GITAM
(DEEMED TO BE UNIVERSITY)
(Estd. u/s 3 of the UGC Act, 1956)

Module 3

Congestion Control

Approaches towards congestion control

two broad approaches towards congestion control:

end-end congestion control:

- no explicit feedback from network
- congestion inferred from end-system observed loss, delay
- approach taken by TCP

network-assisted congestion control:

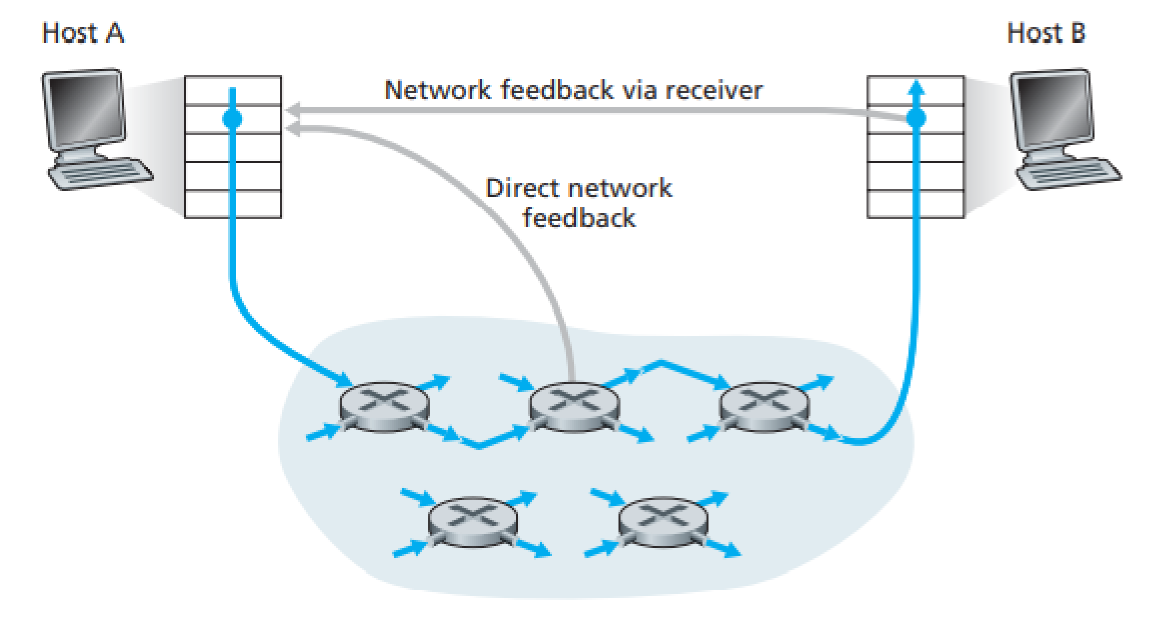
- routers provide feedback to end systems
  - single bit indicating congestion (SNA, DECbit, TCP/IP ECN, ATM)
  - explicit rate for sender to send at

Network Assisted Congestion Control

- Direct feedback may be sent from a network router to the sender. This form of notification typically takes the form of a **choke packet**.

- The second form : when a router marks/updates a field in a packet to indicate congestion.

Upon receipt of a marked packet, the receiver then notifies the sender of the congestion indication.



