



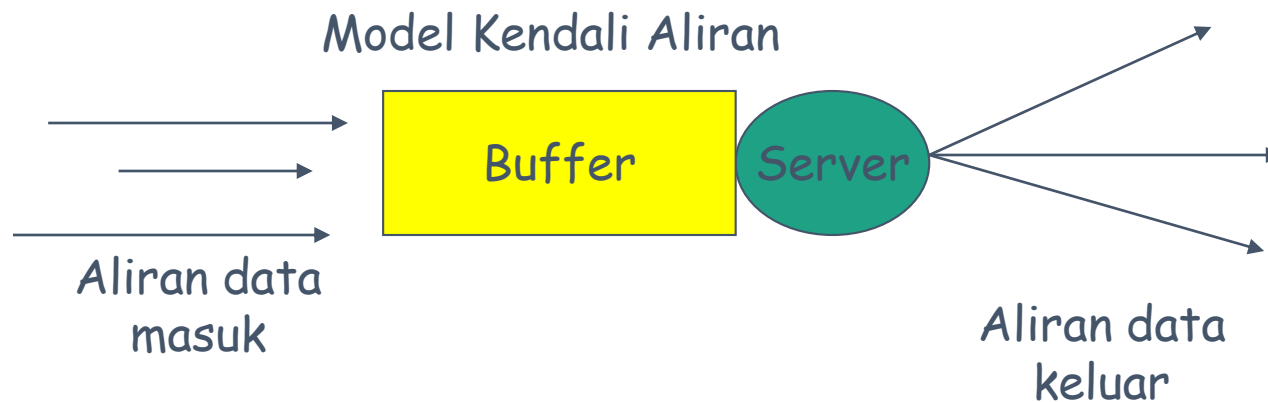
TCP : Flow & Congestion Control

Jaringan Komunikasi Data
Telkom University

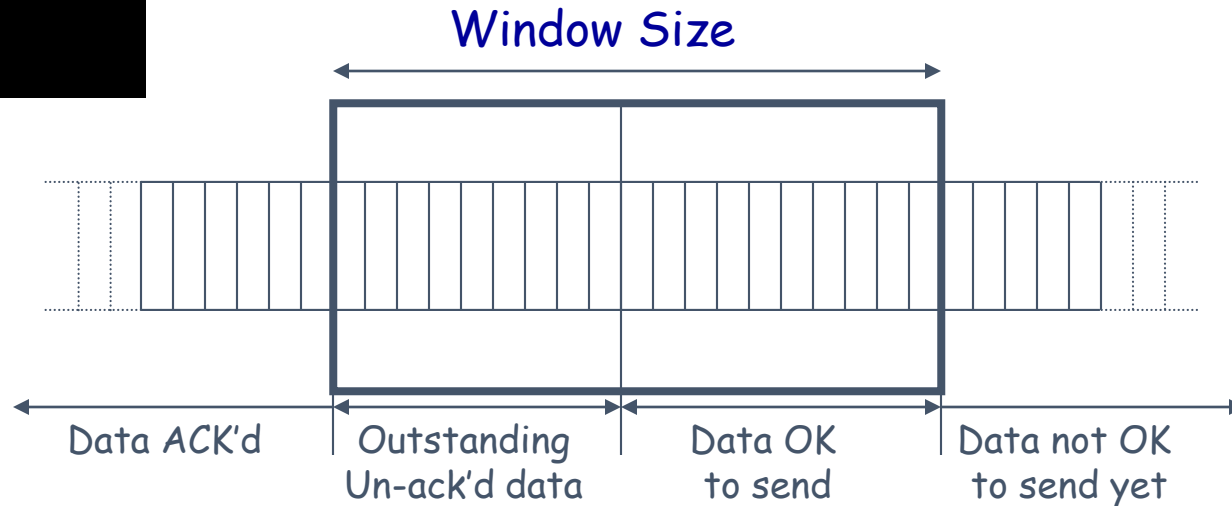


Kendali Aliran (Flow control)

- Fungsi lain yang diperlukan dalam mentransmisikan data di suatu link adalah kendali aliran
- Dibutuhkan terutama jika aliran data dari yang cepat ke yang lambat, dimana aliran data harus diatur agar penerima tidak overflow

