TCP Flow Controls

Matthew Roughan

Adelaide-Melbourne Grampians Workshop

1999

TCP/IP

Primary protocols used in the Internet

- IP (Internet Protocol)
- Transmission Control Protocol (TCP)
- TCP/IP refers to more than just TCP & IP
  - TCP is where flow controls are introduced

Why Use Flow Controls?

- October1986 Internet had its first congestion collapse
- Link LBL to UC Berkeley
  - 400 yards, 3 hops, 32 Kbps
  - throughput dropped to 40 bps
  - factor of ~1000 drop!
- 1988, Van Jacobson proposed TCP flow control

What are we interested in?

- Flow control is now mandatory on TCP connections
- Much is known about the qualitative performance of the Internet
  - the Internet works!
- Little is known about the quantitative performance of the TCP flow controls
  - mostly by simulation, few analytic results

Outline

- TCP/IP and the Internet
  - What is it?
  - How does it work?
- TCP flow controls
  - window flow controls
  - TCP implementations
- State of the art in performance analysis
  - $1/\sqrt{p}$ law

IETF

- Internet Engineering Task Force
  - standards organisation for Internet
  - publishes RFCs - Requests For Comment
    - standards track
    - experimental
    - informational
    - poetry/humour (RFC 1149: Standard for the transmission of IP datagrams on avian carriers)
  - TCP should obey RFC
    - no means of enforcement

RFCs of note

- RFC 791: Internet Protocol
- RFC 793: Transmission Control Protocol
- RFC 1180: A TCP/IP Tutorial
- RFC 2581: TCP Congestion Control
- RFC 2525: Known TCP Implementation Problems
- RFC 1323: TCP Extensions for High Performance

Other Key references

- W. Stevens, “TCP/IP Illustrated”, Vol. 1-3
  - Addison-Wesley, 1994
- Vern Paxson, “Measurements and Analysis of End-to-End Internet Dynamics”
  - PhD Thesis
- Van Jacobson, “Congestion Avoidance and Control”
  - SIGCOMM’88

Internet Protocol (IP)

- packet switched
- unreliable (best effort)
- heterogeneous
- robust
- intelligence is in terminals, not in network

