

Transport Level Congestion Control

Part 2

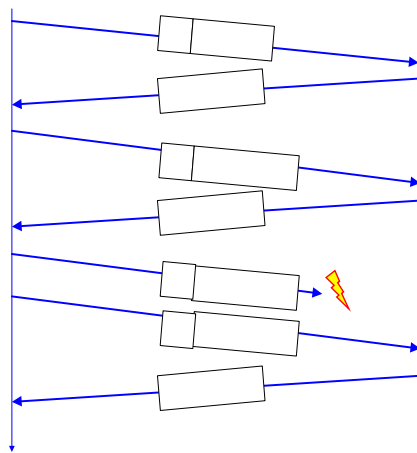
Flavors of TCP Congestion Control

- **TCP Tahoe** (1988, FreeBSD 4.3 Tahoe)
 - Slow Start
 - Congestion Avoidance
 - Fast Retransmit
- **TCP Reno** (1990, FreeBSD 4.3 Reno)
 - Fast Recovery
- **New Reno** (1996)
- **SACK** (1996)
- **Vegas** (Brakmo and Peterson 1994)

- **RED** (Floyd and Jacobson 1993)

Acknowledgments in TCP

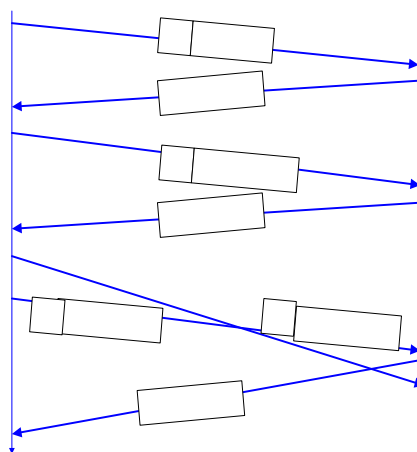
- Receiver sends ACK to sender
 - ACK is used for flow control, error control, and congestion control
- ACK number sent is the next sequence number expected
- Delayed ACK: TCP receiver normally delays transmission of an ACK (for about 200ms)
 - Why?
- ACKs are not delayed when packets are received out of sequence
 - Why?



Lost segment

Acknowledgments in TCP

- Receiver sends ACK to sender
 - ACK is used for flow control, error control, and congestion control
- ACK number sent is the next sequence number expected
- Delayed ACK: TCP receiver normally delays transmission of an ACK (for about 200ms)
 - Why?
- ACKs are not delayed when packets are received out of sequence
 - Why?



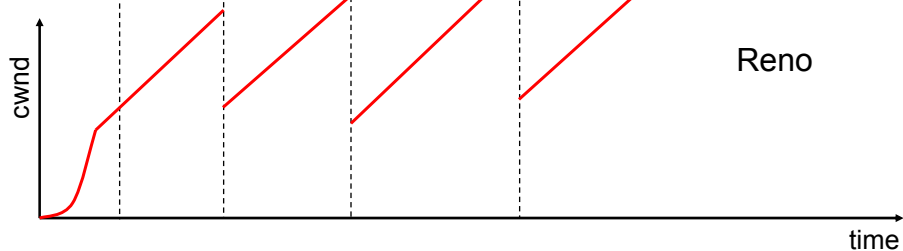
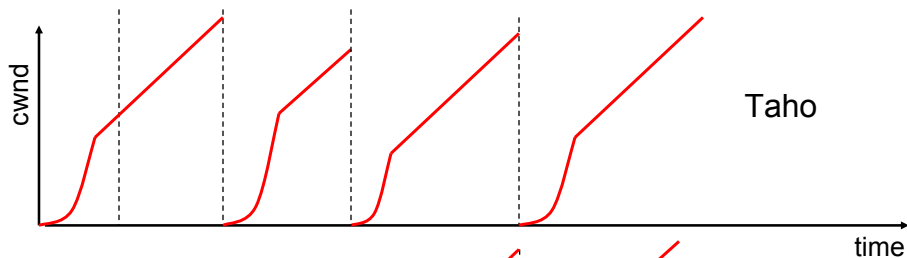
Out-of-order arrivals

TCP Reno

- Duplicate ACKs:
 - Fast retransmit
 - Fast recovery→ Fast Recovery avoids slow start
- Timeout:
 - Retransmit
 - Slow Start
- TCP Reno improves upon TCP Tahoe when a single packet is dropped in a round-trip time.