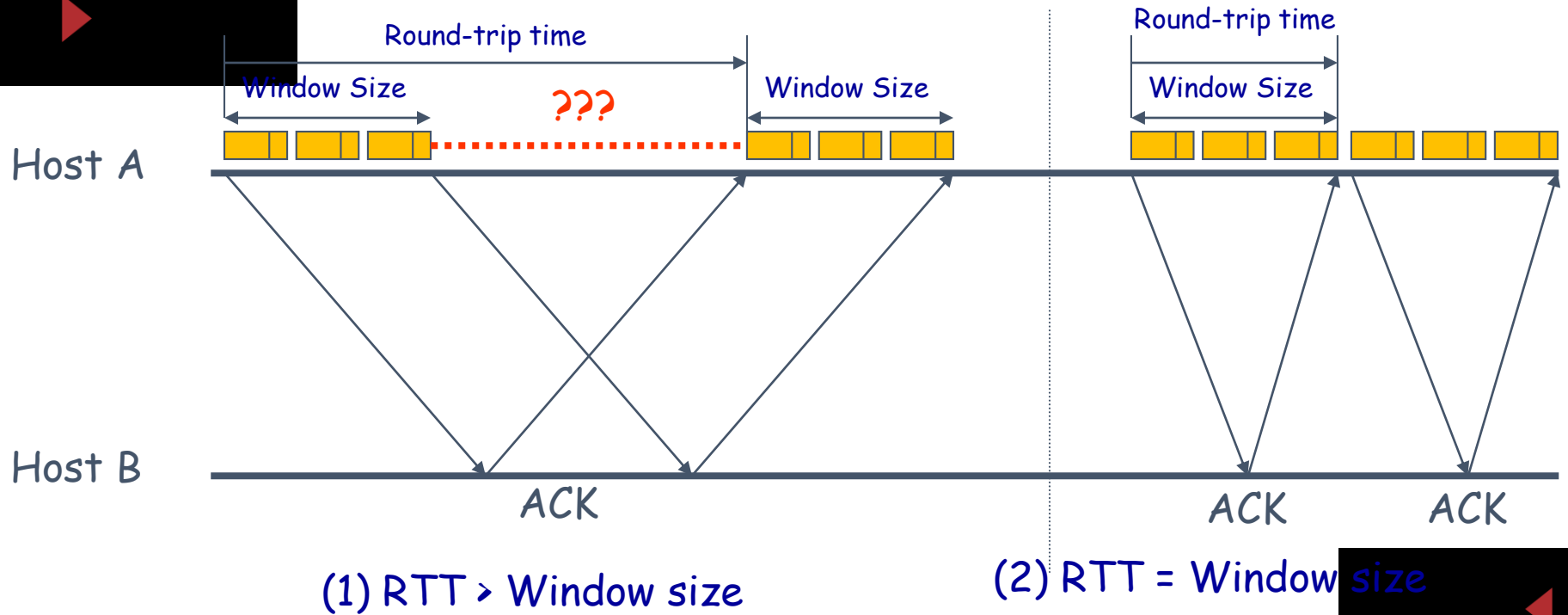


TCP Sliding Window

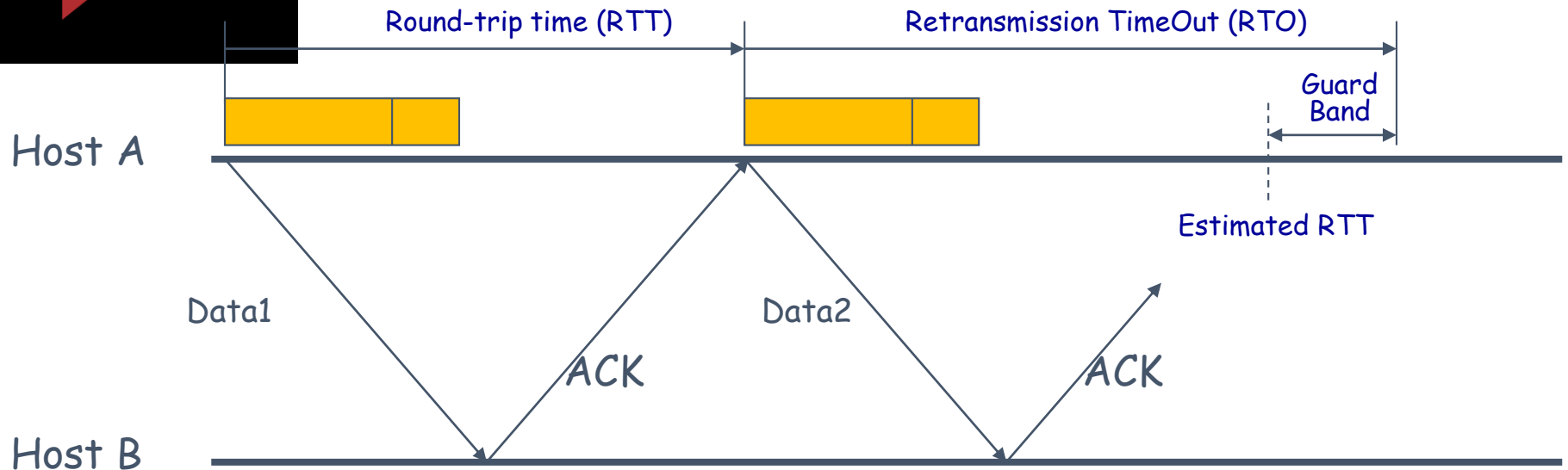


- ❖ Window bermakna utk pengirim/sender.
- ❖ Ukuran window saat ini diberikan/"advertised" oleh penerima (umumnya 4k - 8k Bytes saat connection set-up).
- ❖ Retransmisi pd TCP adalah "Go Back N"

TCP Sliding Window



TCP: Retransmisi dan Timeout




TCP menggunakan nilai waktu timeout utk retransmisi yg adaptif:

Kongesti
Perubahan pd Routing

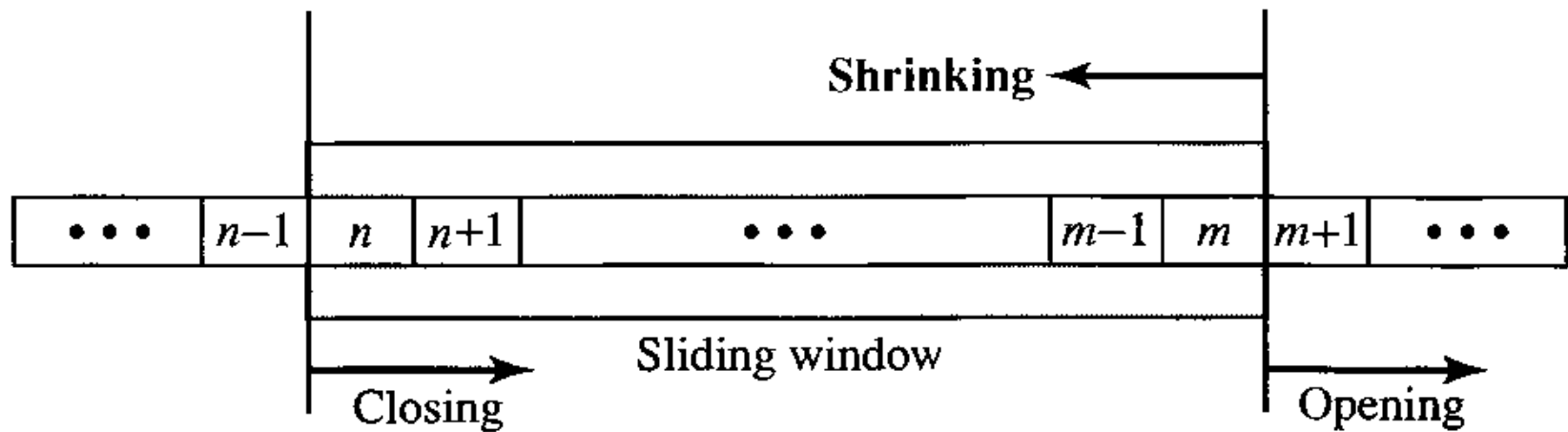
} RTT berubah
cukup sering



Sliding window (transport)

