UNIVERSITY OF Southampton

School of Electronics and Computer Science

Transport Layer 3

ELEC3227/ELEC6255

Alex Weddell
asw@ecs.soton.ac.uk

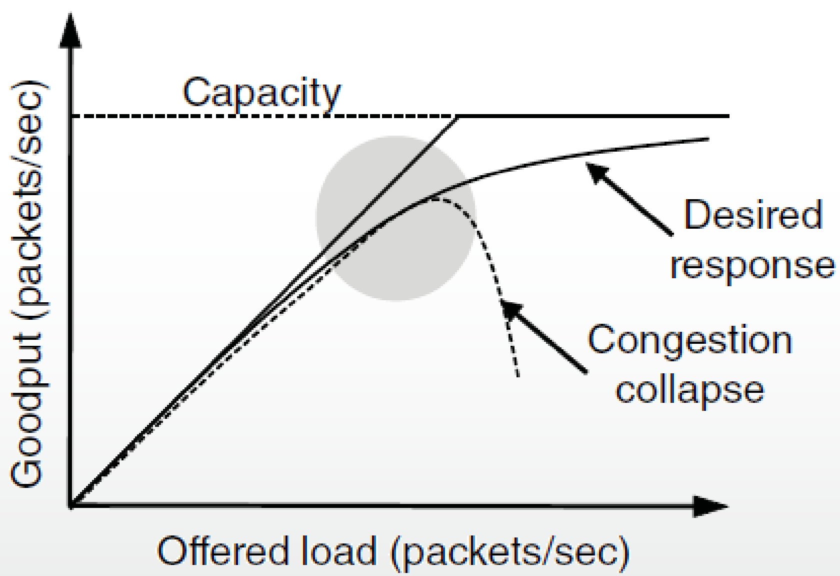
UNIVERSITY OF Southampton
School of Electronics and Computer Science

Overview

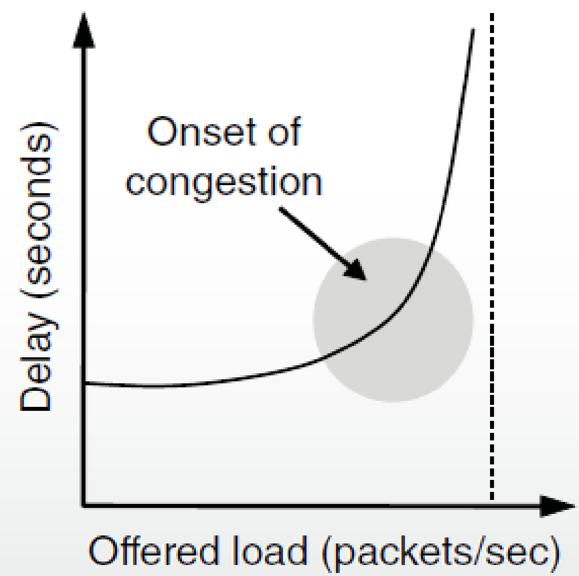
- Regulating the Sending Rate
- TCP Connection State
- TCP Sliding Window
- TCP Congestion Control

Desirable Bandwidth Allocation

- Efficient use of bandwidth gives high goodput, low delay



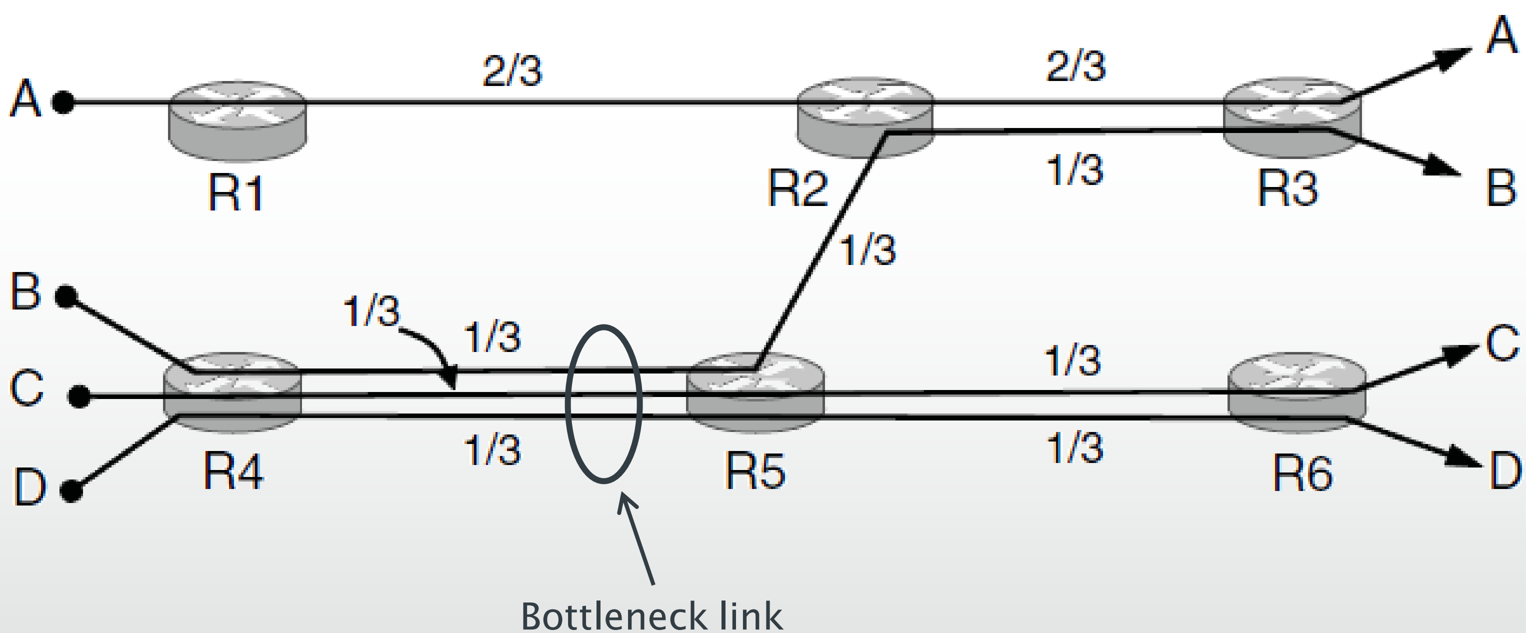
Goodput rises more slowly than load when congestion sets in