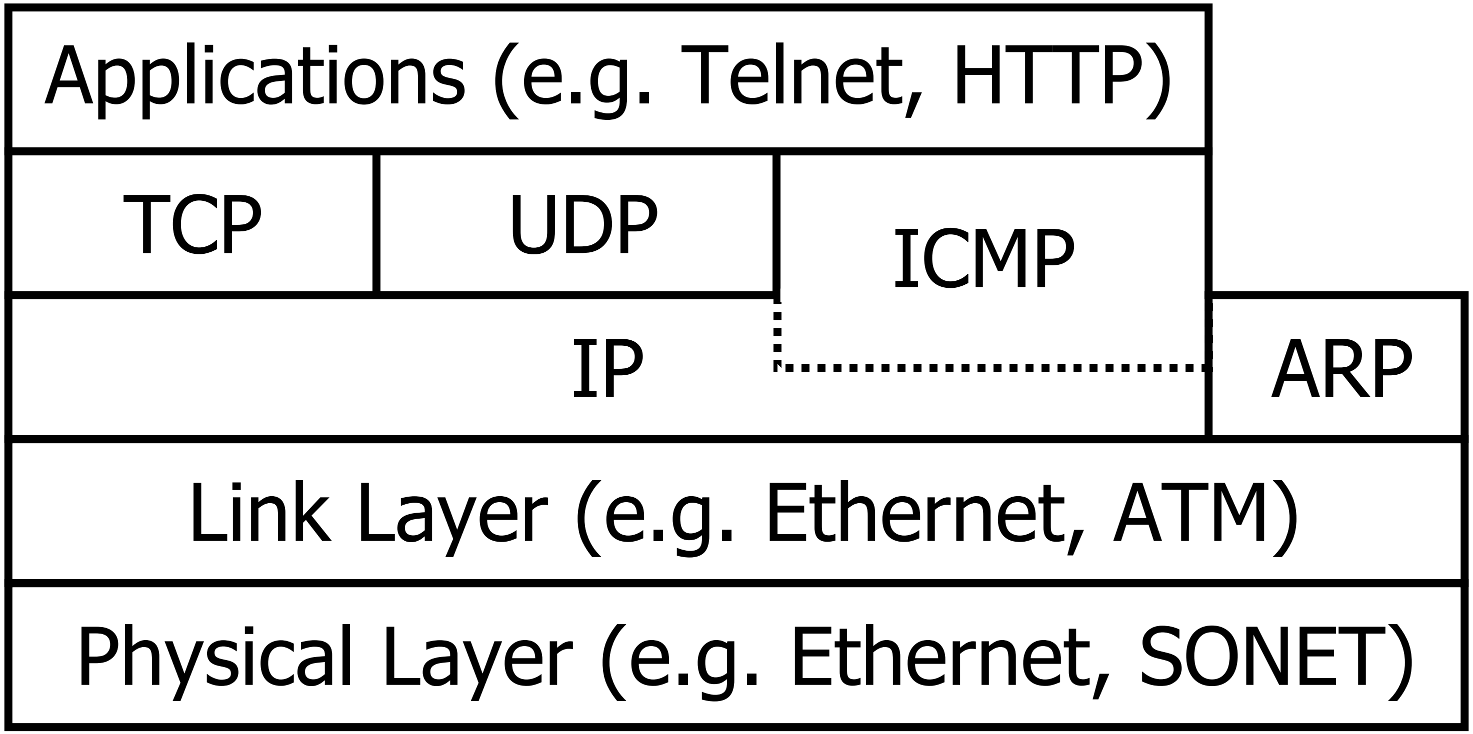
Aims of TCP

- TCP seeks to deliver a byte stream
  - from end-to-end, in order, reliably
  - allowing multiplexing
  - use bandwidth efficiently
- TCP achieves reliability using ACKs
- Robustness Principle