- Window = angka jumlah pengiriman maximum saat ini
 - Window = 3 → satu kali kirim maksimum 3 byte
 - Cara kerja 1:
 - Penerima akan menetapkan jumlah window terimanya berdasarkan tingkat keberhasilan penerimaan paket, kebijakan yang ditetapkan oleh lapis aplikasi, dll
 - Pengirim kemudian akan mengirim paket sesuai dengan jumlah window yang ditetapkan penerima
 - Cara kerja 2:
 - Adaptive mengikuti keadaan jaringan
- 

Window size = minimum (rwnd, cwnd)



SLIDING WINDOW

Window size = minimum (20, 9) = 9

Sent, not
acknowledged

Can be sent immediately




Sent and
acknowledged

Next byte to be sent

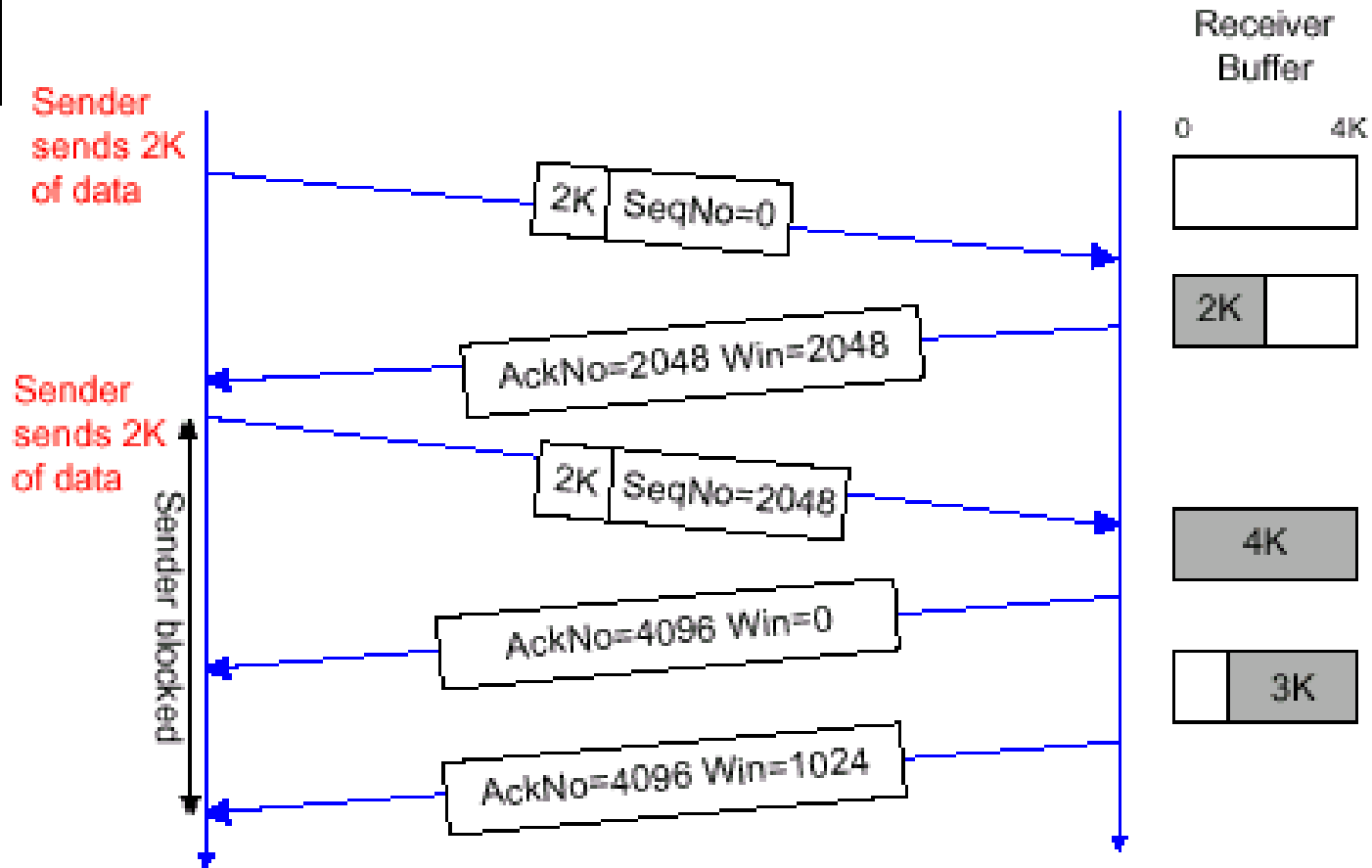
Can't be
sent until window
opens



SLIDING WINDOW

- ❖ Karena byte segment yang berada dalam window pengirim bisa hilang atau rusak, pengirim harus tetap menyimpan byte tersebut dalam memorinya sebagai antisipasi kemungkinan retransmisi.
 - ❖ Piggybacking → teknik penumpangan balasan pada frame data untuk komunikasi 2 arah (menghemat kapasitas komunikasi).
- 

Contoh : Sliding Window





Congestion Control





Masalah

Berapa besar trafik harus dikirim?

