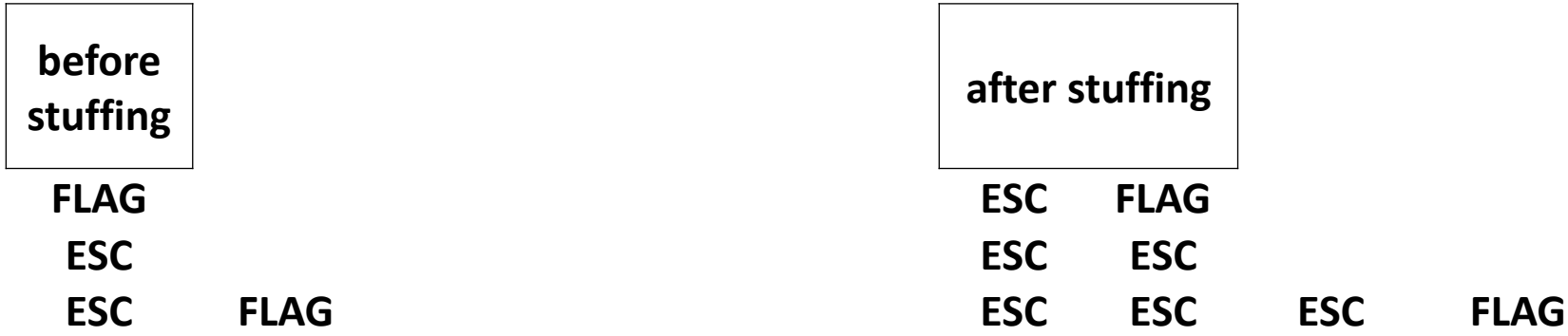


# Framing

- Il livello data link suddivide uno stream di bit in **frame** discreti
- Il destinatario di un frame deve essere in grado di individuarne esattamente i punti iniziale e finale
- Vediamo la tecnica **flag bytes with byte stuffing**

# Flag bytes with byte stuffing

- Con questa tecnica di framing, ciascun frame inizia e termina con un byte speciale, detto **flag byte**
- E se il flag byte fa parte dei dati del frame?
  - il livello data link del mittente inserisce un **escape byte** (ESC)
  - il livello data link del destinatario si occupa di rimuovere gli escape byte
  - questa tecnica prende il nome di **byte stuffing**



# Controllo di flusso (*flow control*)

- Il **controllo di flusso** è la gestione della velocità di trasmissione da mittente a destinatario
- Lo scopo del controllo di flusso è evitare che un mittente veloce inondi di dati un destinatario più lento, il quale non sarebbe in grado di processarli
- Se un mittente invia una quantità eccessiva di dati, la memoria (*buffer*) del destinatario potrebbe riempirsi, portando così a perdere dati

# Utopian simplex protocol

```
utopian_simplex_sender() {
  while (true) {
    packet p = get_from_network_layer();
    frame f = build_frame(p);
    send_to_physical_layer(f);
  }
}
```

**Funzionamento**: il mittente riceve un PDU (*packet*) dal livello superiore, costruisce il frame e lo invia. Il protocollo è *simplex* perché la trasmissione è unidirezionale

**Criticità**:

- il destinatario potrebbe non essere pronto ad accettare il frame (potrebbe essere più lento del mittente)
- il frame potrebbe essere perduto e non arrivare al destinatario. Con questo protocollo, il mittente non ha modo di sapere se la consegna è andata a buon fine

# Utopian simplex protocol

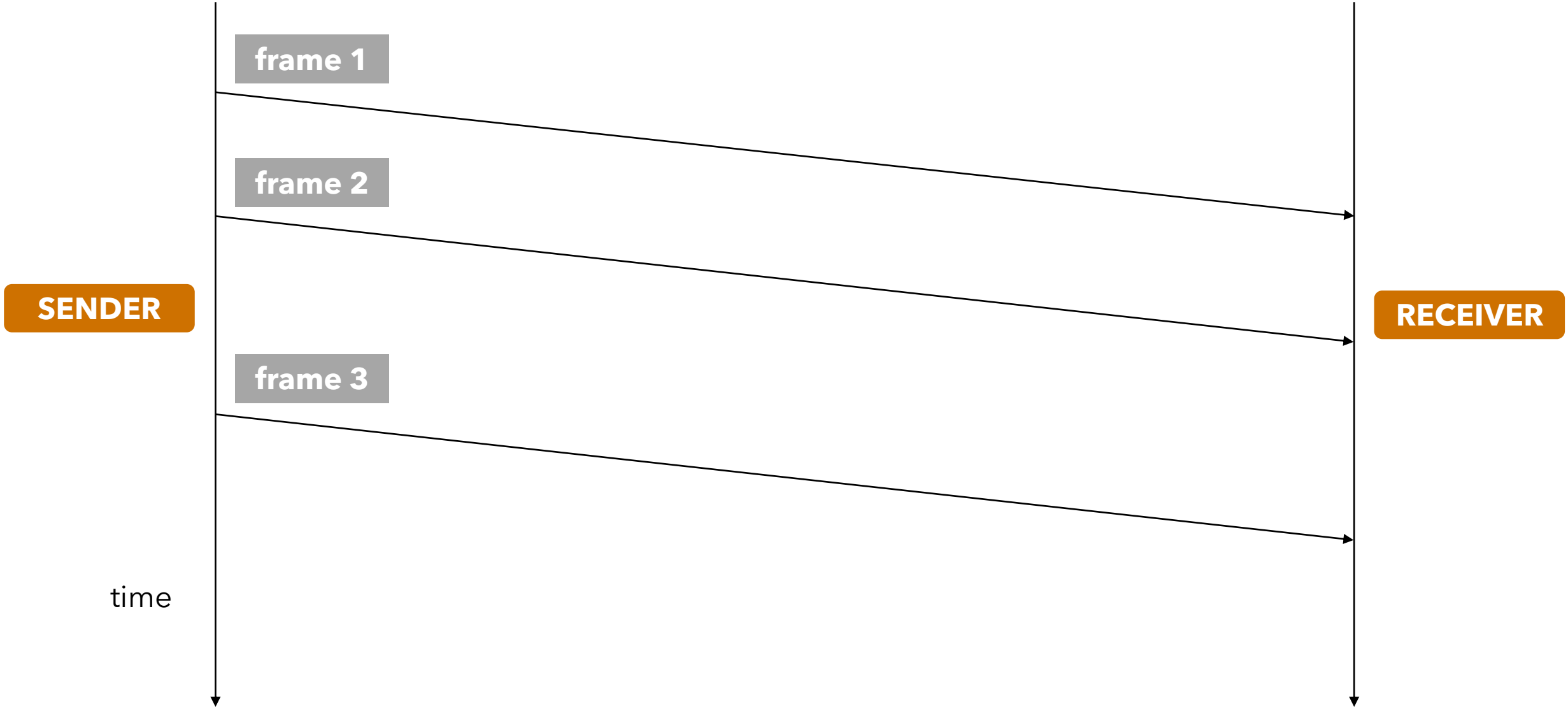
```
utopian_simplex_receiver() {
  while (true) {
    bits b = get_from_physical_layer();
    frame f = get_frame(b);
    send_to_network_layer(f);
  }
}
```

**Funzionamento**: il mittente riceve un PDU (*packet*) dal livello superiore, costruisce il frame e lo invia. Il protocollo è *simplex* perché la trasmissione è unidirezionale

**Criticità**:

- il destinatario potrebbe non essere pronto ad accettare il frame (potrebbe essere più lento del mittente)
- il frame potrebbe essere perduto e non arrivare al destinatario. Con questo protocollo, il mittente non ha modo di sapere se la consegna è andata a buon fine