Delay begins to rise sharply when congestion sets in

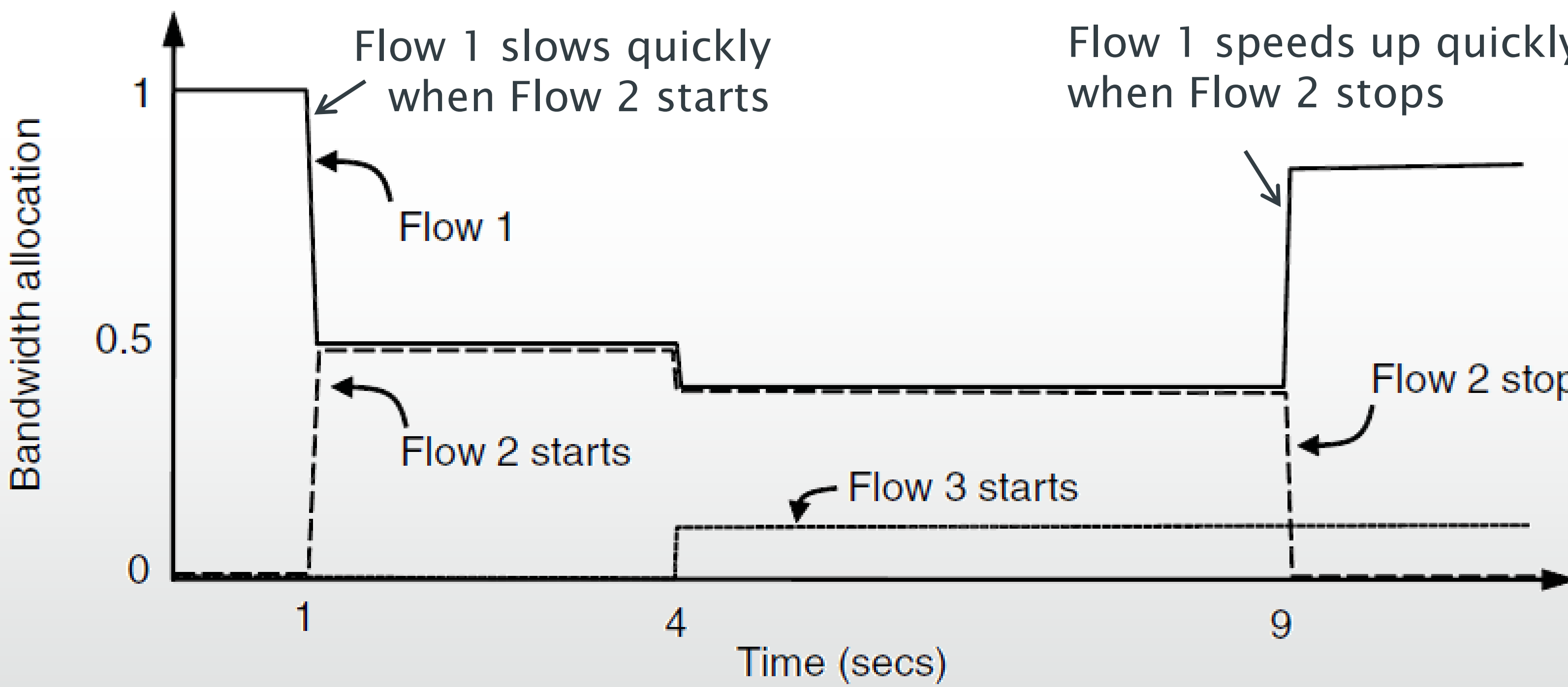
Desirable Bandwidth Allocation

- Fair use gives bandwidth to all flows (no starvation)
  - Max-min fairness gives equal shares of bottleneck



Desirable Bandwidth Allocation

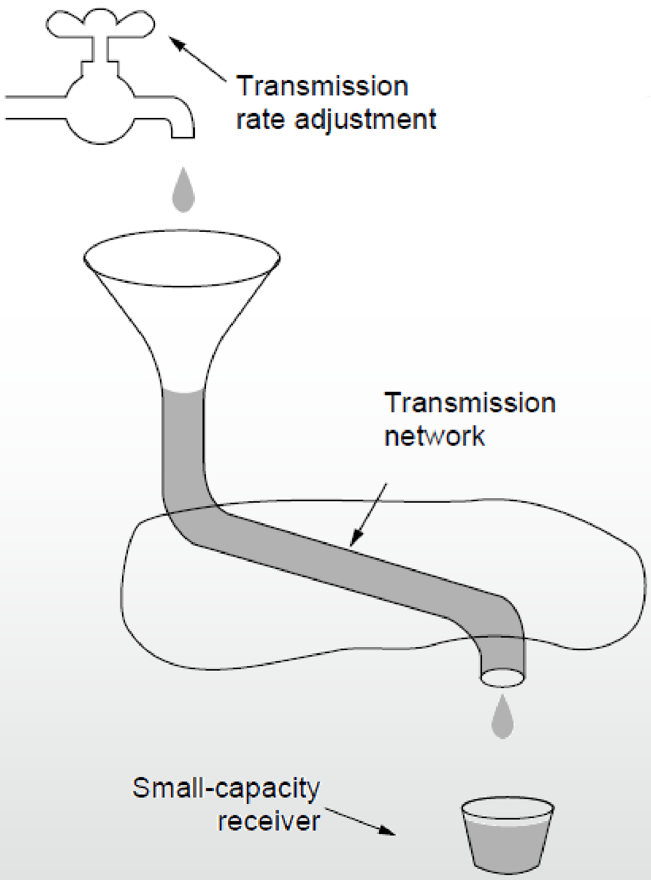
- We want bandwidth levels to converge quickly when traffic patterns change



Regulating the Sending Rate

Sender may need to slow down for different reasons:

- **Flow control**, when the receiver is not fast enough



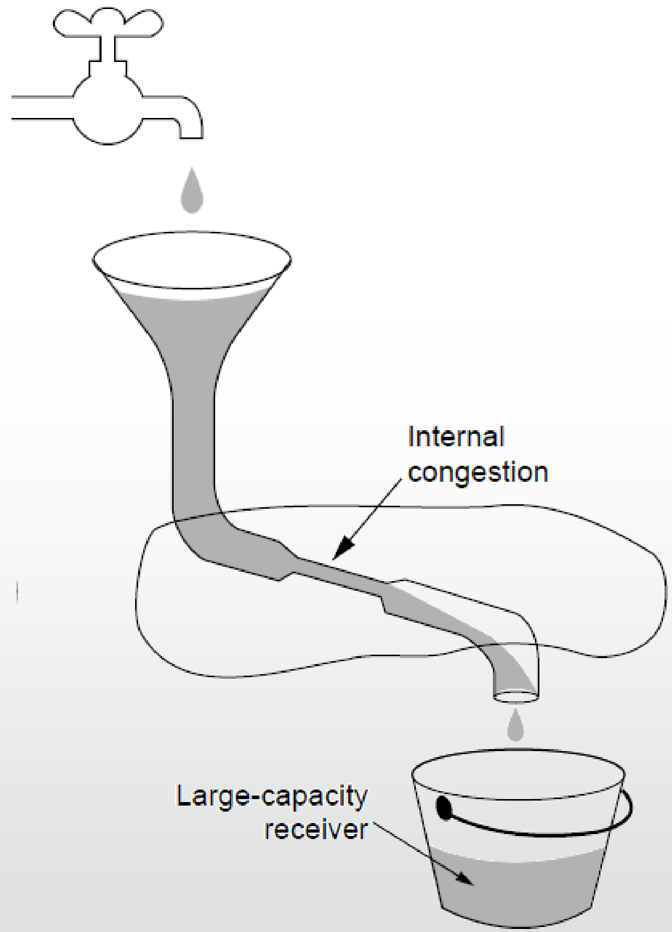
A fast network feeding a low-capacity receiver → flow control is needed

Regulating the Sending Rate

Sender may need to slow down for different reasons:

- **Flow control**, when the receiver is not fast enough
- **Congestion**, when the network is not fast enough

Focus here is on dealing with congestion.



