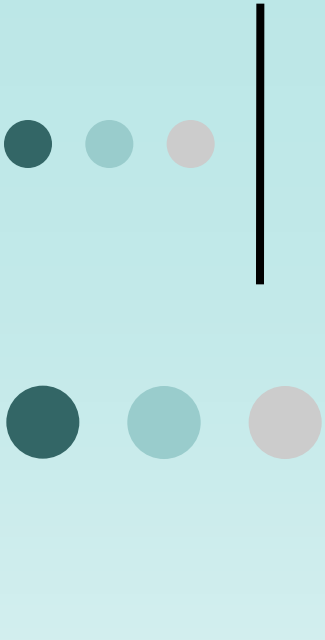
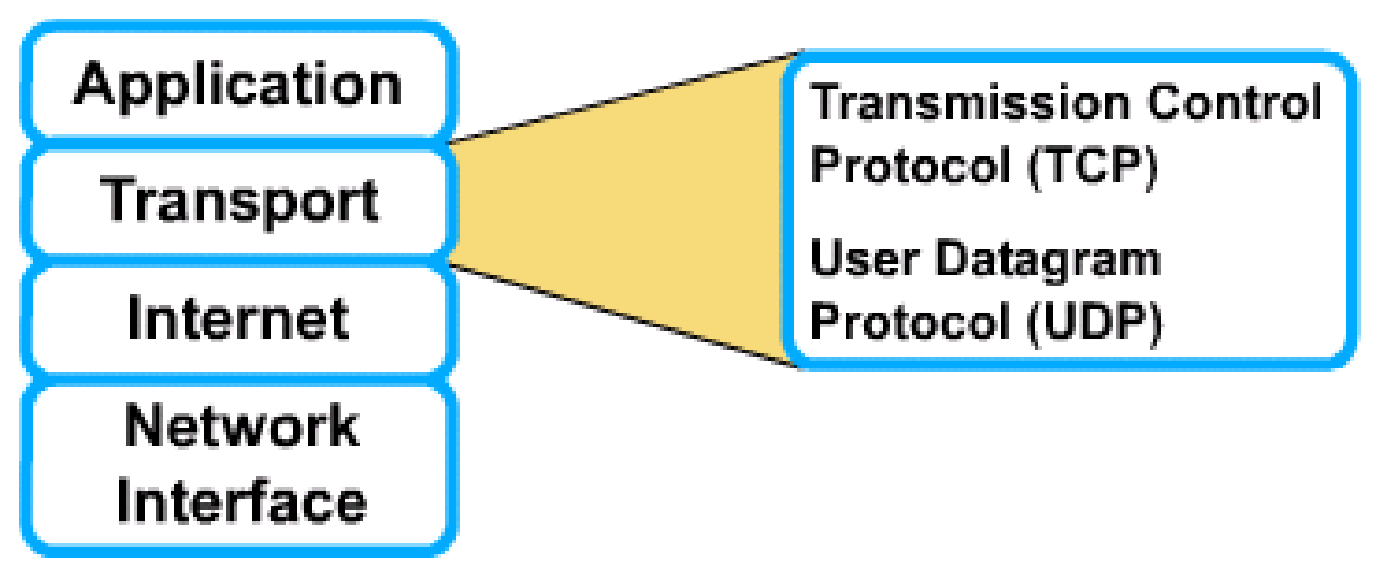


# Transport Layer Protocol

# Pendahuluan

- Protokol pada Transport Layer TCP/IP terdiri atas : TCP dan UDP.



# Pendahuluan

- UDP adalah protokol yang sifatnya *unreliable* dan *connectionless*.
- TCP adalah protokol yang sifatnya *reliable* dan *connection-oriented*.
- Setiap proses pada aplikasi harus mendefinisikan protokol transport mana yang akan digunakan. Contoh: HTTP, FTP, SMTP menggunakan TCP, sedangkan DNS, Internet Telephony menggunakan UDP.