# Utopian simplex protocol



# Stop-and-Wait protocol

```
stop_and_wait_sender() {
  while (true) {
    packet p = get_from_network_layer();
    frame f = build_frame(p);
    send_to_physical_layer(f);
    wait_for_ack();
  }
}
```

**Funzionamento**: il mittente riceve un PDU (*packet*) dal livello superiore, costruisce il frame e lo invia. Si mette in attesa di un riscontro da parte del destinatario (*acknowledgement* - **ACK**). Una volta ricevuto il riscontro, può inviare il prossimo frame

**Criticità**:

- cosa succede se un frame viene perduto?
- cosa succede se un ACK viene perduto?
- il mittente è costretto a inviare un frame alla volta

# Stop-and-Wait protocol

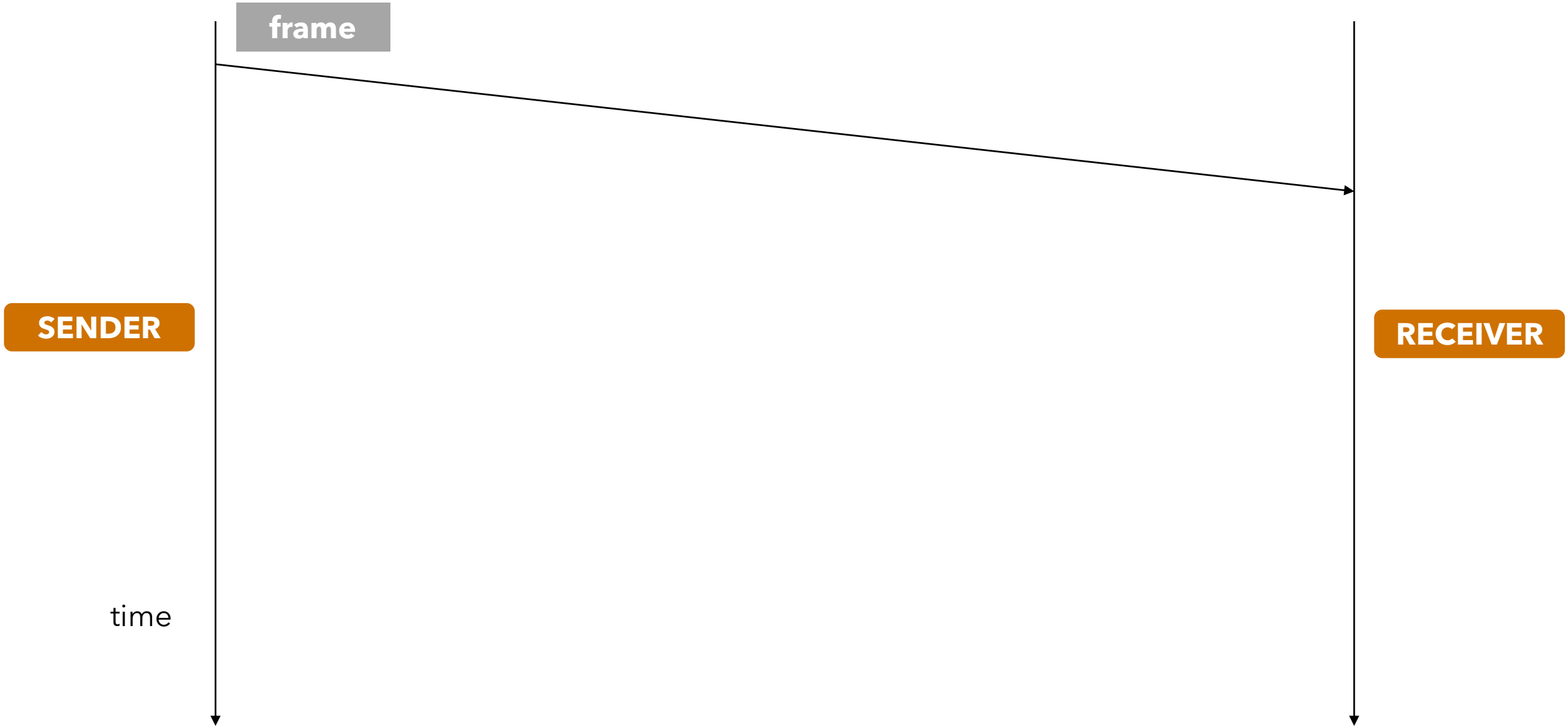
```
stop_and_wait_receiver() {
  while (true) {
    bits b = get_from_physical_layer();
    frame f = get_frame(b);
    send_to_network_layer(f);
    send_ack();
  }
}
```

**Funzionamento**: il mittente riceve un PDU (*packet*) dal livello superiore, costruisce il frame e lo invia. Si mette in attesa di un riscontro da parte del destinatario (*acknowledgement* - **ACK**). Una volta ricevuto il riscontro, può inviare il prossimo frame

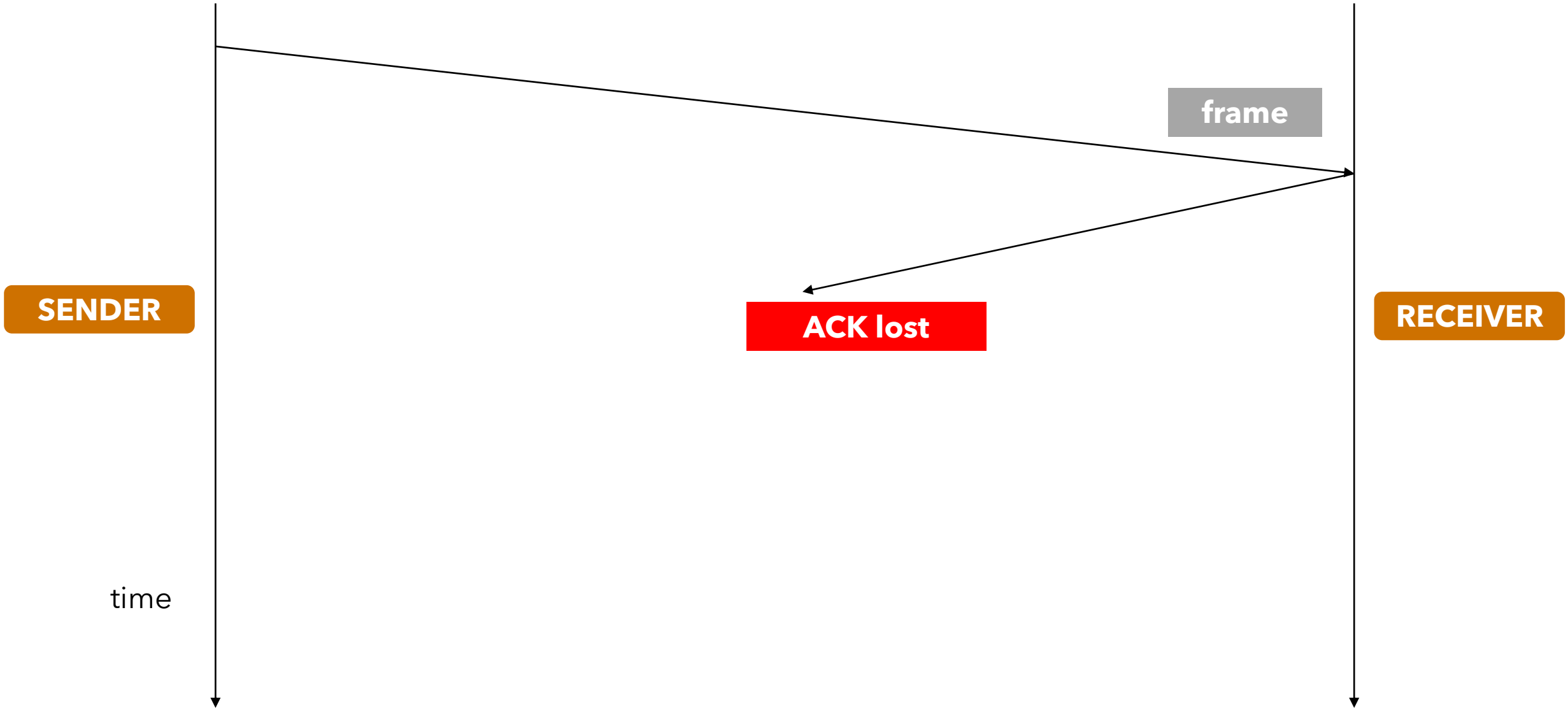
**Criticità**:

- cosa succede se un frame viene perduto?
- cosa succede se un ACK viene perduto?
- il mittente è costretto a inviare un frame alla volta

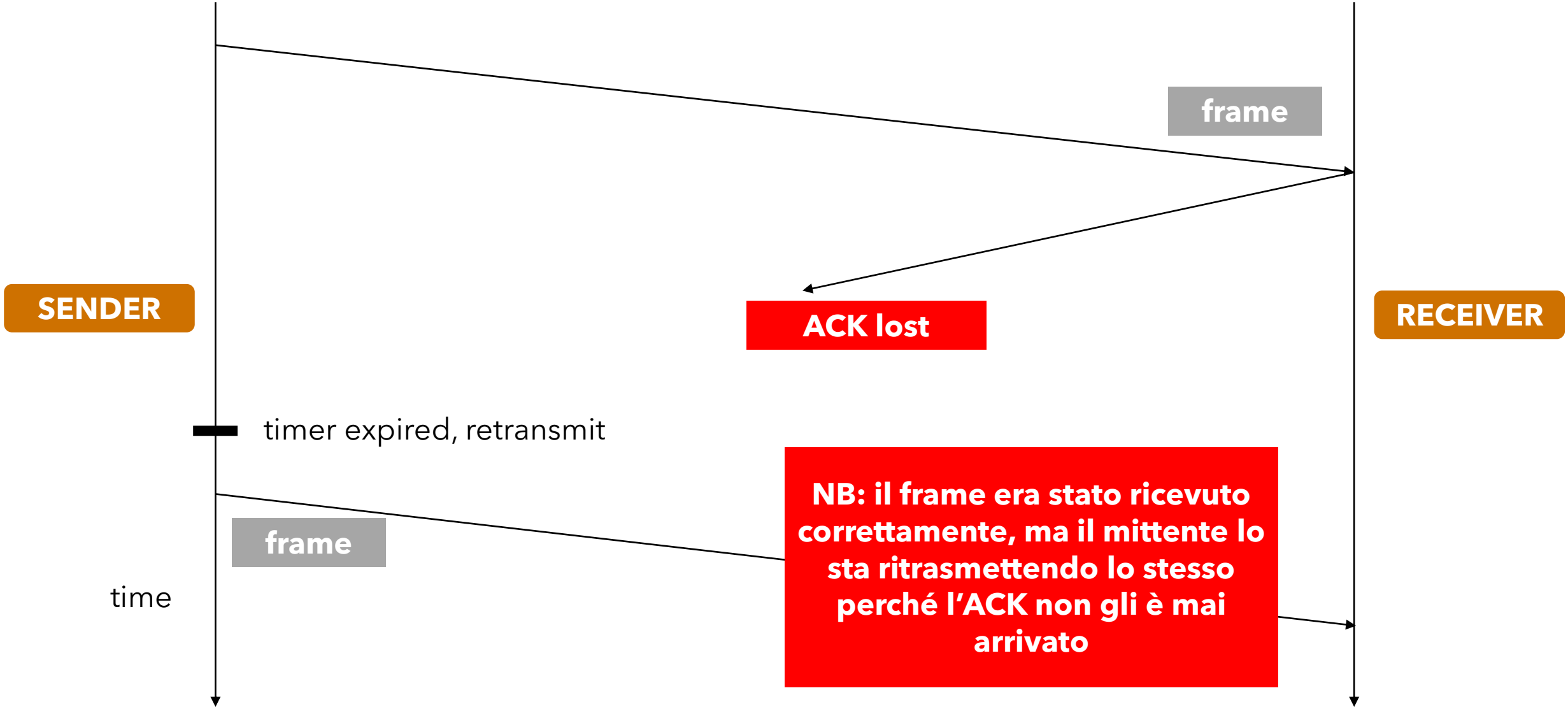
# Stop-and-Wait protocol - frame duplicati



# Stop-and-Wait protocol - frame duplicati



# Stop-and-Wait protocol - frame duplicati



# Stop-and-Wait protocol

```
stop_and_wait_sender() {
  while (true) {
     packet p = get_from_network_layer();
     frame f = build_frame(p);
     send_to_physical_layer(f);
     wait_for_ack();
  }
}
```

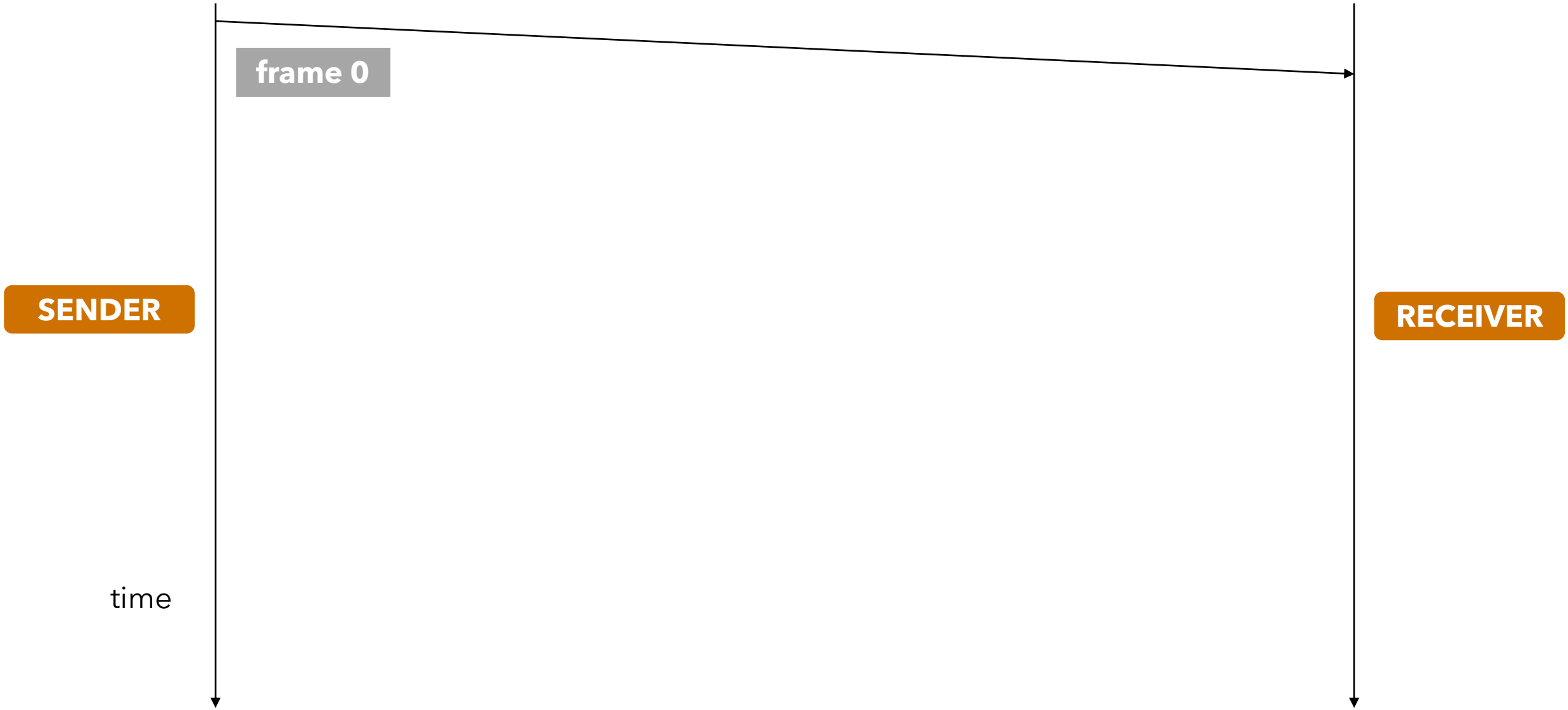
- il destinatario dovrebbe essere in grado di distinguere un nuovo pacchetto dalla ritrasmissione di uno vecchio
- con questo protocollo non è possibile operare questa distinzione
- bisognerebbe aggiungere un **sequence number** all'header del frame