Dua komponen

- Menjamin penerima dp menerima secepat yg dikirim (Receiver)
- Menjamin jaringan mengirimkan paket ke penerima (Jaringan)

Actual window size = minimum (rwnd, cwnd)



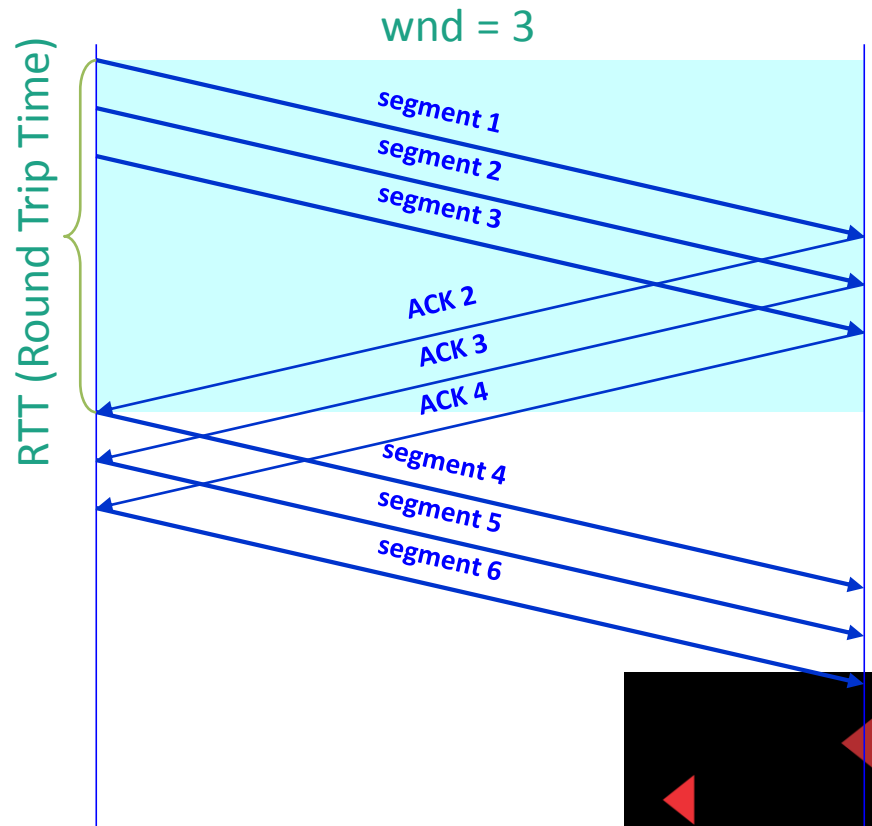
Flow control: Ukuran Window dan Throughput

- Sliding-window based flow control:

- Window lebih besar → throughput lebih tinggi


- $\text{Throughput} = \text{wnd}/\text{RTT}$

- Ingat: ukuran window mengendalikan throughput





Mengapa Kita Peduli Dengan Congestion Control?

- Jika tdk kita akan mengalami **congestion collapse**
 - Bagaimana bisa terjadi?
 - ✓ Mis: Jaringan dlm keadaan kongesti (router membuang paket-paket)
 - ✓ Pengirim tahu penerima tdk menerima paket
 - ✓ dari ACK, NACK, atau Timeout
 - ✓ Apa yg dilakukan sumber? retransmisi paket
 - ✓ Penerima tetap tdk menerima paket (krn jar. kongesti)
 - ✓ Retransmisi paket
 - ✓ dan seterusnya ...
 - ✓ Dan sekarang asumsikan semuanya melakukan hal yg sama!
 - Jaringan akan menjadi bertambah kongesti
 - ✓ Dan ini terjadi dg duplikasi paket (paket-paket retransmisi) dibandingkan oleh paket-paket baru!
- 



Solusi ?

Slow down

Jika kita tahu paket tdk dikirimkan karena jaringan kongesti, turunkan laju pengiriman (slow down)

Pertanyaan:

Bagaimana kita mendeteksi jaringan kongesti?
Seberapa besar kita harus slow down?



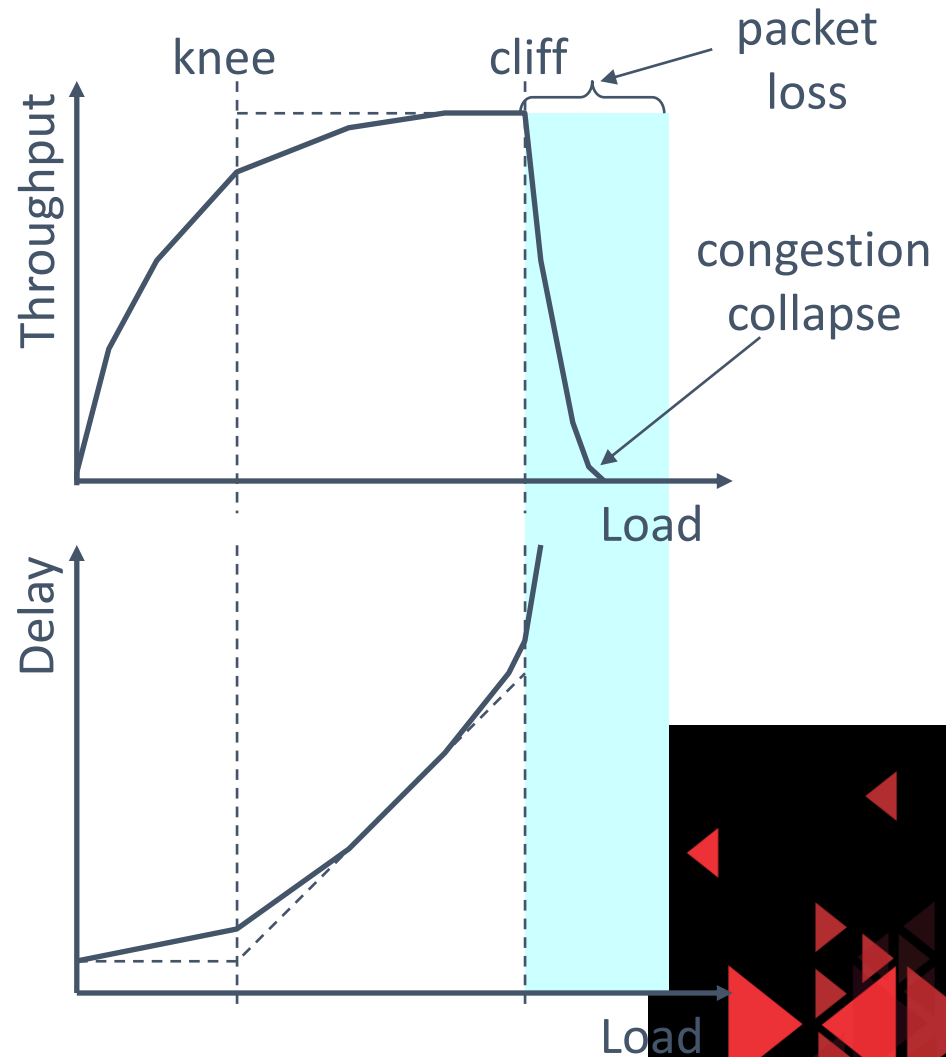
Apa yg Sesungguhnya Terjadi ?