A slow network feeding a high-capacity receiver → congestion control is needed

UNIVERSITY OF
Southampton
School of Electronics
and Computer Science

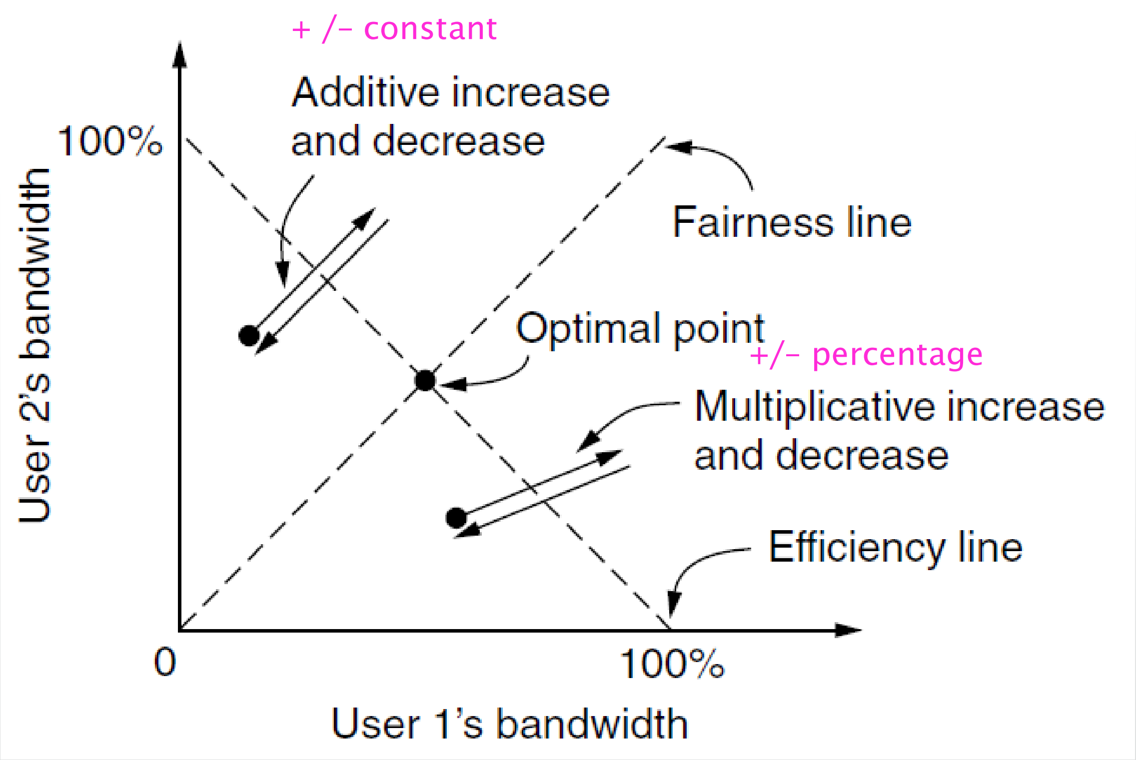
Regulating the Sending Rate

- Different congestion signals the network may use to tell the transport endpoint to slow down (or speed up)

Protocol	Signal	Explicit?	Precise?
XCP	Rate to use	Yes	Yes
TCP with ECN	Congestion warning	Yes	No
FAST TCP	End-to-end delay	No	Yes
CUBIC TCP	Packet loss	No	No
TCP	Packet loss	No	No

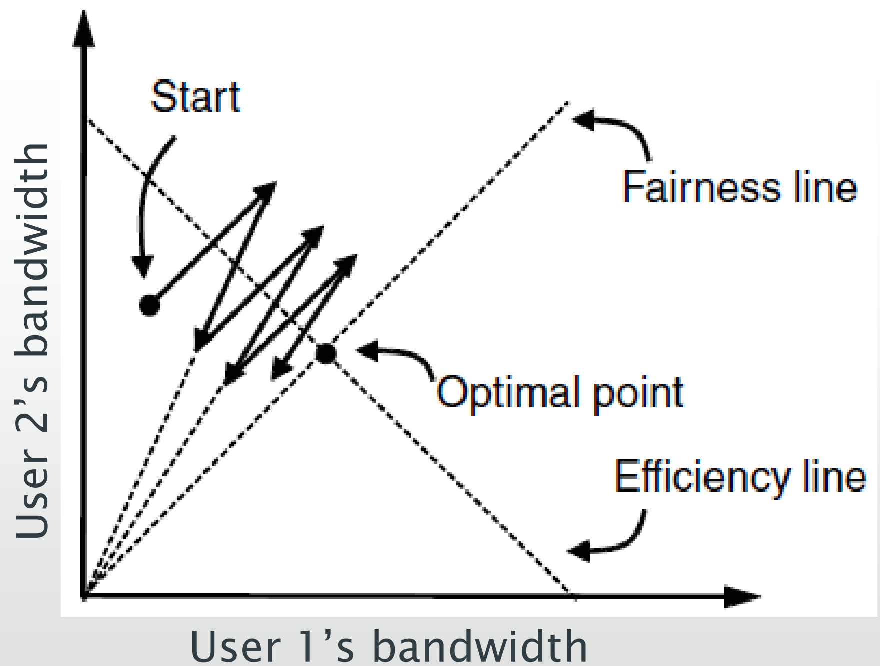
Regulating the Sending Rate

- If two flows increase/decrease their bandwidth in the same way when the network signals free/busy they will not converge to a fair allocation



Regulating the Sending Rate

- The AIMD (Additive Increase Multiplicative Decrease) control law does converge to a fair and efficient point!
  - TCP uses AIMD for this reason



TCP Service Model

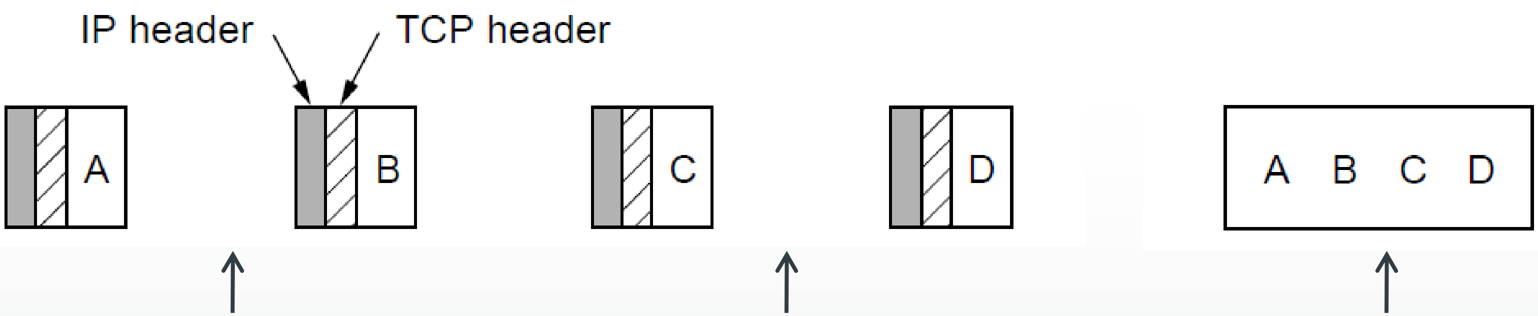
- TCP provides applications with a reliable byte stream between processes; it is the workhorse of the Internet
  - Popular servers run on well-known ports

Port	Protocol	Use
20, 21	FTP	File transfer
22	SSH	Remote login, replacement for Telnet
25	SMTP	Email
80	HTTP	World Wide Web
110	POP-3	Remote email access
143	IMAP	Remote email access
443	HTTPS	Secure Web (HTTP over SSL/TLS)
543	RTSP	Media player control
631	IPP	Printer sharing

University of Southampton
School of Electronics and Computer Science

TCP Service Model

- Applications using TCP see only the byte stream [right] and not the segments [left] sent as separate IP packets