# S&W – ARQ (Automatic Repeat reQuest)

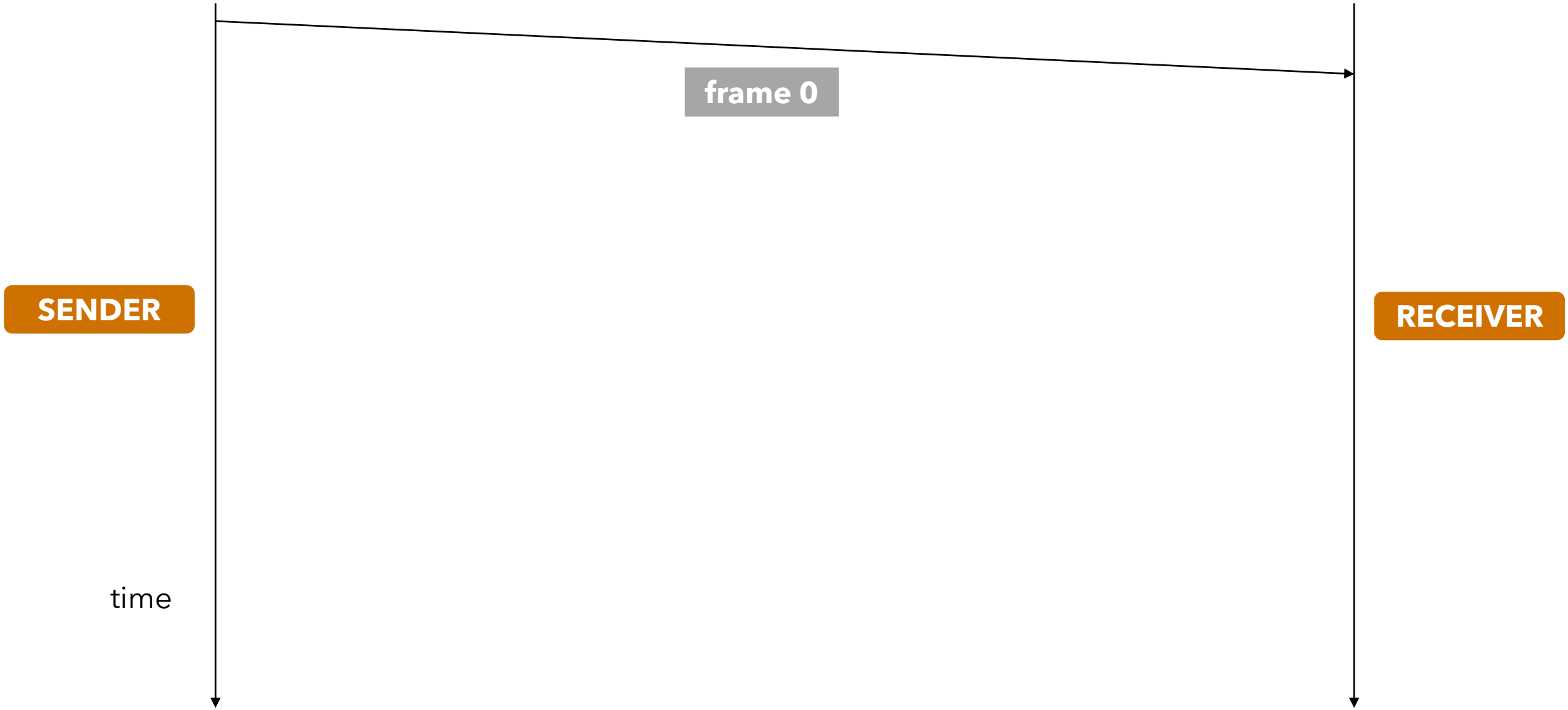
- **Domanda**: con i protocolli che stiamo vedendo, quanti frame transitano contemporaneamente sul canale?
- **Risposta**: se prima di inviare il prossimo frame bisogna aspettare l'ACK, allora transita 1 frame alla volta
- Per risolvere il problema dei duplicati, potremmo numerare i frame così (è sufficiente aggiungere un 1 bit al frame):
  - `frame 0`
  - `frame 1`
  - `frame 0`
  - `frame 1`
  - `frame 0`
  - …

**l'identificativo numerico del frame viene detto sequence number**

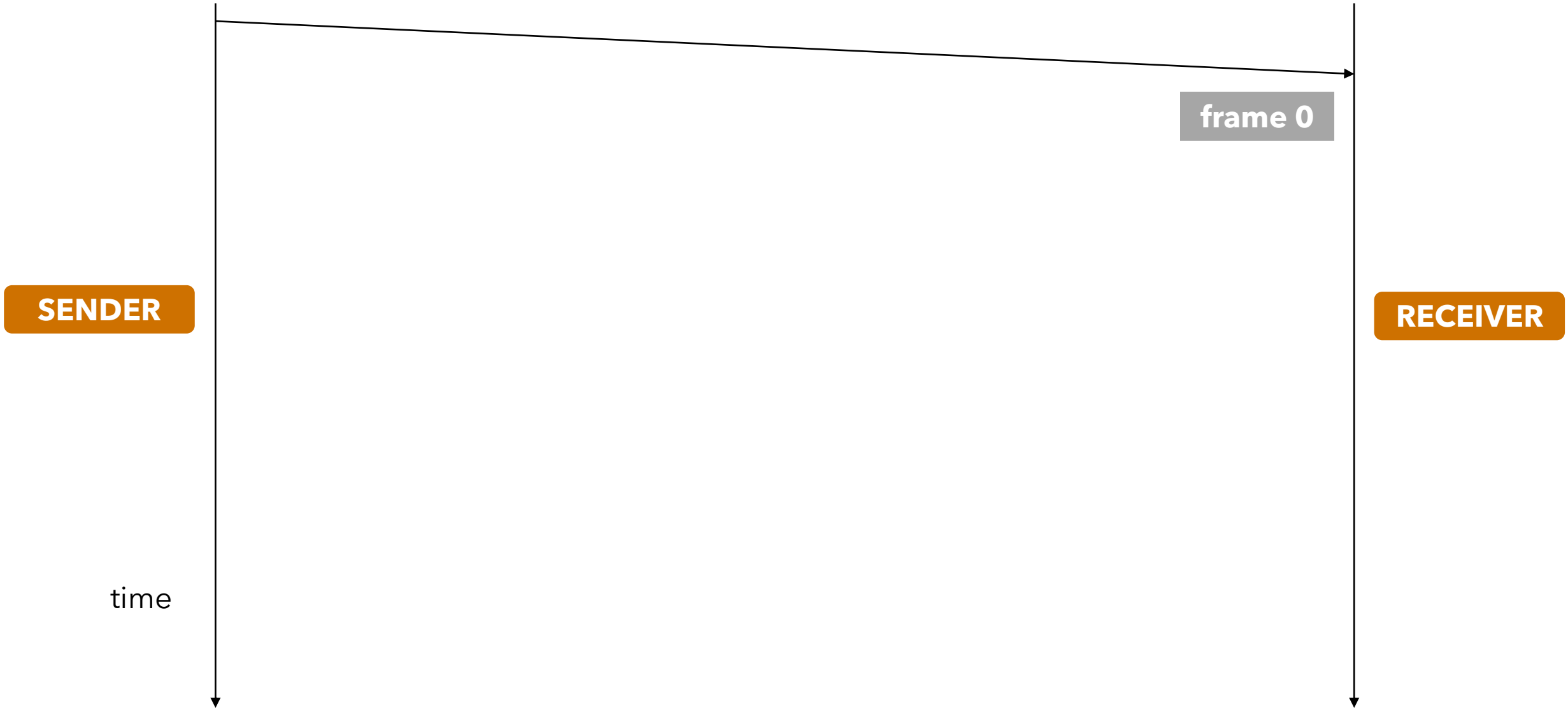
# S&W – ARQ (Automatic Repeat reQuest)