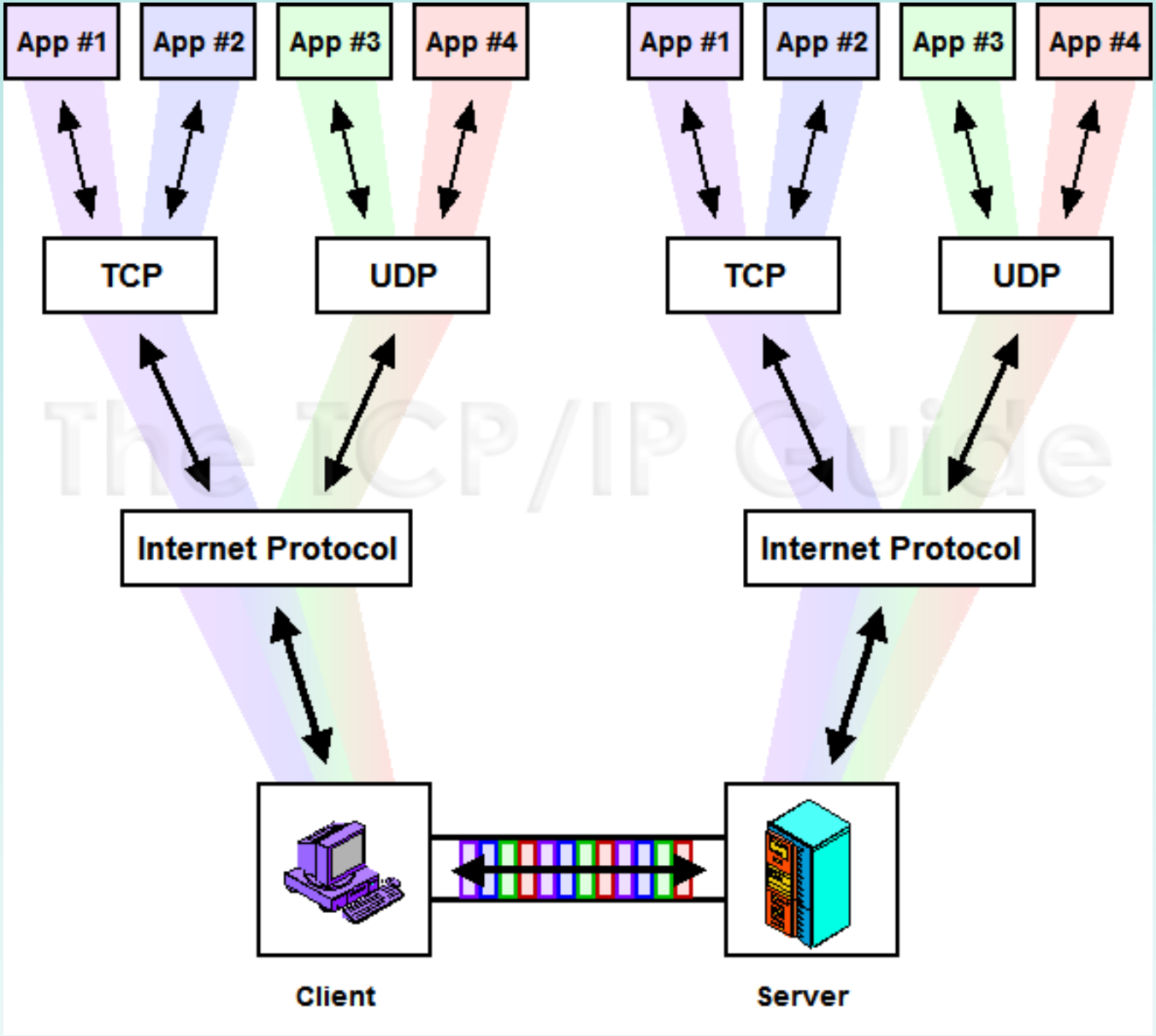
# Pendahuluan

- Sebuah host dapat memberikan layanan lebih dari satu proses, hal ini terjadi karena Transport Layer mampu memberikan layanan *multiplexing* dan *demultiplexing*.
- TCP merupakan reliable data transfer karena TCP menjamin pengiriman pesan sampai ditempat tujuan melalui: flow control, congestion control, acknowledgment, timer.

# Multiplexing - Demultiplexing

- Dalam jaringan TCP/IP beberapa proses dapat dikirimkan secara bersama-sama dari sebuah host melalui multiplexing.
- Pada sisi penerima, demultiplexing mengijinkan alokasi pesan pada proses yang sesuai. Lihat Gambar.
- Seperti diketahui, masing-masing proses dibedakan berdasarkan nomor Port.

# Ilustrasi Multiplexing-Demultiplexing




*Source: www.tcpipguide.com*

# User Datagram Protocol

- Didefiniskan dalam RFC 768.
- *Pada sisi pengirim*:

  - UDP mengambil pesan dari proses aplikasi,

  - menambahkan sumber dan tujuan port (untuk multiplexing/demultiplexing),

  - menambahkan beberapa header kecil,

  - mengirimkan segment tersebut ke network layer untuk dikirimkan.

# User Datagram Protocol

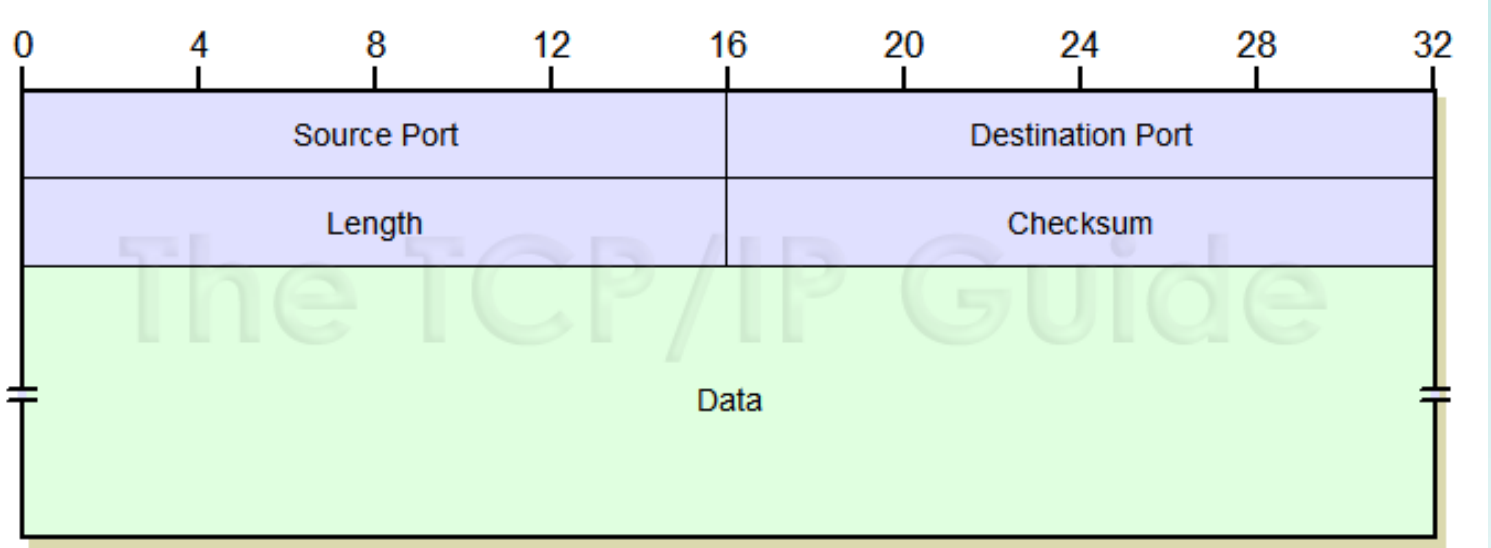
- Pada sisi penerima:

  - Segment tiba di sisi penerima,

  - UDP mengidentifikasi alamat port tujuan untuk dapat mengalokasikan pesan pada proses yang sesuai.

- Terlihat bahwa UDP tidak menggunakan proses inisialisasi koneksi sebelum sebuah segment dikirimkan. Karena itu UDP disebut sebagai connectionless protocol.

# Struktur Segment UDP




*Source: www.tcpipguide.com*

# Alasan Menggunakan UDP

- Tidak ada proses penetapan (establishment) koneksi → memperkecil delay.
- Memiliki jumlah header lebih kecil daripada TCP → memperkecil delay, mempercepat proses transmisi.
- Tidak ada kondisi-kondisi tertentu yang harus dilakukan, misalnya congestion control, sequence, acknowledgment seperti halnya pada TCP → memperkecil delay.

# UDP Checksum

- Misalkan terdapat 3 buah 16-bit pesan sbb:

  0111010101011101

  0001101101100110

  0101100101011011

- Ketiga pesan dijumlahkan menjadi: 1110101000011110
- Hasil jumlahan ini dioperasikan melalui 1's komplemen menghasilkan *checksum*: 0001010111100001.