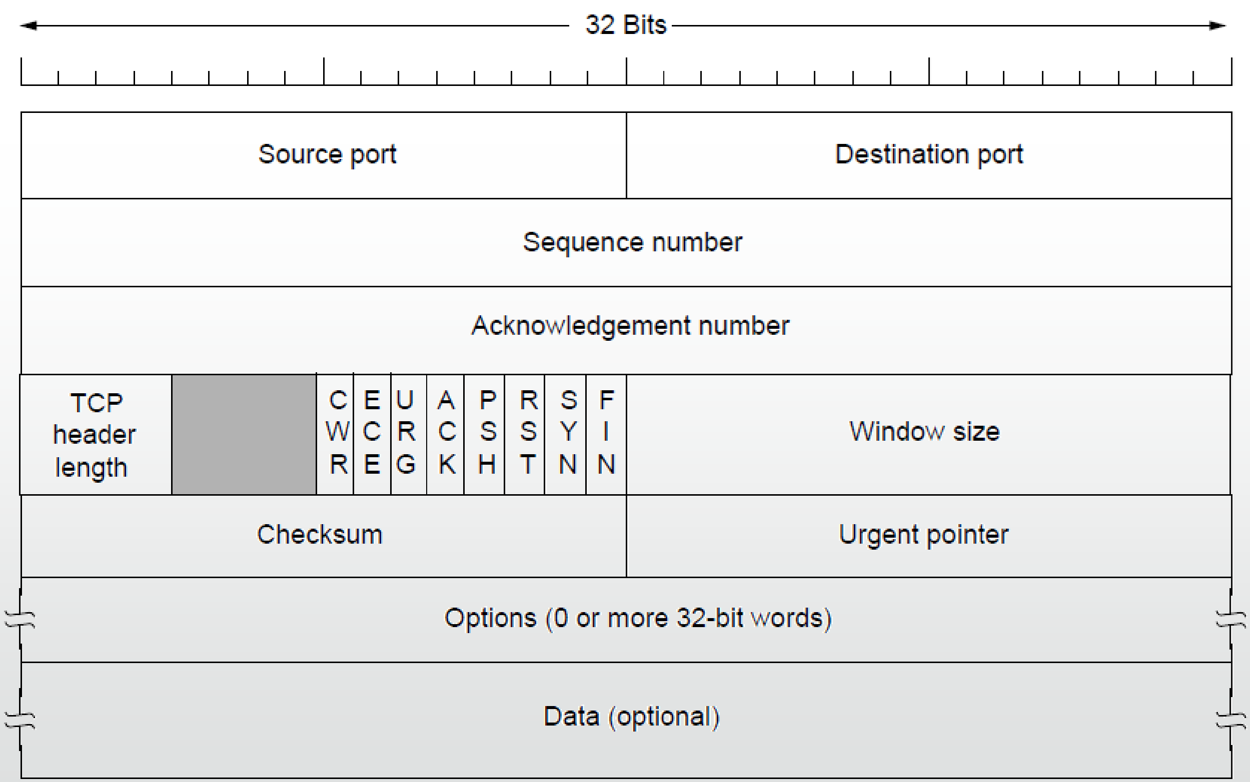
Four segments, each with 512 bytes of data and carried in an IP packet

2048 bytes of data delivered to application in a single READ call

UNIVERSITY OF Southampton
School of Electronics and Computer Science

TCP Segment Header

- TCP header includes addressing (ports), sliding window (seq. / ack. number), flow control (window), error control (checksum) and more.



UNIVERSITY OF Southampton
School of Electronics and Computer Science

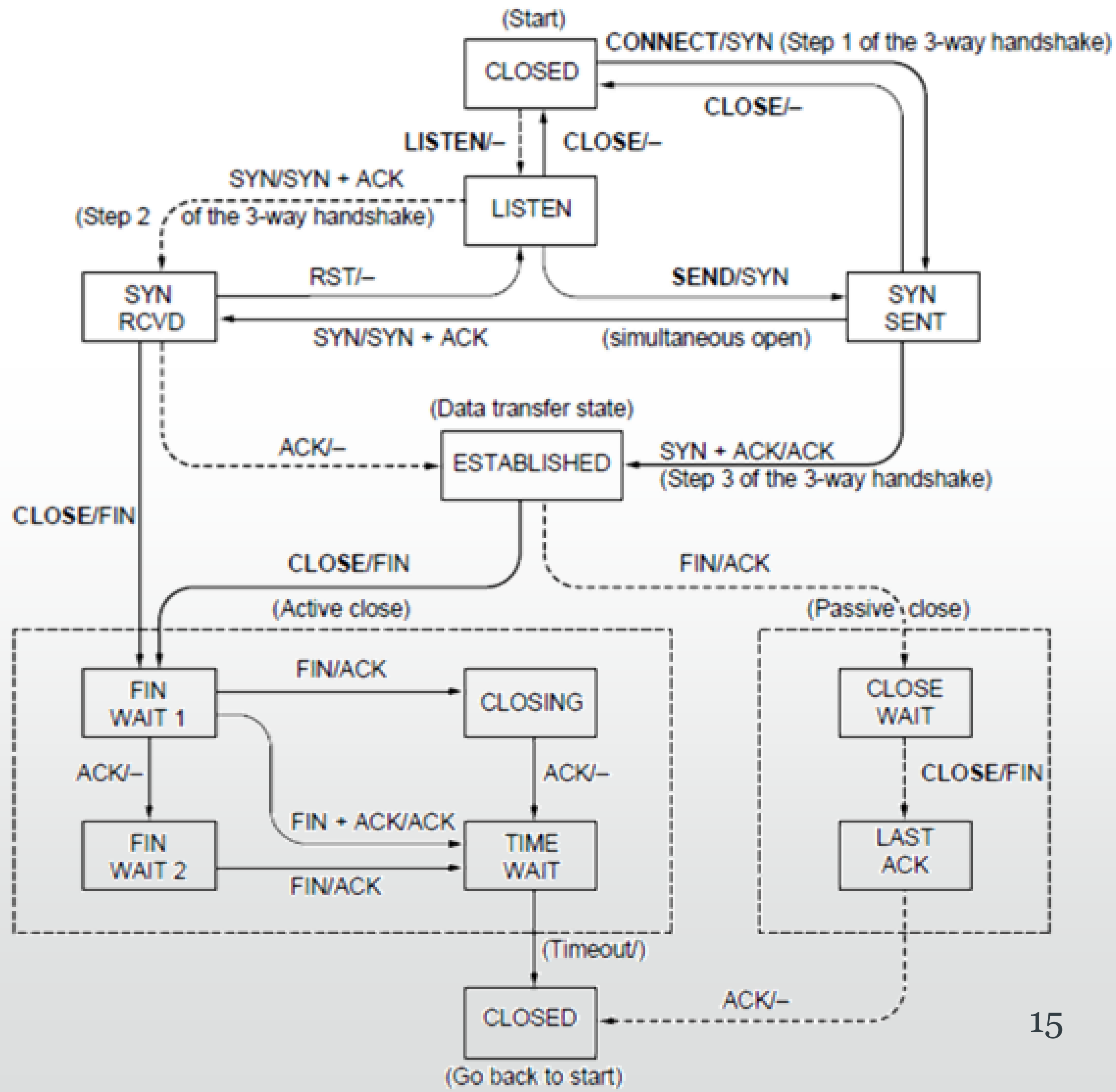
TCP Connection State Modelling

- The TCP connection finite state machine has more states than the simple example from earlier.

