End-to-End Network Issues

Matt Mathis
mathis@psc.edu

20 June 2003
http://www.psc.edu/~mathis/papers/DoE200306/
What causes end-to-end bottlenecks?

- Background

- Technical problems
  - Tuning requires too much expertise
  - Congestion Control
  - Remote Direct Data Placement (RDDP)
  - Maximum Transmission Unit

- Epilogue
Imagine closing the wizard gap

- **Deployed:** 1 Gb/s
  - 1 second: 125 Megabytes
  - 1 minute: 7.5 Gigabytes
  - 1 hour: 450 Gigabytes
  - 1 day: 10.8 Terabytes

- **New:** 10 Gb/s (2002)
  - 1 second: 1.25 Gigabytes
  - 1 minute: 75 Gigabytes
  - 1 hour: 4.5 Terabytes
  - 1 day: 108 Terabytes

- This solves bulk application performance problems!
What causes of the wizard gap?

- Should we overhaul TCP/IP?
- Why?

But not single end-to-end applications

- Aggregate traffic (even steeped)
- Raw end-to-end Internet path capacity
- All computer, instrument, and network components

Moore's Law applies to:

For networks: a factor of 10 every 3 years
The spoiler

- TCP is easy to blame, but...

- I will show:
  - Nearly all problems are not due to TCP but rather how TCP is integrated into end systems and the Internet.

  - As a result, non-integrated experimental protocols tend to give false positive performance improvements.

  - ...until deployed at production scale, where they will suffer many of the existing bottlenecks.
Background: What is a protocol?

- A protocol is a packet format plus algorithms:
  - Required for correct operation (e.g. data retransmission)
  - Required for Internet stability (e.g. congestion control)
  - For efficiency (e.g. delayed ack)

- Many good algorithms are portable between protocols
  - Bad algorithms can be dangerous, even on a small scale

- Protocols that share algorithms share properties
  - Even if they differ in other areas

- My talk is really about algorithms, not protocols
Example: Fat TCP

"TCP Extensions Considered Harmful"
- RFC1263 (O’Malley, 1991)
- Defines a 64 bit TCP

Same properties as TCP except larger sequence space
- Does not need PAWS
- Does not need window scaling
- (An alternate to RFC1072/RFC1323)

But not wire compatible
- Therefore hard to deploy

Is it TCP?
Protocol changes - especially packet format changes - are extremely hard to deploy.

But we want to recapture the huge "intellectual investment" in the Internet by preserving as much as possible of the existing algorithms.
Background: What is the Internet?

- Shared infrastructure
  - Economics of the commons
  - Playing nice is required

- Traffic isolation changes the rules
  - e.g. switched optical networks

- But not as much as you might think
  - Congestion control is still needed to prevent brittleness
Technical Problems that contribute to the Wizard Gap
"Tuning" requires too much expertise

- Protocols hide the net from the applications
  - Provide uniform services to upper layers
  - Independent of the details of the link layers
  - This is the Internet "hourglass"

- This is good for the growth of the Internet
  - The hourglass decouples network and application deployment

- But all bugs have the same symptom: less than expected performance!

- Intrinsic property of an hourglass w/ recovery
  - Effecting all transport protocols, including TCP
Tuning is really debugging

- Six classes of bugs limit performance
  - Packet losses, corruption, congestion, lame HW
  - IP Routing, long round trip times
  - Packet reordering
  - Inappropriate buffer space
  - Inappropriate packet sizes
  - Inefficient applications
Tuning is painful debugging

- Any one problem can mask all other problems, confounding all but the best experts

- Akin to finding the weakest link of an invisible chain

- We need better diagnostic visibility

- Instrument the protocol to diagnose the network
Web100 Instrumentation

- About 120 TCP "test points" to diagnose network problems

- Prototype TCP MIB on the standards track: draft-ietf-tsvwg-tcp-mib-extension-03.txt

- When there is a problem, just ask TCP
False results

- The hourglass creates an intrinsic problem
  - By nature transport protocols hide bugs

- New protocols are tested with full debugging code
  - The experts debug everything along with the protocol

- Resulting performance improvements don’t hold in the field

- Laboratory results are overly optimistic real world
Congestion Control

\[
Rate = \frac{MSS}{RTT} \times \frac{0.7}{\sqrt{p}}
\]

[MSMO, July'97 CCR]

- MTU (MSS) and RTT are determined by the path
  - The loss rate, \( p \), is the primary control

- If data rate spans 3 orders of magnitude then loss rate needs to span 6 orders of magnitude

- Since finding residual packet loss in the production Internet can be very hard, it is often nearly impossible to get full data rates.
Anybody can design an aggressive protocol, with insufficient congestion control, that out performs others.

