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?

- October 1986 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

Applications (e.g. Telnet, HTTP)

TCP  |  UDP  |  ICMP

IP

Link Layer (e.g. Ethernet, ATM)

Physical Layer (e.g. Ethernet, SONET)

ARP
Packet Terminology

Application Message

TCP segment

TCP header

TCP data

MSS

IP Packet

IP Header

IP Data

20 bytes

Ethernet Frame

Ethernet

Ethernet Data

20 bytes

14 bytes

MTU 1500 bytes

4 bytes
### IP Header Format

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vers(4)</td>
<td>Version number</td>
</tr>
<tr>
<td>H len</td>
<td>Header length</td>
</tr>
<tr>
<td>Type of Service</td>
<td>Type of service</td>
</tr>
<tr>
<td>Total Length</td>
<td>Total length</td>
</tr>
<tr>
<td>Identification</td>
<td>Identification</td>
</tr>
<tr>
<td>Flags</td>
<td>Flags</td>
</tr>
<tr>
<td>Fragment Offset</td>
<td>Fragment offset</td>
</tr>
<tr>
<td>Time to Live</td>
<td>Time to live</td>
</tr>
<tr>
<td>Protocol (TCP=6)</td>
<td>Protocol (TCP=6)</td>
</tr>
<tr>
<td>Header Checksum</td>
<td>Header checksum</td>
</tr>
<tr>
<td>Source IP Address</td>
<td>Source IP Address</td>
</tr>
<tr>
<td>Destination IP Address</td>
<td>Destination IP Address</td>
</tr>
<tr>
<td>Options</td>
<td>Options</td>
</tr>
<tr>
<td>Padding</td>
<td>Padding</td>
</tr>
<tr>
<td>IP data</td>
<td>IP data</td>
</tr>
</tbody>
</table>
## TCP Header Format

<table>
<thead>
<tr>
<th>Field</th>
<th>Bits</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source Port</td>
<td>16</td>
<td>Source port</td>
</tr>
<tr>
<td>Destination Port</td>
<td>16</td>
<td>Destination port</td>
</tr>
<tr>
<td>Sequence Number (32 bits)</td>
<td>32</td>
<td>Sequence number</td>
</tr>
<tr>
<td>Acknowledgement Number (32 bits)</td>
<td>32</td>
<td>Acknowledgement number</td>
</tr>
<tr>
<td>Data Offset</td>
<td>4</td>
<td>Data offset</td>
</tr>
<tr>
<td>Reserved</td>
<td>4</td>
<td>Reserved</td>
</tr>
<tr>
<td>URG</td>
<td>1</td>
<td>Urgent</td>
</tr>
<tr>
<td>ACK</td>
<td>1</td>
<td>Acknowledgement</td>
</tr>
<tr>
<td>PSH</td>
<td>1</td>
<td>Push</td>
</tr>
<tr>
<td>RST</td>
<td>1</td>
<td>Reset</td>
</tr>
<tr>
<td>SYN</td>
<td>1</td>
<td>Synch</td>
</tr>
<tr>
<td>FIN</td>
<td>1</td>
<td>Finish</td>
</tr>
<tr>
<td>Receive Window (16 bits)</td>
<td>16</td>
<td>Receive window</td>
</tr>
<tr>
<td>Checksum</td>
<td>16</td>
<td>Checksum</td>
</tr>
<tr>
<td>Urgent Pointer</td>
<td>16</td>
<td>Urgent pointer</td>
</tr>
<tr>
<td>Options</td>
<td>16</td>
<td>Options</td>
</tr>
<tr>
<td>Padding</td>
<td>16</td>
<td>Padding</td>
</tr>
<tr>
<td>TCP data</td>
<td></td>
<td>TCP data</td>
</tr>
</tbody>
</table>
How TCP works

Connection

Client

SYN

Internet

SYN-ACK

ACK

Server

SYN-ACK Handshake established route MTU
How TCP works

Reliable Data Transport

Client

Data

ACK

Data

ACK

Internet

Server

ACKs ensure reliability with retransmission of unacknowledged data
TCP example client/server