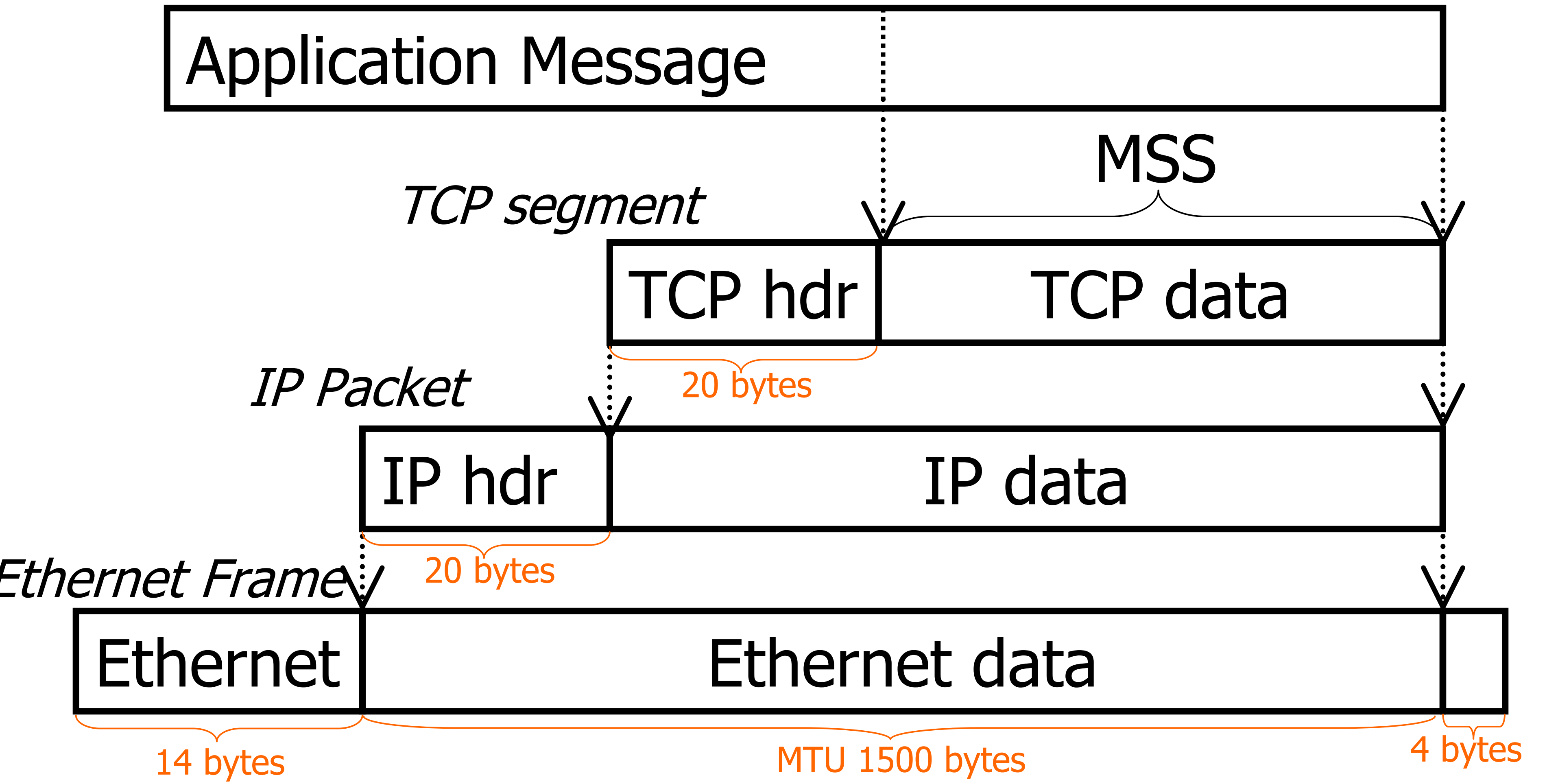
  be conservative in what you do,

  be liberal in what you accept from others

TCP/IP Protocol Stack



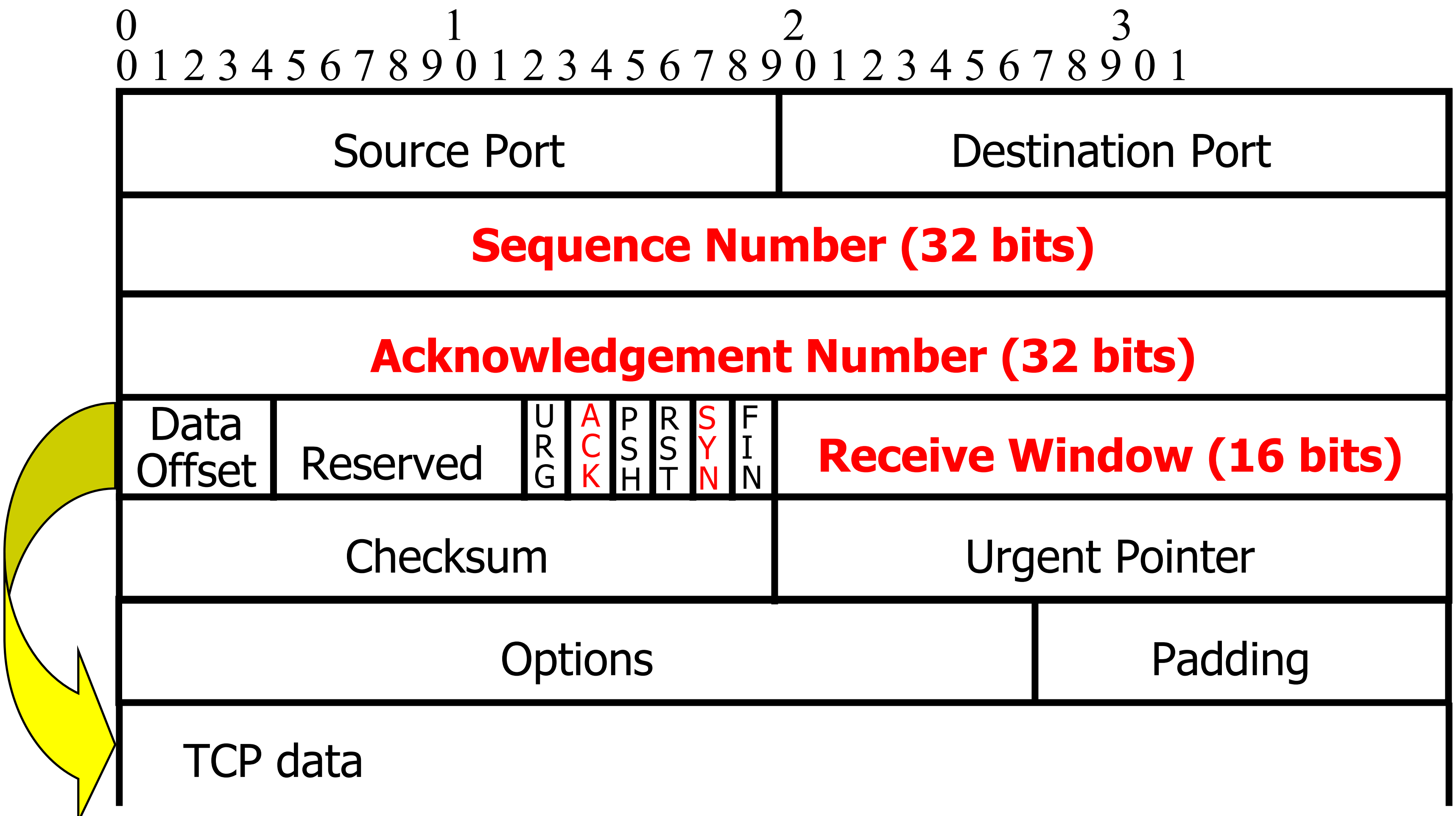
Packet Terminology



IP Header Format

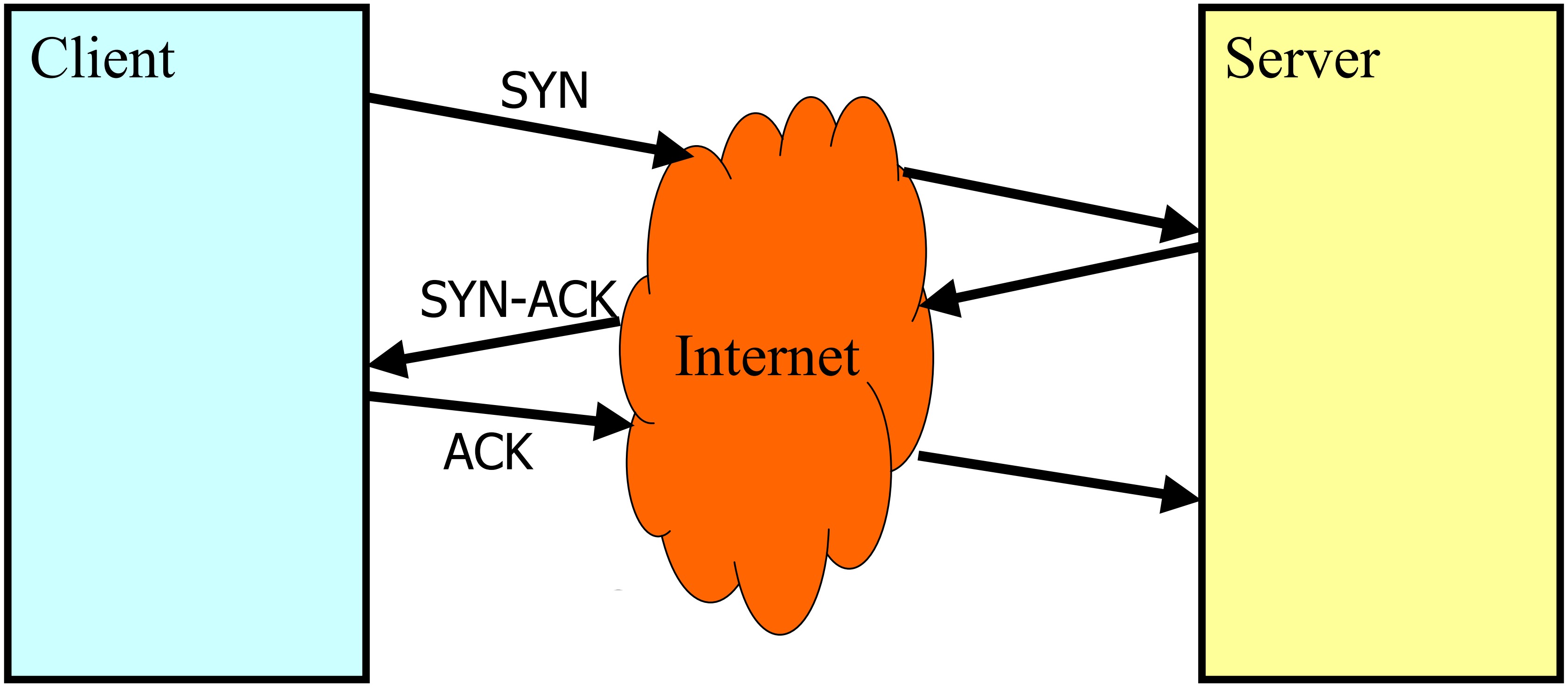
<table>
<tr><td colspan="10">0</td><td colspan="10">1</td><td colspan="10">2</td><td colspan="2">3</td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>0</td><td>1</td></tr>
<tr><td colspan="4">Vers(4)</td><td colspan="4">H len</td><td colspan="8">Type of Service</td><td colspan="16">Total Length (16 bits)</td></tr>
<tr><td colspan="16">Identification</td><td colspan="3">Flags</td><td colspan="13">Fragment Offset</td></tr>
<tr><td colspan="8">Time to Live</td><td colspan="8">Protocol
(TCP=6)</td><td colspan="16">Header Checksum</td></tr>
<tr><td colspan="32">Source IP Address</td></tr>
<tr><td colspan="32">Destination IP Address</td></tr>
<tr><td colspan="22">Options</td><td colspan="10">Padding</td></tr>
<tr><td colspan="32">IP data</td></tr>
</table>