This can be a win when hand tuned for an isolated network.

However if there is no congestion control, it makes applications brittle in the presence of even minor problems, such as unexpected cross traffic.
<table>
<thead>
<tr>
<th>Change global congestion control?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Need to address economics of the commons</td>
</tr>
<tr>
<td>- Global fairness (not specific to TCP)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Floyd’s high speed TCP and Kelly’s Scalable TCP</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Preserve strict TCP fairness at small windows</td>
</tr>
<tr>
<td>- More aggressive at larger windows</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Caltech FAST and Katabi XCP</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Use delay sensing as primary control (new headers?)</td>
</tr>
<tr>
<td>- Must still respond to losses</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Net100 Work Around Daemon</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Explicit controls to alter congestion control</td>
</tr>
</tbody>
</table>
A better solution

- The Net100 Work Around Daemon
  - See www.net100.org

- Supports a wide range of congestion control tweaks
  - Settable AIMD parameters
  - Floyd HSTCP
  - Kelly Scalable TCP
  - Pseudo Vegas (hybrid user-mode implementation)
  - (Future) FAST and XCP

- Infrastructure to support experimental algorithms
  - Easily monitored and controlled
  - Can be arbitrarily abusive
Key point

- Congestion control algorithms are substantially portable between protocols

- Any good algorithm which can be deployed on the Internet can be ported into all standard protocols, including TCP

- All bad algorithms are risky in any environment
False Results

- Current Congestion Control practices are a matter of public policy to protect the commons, and not an issue of protocol design.
- Efforts to implement non-sharing protocols create exposures to the internet due to crossover usages.
- This Pandora’s box is already open.
Remote Direct Data Placement (RDDP)

- Also called Remote Direct Memory Access (RDMA)

- Many "back-end" protocols support RDDP and are zero copy by default
  - Fiber channel, Myrinet, Infiniband, Quadrix, etc

- Not supported by standard Internet protocols
  - require extra copies through system memory
Technical pieces

- Sending side is "just a matter of engineering"
  - Zero copy TCP, etc

- Receiving side requires application buffer addresses in the NIC
  - Look at control fields in every packet
  - Place headers and data directly into final buffers

- The application buffer addresses:
  - Can be carried end-to-end
  - Or rendezvous in the NIC
RDDP Implementation

- Easy to do in proprietary protocols

- Not too hard in block oriented protocols such as SCTP
  - possible next generation bulk transport protocol

- Harder in byte oriented protocols such as TCP

- It is generally hard to add as an after thought
False Results

- Many experimental NICs are not fully integrated into the OS

- Test software controls the NIC directly
  - OS bypass by default
  - Does not support all OS services
  - Not usable by non-experts

- A full production quality driver requires an extra copy
TCPs 32 bit Sequence Limit

- TCP sequence numbers "overflow"
  - every 4,294,967,296 bytes
  - This takes only 3.5 seconds at 10 Gbit/s

- Use TCP timestamps to extend sequence space
  - PAWS in RFC1312 is good for *100
  - Altered PAWS might solve the problem

- Other protocols have lower limits
  - SCTP - whoops!
  - Nobody is fretting about this
What about MTU?

- Maximum Transmission Unit or "packet size"
- Determines the TCP Maximum Segment Size (MSS)

- Predominant MTU is defined by Ethernet (1500)
  - More than 2 orders of magnitude too small at 10 Gb/s

- See http://www.psc.edu/~mathis/MTU
Revisit congestion control

\[ \text{Rate} = \frac{MSS}{RTT} \times \frac{0.7}{\sqrt{p}} \]

[MSMO, July'97 CCR]

- Coast-to-coast 100 Mb/s is reasonable:
  - 1 ppm losses to get 100 mb/s over 70 ms at 1500

- But 10 Gb/s (100 times faster) requires 10,000 times less loss
  - 0.1 ppb (i.e. 1 in 1e10) over the same path

- EE jargon: Noise immunity goes as the square of the window size in packets
Possible approaches

- Disable/evade congestion control
  - Start with UDP.....

