Chapter Outline

- How TCP performs Flow Control?
- Understand the need for TCP Congestion Control
- Dynamic Window Management
- Understand the issues of TCP over ATM
TCP Flow Control

- Uses a form of sliding window
- Differs from mechanism used in LLC, HDLC and others:
  - Decouples acknowledgement of received data units from granting permission to send more
- TCP’s flow control is known as a credit allocation scheme (i.e. credit-based):
  - And each transmitted octet has a sequence number
TCP Header Fields for Flow Control

- Sequence number (SN) of first octet/byte in data segment
- Acknowledgement (ACK) number (AN) → next octet to receive, (if any)
- Window (W)
- If ACK contains AN = i, W = j:
  - Octets through SN = i - 1 acknowledged
  - Permission is granted to send W = j more octets, i.e., from octets i through i + j - 1
Chapter 12 TCP Traffic Control

(a) Send sequence space

(b) Receive sequence space
Chapter 12 TCP Traffic Control

TCP Credit Allocation Mechanism

Transport Entity A

...1000 1001 2401...
A may send 1400 octets

...1000 1001 1601 2401...
A shrinks its transmit window with each transmission

...1000 1001 2001 2401...
A adjusts its window with each credit

...1600 1601 2601 2601...
A exhausts its credit

...2600 2601 4000 4001...
A receives new credit

Transport Entity B

...1000 1001 2401...
B is prepared to receive 1400 octets, beginning with 1001

...1600 1601 2601...
B acknowledges 3 segments (600 octets), but is only prepared to receive 200 additional octets beyond the original budget (i.e., B will accept octets 1601 through 2600)

...1600 1601 2001 2601...
B acknowledges 5 segments (1000 octets) and restores the original amount of credit

AN = 2601, W = 1400
Credit Allocation is Flexible

Suppose last message B issued was AN = i, W = j:

- To increase credit to k (k > j) when no new data, B issues AN = i, W = k
- To acknowledge a segment containing m octets (m < j) without allocating more credit, B issues AN = i + m, W = j − m
Credit Policy

- Receiver needs a policy for how much credit to give sender
- Conservative approach: grant credit up to limit of available buffer space
  - May limit throughput in long-delay situations
- **Optimistic** approach: grant credit based on expectation of freeing space before data arrives
Effect of Window Size

\[ W = \text{TCP window size (octets)} \]
\[ R = \text{Data rate (bps) at TCP source} \]
\[ D = \text{Propagation delay (seconds)} \]

- After TCP source begins transmitting, it takes \( D \) seconds for first octet to arrive, and \( D \) seconds for acknowledgement to return.
- TCP source could transmit at most \( 2RD \) bits, or \( RD/4 \) octets (i.e. rate-delay product).
Normalized Throughput $S$

$$S = \begin{cases} 
1 & \text{W} \geq \text{RD}/4 \\
\frac{4W}{\text{RD}} & \text{W} < \text{RD}/4
\end{cases}$$
Default max wnd size = $2^{16} - 1$

Scaled Wnd size = $2^{16} \times 2^4 - 1 = 2^{20} - 1$
Complicating Factors

- Multiple TCP connections are multiplexed over same network interface, reducing available R and efficiency.
- For multi-hop connections, D is the sum of delays across each network plus delays at each router.
- If source data rate R exceeds data rate on one of the hops, that hop will be a bottleneck.
- Lost segments are retransmitted, reducing throughput.
  - Impact depends on retransmission policy.
Retransmission Strategy

- TCP relies exclusively on positive ACKs and retransmission on ACK timeout.
- There is no explicit negative ACK.
- Retransmission required when:
  - Segment arrives damaged, as indicated by checksum error, causing receiver to discard segment.
  - Segment fails to arrive, lost!
Timers

- A timer is associated with each segment as it is sent, i.e. retransmission timer (RTO)
- If timer expires before segment ACKed, sender must retransmit
- Key Design Issue:
  - Find a suitable value of retransmission timer??
  - Too small: many unnecessary retransmissions, wasting network bandwidth
  - Too large: delay in handling lost segment
Two Strategies

- Timer should be longer than round-trip time (RTT) (send segment, receive ACK)
- Also, remember delay is variable

Strategies:

- Fixed timer
- Adaptive
Some problems with Adaptive Schemes

- Peer TCP entity performs cumulative acknowledgements and does not acknowledge immediately.
- For retransmitted segments, cannot tell whether ACK is response to original transmission or retransmission.
- Network conditions may change suddenly.
- However, adaptive is still better than fixed timer!
First try: Adaptive Retransmission Timer