TCP Header Format



How TCP works

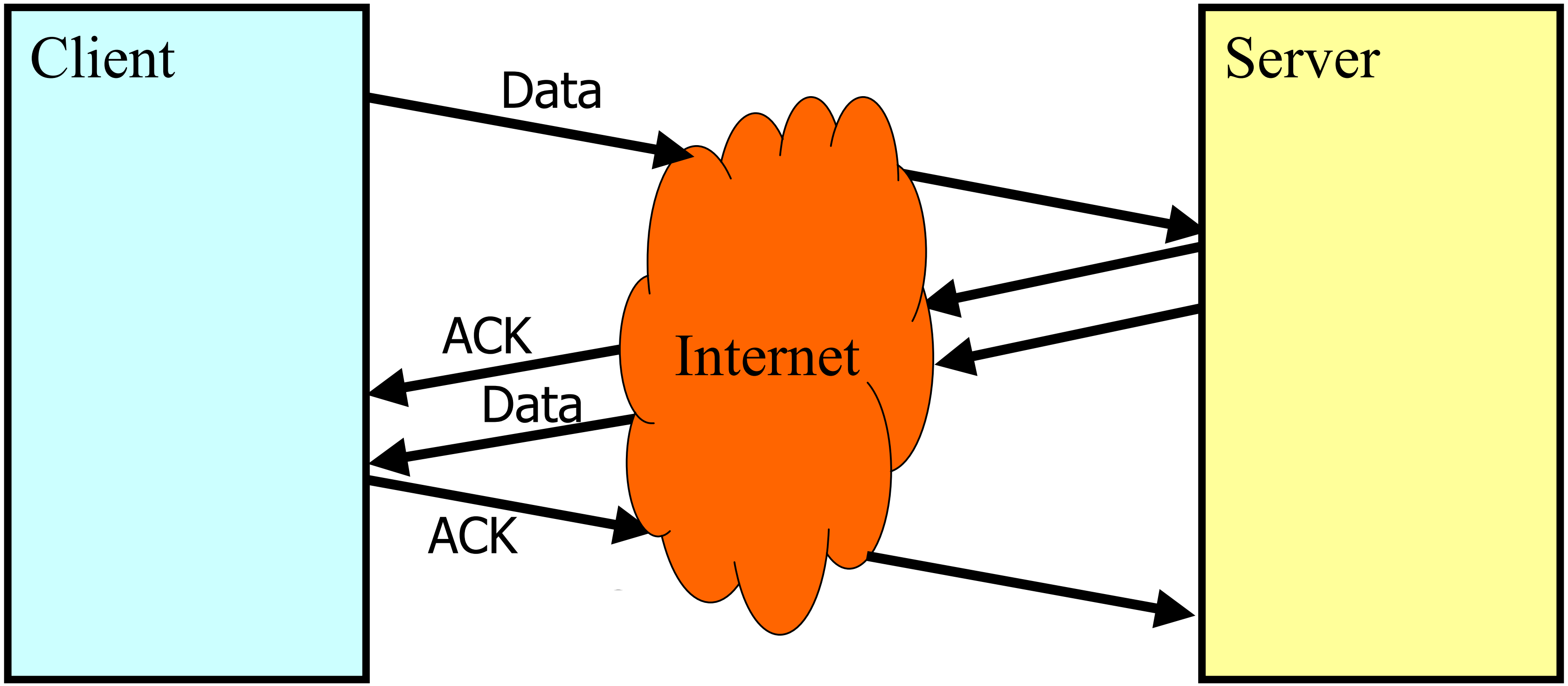
Connection



SYN-ACK Handshake established route MTU

How TCP works

Reliable Data Transport



ACKs ensure reliability
with retransmission of unacknowledged data

TCP example client/server

Client

```
struct sockaddr_in
   servaddr;
s = socket(flags);
connect(s,
   &servaddr,
   sizeof(servaddr));
```