Client

```c
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

```c
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)

4.3 BSD Reno (1990)

IRIX

Digital OSF

HP/UX

NewReno (1999)

Windows 95

Windows NT

SunOS 4.1.3, 4.1.4

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 = $\frac{W \times MSS}{RTT}$

  If $W$ is too small then throughput $\ll$ bandwidth
  
  If $W$ is too big then load $>\$ bandwidth
  
  $\Rightarrow$ 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

- \texttt{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 $cwnd$
  $$cwnd \leftarrow cwnd + \frac{MSS^2}{cwnd}$$
- The effect is linear growth 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)

Packets

1 2 3 4 5 6 7

Acknowledgements

1 2 3 3 3 3 3
Fast Recovery/Fast Retrans.

- When a packet loss is detected
  \[ \text{ssthresh} \leftarrow \max(\text{flightsize}/2, \ 2\times\text{MSS}) \]

- packet loss detected by a timeout go into Slow Start \((\text{cwnd} = 1)\)

- packet loss detected by Dup ACKs
  - Fast Recovery/Fast Retransmission
    \[ \text{cwnd} \leftarrow \text{cwnd}/2 \]
Implementation Dependence

- ssthresh initialisation (not standardised)
  - Reno \( ssthresh_{\text{init}} = \infty \)
  - Solaris \( ssthresh_{\text{init}} = 8 \)
  - Linux \( ssthresh_{\text{init}} = 1 \)

- 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
  - \( \Delta W = \frac{MSS^2}{\text{cwnd}} + \frac{MSS}{8} \)
- 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 packet's ACK
- Use a smoothed RTT, $S_{RTT}$ and the smoothed mean deviation $D_{RTT}$

\[ RTO = S_{RTT} + 4 \ D_{RTT} \]
The measurement of RTT

\[ S_{\text{RTT}} = S_{\text{RTT}} + g (M_{\text{RTT}} - S_{\text{RTT}}) \]
\[ D_{\text{RTT}} = D_{\text{RTT}} + h (|M_{\text{RTT}} - S_{\text{RTT}}| - D_{\text{RTT}}) \]

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 parameters
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

<table>
<thead>
<tr>
<th>Network</th>
<th>Bandwidth</th>
<th>Delay</th>
<th>BWxdelay</th>
</tr>
</thead>
<tbody>
<tr>
<td>10baseT Ethernet</td>
<td>10 Mbps</td>
<td>3 ms</td>
<td>3,750 B</td>
</tr>
<tr>
<td>T1 (satellite)</td>
<td>1.544 Mbps</td>
<td>500 ms</td>
<td>96,500 B</td>
</tr>
<tr>
<td>GB (transcontinental)</td>
<td>1 Gbps</td>
<td>60 ms</td>
<td>7,500,500 B</td>
</tr>
</tbody>
</table>
TCP Options

- Standard TCP performance is limited
  - max window size ($2^{16}-1 = 65,535$ bytes)
  - max sequence numbers ($2^{32}-1 ≈ 4\text{GB}=32\text{ Gb}$)

- 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

\[
\begin{aligned}
\quad w_{n+1} &= \begin{cases} 
  w_n/2, & \text{with probability } p \\
  w_n + 1/w_n, & \text{with probability } 1-p 
\end{cases} \\
\quad \bar{w} &= p\bar{w}/2 + (1-p)(\bar{w} + 1/\bar{w}) \\
\quad \bar{w} p/2 &= (1-p)/\bar{w} \\
\quad \bar{w}^2 &= 2(1-p)/p \\
\end{aligned}
\]

\[ p \ll 1 \quad \bar{w} \approx \sqrt{\frac{2}{p}} \]
Refinement

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

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

Include Receive window

\[ B(p) \approx \min\left(W_r, \frac{MSS}{RTT}, B(p)\right) \]
Possible future work

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