- Knee – titik dimana

- Throughput **naik secara perlahan**
- Delay **naik secara cepat**

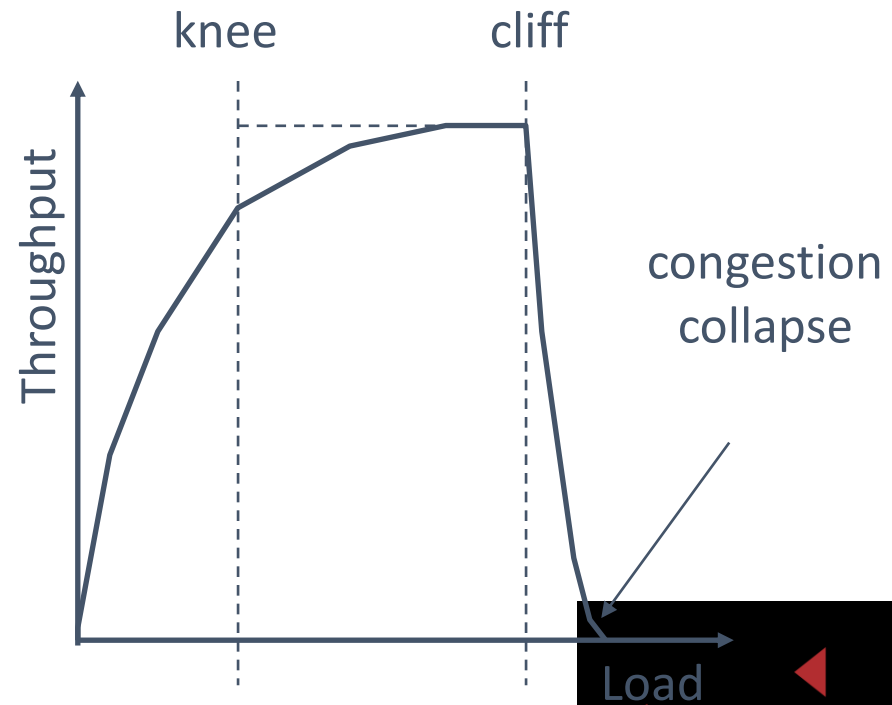
- Cliff – titik dimana

- Throughput mulai **menurun secara cepat ke nol** (congestion collapse)
- Delay **menuju tak hingga**




Congestion Control vs. Congestion Avoidance

- Tujuan congestion control
 - Tetap di sebelah kiri cliff
- Tujuan congestion avoidance
 - Tetap di sebelah kiri knee





Bagaimana Melakukannya ?

- Deteksi saat jaringan mendekati/mencapai *knee point*
 - Tetap disana
 - Pertanyaan
 - Bagaimana mencapai kesana?
 - Bagaimana jika *overshoot* (pergi melebihi *knee point*)?
 - Solusi yg mungkin:
 - Naikkan ukuran window sampai teridentifikasi kongesti
 - Turunkan ukuran window jika jaringan kongesti
- 



Congestion Policy

Slow Start

Congestion Avoidance

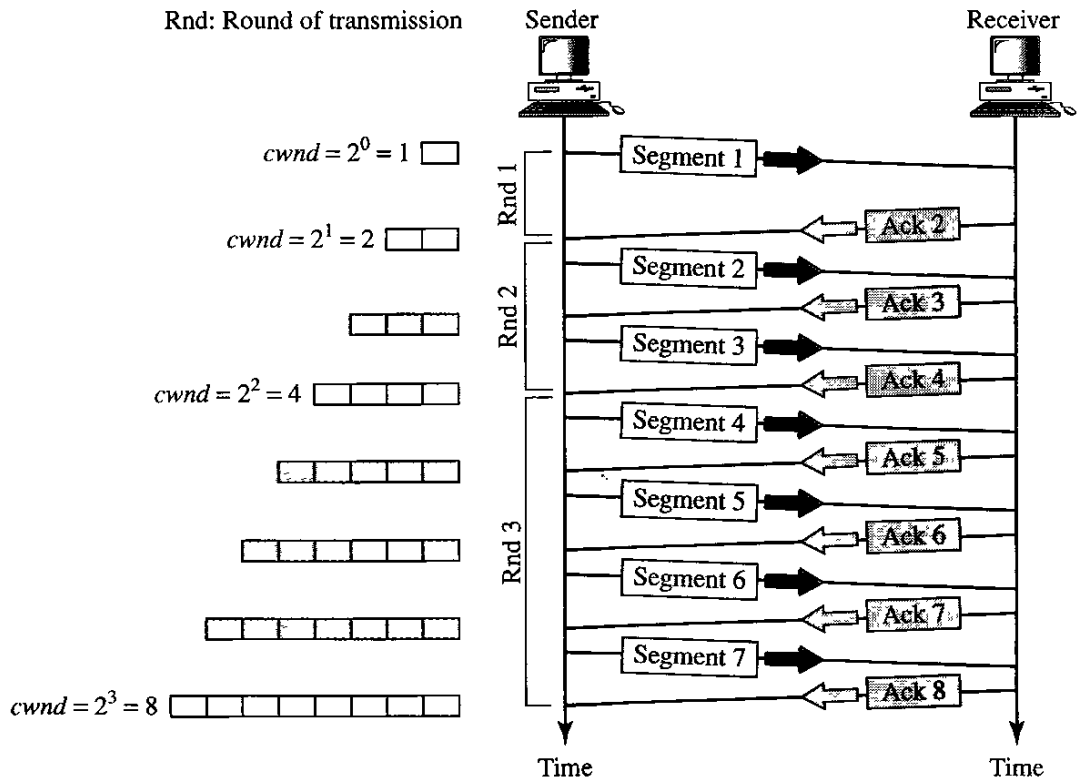
Congestion detection



Slow-Start


- $cwnd$ dinaikkan secara exponential: $cwnd$ jadi double tiap full $cwnd$ paket telah dikirim
- Slow-start disebut “slow” krn starting point