`servaddr` is a structure which contains the IP address and TCP port number of the server

Server

```
l = socket(flags);
bind(l, &servaddr,
   sizeof(servaddr));
listen(l, LISTENQ);

for ( ; ; ) {
   c = accept(l,
      &cliaddr,
      &clilen);
}
```

TCP versions

- TCP is not perfectly homogenous (200+)

4.2 BSD first widely available release of TCP/IP (1983)

4.3 BSD (1986)

4.3 BSD **Tahoe** (1988)

SunOS 4.1.3,4.1.4

4.3 BSD **Reno** (1990)

IRIX

HP/UX

Digital OSF

NewReno (1999)

?

Windows 95

Windows NT

Solaris 2.3,2.4

Linux 1.0

Vegas

Window Flow Controls

- Limit the number of packets in the network to be less than some window W
- offered load = $\dfrac{W \times MSS}{RTT}$

If W is too small then throughput « bandwidth

If W is too big then load > bandwidth

=> congestion occurs

Effect of Congestion

- congestion causes packet loss
- results in retransmission
- reduces data throughput
- in extremes it can cause a collapse which persists much longer than the original overload

Congestion Control

- IP networks are heterogeneous
  - bandwidth ranges from 1200 bps to 10 Gbps
  - network delays range from < 1ms to ~ 1s
- TCP seeks to use BW
  - with high utilisation
  - without congestion
- Window Flow Control
  - Must choose the window size W correctly

TCP Window Flow Controls

- TCP separates receiver congestion from network congestion, and uses window flow controls for each
  - rwnd: receiver window
  - cwnd: congestion window
- TCP must not send data with a higher sequence number than the sum of the highest acknowledged sequence number and min(cwnd, rwnd)

TCP Receiver Flow Control

- prevent receiver from becoming overloaded
- receiver advertises a window rwnd with each acknowledgement
- Window
  - closed (by sender) when data is sent and ack'd
  - opened (by receiver) when data is read
- The size of this window can be *the* performance limit (e.g. on a LAN)
  - sensible default ~16kB

TCP Congestion Control

- Has four parts
  - Slow Start
  - Congestion Avoidance
  - Fast Recovery/Fast Retransmit
- ssthresh: slow start threshold determines whether to use slow start or congestion avoidance
- Assume packet losses are caused by congestion

Slow Start

- Slow start is used if cwnd < ssthresh
- Slow start named because it starts with the congestion window cwnd = 1
- On each successful ACK increase cwnd