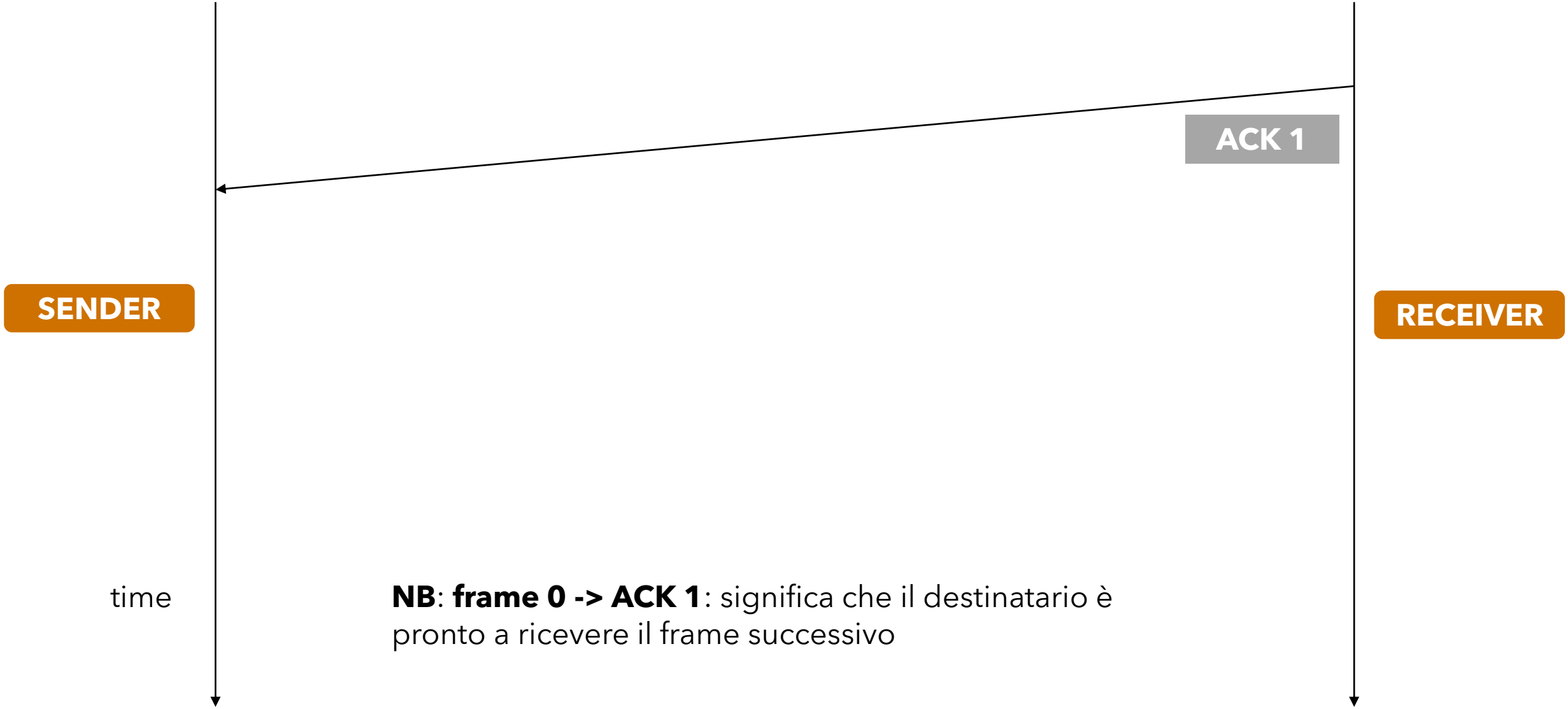
# S&W – ARQ (Automatic Repeat reQuest)



# S&W – ARQ (Automatic Repeat reQuest)



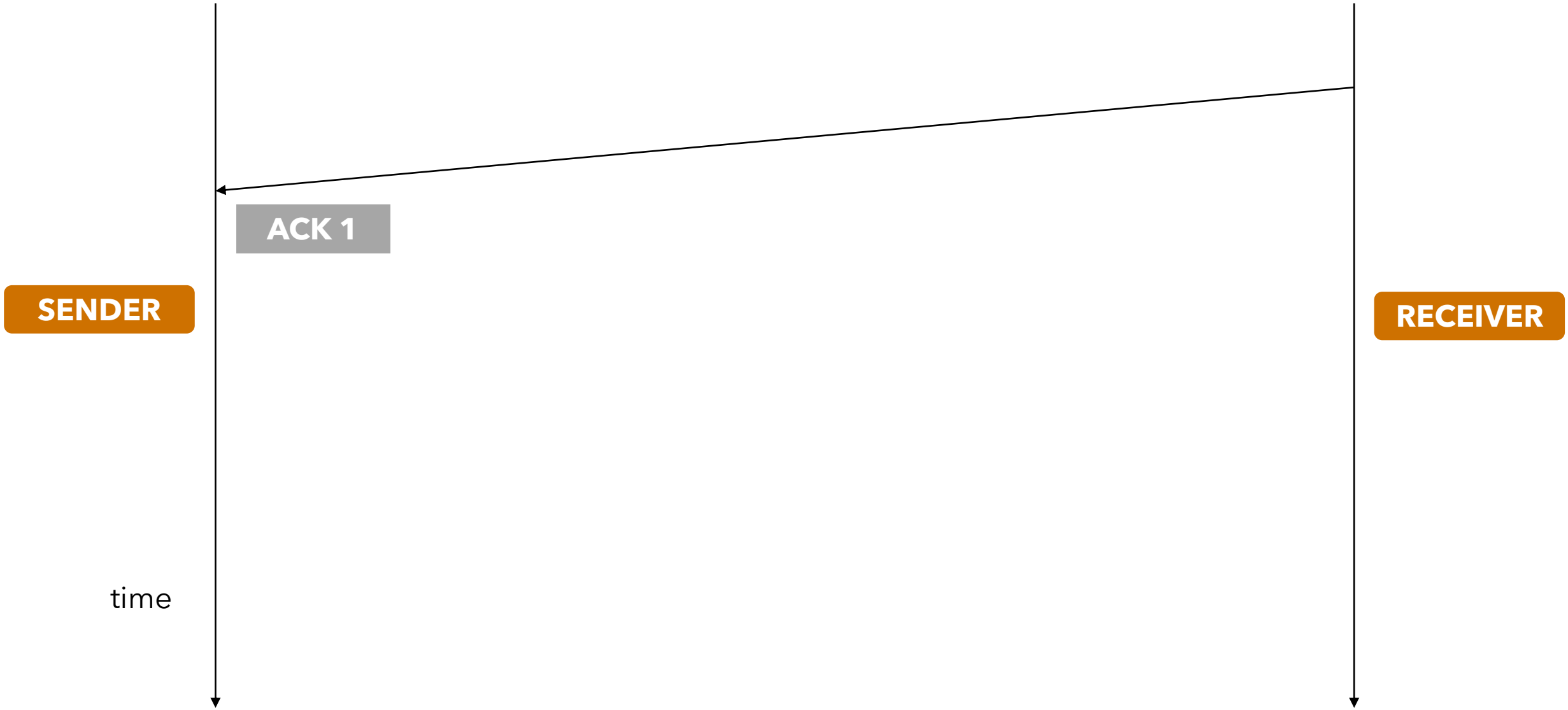
# S&W – ARQ (Automatic Repeat reQuest)