Two feedback pathways for network-indicated congestion information

Case study: ATM ABR congestion control

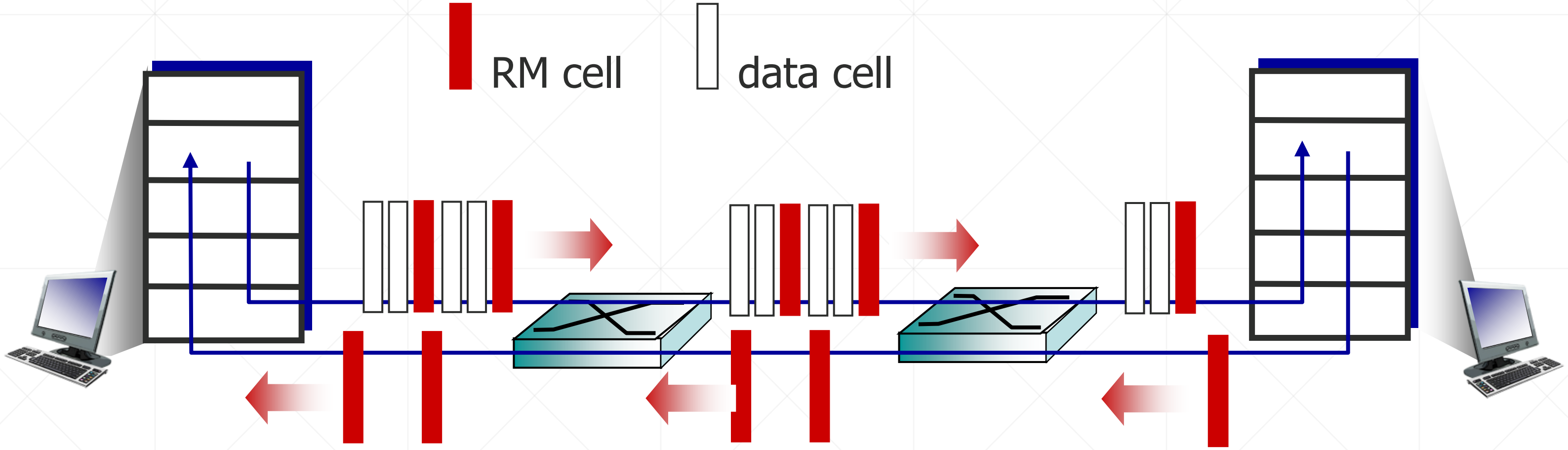
ABR: available bit rate:

- "elastic service"
- if sender's path "underloaded" :
  - sender should use available bandwidth
- if sender's path congested:
  - sender throttled to minimum guaranteed rate

RM (resource management) cells:

- sent by sender, interspersed with data cells
- Each data cell contains an explicit forward congestion indication (EFCI) bit.
- bits in RM cell set by switches ("*network-assisted*")
  - *NI bit:* no increase in rate (mild congestion)
  - *CI bit:* congestion indication
- RM cells returned to sender by receiver, with bits intact

Case study: ATM ABR congestion control

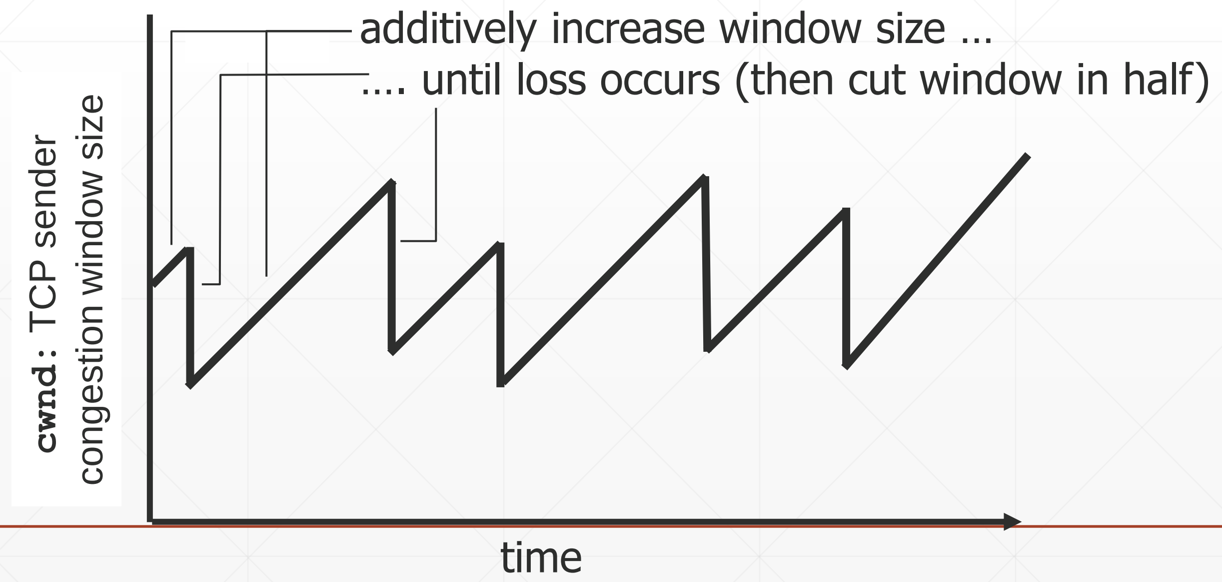


- two-byte ER (explicit rate) field in RM cell
  - congested switch may lower ER value in cell
  - Sender's send rate thus max supportable rate on path
- EFCI bit in data cells: set to 1 in congested switch
  - if data cell preceding RM cell has EFCI set, receiver sets CI bit in returned RM cell

TCP congestion control: additive increase multiplicative decrease

- *approach:* sender increases transmission rate (window size), probing for usable bandwidth, until loss occurs
  - *additive increase:* increase **`cwnd`** by 1 MSS every RTT until loss detected
  - *multiplicative decrease:* cut **`cwnd`** in half after loss