- Average Round-Trip Time (ARTT)

\[
ARTT(K + 1) = \frac{1}{K + 1} \sum_{i=1}^{K+1} RTT(i)
\]

\[
= \frac{K}{K + 1} ARTT(K) + \frac{1}{K + 1} RTT(K + 1)
\]

- Each term given same weight \([1/(K+1)]\)
- May not adjust properly to recent changes as this more likely reflect future behaviour
  - Give more weight to recent values!
RFC 793 Exponential Averaging

- Smoothed Round-Trip Time (SRTT)

\[ SRTT(K + 1) = \alpha \times SRTT(K) + (1 - \alpha) \times RTT(K + 1) \]

- To see the effect of \( \alpha \), let's see SRTT expansion:

\[ SRTT(K + 1) = (1 - \alpha) \times RTT(K + 1) + \alpha(1 - \alpha) \times RTT(K) + \alpha^2(1 - \alpha) \times RTT(K-1) + \ldots + \alpha^K(1 - \alpha) \times RTT(1) \]

- E.g., if \( \alpha = 0.8 \):

\[ SRTT(K + 1) = 0.2 \times RTT(K + 1) + 0.16 \times RTT(K) + 0.128 \times RTT(K-1) \ldots \]

- The older the observation, the less it is counted in the average (\( 0 \leq \alpha \leq 1 \))
Exponential Smoothing Coefficients
Figure 12.5 Use of Exponential Averaging
RFC 793 Retransmission Timeout

- RTO should be set slightly > current SRTT

\[ RTO(K + 1) = \text{Min}(UB, \text{Max}(LB, \beta \times SRTT(K + 1))) \]

UB, LB: prechosen fixed upper and lower bounds

Example values for \( \alpha, \beta \):

\[ 0.8 < \alpha < 0.9 \quad 1.3 < \beta < 2.0 \]
TCP Implementation Policy Options

- **Send** – may transmit per batch or buffer limit
- **Deliver** – similar as send primitive
- **Accept**
  - In-order – discard out of order segments (Go Back N)
  - In-window – accept segments in receive window (SR)
- **Retransmit**
  - First-only – 1 timer for a queue; expire – retransmit front of queue
  - Batch – 1 timer for a queue; expire – retransmit all in queue
  - Individual – 1 timer for each segment
- **Acknowledge**
  - immediate
  - cumulative
TCP Congestion Control

- Dynamic routing can alleviate congestion by spreading load more evenly
- But only effective for unbalanced loads and brief surges in traffic
- Congestion can only be controlled by limiting amount of data entering network!!
- RSVP signalling may help but not widely implemented (see later)
TCP Congestion Control is Difficult

- IP is connectionless and stateless, with no provision for detecting or controlling congestion
- TCP only provides end-to-end flow control
- No cooperative, distributed algorithm to bind together various TCP entities
TCP Flow and Congestion Control

- Need to distinguish them; not same!
- The rate at which a TCP entity can transmit is determined by rate of incoming ACKs to previous segments with new credit
- Rate of ACK arrival determined by round-trip path between source and destination
- Bottleneck may be destination or internet
  - Sender cannot tell which
- **And only the internet bottleneck can be due to congestion**
Figure 12.6  TCP Segment Pacing
TCP Flow and Congestion Control

- Ideally, returning ACKs pace the signal in steady state
- The sender rate matches the ACK arrival rate
  - Equals to the slowest link on the path
  - Such regulation of flow is known as TCP *self-clocking* behaviour
- However, this is rarely achieved due to fluctuation of queueing delays at the routers and switches
- So, using a constant factor for RTO may not be suitable (RFC 793) – need to consider RTT variability
Retransmission Timer Management

Three Techniques to calculate RTO:
- RTT Variance Estimation
- Exponential RTO Backoff
- Karn’s Algorithm
RTT Variance Estimation (Jacobson’s Algorithm)

3 sources of high variance in RTT:

- If data rate relatively low, then transmission delay (T) will be relatively large, with larger variance due to variance in packet size
- Load may change abruptly due to other sources
- Peer may not acknowledge segments immediately
Jacobson’s Algorithm

Introduced a variation measure called mean deviation
\[ \text{MDEV}(X) = \mathbb{E}[|X - \mathbb{E}[X]|] \]

\[ SRTT(K + 1) = (1 - g) \times SRTT(K) + g \times RTT(K + 1) \]

\[ SERR(K + 1) = RTT(K + 1) - SRTT(K) \]

\[ SDEV(K + 1) = (1 - h) \times SDEV(K) + h \times |SERR(K + 1)| \]