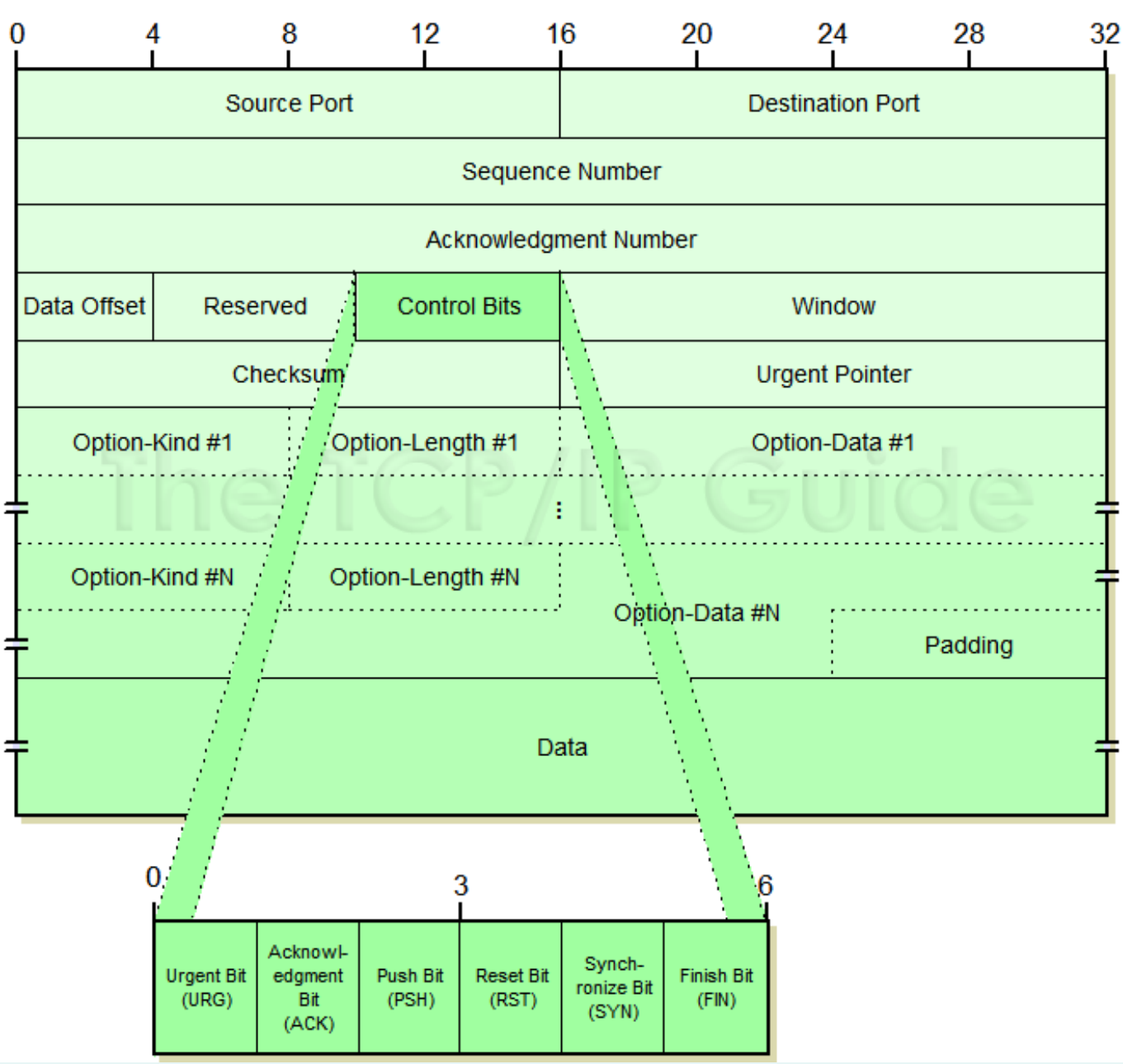
# UDP Checksum

- Pada sisi penerima semua 16-bit pesan dijumlahkan bersama-sama dengan checksum.
- Apabila tidak ada kesalahan bit pada saat pengiriman hasil penjumlahan akan menjadi: 1111111111111111.
- *UDP mendeteksi kesalahan, tetapi tidak mengkoreksi kesalahan.*

# Transport Control Protocol

- TCP adalah connection-oriented protocol.
- TCP menjamin bahwa data yang dikirimkan pasti diterima oleh receiver dengan benar (reliable data transfer).
- TCP memberi layanan komunikasi *full-duplex*.
- Koneksi pada TCP merupakan koneksi *point-to-point*.

# Struktur Segment TCP




*Source: www.tcpipguide.com*

# Model Koneksi TCP

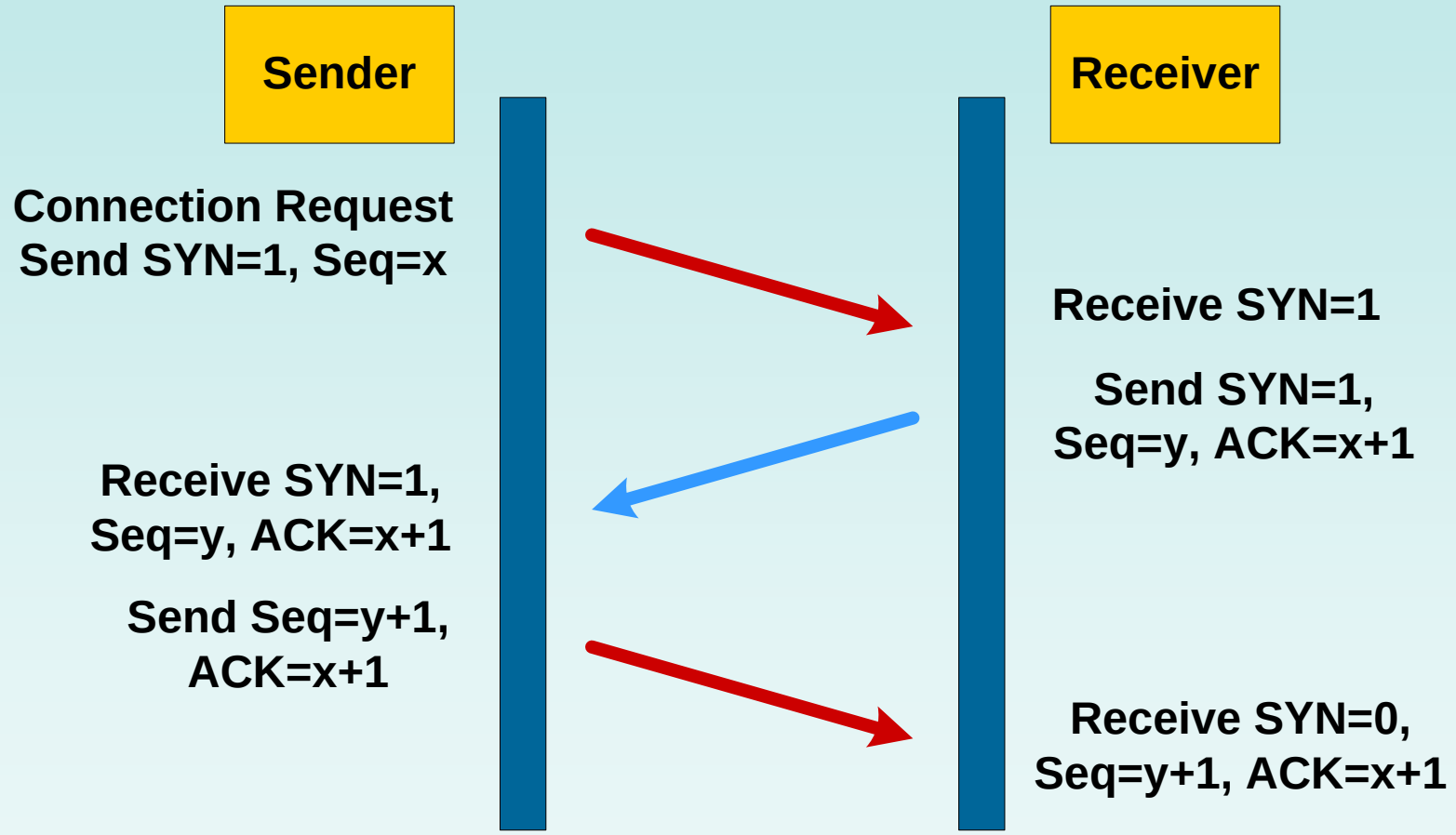
- Model koneksi TCP protocol terdiri atas 3 fase, yaitu:

  - Fase penetapan (establishment) koneksi.

  - Fase transaksi pesan.

  - Fase penutupan koneksi.

# Fase Penetapan Koneksi TCP



# Fase Penetapan Koneksi TCP

- Fase penetapan koneksi TCP disebut sebagai: ***three-way handshake***.
- Tahapan:

  1. Sender mengirimkan TCP segment dengan nilai SYN=1. Sender juga mengirimkan *informasi sequence number (isn)* yang digenerate secara random.

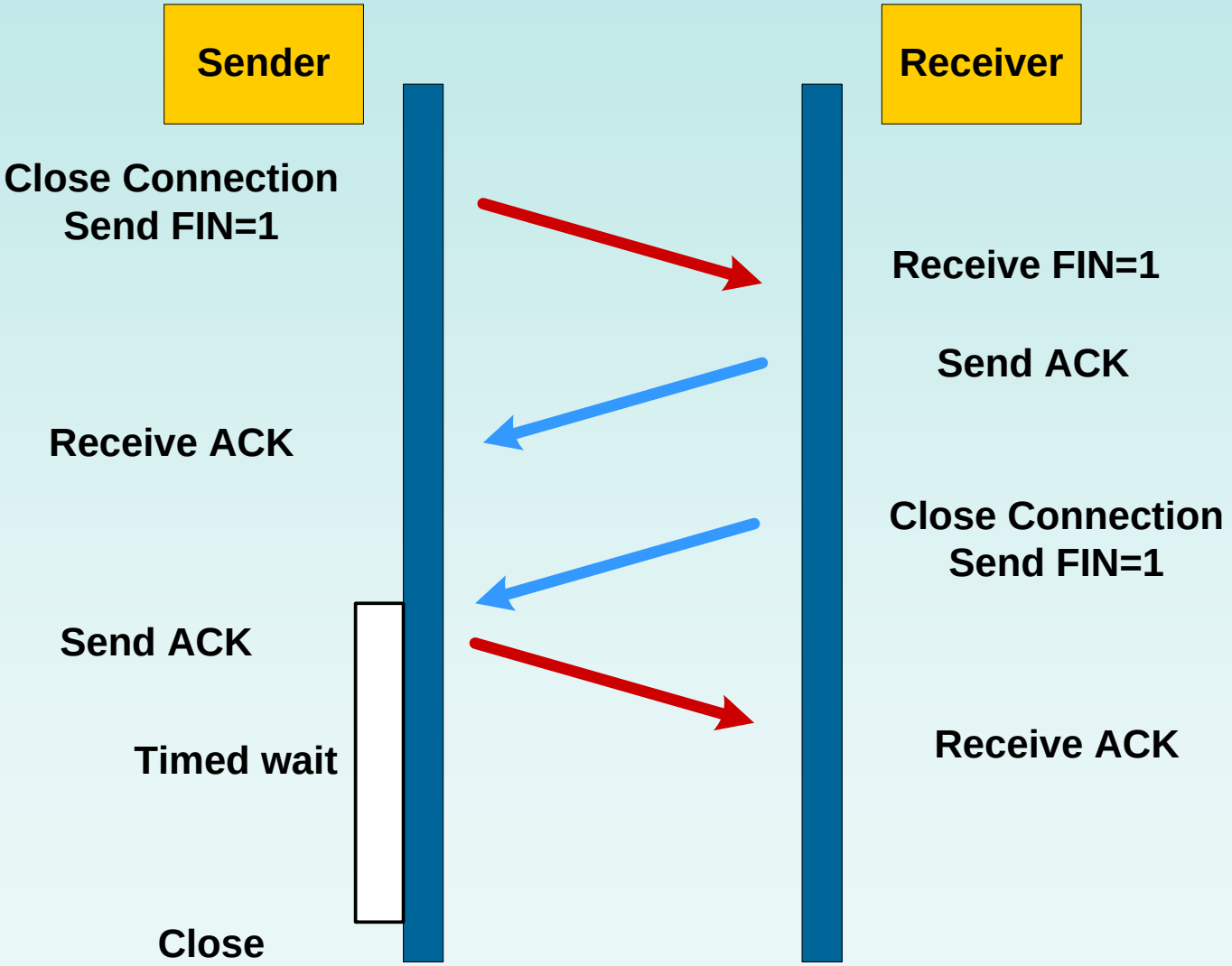
# Fase Penetapan Koneksi TCP

2. TCP SYN segment diterima oleh receiver. Receiver mengalokasikan buffer memory dan variable untuk koneksi yang diminta serta mengirimkan segment TCP berisi: **SYN=1**, **Acknowledgment field berisi isn dari sender + 1** dan **sequence number berisi sembarang angka** yang digenerate secara random oleh receiver.