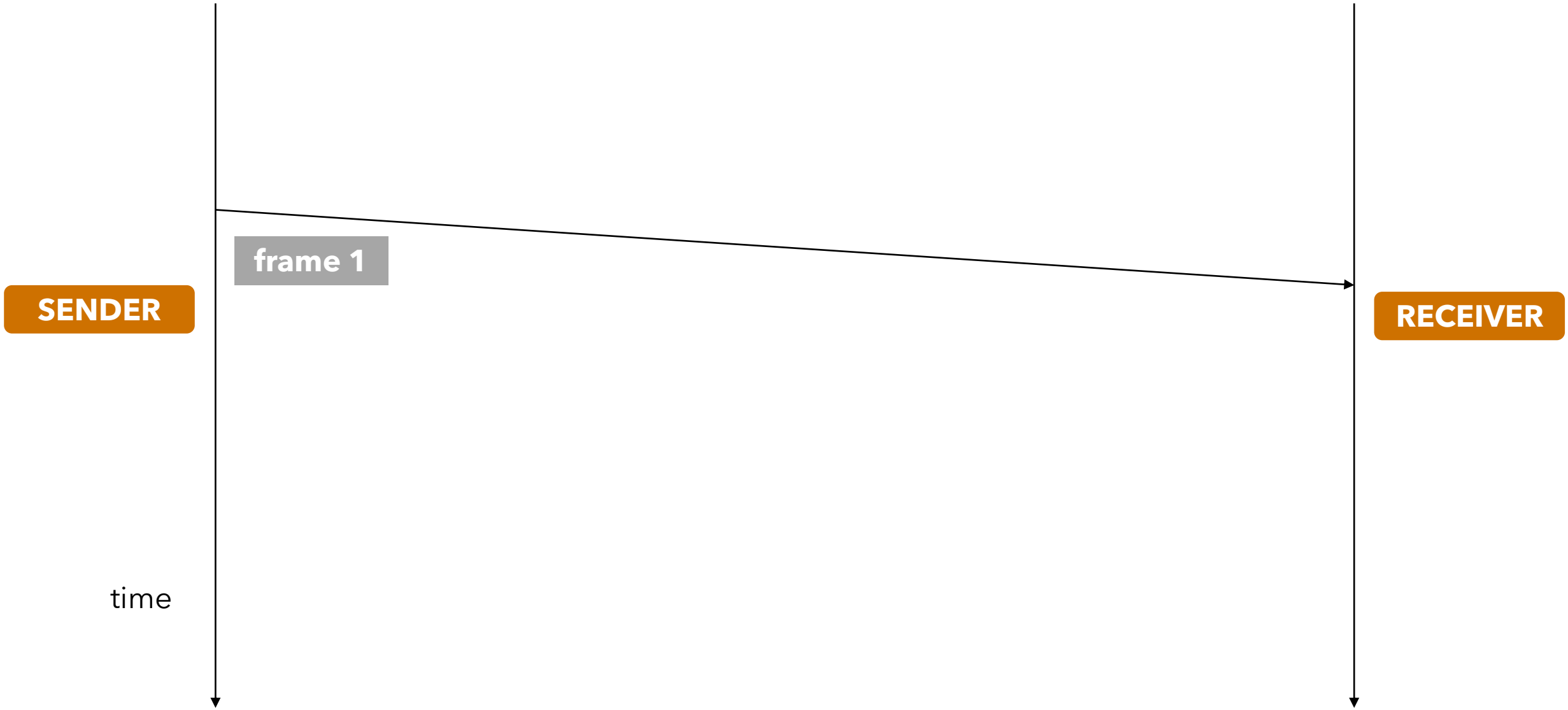

**NB**: **frame 0 -> ACK 1**: significa che il destinatario è pronto a ricevere il frame successivo

# S&W – ARQ (Automatic Repeat reQuest)

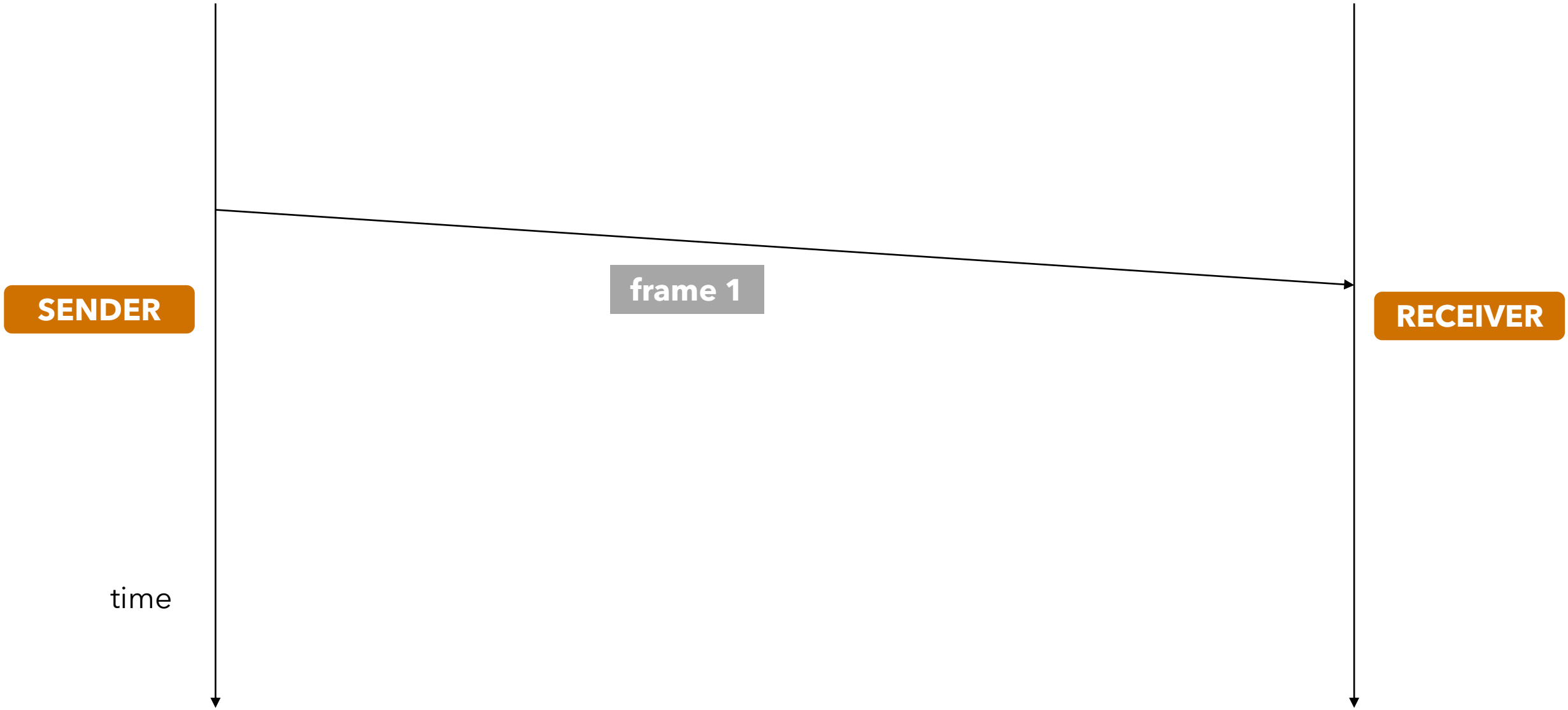
# S&W – ARQ (Automatic Repeat reQuest)