TCP Tahoe and TCP Reno

(for single segment losses)



TCP New Reno

- When multiple packets are dropped, Reno has problems
- Partial ACK:
 - Occurs when multiple packets are lost
 - A partial ACK acknowledges some, but not all packets that are outstanding at the start of a fast recovery, takes sender out of fast recovery
 - Sender has to wait until timeout occurs
- **New Reno:**
 - Partial ACK does not take sender out of fast recovery
 - Partial ACK causes retransmission of the segment following the acknowledged segment
- New Reno can deal with multiple lost segments without going to slow start

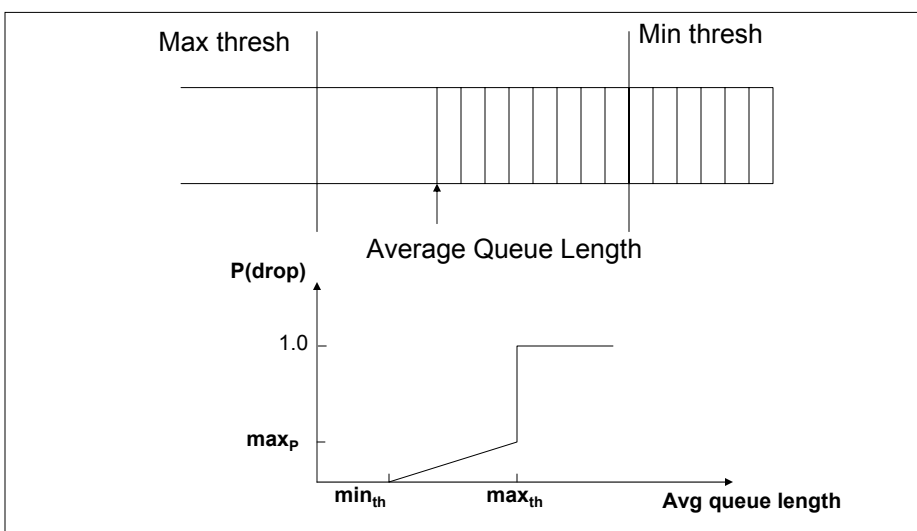
SACK

- SACK = Selective acknowledgment
- Issue: Reno and New Reno retransmit at most 1 lost packet per round trip time
- **Selective acknowledgments:** The receiver can acknowledge non-continuous blocks of data (SACK 0-1023, 1024-2047)
- Multiple blocks can be sent in a single segment.
- TCP SACK:
 - Enters fast recovery upon 3 duplicate ACKs
 - Sender keeps track of SACKs and infers if segments are lost. Sender retransmits the next segment from the list of segments that are deemed lost.

Active queue management (AQM)

- How do packets get dropped?
 - Buffer overflow at router
 - **Droptail:** A packet that arrives to a full buffer is dropped.
- Disadvantages of droptail:
 - Can be unfair to some flows
 - Buffers need to be large
- Solutions:
 - Drop a packet that is randomly chosen
 - Drop packets before queue is full
- AQM: Manage dropping behaviors of routers, often with the goal of improving TCP performance

Random Early Detection (RED)



Random Early Detection (RED)

- Maintain moving average of queue length Q
- If $\text{avg } Q < \text{min}_{th}$, do nothing
If $\text{avg } Q > \text{max}_{th}$, drop packet
Else mark (or drop) packet with a probability proportional to queue length
- Turns out that RED is difficult to tune

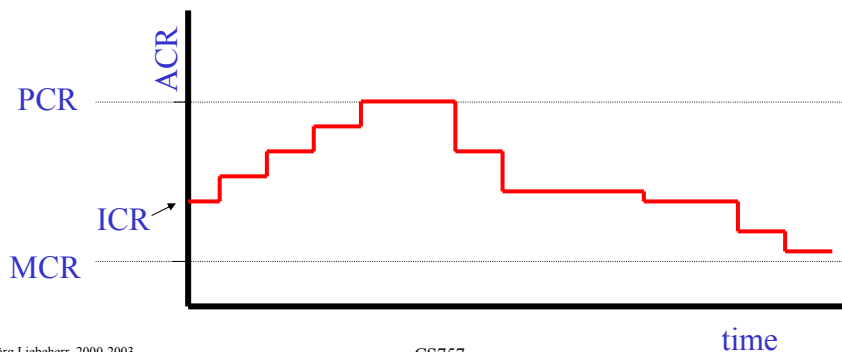
ABR Rate Control

ABR Rate Control

- ABR provides a closed loop rate control mechanism
- Developed by the ATM Forum in mid-1990s:
 - Too complex (more than 20 parameters)
 - Not widely used
- Goal: Set the rate of an ABR connection such that
 - it adapts to network congestion
 - bandwidth is shared fairly with other ABR connections
- Note: Compare to TCP congestion control (slow start/congestion avoidance)

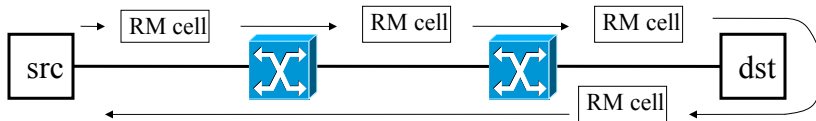
Setting the Rate

- Allowed Cell Rate (ACR): maximum current rate for source
- Initial Cell Rate (ICR): initial value for ACR
- PCR, MCR: upper and lower bound for ACR