AIMD saw tooth behavior: probing for bandwidth

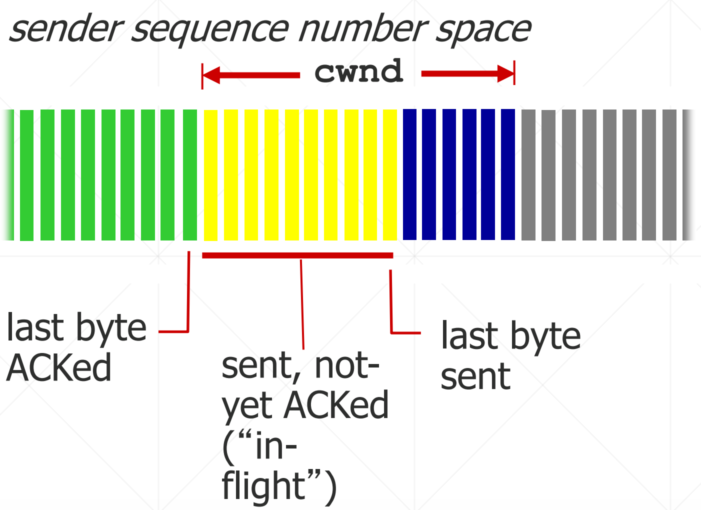


Congestion Control Introduction

- If no congestion : increase the rate
- If congestion : reduces the send rate
- This approach raises three questions:
  - how does a TCP sender limit the rate at which it sends traffic into its connection
  - how does a TCP sender perceive that there is congestion on the path between itself and the destination
  - what algorithm should the sender use to change its send rate

TCP Congestion Control: details

How does a TCP sender limit the rate at which it sends traffic into its connection



- sender limits transmission:

$$\texttt{LastByteSent-LastByteAcked} \leq \texttt{cwnd}$$

- the amount of unacknowledged data at a sender may not exceed the minimum of cwnd and rwnd
- `cwnd` is dynamic, function of perceived network congestion

TCP sending rate:

- *roughly:* send cwnd bytes, wait RTT for ACKS, then send more bytes

$$\text{rate} \approx \frac{\text{cwnd}}{\text{RTT}} \text{ bytes/sec}$$

Congestion details

- **How a TCP sender perceives that there is congestion on the path between itself and the destination**
- “**loss event**” at a TCP sender
  - occurrence of either a timeout or
  - the receipt of three duplicate ACKs from the receiver
- Loss event is the indication of congestion on the sender-to-receiver path
- When a loss event doesn’t occur
  - acknowledgments for previously unacknowledged segments will be received at the TCP sender

Congestion Details

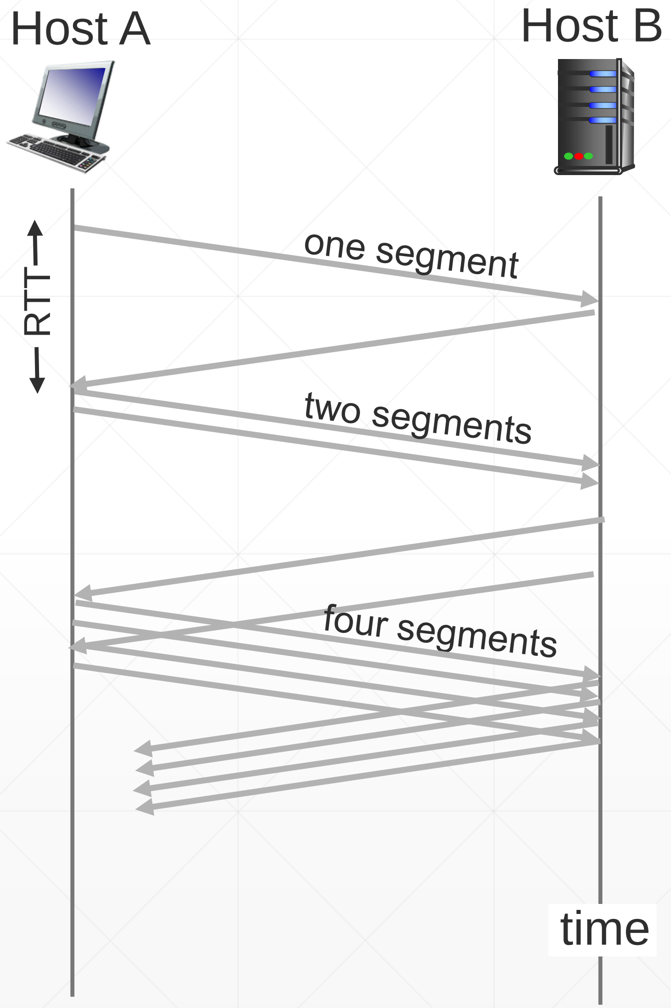
- **How should a TCP sender determine the rate at which it should send**
- Three guiding principles:
  - A lost segment implies congestion, and hence, the TCP sender’s rate should be decreased when a segment is lost
  - An acknowledged segment indicates that the network is delivering the sender’s segments to the receiver, and hence, the sender’s rate can be increased when an ACK arrives for a previously unacknowledged segment.
  - Bandwidth probing :
    - ACKs – increase rate
    - Loss event – decrease rate