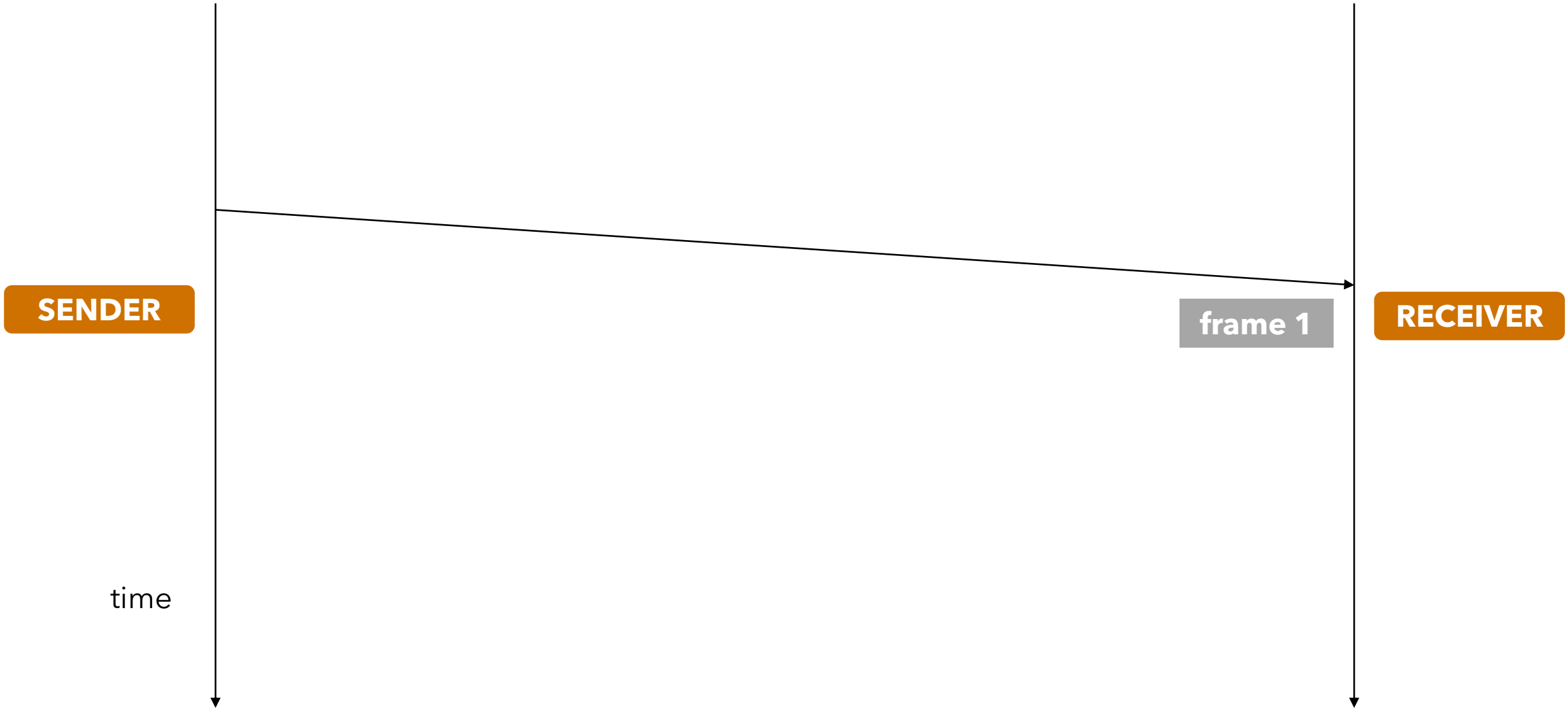
# S&W – ARQ (Automatic Repeat reQuest)



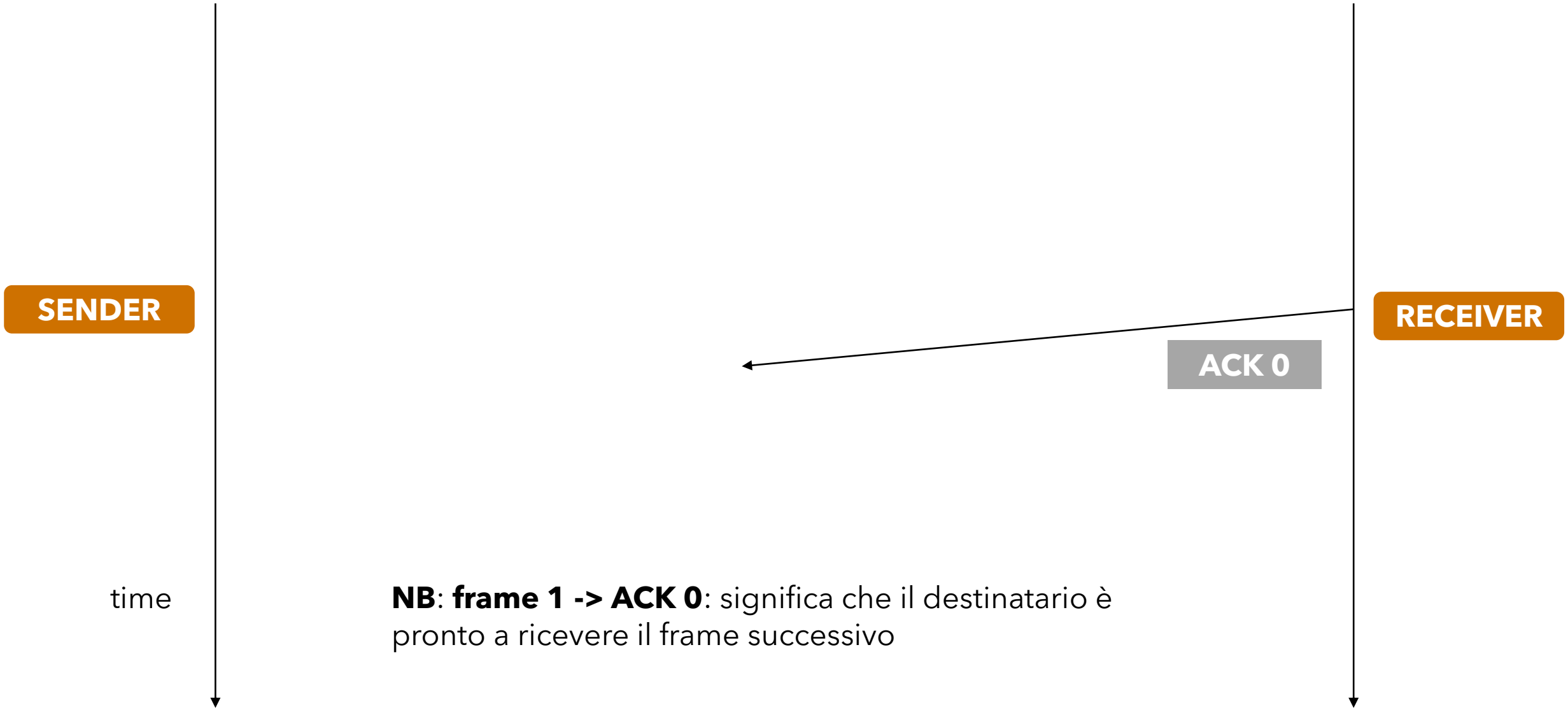
# S&W – ARQ (Automatic Repeat reQuest)