SDEV is a RTT variability factor, similar to mdev

\[ \text{RTO}(K + 1) = SRTT(K + 1) + f \times SDEV(K + 1) \]

\[ g = 0.125 \]
\[ h = 0.25 \]
\[ f = 2 \text{ or } f = 4 \] (most current implementations use \( f = 4 \))
Figure 12.8  Jacobson’s RTO Calculation
Two Other Factors

Jacobson’s algorithm can significantly improve TCP performance, but:

- What RTO to use for retransmitted segments?
  ANSWER: exponential RTO backoff algorithm

- Which round-trip samples to use as input to Jacobson’s algorithm?
  ANSWER: Karn’s algorithm
Exponential RTO Backoff

- Increase RTO each time the same segment retransmitted – backoff process
- Multiply RTO by constant:
  \[ RTO = q \times RTO \]
- When \( q = 2 \) is called binary exponential backoff (similar to Ethernet backoff)
Which Round-trip Samples?

- If an ack is received for retransmitted segment, there are 2 possibilities:
  - Ack is for first transmission
  - Ack is for second transmission
- TCP source cannot distinguish these 2 cases
- No valid way to calculate RTT:
  - From first transmission to ack, or
  - From second transmission to ack?
Karn’s Algorithm

- Do **not** use measured RTT of retransmitted segments to update SRTT and SDEV
- Calculate backoff RTO when a retransmission occurs
- Use backoff RTO for segments until an ACK arrives for a segment that has not been retransmitted
  - Then Jacobson’s algorithm is reactivated to calculate RTO
Window Management

- Besides improving RTO calculation, we need to know how to manage sender window => has direct effect on congestion

- Five important techniques are:
  - Slow start
  - Dynamic window sizing on congestion
  - Fast retransmit
  - Fast recovery
  - Limited transmit
1. Slow Start

- TCP uses 2 windows to determine how much data to transmit (i.e. a probing approach)

\[ \text{awnd} = \text{MIN}[\text{credit}, \text{cwnd}] \]

where

- \( \text{awnd} = \) allowed window in segments
- \( \text{cwnd} = \) congestion window in segments
- \( \text{credit} = \) amount of unused credit granted in most recent ACK in segments (i.e. advertised receiver window)

- \( \text{cwnd} = 1 \) for a new connection and increased by 1 for each ACK received, up to a threshold (\( \text{ssthresh} \))
Effect of Slow Start
2. Dynamic Window Sizing on Congestion

- A lost segment indicates congestion (implicit approach)
- Prudent to reset cwnd = 1 and restart slow start process
- May not be conservative enough: “easy to drive a network into saturation but hard for the net to recover” (Jacobson)
- Instead, use slow start up to a smaller ssthresh (cwnd/2) and then, linear growth in subsequent period
Figure 12.11 Illustration of Slow Start and Congestion Avoidance

- 4 RTTs to reach 16
- (3+8) RTTs to reach 16

Congestion Avoidance

SS
3. Fast Retransmit

- RTO is generally noticeably longer than actual RTT
- If a segment is lost, TCP may be slow to retransmit
- TCP rule: if a segment is received out of order, an ACK must be issued immediately by rcvr for the last in-order segment
- **Fast Retransmit rule**: if 4 ACKs received for same segment, very likely it was lost, so retransmit immediately, rather than waiting for timeout
4. Fast Recovery

- When TCP retransmits a segment using Fast Retransmit, a segment was assumed lost.
- *Congestion avoidance* measures are appropriate at this point.
  - E.g., slow-start/congestion avoidance procedure.
- However, this may be too conservative since multiple ACKs indicate other segments are getting through.
- **Fast Recovery**: retransmit lost segment, cut cwnd in half (ssthresh also halved), proceed with linear increase of cwnd.
  - Use only congestion avoidance procedure.
5. Limited Transmit

- If congestion window at sender is small, fast retransmit may not get triggered, e.g., cwnd = 3
  - Under what circumstances does sender have small congestion window?
    - Limited data to tx, small RD product connection
  - Is the problem common?
    - Yes! More RTO expiration than Fast Retransmit.
  - If the problem is common, why not reduce number of duplicate acks needed to trigger retransmit?
    - May lead to unnecessary retransmissions
Limited Transmit Algorithm