State	Description
CLOSED	No connection is active or pending
LISTEN	The server is waiting for an incoming call
SYN RCVD	A connection request has arrived; wait for ACK
SYN SENT	The application has started to open a connection
ESTABLISHED	The normal data transfer state
FIN WAIT 1	The application has said it is finished
FIN WAIT 2	The other side has agreed to release
TIME WAIT	Wait for all packets to die off
CLOSING	Both sides have tried to close simultaneously
CLOSE WAIT	The other side has initiated a release
LAST ACK	Wait for all packets to die off

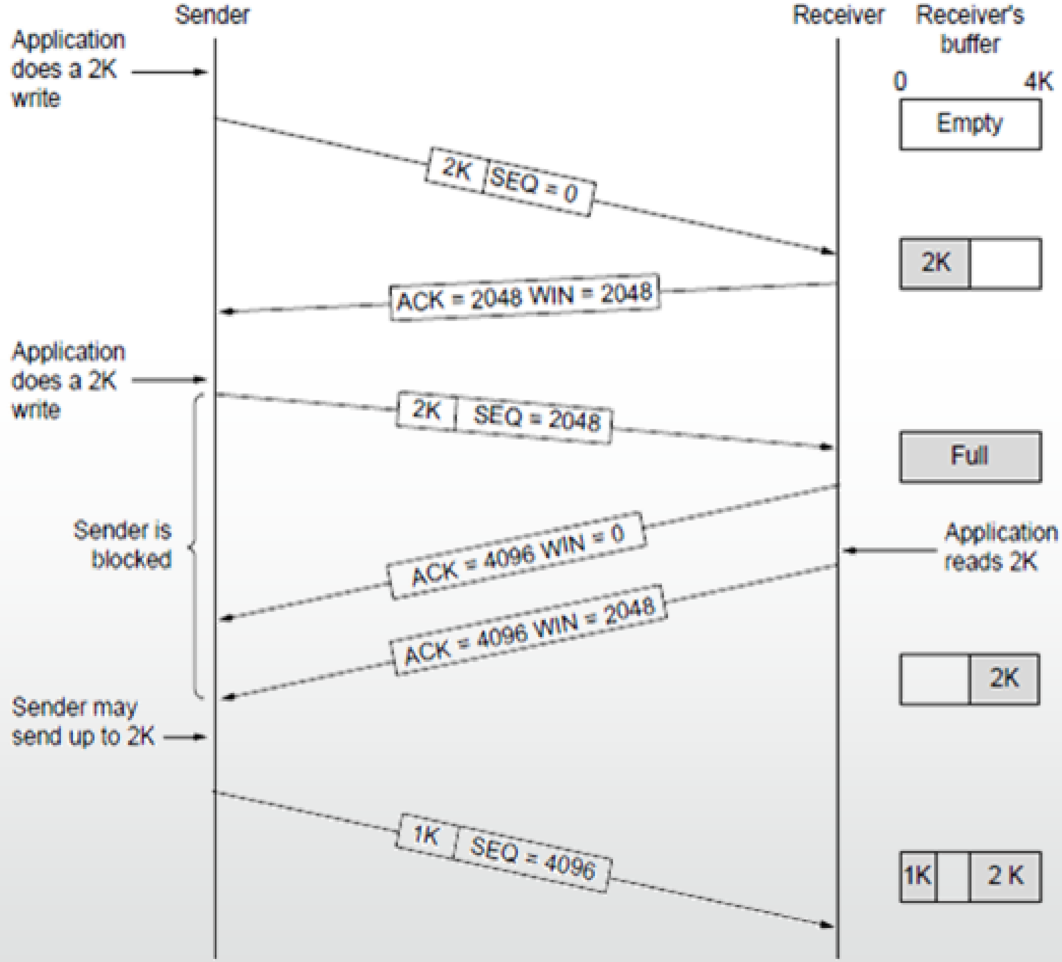
TCP Connection State Modelling

- Solid line is the normal path for a client.
- Dashed line is the normal path for a server.
- Light lines are unusual events.
- Transitions are labeled by the cause and action, separated by a slash.



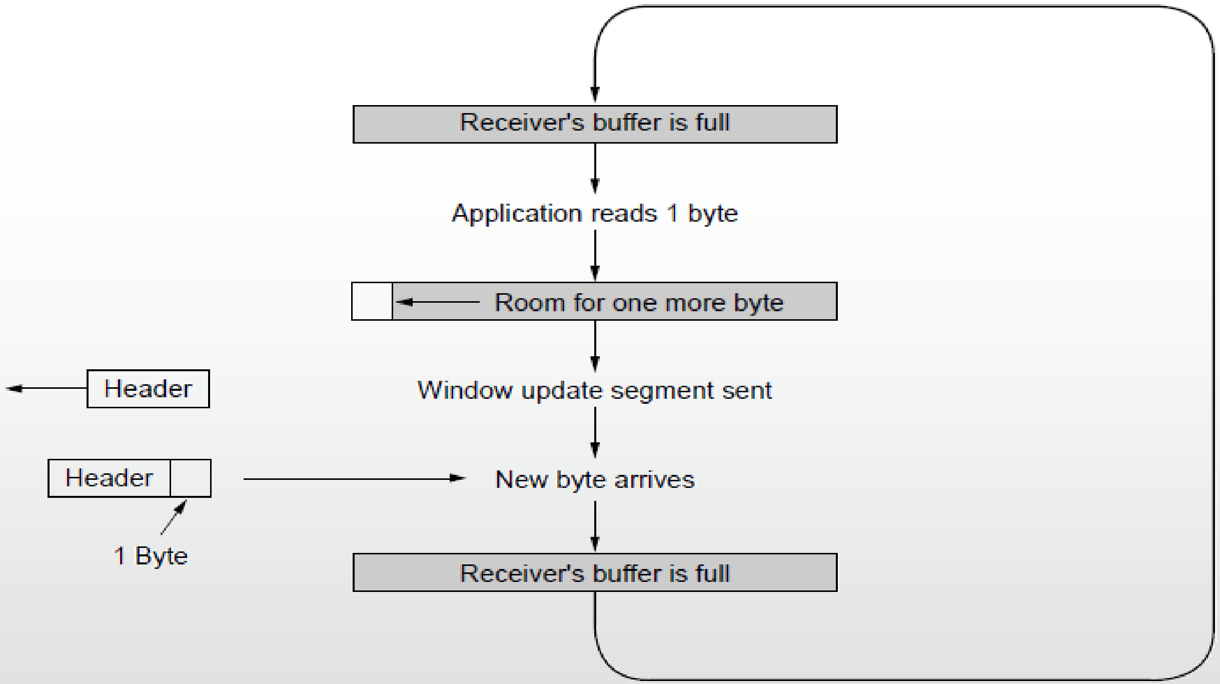
TCP Sliding Window

- TCP adds flow control to the sliding window as before
  - ACK + WIN is the sender's limit



TCP Sliding Window

- Need to add special cases to avoid unwanted behavior
  - e.g., silly window syndrome

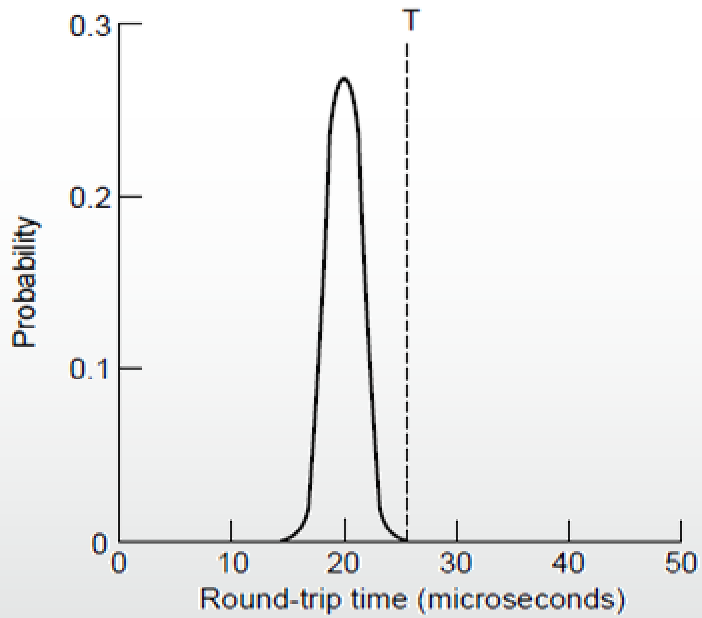


Receiver application reads single bytes, so sender always sends one byte segments