Start → $cwnd = 1$
After round 1 → $cwnd = 2^1 = 2$
After round 2 → $cwnd = 2^2 = 4$
After round 3 → $cwnd = 2^3 = 8$






Masalah dengan Slow-Start

- Slow-start dp menyebabkan banyak losses
 - secara kasar ukuran cwnd \sim BW*RTT
 - Contoh:
 - pd suatu titik, cwnd cukup utk mengisi “pipe”
 - setelah RTT berikutnya, cwnd menjadi dua kali harga sblmnya
 - semua kelebihan paket di-dropped!
 - Krnnya, perlu adjustment algorithm yg lebih ‘gentle’ begitu telah mengetahui estimasi kasar bandwidth
- 





Congestion Avoidance


1. Menghindari terjadinya kongesti
 2. Additive Increase
 3. Slow start hingga slow-start threshold (sssthres), slow-start phase stops mulai additive phase begins.
- 

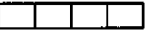


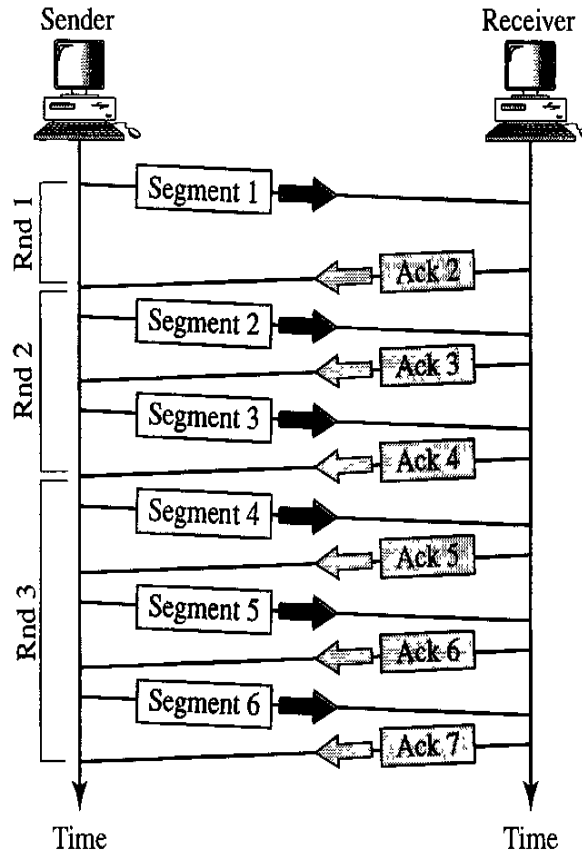
Rnd: Round of transmission

$cwnd = 1$ 

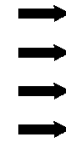
$cwnd = 1 + 1 = 2$ 

$cwnd = 2 + 1 = 3$ 

$cwnd = 3 + 1 = 4$ 



Start
After round 1
After round 2
After round 3



$cwnd = 1$
 $cwnd = 1 + 1 = 2$
 $cwnd = 2 + 1 = 3$
 $cwnd = 3 + 1 = 4$





Problem : Menentukan ssthres

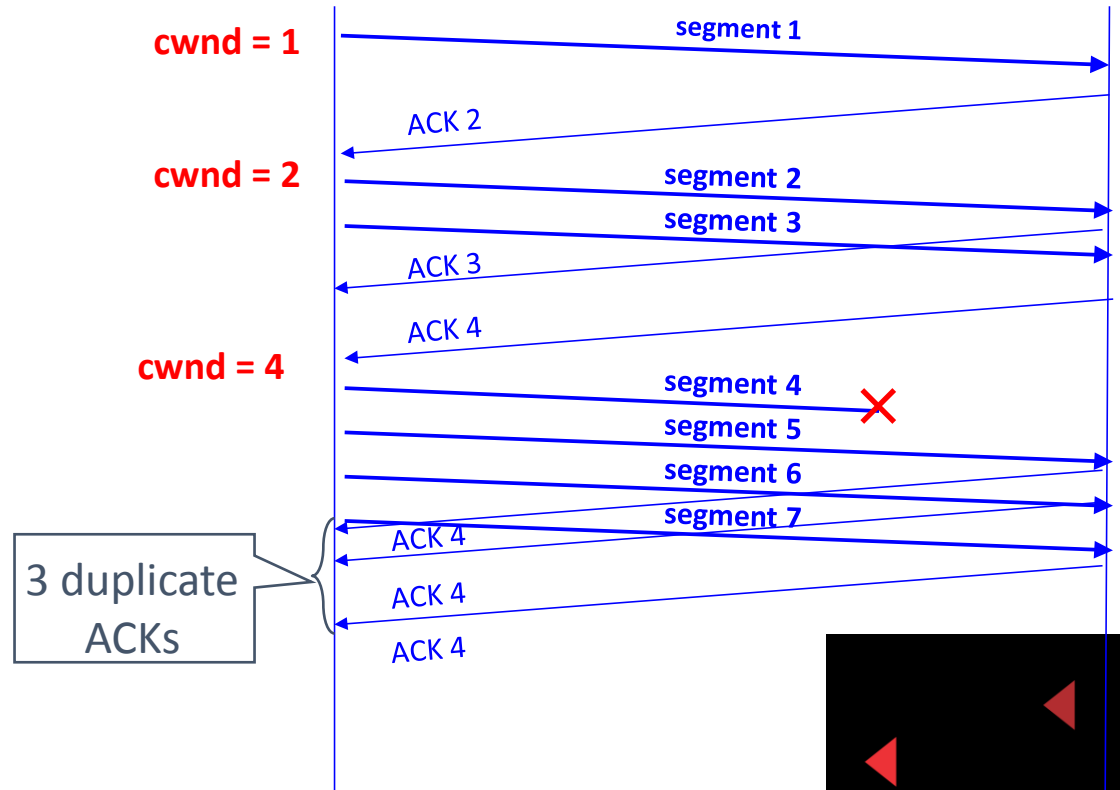


Congestion Detection

How to detect ?