# Fase Penetapan Koneksi TCP

3. Setelah menerima TCP segment dari receiver, Sender mengirimkan kembali TCP segment ketiga yang berisi **SYN=1**, **sequence number berisi isn dari receiver + 1** dan **Acknowledgment field berisi isn dari sender + 1**.

# Fase Penutupan Koneksi TCP



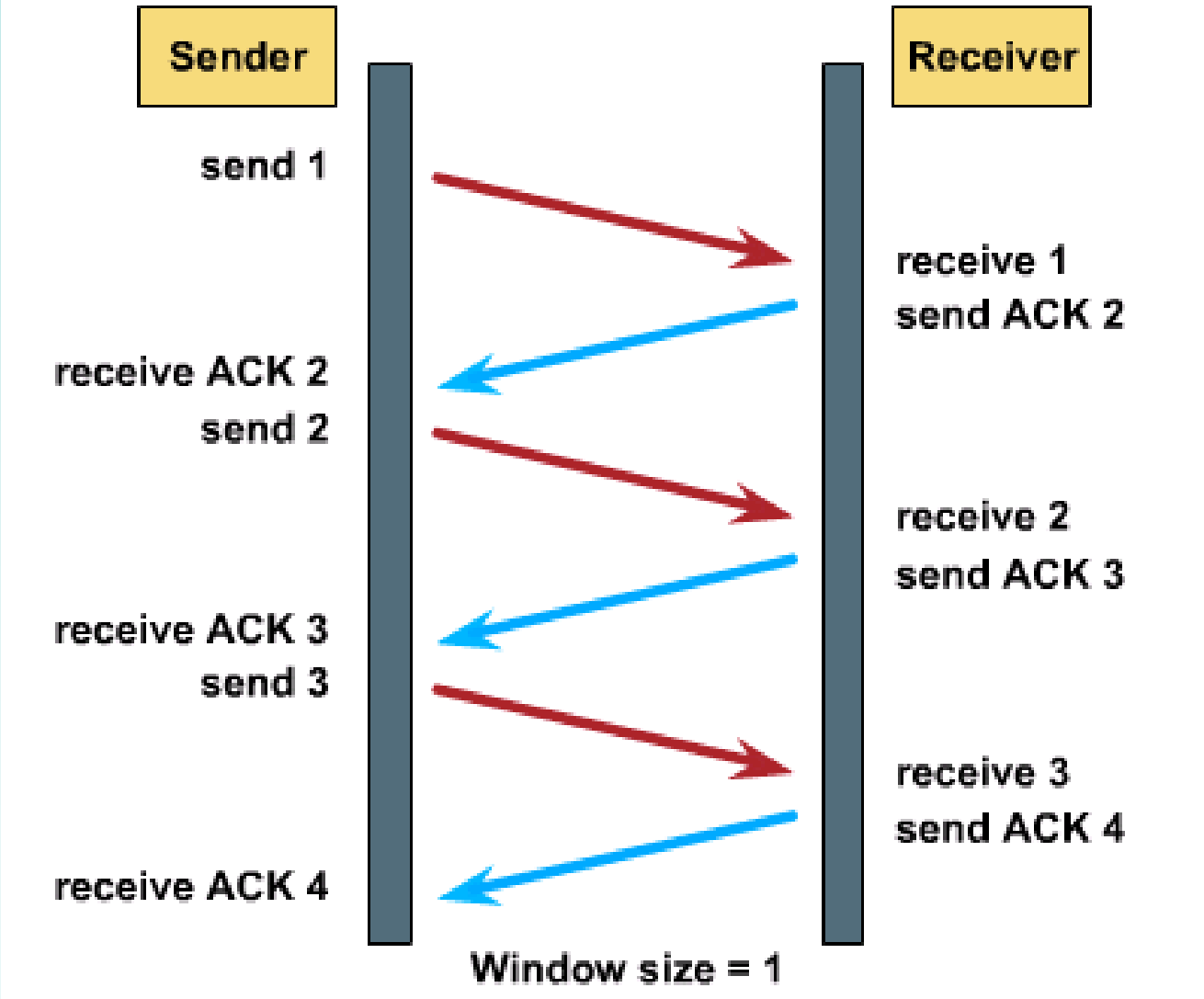
# Fase Penutupan Koneksi TCP

- Sender memulai penutupan koneksi dengan mengirimkan TCP segment dengan nilai FIN=1.
- Receiver mengirimkan TCP segment ACK.
- Sender menunggu sampai Receiver mengirimkan TCP segment berikutnya.
- Sender mengirim TCP segment ACK.
- Setelah periode waktu tertentu koneksi tertutup.