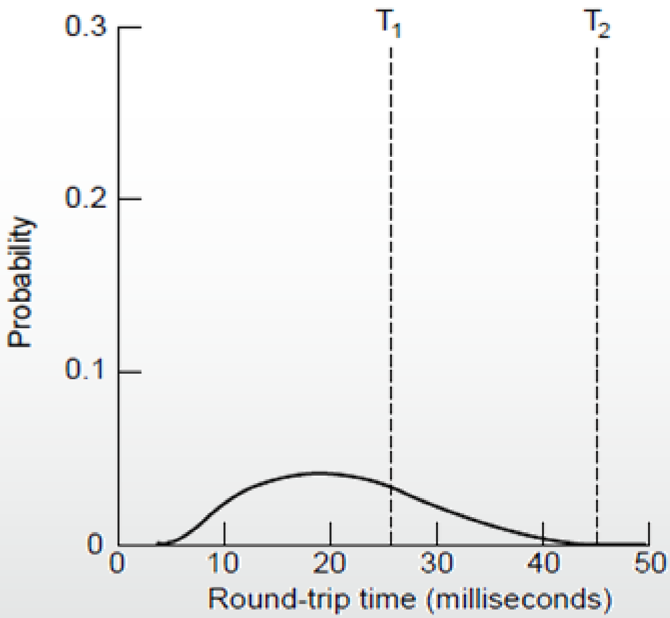
UNIVERSITY OF Southampton
School of Electronics and Computer Science

TCP Timer Management

- TCP estimates retransmit timer from segment RTTs
  - Tracks both average and variance (for Internet case)
  - Timeout is set to average plus 4 x variance



LAN case – small, regular RTT



Internet case – large, varied RTT

UNIVERSITY OF Southampton
School of Electronics and Computer Science

TCP Congestion Control

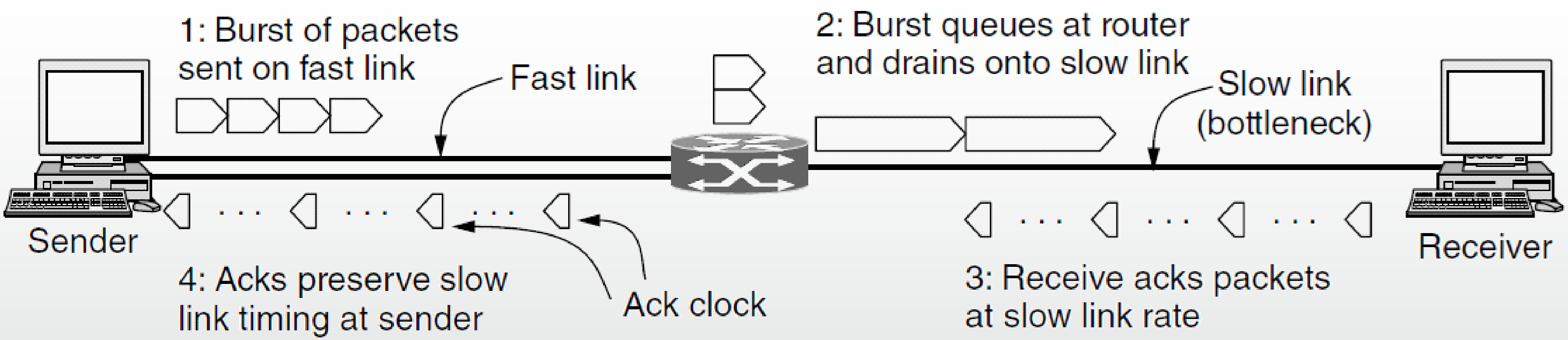
TCP uses AIMD with loss signal to control congestion

- Implemented as a congestion window (cwnd) for the number of segments that may be in the network
- Uses several mechanisms that work together

Name	Mechanism	Purpose
ACK clock	Congestion window (cwnd)	Smooth out packet bursts
Slow-start	Double cwnd each RTT	Rapidly increase send rate to reach roughly the right level
Additive Increase	Increase cwnd by 1 packet each RTT	Slowly increase send rate to probe at about the right level
Fast retransmit / recovery	Resend lost packet after 3 duplicate ACKs; send new packet for each new ACK	Recover from a lost packet without stopping ACK clock

TCP Congestion Control

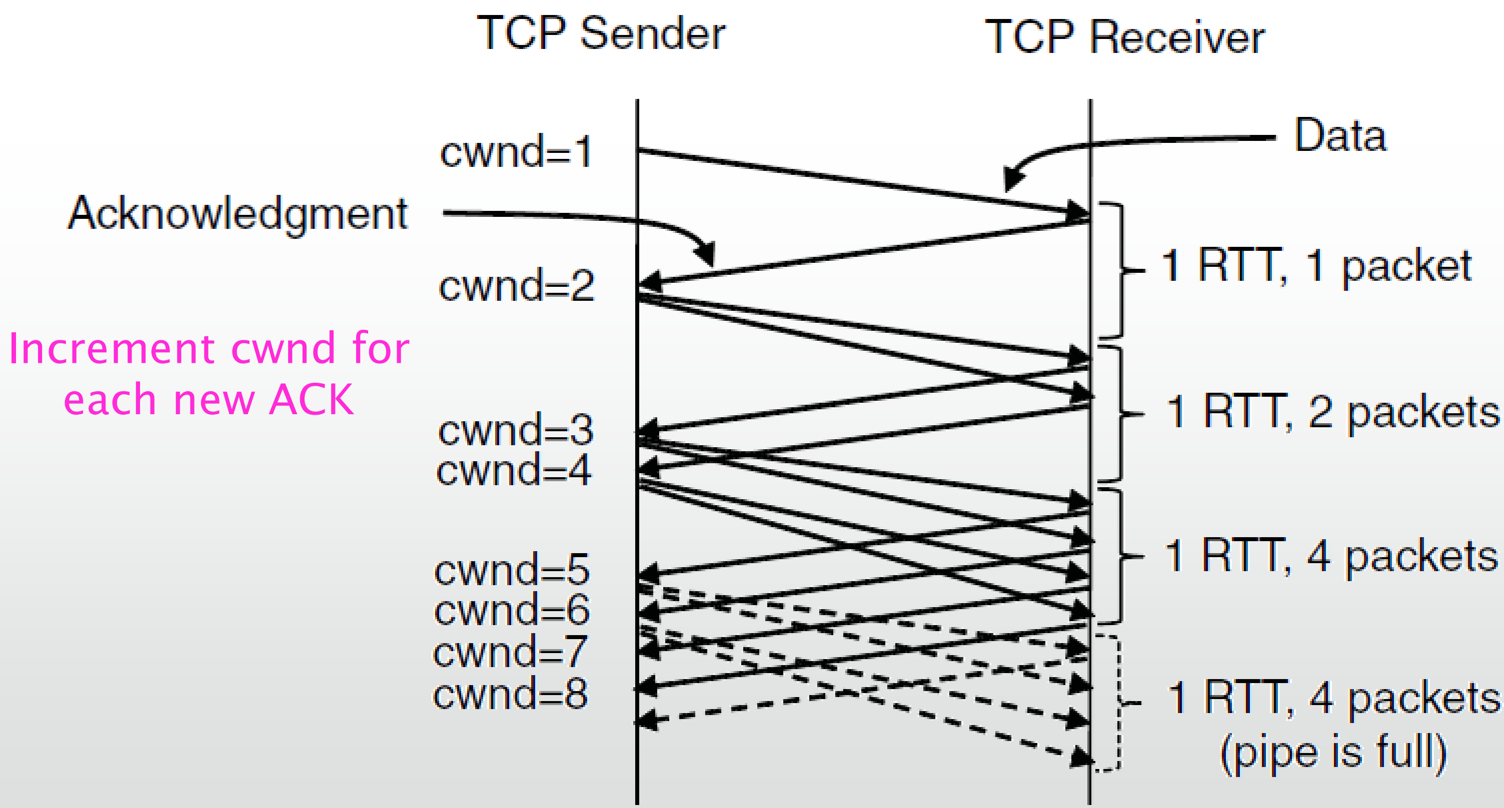
- Congestion window controls the sending rate
  - Rate is cwnd / RTT; window can stop sender quickly
  - ACK clock (regular receipt of ACKs) paces traffic and smoothes out sender bursts