RTO

Three ACK arrive

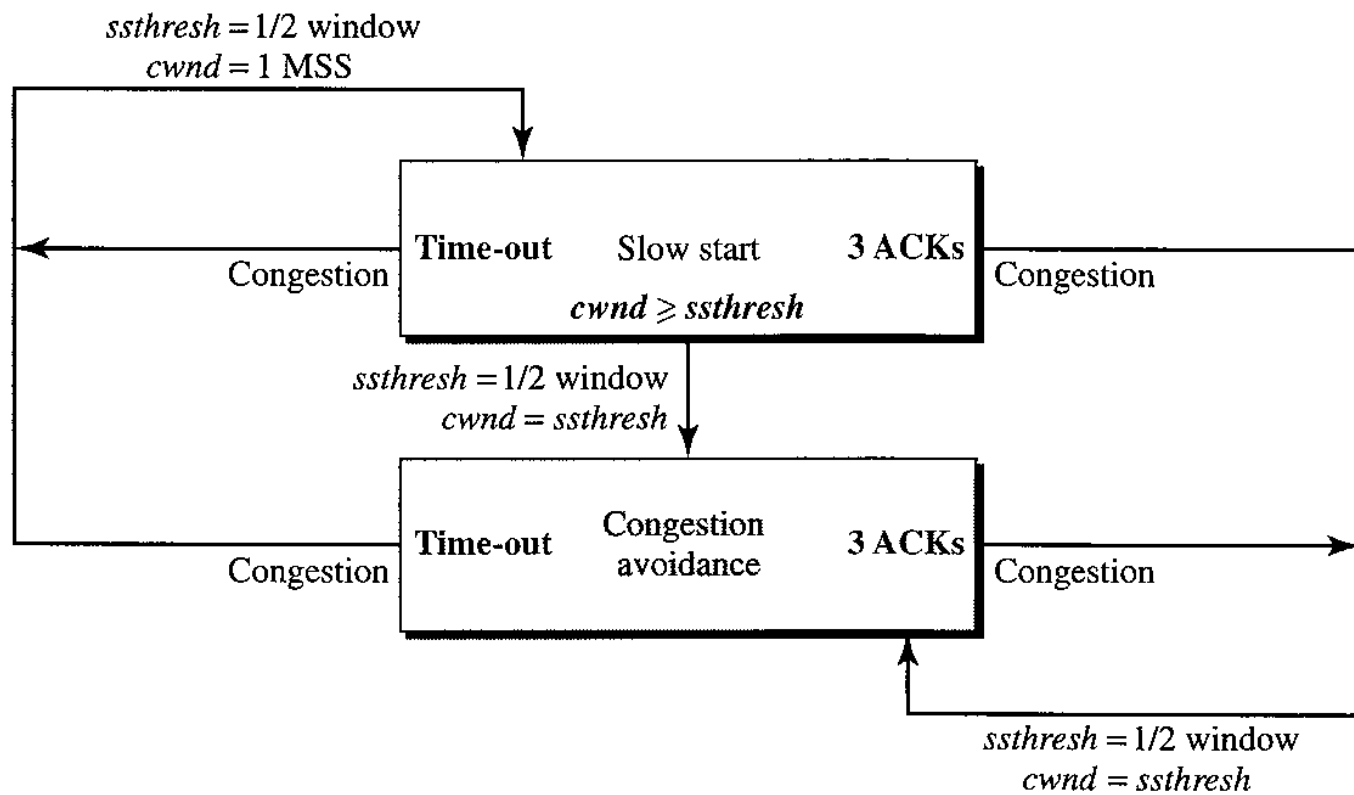


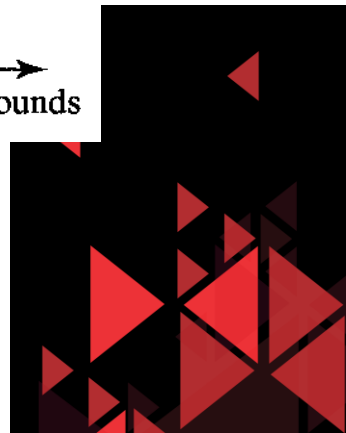
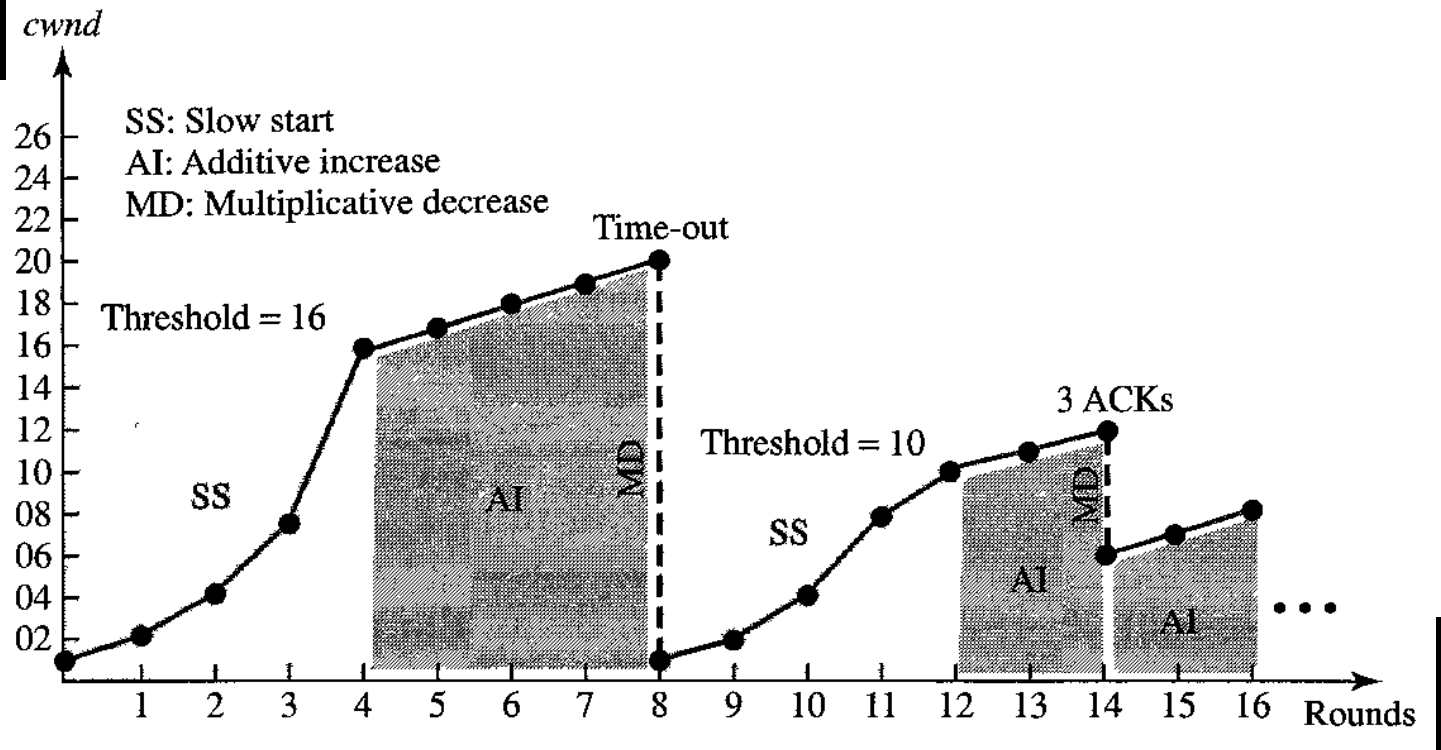


