Network Research and Linux at the Hamilton Institute, NUIM.

David Malone

4 November 2006
What has TCP ever done for Us?

- Demuxes applications (using port numbers).
- Makes sure lost data is retransmitted.
- Delivers data to application in order.
- Engages in congestion control.
- Allows OOB data.
- Some weird stuff with TCP options.
Standard Picture of TCP

Connector
SYN
SYN+ACK
ACK
ACKs
Data
FIN
High Port (Usually)

Listener

Connector
SYN
RST

No Listener
Known Port (Usually)

Known Port (Usually)
High Port (Usually)
Other Views of TCP

Programmer’s View

Network View

Packets Out (at link rate)
Congestion Control

- TCP controls number of packets in network.
- Packets are acknowledged, so flow of ACKs.
- Receiver advertises window to avoid overflow.
- Congestion window tries to adapt to network.
- Slow start to roughly find capacity.
- Congestion avoidance gradually adapts.
The Congestion Window

- Additive increase, multiplicative decrease (AIMD).
- To fill link need to reach $BW \times \tau$.
- Backoff by $1/2$, implies buffer is $BW \times \tau$.
- Fairness, responsiveness, stability, …
TCP On Linux

- Network stack buffers in-flight data.
- Socket buffer must be $BW \times \tau$.
- `/proc/net/core/{r,w}mem_max` → sockbuf sizes.
- `/proc/net/ipv4/tcp_{r,w,}mem` → min/def/max tcp window.
- Trade off — memory is wired, so valuable.
- Defaults have recently been increased.
Research Work

- Packet loss not caused by congestion.
- Filling big $BW \times \tau$ product packet at a time.
- Bad for long-distance high-bandwidth links.
- Various solutions in pipeline (BIC, Scalable, High-Speed, FAST, H-TCP).
- Pluggable congestion control in Linux (behind TCP_CONG_ADVANCED).
- `/proc/sys/net/ipv4/tcp_congestion_control`
- Working on other congestion detection techniques.
Throughput

Throughput (KBytes)

Throughput Flow 1

Throughput Flow 2

Throughput (KBytes)

time (sec)

Throughput Flow 1

Throughput Flow 2

time (sec)
Cwnd

Cwnd flow 1
Cwnd flow 2

Cwnd (packets)

time (sec)
Practical Stuff

- Other issues at play, such as implementation quality.
- For example ACK processing and queueing problems.
- Testing is important: land speed records.
- Project with OSDL to build validation suite.
Before

Segments (cwnd, ssthresh)

Bytes (sequence)

Distribution

time (seconds)

microsecond

snd_cwnd
snd_rsts
throttle

slow path

0.1 1 10 100 1000 10000

0.1 1 10 100 1000 10000
After
802.11(e) MAC

Summary

• After TX choose $\text{rand}(0, CW - 1)$.
• Wait until medium idle for DIFS($50\mu s$),
• While idle count down in slots ($20\mu s$).
• TX when counter gets to 0, ACK after SIFS ($10\mu s$).
• If ACK then $CW = CW_{\text{min}}$ else $CW* = 2$.

Ideally produces even distribution of packet TX.
In 11e have multiple queues. Each has own $CW_{\text{min}}$, DIFS(aka AIFS) and can have TXOP.
**Testbed setup**

Multiple STA (Linux) connected to AP (Linux hostap).

<table>
<thead>
<tr>
<th>Hardware</th>
<th>model</th>
</tr>
</thead>
<tbody>
<tr>
<td>1× AP</td>
<td>Desktop PC</td>
</tr>
<tr>
<td>18× STA</td>
<td>Soekris</td>
</tr>
<tr>
<td>1× STA</td>
<td>Desktop PC</td>
</tr>
<tr>
<td>WLAN NIC</td>
<td>Atheros AR5212</td>
</tr>
</tbody>
</table>

External antenna, PCI interface, Madwifi driver with local patches for 11e parameter setting.
Measure relative performance of two saturated flows while varying TXOP, AIFS and CW$_{\text{min}}$. Compare to well-known models.
Before

12 TCP Uploads to AP for 180 sec

STA id

Throughput (kbits/sec)

6 Uploads and 6 Downloads for 180 sec

STA id

Throughput (kbits/sec)
After

12 TCP Uploads to AP for 180 sec

STA id

Throughput (kbits/sec)

6 Uploads and 6 Downloads

STA id

Throughput (kbits/sec)
Unprioritised Voice

Throughput

Delay

Throughput (kbps) vs. Number of competing stations

Delay (seconds x 10^{-6}) vs. Number of competing stations

Unprioritised Station
Ideal Throughput
90% Throughput

Delay bound for queue stability
Want to measure one-way MAC delay.

NTP slow and insufficiently accurate.

Simultaneously observable TX better, largish noise.
Delay Technique

- Transmission not complete until MAC ACK.
- Hardware supports interrupt after ACK.

1. Driver notes enqueue time.
2. Hardware contends until ACK received.
3. Hardware interrupts driver.
4. Driver notes completion time.

Packet transmitted
ACK received
Validation

Median delay (seconds x 10^-6) vs. Packet size (bytes, excluding headers)

Fit line: 11.0012Mbps x + 484.628us

Long preamble

Fit line: 11.0016Mbps x + 292.565us

Short preamble
AIFS Impact

Throughput

Delay

Throughput (kbps)

Number of competing stations

Unpriority Station
Competitors AIFS = 4
Competitors AIFS = 6
Ideal Throughput
90% Throughput

Average Delay (seconds x 10^-6)

Number of competing stations

Unpriority Station
Competitors AIFS = 4
Competitors AIFS = 6
Delay bound for queue stability

25