ACKs pace new segments into the network and smooth bursts

TCP Congestion Control

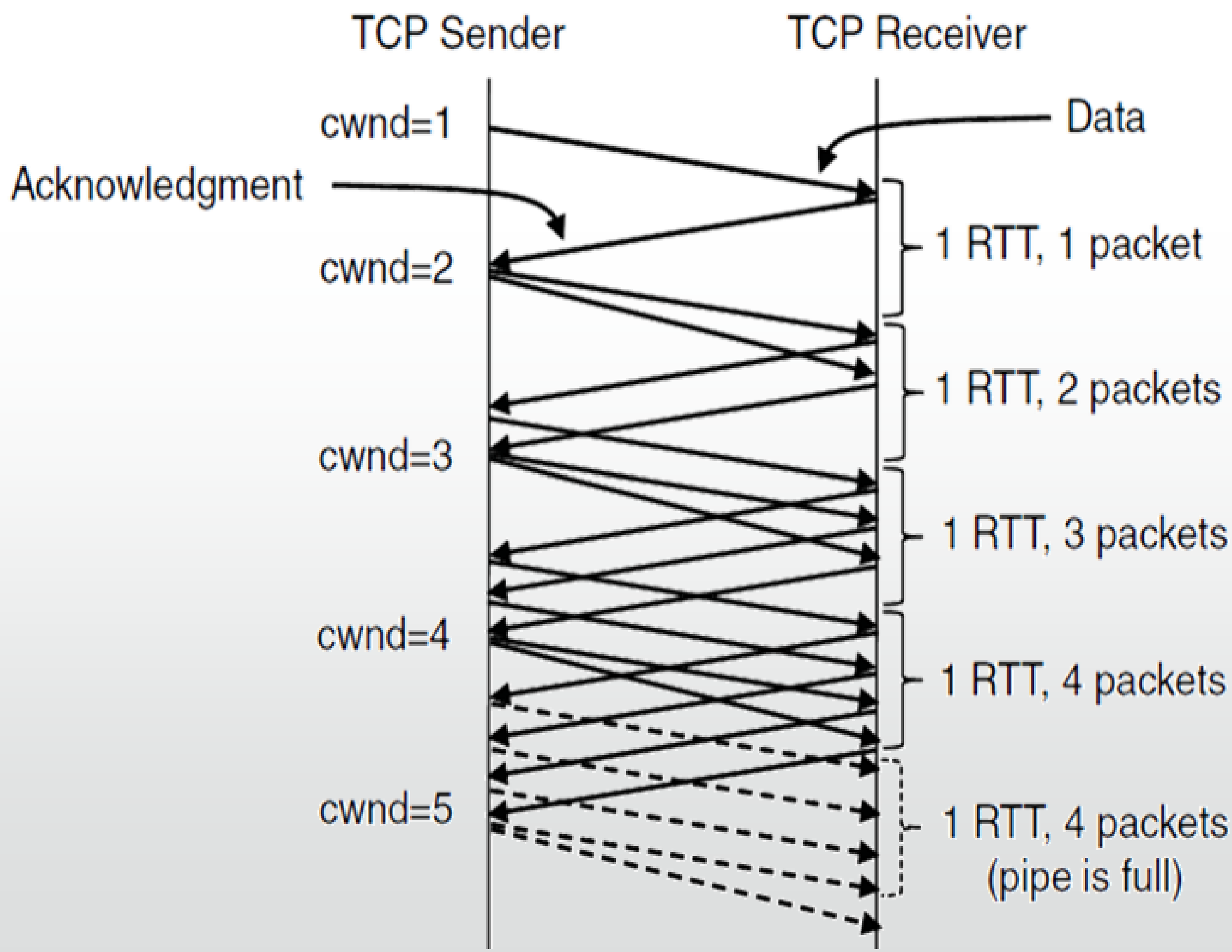
- Slow start grows congestion window exponentially
  - Doubles every RTT while keeping ACK clock going



TCP Congestion Control

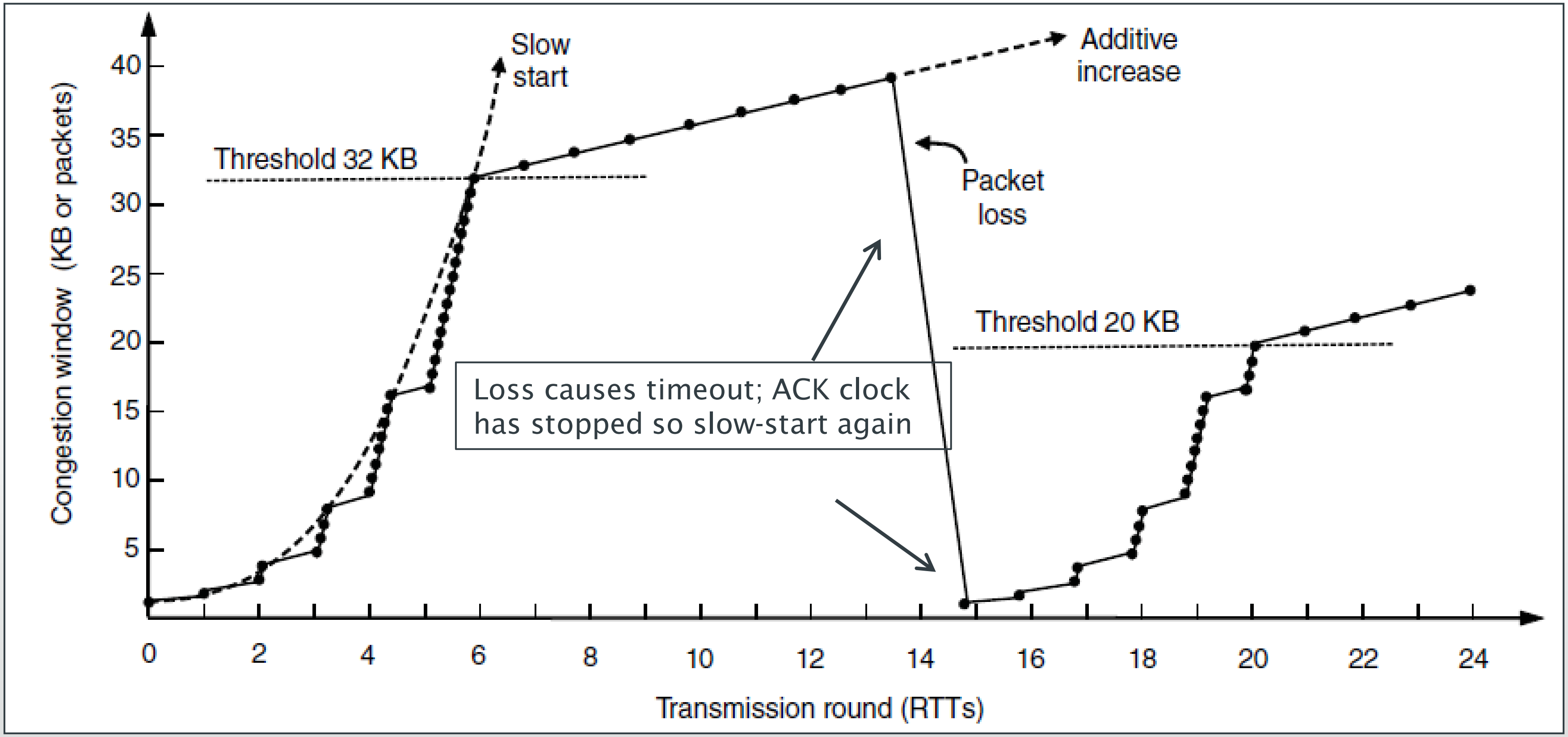
UNIVERSITY OF Southampton
School of Electronics and Computer Science