$$cwnd \leftarrow cnwd + MSS$$

- The effect is exponential growth of cwnd

  each RTT: $cwnd \leftarrow 2 \times cnwd$

Congestion Avoidance

- Congestion Avoidance is used if

  $cwnd > ssthresh$

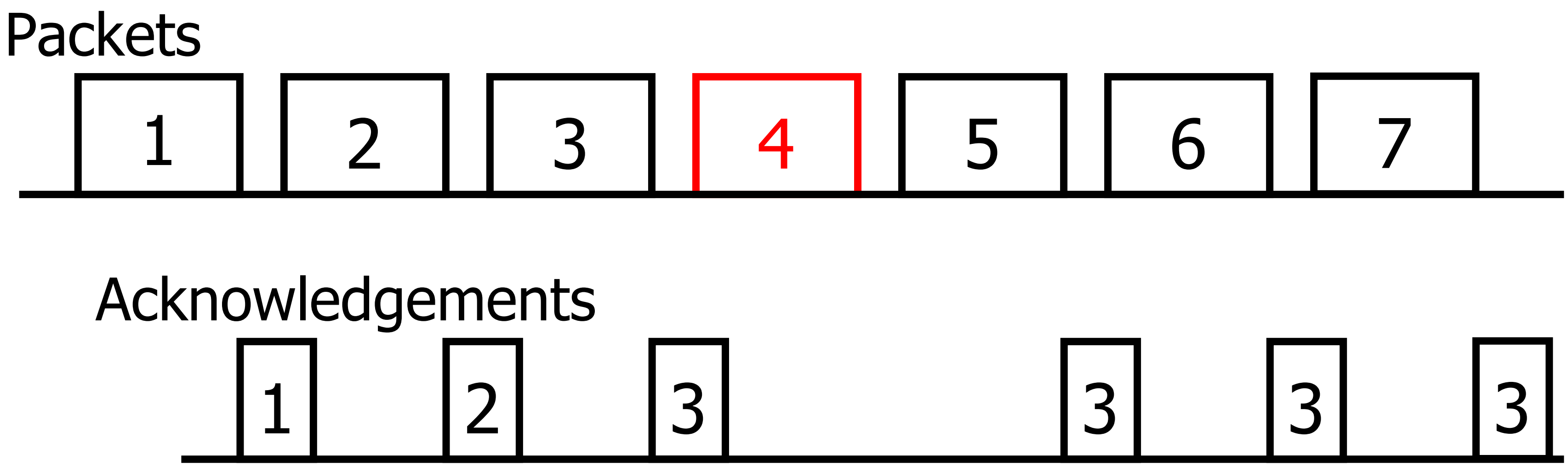
- On each successful ACK increase cwnd

  $cwnd \leftarrow cwnd + MSS^2/cwnd$

- The effect is <u>linear growth</u> of cwnd each RTT: $cwnd \leftarrow cwnd + MSS$

Packet Losses

- Packet losses may be detected by
  - Retransmission timeouts (RTO timer)
  - Duplicate Acknowledgements (at least 3)



Fast Recovery/Fast Retrans.

- When a packet loss is detected
  ssthresh $\leftarrow$ max(flightsize/2, 2xMSS)
- packet loss detected by a timeout go into Slow Start (cwnd = 1)
- packet loss detected by Dup ACKs
  - Fast Recovery/Fast Retransmission

  cwnd $\leftarrow$ cwnd/2

Implementation Dependence

- ssthresh initialisation (not standardised)
  - Reno $ssthresh_{init} = \infty$
  - Solaris $ssthresh_{init} = 8$
  - Linux $ssthresh_{init} = 1$
- algorithm for incrementing cwnd in CA
- Tahoe went into slow start after Dup.ACKs
  - no Fast Recovery (cwnd = 1)
- 1990 Reno had CA window increase
  - $\Delta W = MSS^2/cwnd + MSS/8$
- Inspect route cache for history

Bugs

- BSDI incorrectly initialised cwnd to $2^{30}-2^{14}$
- HP/UX doesn’t clear Dup.ACK counter on timeout
- Linux 1.0 no FR/FR (more like Tahoe)
- Linux/Solaris retransmit behaviour was broken
  - retransmits every unACK’d packet

Bugs

- Windows 95
  - often when 2 packets are sent one is lost somewhere in the NIC, so that only 1 is sent. The second is later sent by retransmission.
- Windows NT
  - no fast retransmit

Timers

- An accurate RTT measure is required to judge timeouts
- We can measure RTT by measuring the time to receive a packets ACK
- Use a smoothed RTT, S_{RTT} and the smoothed mean deviation D_{RTT}

$$RTO = S_{RTT} + 4\ D_{RTT}$$

Implementation dependence

- The measurement of RTT

  $$S_{RTT} = S_{RTT} + g\,(M_{RTT} - S_{RTT})$$
  $$D_{RTT} = D_{RTT} + h\,(\,|M_{RTT} - S_{RTT}| - D_{RTT})$$

- Need to minimize processing requirements
  - Only 1 counter (regardless of how many packets are extant)
  - Counter granularity is typically 500 ms
- Measurement equations have gain par.s

Implementation Dependence

- Retransmission Timeout

  $RTO = \beta \, S_{RTT}$
- Initial RTO (should be > 3 s)
- measurement of RTT of retransmitted packets
  - from first transmission
  - from final retransmission
  - ignore RTT for retransmitted packets (Karn)

Timers on a packet loss

- If a timeout occurs, double the RTO and retransmit the lost packet
  - results in exponential back-off
  - recalculate S_{RTT} only when a packet gets through

- RTT is lost if several packets are lost

Delayed Acknowledgements

- ACKs may be delayed to ‘piggy-back’ on returning data packets (by no more than 500ms, typically 200ms)
- If multiple packets arrive near to each other, a single ACK can be used to acknowledge up to 2 packets
- Slow Start and Congestion Avoidance increment cwnd per ACK, *not* per ACK’d packet