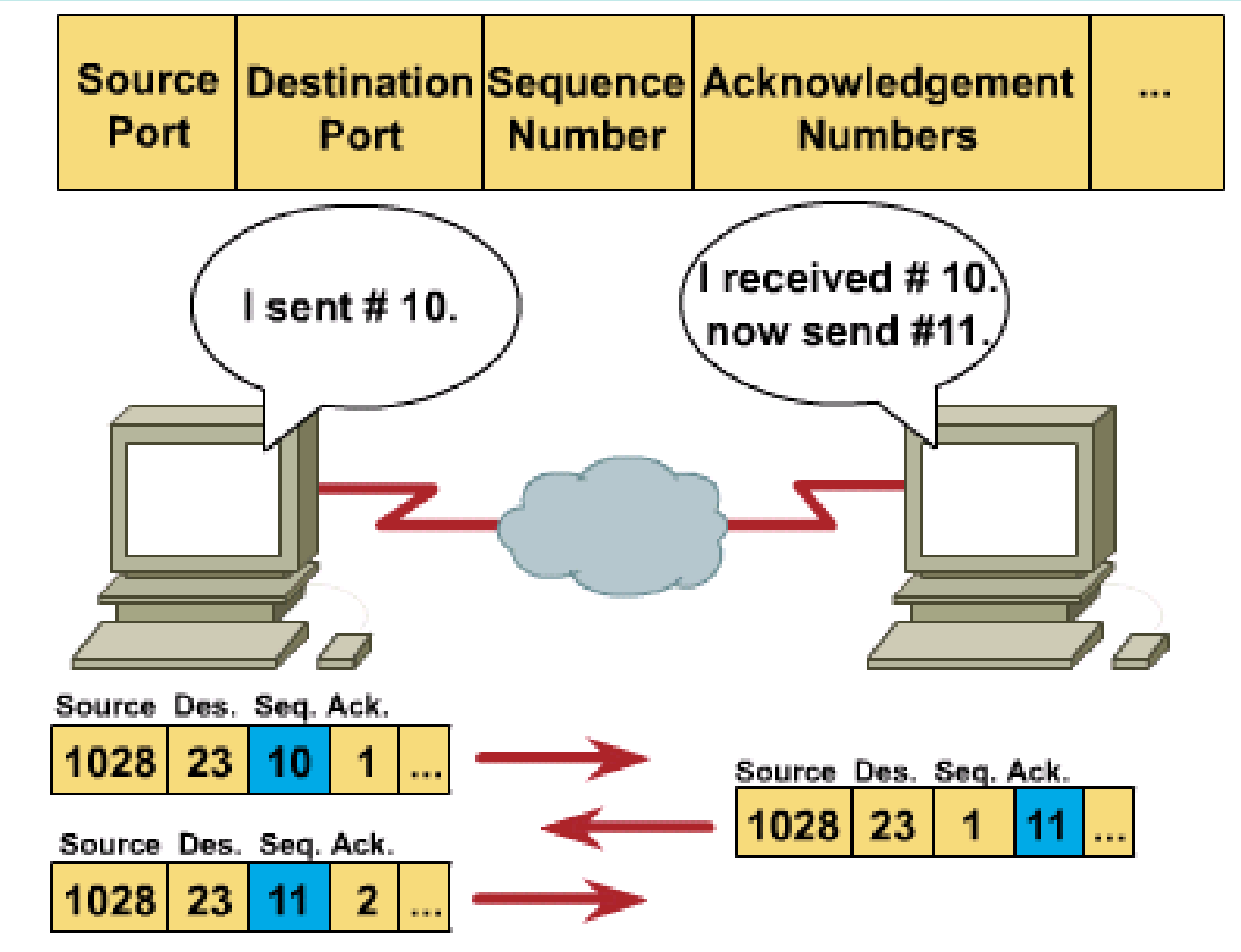
# Fase Transaksi Pesan TCP

- TCP melakukan transaksi pesan dengan menggunakan *sliding-window* protocol.
- Saat pengirim mengirim pesan, misalnya dengan isn=1, receiver mengirim acknowledgment dengan ACK=2. ***Artinya, pesan dengan isn=1 telah deterima, selanjutnya kirimkan pesan dengan isn=2***.
- Dan seterusnya sampai semua TCP segment dikirimkan.

# Fase Transaksi Pesan TCP



# Fase Transaksi Pesan TCP



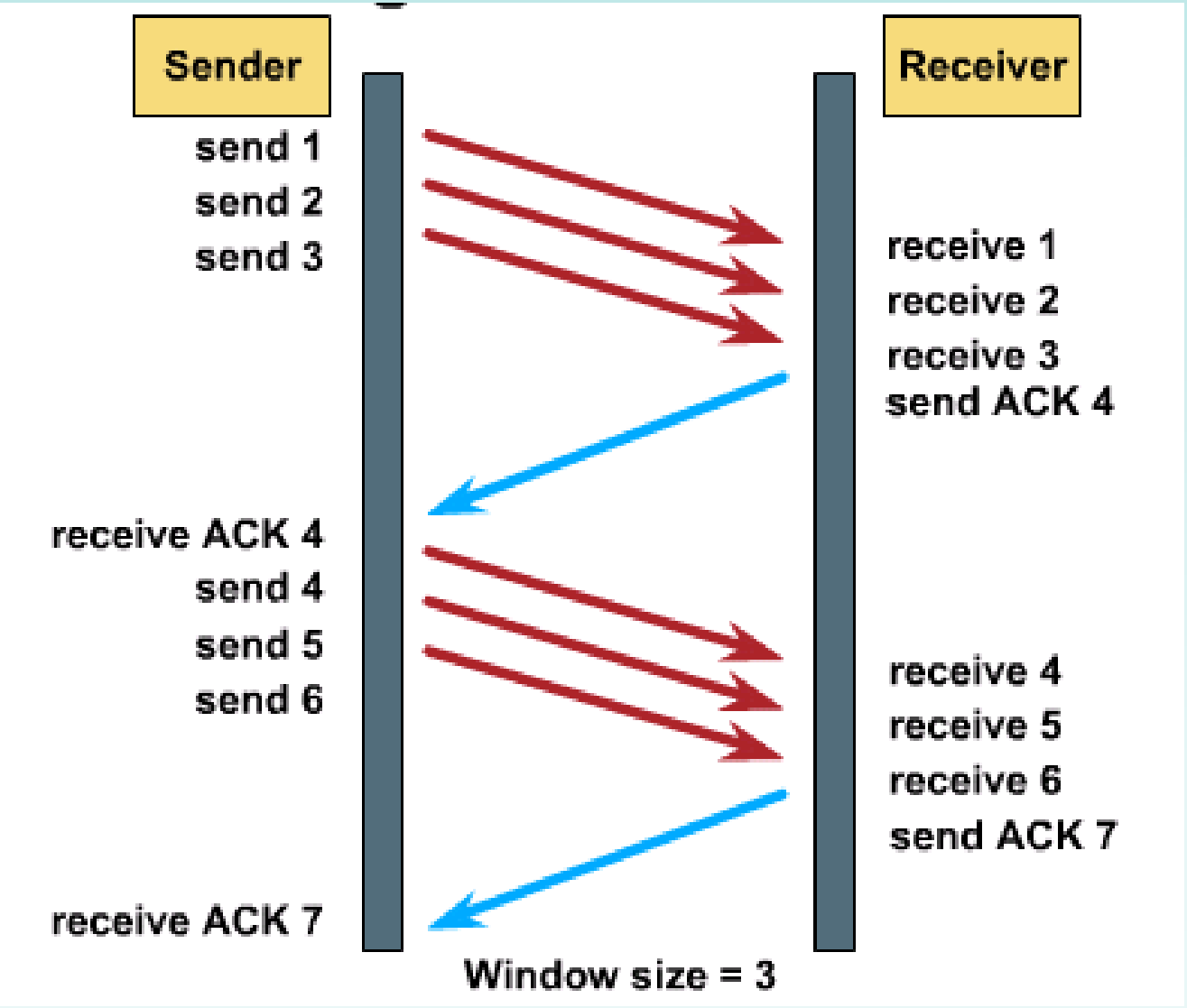
# Fase Transaksi Pesan TCP

- Pengiriman TCP segment (pesan) seperti pada contoh di atas menggunakan window size=1.
- Model ini disebut juga sebagai *Stop-And-Wait*.
- Kelemahan model ini adalah munculnya delay yang cukup besar karena pesan berikut baru bisa dikirimkan setelah sender menerima ACK.