- Additive increase grows cwnd slowly
  - Adds 1 every RTT
  - Keeps ACK clock



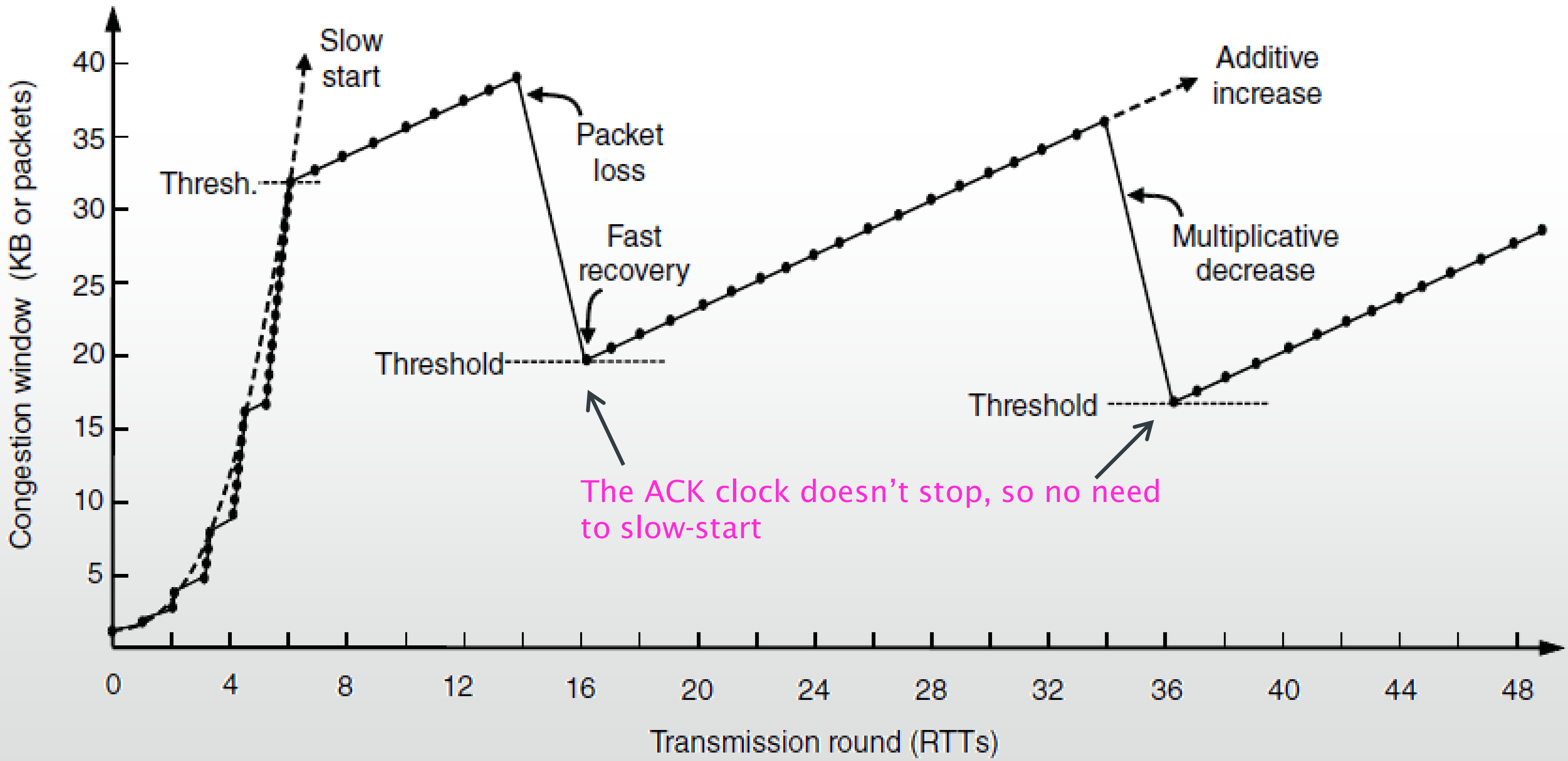
TCP Congestion Control

- Slow start followed by additive increase (TCP Tahoe)
  - Threshold is half of previous loss cwnd



TCP Congestion Control

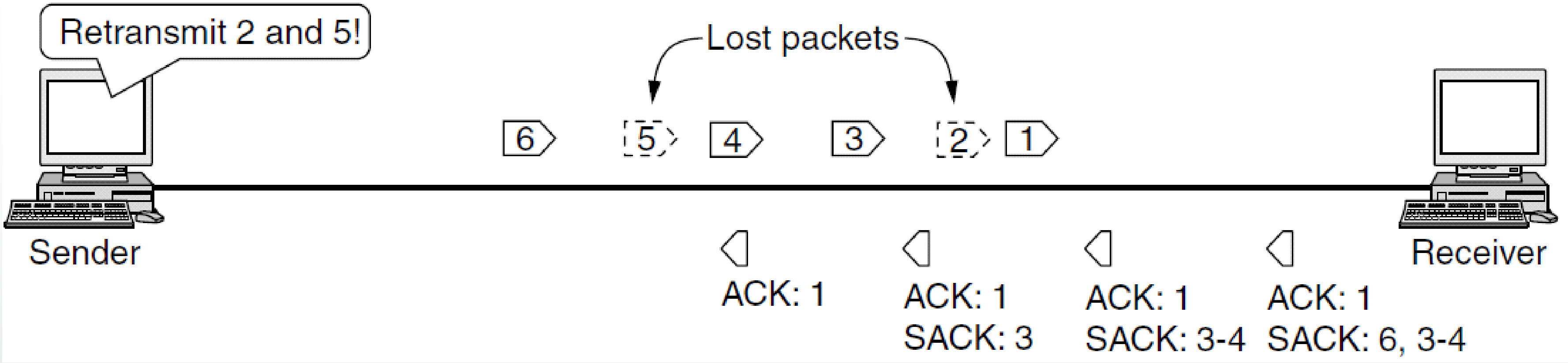
- With fast recovery, we get the classic sawtooth (TCP Reno)
  - Retransmit lost packet after 3 duplicate ACKs
  - New packet for each dup. ACK until loss is repaired



UNIVERSITY OF Southampton
School of Electronics
and Computer Science

TCP Congestion Control

- SACK (Selective ACKs) extend ACKs with a vector to describe received segments and hence losses
  - Allows for more accurate retransmissions / recovery



No way for us to know that 2 and 5 were lost with only ACKs

Summary

- Regulating the Sending Rate
- TCP Connection State
- TCP Sliding Window
- TCP Congestion Control