TCP Congestion Control Algorithm

- First described in [Jacobson 1988]
- Standardized in [RFC 5681].
- The algorithm has three major components :
  - (1) **slow start :**
    - value of cwnd begins at 1 MSS and increases by 1 MSS every time a transmitted segment is first acknowledged
  - (2) **congestion avoidance :**
    - value of cwnd equals ssthresh, congestion avoidance mode starts. Tcp increases cwnd more cautiously
  - (3) **fast recovery :**
    - if three duplicate ACKs are detected, in which case TCP performs a fast retransmit

TCP Slow Start

- when connection begins, increase rate exponentially until first loss event:
  - initially **`cwnd`** = 1 MSS
  - double **`cwnd`** every RTT
  - done by incrementing **`cwnd`** for every ACK received
- *summary:* initial rate is slow but ramps up exponentially fast
- Exponential growth ends:
  - Loss event occurs
  - Cwnd equals ssthresh, slow start ends and TCP enters **congestion avoidance**



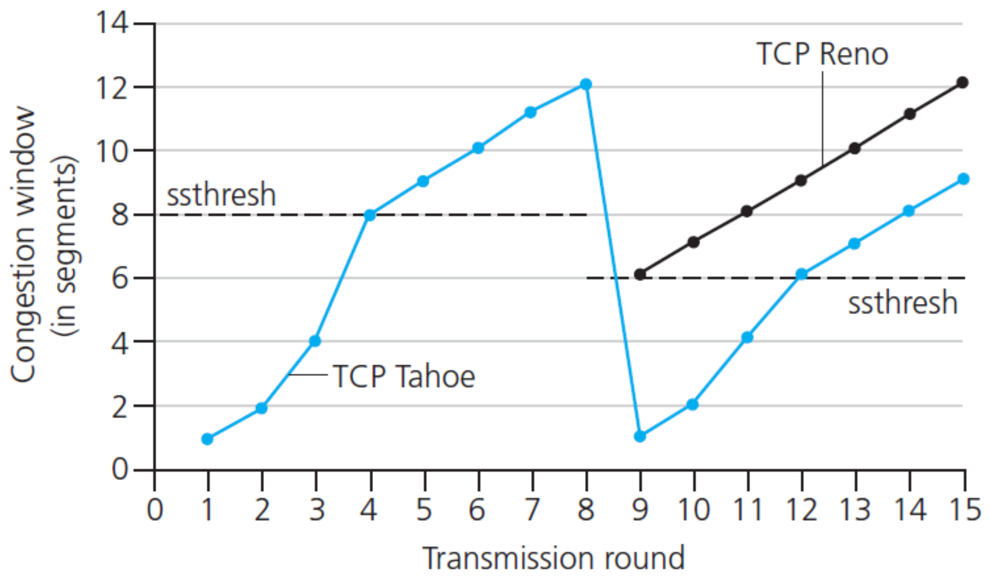
TCP: detecting, reacting to loss

- loss indicated by timeout:
  - **`cwnd`** set to 1 MSS;
  - window then grows exponentially (as in slow start) to threshold, then grows linearly
- loss indicated by 3 duplicate ACKs: TCP RENO
  - dup ACKs indicate network capable of delivering some segments
  - **`cwnd`** is cut in half window then grows linearly
- TCP Tahoe always sets **`cwnd`** to 1 (timeout or 3 duplicate acks)

TCP: switching from slow start to CA

Q: when should the exponential increase switch to linear?

A: when **`cwnd`** gets to 1/2 of its value before timeout.



<u>Implementation:</u>

- variable **`ssthresh`**
- on loss event, **`ssthresh`** is set to 1/2 of **`cwnd`** just before loss event

Summary: TCP Congestion Control