RFC Index

4655 A Path Computation Element (PCE)-Based Architecture A. Farrel, J.-P. Vasseur, J. Ash [ August 2006 ] (TXT = 97561 bytes)
4647 Matching of Language Tags A. Phillips, M. Davis [ September 2006 ] (TXT = 45595 bytes)(Obsoletes RFC3066) (Also BCP47)
4646 Tags for Identifying Languages A. Phillips, M. Davis [ September 2006 ] (TXT = 135810 bytes)(Obsoletes RFC3066) (Also BCP47)
4645 Initial Language Subtag Registry D. Ewell [ September 2006 ] (TXT = 15517 bytes)
4633 HMAC SHA (Hashed Message Authentication Code, Secure Hash Algorithm) TSIG Algorithm Identifiers D. Eastlake 3rd [ August 2006 ] (TXT = 16523 bytes)
4632 Experiment in Long-Term Suspensions From Internet Engineering Task Force (IETF) Mailing Lists S. Hartman [ August 2006 ] (TXT = 14979 bytes)
4631 Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan V. Fuller, T. Li [ August 2006 ] (TXT = 66944 bytes)(Obsoletes RFC1222)
4630 Update to DirectoryString Processing in the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile R. Housley, S

...A snapshot from www.rfc-editor.org

Chapter 12 TCP Traffic Control
Performance of TCP over ATM

- How best to manage TCP’s segment size, window management and congestion control...
- ...at the same time as ATM’s quality of service and traffic control policies
- TCP may operate end-to-end over one ATM network, or there may be multiple ATM LANs or WANs with non-ATM networks
E.g.: TCP over AAL5/ATM Network
TCP Performance using UBR class

- **Buffer capacity** at ATM switches is a critical parameter in assessing TCP throughput performance.
- Insufficient buffer capacity results in lost TCP segments and retransmissions.
- Certain informed dropping policies should be adopted, like PPD or EPD.
Partial Packet and Early Packet Discard Schemes

- Designed to reduce the transmission of useless cells
- Work on a per-virtual circuit basis
- Partial Packet Discard (PPD)
  - If a cell is dropped, then drop all subsequent cells in that segment (i.e., look for cell with SDU type bit set to one)
- Early Packet Discard (EPD)
  - When a switch buffer reaches a threshold level, preemptively discard all cells in a segment
  - More effective
Effect of Switch Buffer Size

An experiment performed for this scenario:

- Data rate of 141 Mbps
- End-to-end propagation delay of 6 μs
- IP packet sizes of 512 to 9180 octets
- TCP window sizes from 8 KB to 64 KB
- ATM switch buffer size per port from 256 to 8000 cells
- One-to-one mapping of TCP connections to ATM virtual circuits
- TCP sources have infinite supply of data ready
Chapter 12 TCP Traffic Control

- Reactive
  - Proactive

(a) Packet TCP over non-ATM
(b) TCP over plain ATM
(c) Partial Packet Discard
(d) Early Packet Discard
Observations

- If a single cell is dropped, other cells in the same IP datagram are unusable, yet plain ATM network forwards these useless cells to destination (Fig. (b))
- Smaller buffer increase probability of dropped cells
- Larger segment size increases number of useless cells transmitted if a single cell dropped
- EPD bias towards shorter IP packets and connections that pass through multiple congested switches – fairness issue
ATM Switch Buffer Layout for fair schemes

\[ B = \text{Buffer capacity of the switch, in cells} \]
\[ R = \text{Threshold parameter; } R < B \]
\[ N = \text{Current # of cells; } N \leq B \]
\[ N(i) = \text{Current # of cells for VC } i \]
\[ V = \text{# of active VCs} \]
Selective Drop

- Ideally, N/V cells should be buffered for each of the V virtual circuits
  \[ W(i) = \frac{N(i)}{N/V} = \frac{N(i) \times V}{N} \]
  - If \( N > R \) and \( W(i) > Z \)
    - then drop next new packet on \( VC_i \)

- \( W(i) > 1 \) \( \Rightarrow \) \( VC_i \) *unfairly* getting more bandwidth than others
- \( Z \) is a parameter to be chosen, close to 1
Fair Buffer Allocation (FBA)

- Need to be more aggressive dropping packets as congestion increases.
- Drop new packet when:
  \[ N > R \text{ and } W(i) > Z \times \frac{B - R}{N - R} \]
  where \((B - R)\) is constant.
- When buffer just passed \(R\), only misbehaving VCs are affected.
- As buffer fills up, situation is serious and more VCs are affected.
TCP over ABR

- Good performance of TCP over UBR can be achieved with minor adjustments to switch mechanisms.
- This reduces the incentive to use the more complex and more expensive ABR service.
- Performance and fairness of ABR is quite sensitive to some ABR parameter settings.
- Overall, ABR does not provide significant performance over simpler and less expensive UBR-EPD or UBR-EPD-FBA.
Summary

- TCP flow control is a credit-based scheme
- An adaptive RTO (timer) is necessary
- TCP congestion control requires proper window management
- Performance of IP over ATM strongly depends on switch buffer size and the dropping policy
- UBR is an simpler and less costly class
- Next: Congestion control in ATM