Collect rate information with RM Cells

- ABR source sends periodically Resource Management (RM) cells
- RM cells:
 - interleaved with data cells (default: 1 RM cell for 32 data cells)
 - Destination writes information in RM cells and returns them to source

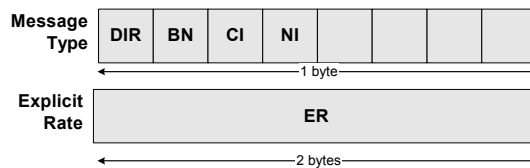


© Jörg Liebeherr, 2000-2003

CS757

RM Cells

- ABR rate control uses two fields in the RM cell:



DIR (direction): DIR = 0 (1): forward (backward) RM cell

BN (backward notification): BN = 0 (1): RM generated by source (switch)

CI (congestion indication): CI = 0 (1): no congestion (congestion)

NI (no increase): NI = 0 (1): may increase rate (don't increase rate)

Explicit rate: Maximum value for ACR

© Jörg Liebeherr, 2000-2003

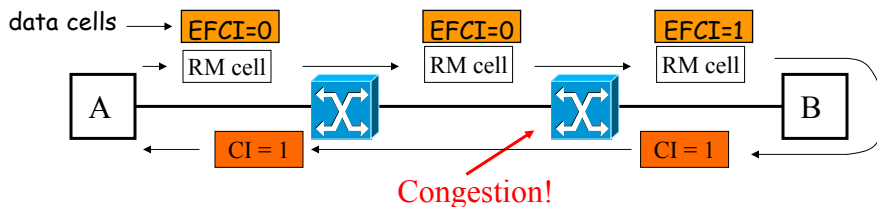
CS757

Two mechanisms for rate control

- There are two mechanisms for ABR rate control
- The ACR is set to the smallest value of the two
- **Binary Feedback**
 - FECN: Forward Explicit Congestion Notification
 - BECN: Backward Explicit Congestion Notification
 - Variant of additive increase / multiplicative decrease algorithm
- **Explicit Rate Feedback:** Fair share calculation

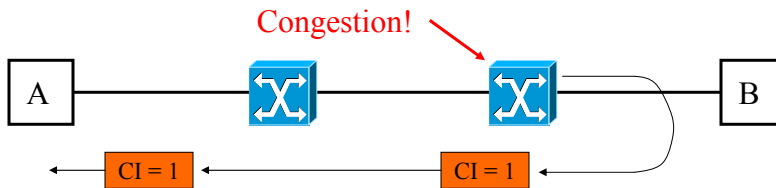
Forward Explicit Congestion Notification (FECN)

- Congested switch sets EFCI bit in data cell
- IF EFCI bit was set destination sets CI=1 in next RM cell



Backward Explicit Congestion Notification (BECN)

- Congested switch generates RM cell with CI=1 or NI=1



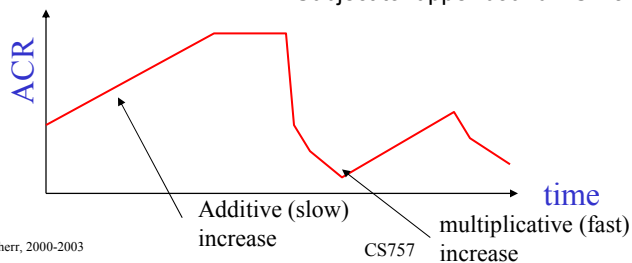
© Jörg Liebeherr, 2000-2003

CS757

Rate Adaptation at Sender

- Additive increase:
 - If CI=0 and NI=0 increase rate by a constant
$$ACR = ACR + \text{Constant}$$
- Multiplicative decrease:
 - If CI=1, decrease rate proportional to rate
$$ACR = ACR \times (1 - \text{Factor})$$

Subject to: upper bound PCR and lower bound MCR



© Jörg Liebeherr, 2000-2003

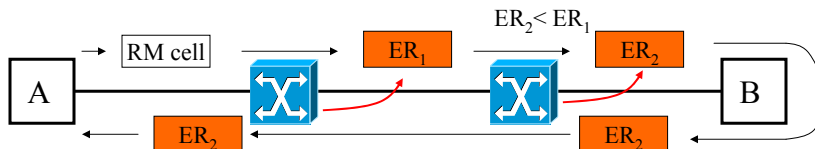
CS757

Explicit Rate Feedback (2)

- Each switch calculates a **fair share** for each ABR connection

$$FairShare = \frac{Target\ Rate}{\#ABR\ connections}$$

- A switch writes its fair share value in the ER field of an RM cell, if it reduces the current value of the ER field



- At the Sender: $ACR = \max(MCR, \min(PCR, ER))$