- Jika RTO :
 - a. Set threshold setengah window size.
 - b. Set cwnd = 1.
 - c. Kembali ke slow-start phase.

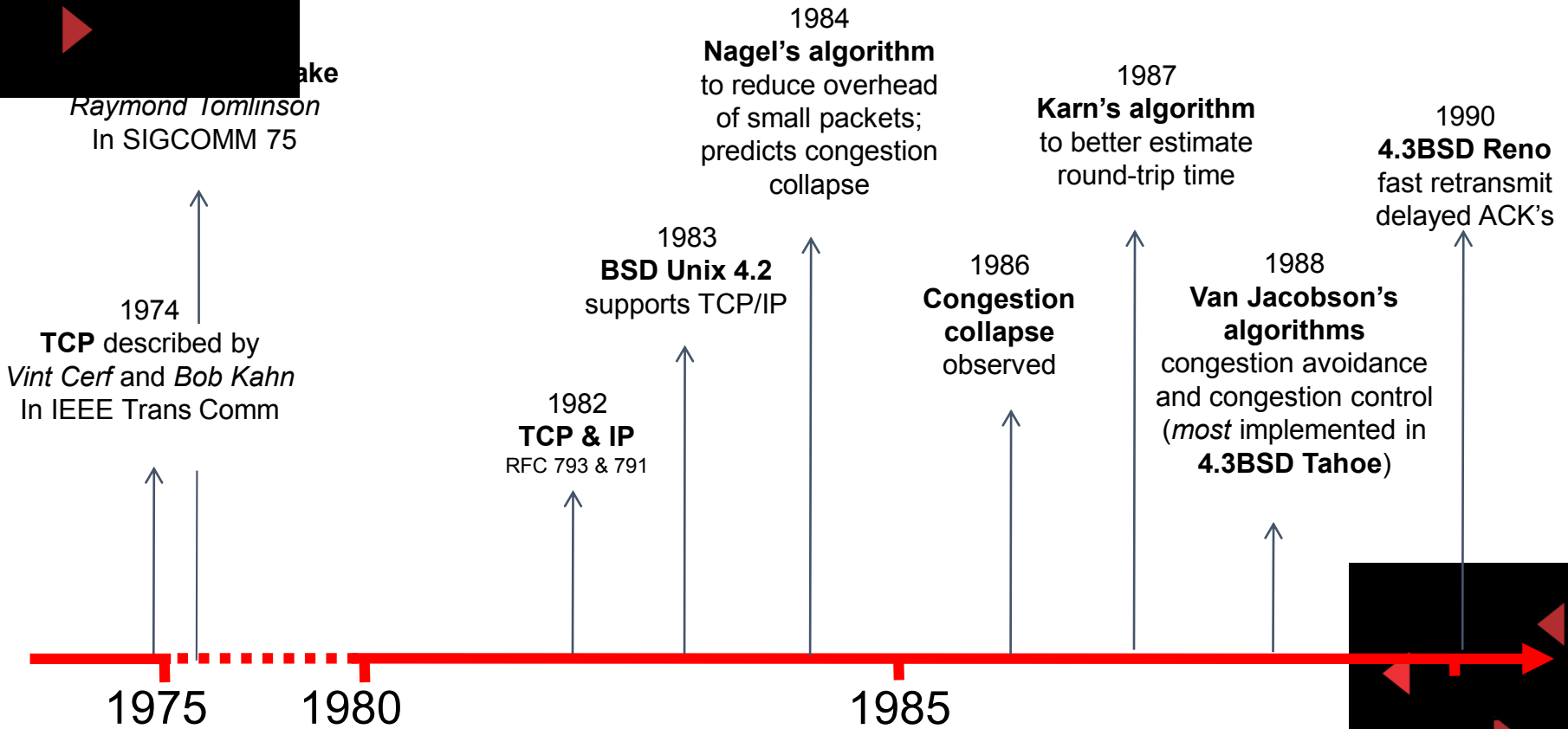
- Duplikasi 3 ACK (fast retransmit & fast recovery) :
 - a. Set threshold setengah window size.
 - b. Set cwnd = threshold
 - c. Start congestion avoidance phase.







Evolution of TCP



TCP Through the 1990s