# Fase Transaksi Pesan TCP

- Model Stop-And-Wait diperbaiki dengan menggunakan model *Go-Back-N*.
- Dengan window size sebesar N, sebanyak N-pesan dapat dikirimkan pada satu saat.
- Model ini juga dipakai untuk *flow-control*.

# Fase Transaksi Pesan TCP



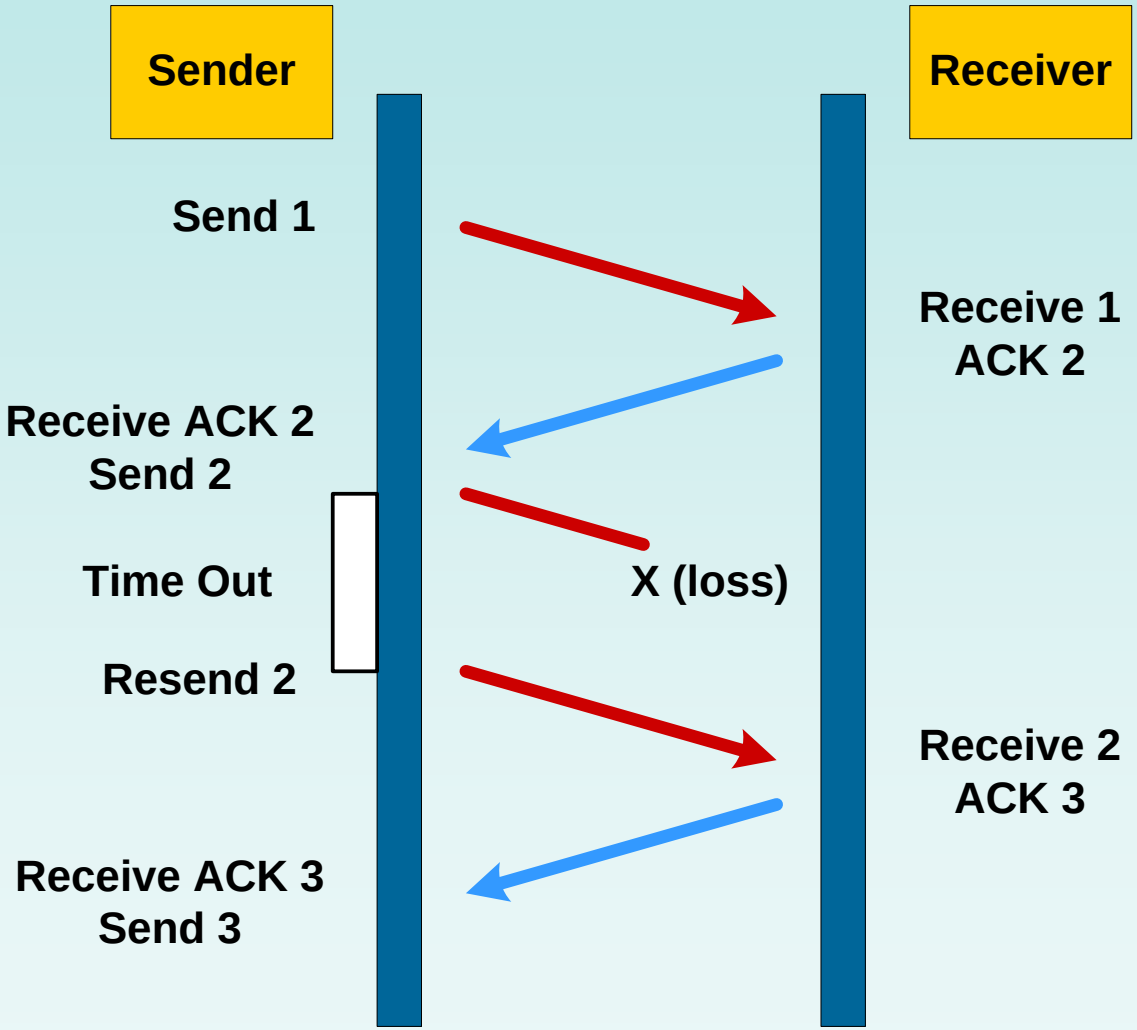
# Fase Transaksi Pesan TCP

- Gambar di atas adalah contoh transaksi TCP dengan window size=3.
- Sender mengirimkan 3 pesan sekaligus.
- Receiver menerima 3 pesan tersebut, selanjutnya receiver mengirimkan ACK=4. ***Artinya, pesan dengan isn=1,2 dan 3 telah deterima, selanjutnya kirimkan pesan dimulai dengan isn=4.*** Dan seterusnya.

# Lost Segment

*Apa yang harus dilakukan oleh TCP apabila terjadi kehilangan TCP segment di tengah jalan ?*