Typical networks

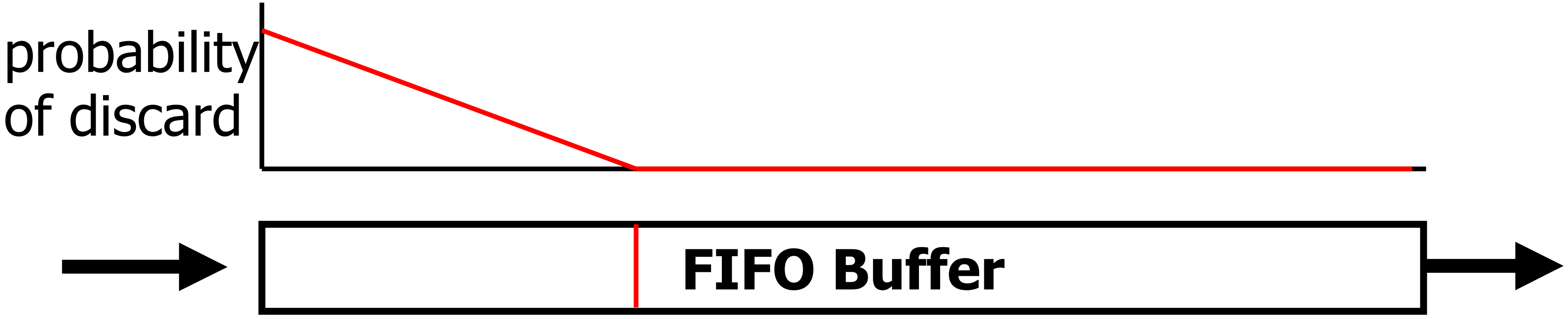
Network	Bandwidth	Delay	BWxdelay
10baseT Ethernet	10 Mbps	3 ms	3,750 B
T1 (satellite)	1.544 Mbps	500 ms	96,500 B
GB (transcontinental)	1 Gbps	60 ms	7,500,500 B

TCP Options

- Standard TCP performance is limited
  - max window size (2^{16}-1 = 65,535 bytes)
  - max sequence numbers (2^{32}-1 $\cong$ 4GB=32 Gb)
- Options for improved performance
  - Window scaling (RFC 1323)
  - Timestamps (RFC 1323)
  - Selective ACKs (RFC 2018)
  - larger initial window (RFC 2414, 2415, 2416)

Network management

- Explicit Congestion Notification (ECN)
  - explicit notification of congestion (RFC 2481)
- Random Early Detection (RED)
  - prevent burst of losses when buffers overflow
  - randomly discard some packets (RFC 2309)



Performance Analysis

- Typical Assumptions
  - Greedy sources
    - source always has data to send
  - Independent losses
    - packets are lost with probability p, independently
  - Examine equilibrium behaviour of bulk transport

Rough Calculation

$$w_{n+1} = \begin{cases} w_n/2, & \text{with probability } p \\ w_n + 1/w_n\ , & \text{with probability } 1\text{-}p \end{cases}$$

$$\begin{aligned} \overline{w} &= p\overline{w}/2 + (1\text{-}p)(\overline{w} + 1/\overline{w}) \\ \overline{w}\ p/2 &= (1\text{-}p)/\overline{w} \\ \overline{w}^2 &= 2(1\text{-}p)/p \end{aligned}$$

p«1 $\qquad \overline{w} \approx \sqrt{\dfrac{2}{p}}$

Refinement

Padhye, Firoin, Towsley and Kurose
SIGCOMM'98

- Treat as a Renewal Reward Process Take into account delayed ACKs

$$B(p) = \frac{MSS}{RTT}\sqrt{\frac{3}{2bp}}$$

- Include timeouts as well as Dup.ACKs

$$B(p) = \frac{MSS}{RTT\sqrt{\frac{2bp}{3}} + T_0 \min\left(1, 3\sqrt{\frac{3bp}{2}}\right) p\left(1 + 32p^2\right)}$$

- Include Receive window

$$B(p) \cong \min\left(W_r \frac{MSS}{RTT}, B(p)\right)$$

Possible future work

- more realistic loss model
- finite sources (not greedy)
- investigation of interaction
  - people have noticed synchronisation
  - chaotic behaviour
  - LRD arising from retransmissions