# S&W – ARQ (Automatic Repeat reQuest)

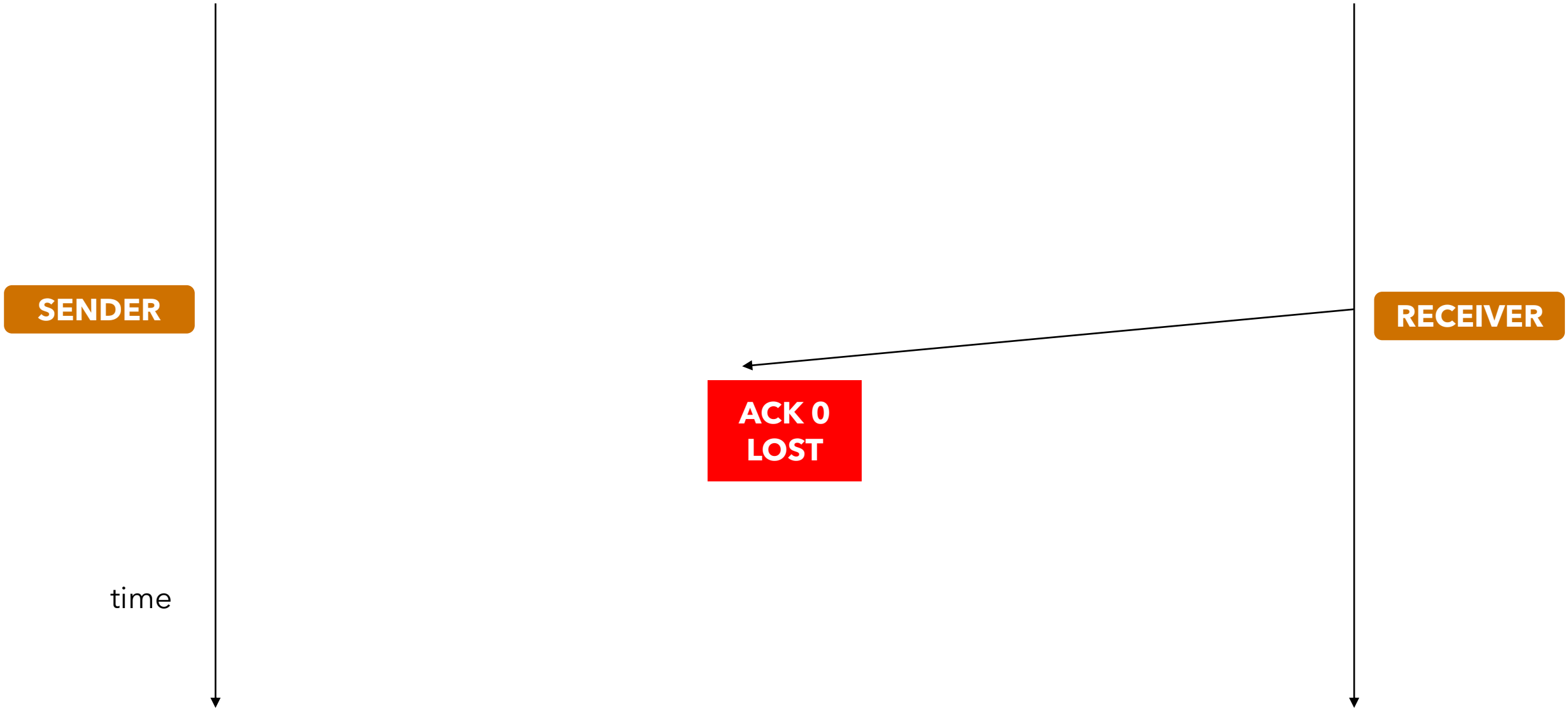


# S&W – ARQ (Automatic Repeat reQuest)

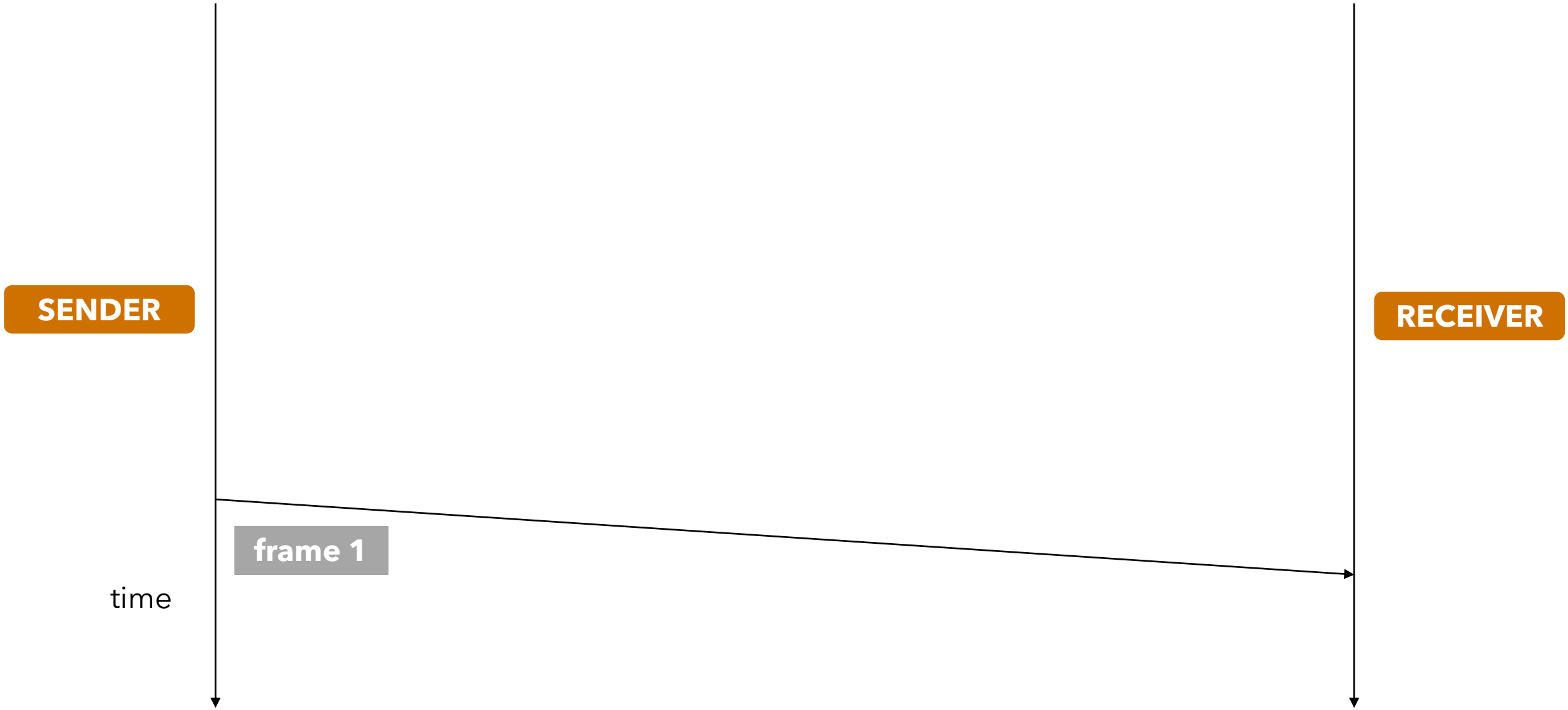



**NB**: **frame 1 -> ACK 0**: significa che il destinatario è pronto a ricevere il frame successivo

# S&W – ARQ (Automatic Repeat reQuest)



# S&W – ARQ (Automatic Repeat reQuest)



# S&W – ARQ (Automatic Repeat reQuest)