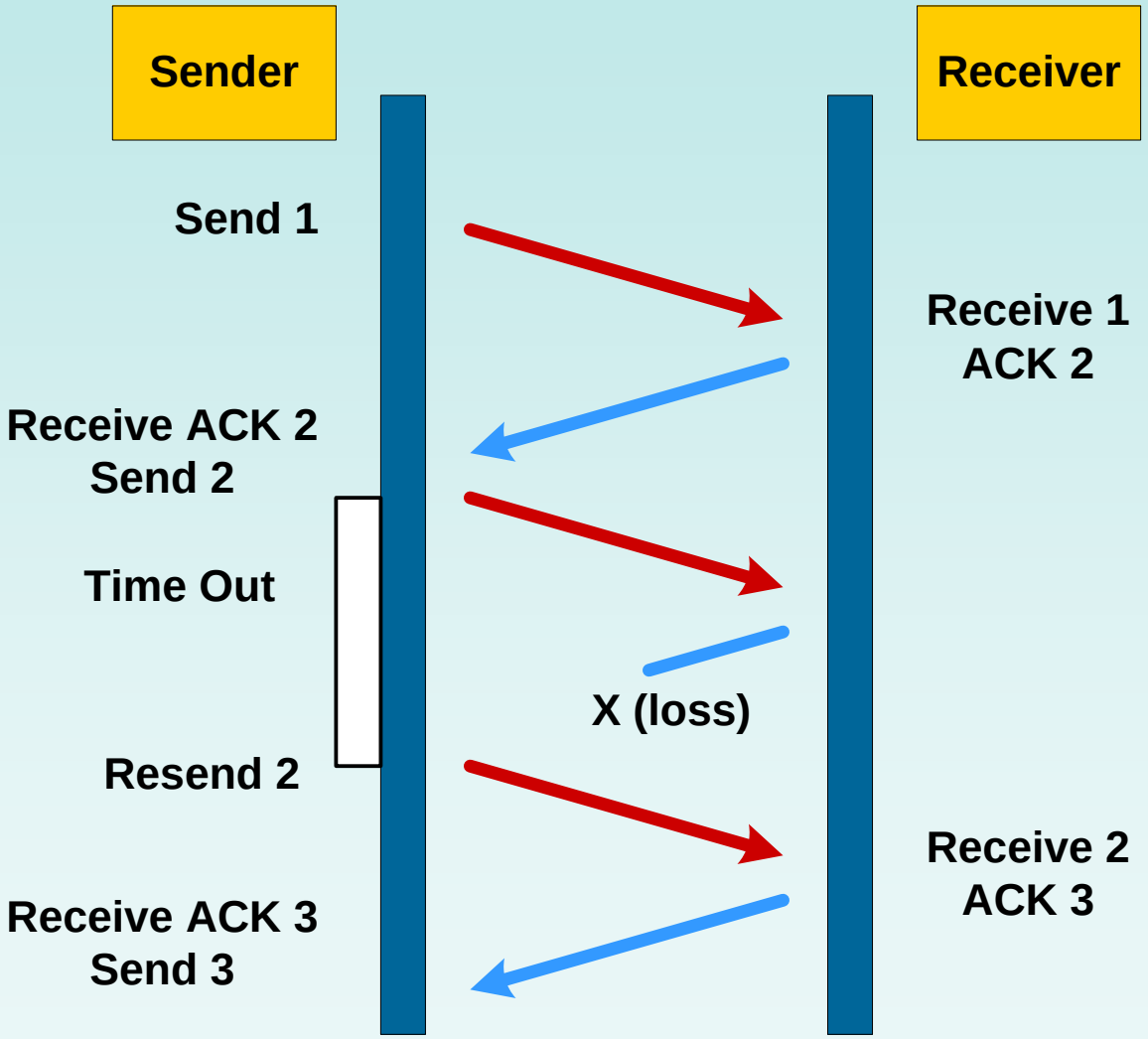
# Lost Segment



# Lost ACK

*Apa yang harus dilakukan oleh TCP apabila terjadi kehilangan ACK segment di tengah jalan ?*

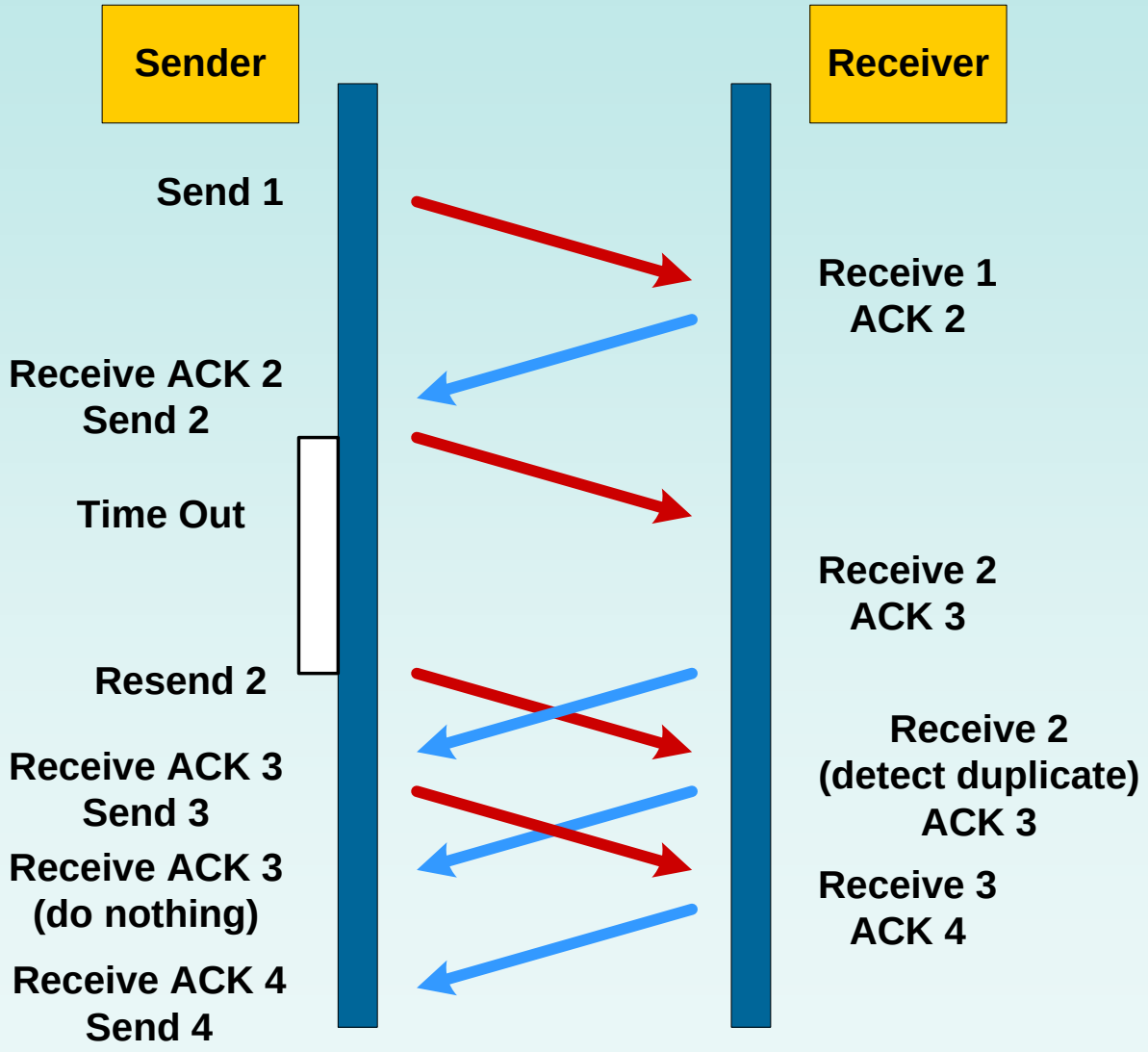
# Lost ACK


Sender
Receiver
Send 1
Receive 1
ACK 2
Receive ACK 2
Send 2
Time Out
X (loss)
Resend 2
Receive 2
ACK 3
Receive ACK 3
Send 3

# Prematur Time Out

*Apa yang terjadi apabila sender menganggap terjadi kehilangan ACK tetapi sebenarnya hanya terjadi delay pengiriman ACK ?*

# Prematur Time Out



# The End