- Change TCP congestion control
  - Floyd and others

- Rescale IP
  - Approximately constant window in packets
  - Approximately constant time packets
  - Approximately constant protocol dynamics
Apply Moore’s law to packet sizes

- Split *10 steps into *8 size and 20% shorter times

<table>
<thead>
<tr>
<th>Rate</th>
<th>MTU</th>
<th>Pkt time (Actual)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10 Mb/s</td>
<td>1.5kBytes</td>
<td>1200 uS (1982 - 1200uS)</td>
</tr>
<tr>
<td>100 Mb/s</td>
<td>12kBytes</td>
<td>960 uS (1995 - 120uS)</td>
</tr>
<tr>
<td>1 Gb/s</td>
<td>96 kBytes</td>
<td>768 uS (1998 - 12uS)</td>
</tr>
<tr>
<td>10 Gb/s</td>
<td>750 kBytes</td>
<td>600 uS (2002 - 1.2 uS)</td>
</tr>
<tr>
<td>100 Gb/s</td>
<td>6 MBytes</td>
<td>480 uS</td>
</tr>
<tr>
<td>1 Tb/s</td>
<td>50 MBytes</td>
<td>400 uS</td>
</tr>
</tbody>
</table>

- These are subject to change as we get better data
A different view

Which has less total overhead? 1 Terabyte of data:
- 1,000,000,000 1kB packets
- 1,000,000 1MB packets

The costs are in different layers!
- Small packets have 1000 times more software overhead
- Double HW costs for large packets
- (500 times less overhead per unit cost)

The LAN industry has optimized their part of the cost, at the expense of other parts of the stack
A serious legacy bug

- Path MTU Discovery (RFC1191) does not work well
  - It requires ICMP messages from routers
    - Many problems - outlined in RFC2923
  - When it fails, the symptoms are hung connections
- Partially disabled to prevent bad PR
  - 1500 byte default MTU
Impact of disabling pMTU discovery

- Chronic problems for tunneled protocols
  - PPPOE, VPNs, mobility, IPv6 migration, etc
  - Useful MTU is 1500 minus tunnel headers

- Furthermore nobody noticed that FDDI is faster than 100 Mb/s Ethernet

- Totally eliminated the demand for larger MTUs
  - because nobody could tell that they make a difference
Two intertwined subtasks

- Fix RFC 1191 Path MTU discovery
  - New algorithm under study
  - Already have four prototype implementations
  - IETF WG launch next month

- Create a market for larger MTUs
  - Reconfigure existing infrastructure
  - Collect hard data on the value of big packets
  - Identify cost effective LAN technologies for large MTUs
  - RFQ language at all levels
The new algorithm

- The basic idea
  - Start "small" (1kB?)
  - Probe with successively larger segments
  - Probes are dropped if too large
  - If a probe is delivered, raise the MTU for the connection

- Does not rely on messages from the net

- Solves tunneled protocol problems too!
  - Independent market push for deployment
Running code

10.0.0.4:65533 ==> 10.0.0.1:5001 (time sequence graph)
LAN vendor resistance

- The LAN vendors want tiny jumbograms (9k)

- Vendors are focused on the mass market
  - Current total costs are below $50 per port
  - We are at best only a few percent of the market

- Current HW is heavily optimized
  - Some of the current optimizations depend on small packets

- We need to explicitly request larger MTUs
  - Provide example RFQ language
Huge payback

- Deploy 9kB MTUs ASAP
  - Makes high performance 6 times easier

- Push IPv4 to 64kB
  - Makes high performance 40 times easier

- Push for even larger IPv6 MTUs
Maximizing DoE Impact

- Need hard (DoE) data: the value of large MTUs
  - Justify not buying excessively cheap gear

- Draft formula based procurement language
  - Exclude smaller than 4kB gear entirely
  - Score based on MTU
  - Deduct points for non-standard solutions

- Request improved Path MTU discovery in hosts
Epilogue
TCP is easy to blame, but...

- Most bottlenecks apply to all protocols when deployed
  - Not diagnosable due to an hourglass
  - Remote Required Congestion Control
  - No Direct Data Placement
  - Tiny Maximum Transmission Units

- These are often fixed in test environments
  - but do not carry over to production deployment

- As a consequence prototype experimental protocols tend to exhibit unrealistic performance gains that evaporate in actual use
Fix the real problems

(in all protocols)

- Built in diagnostic instrumentation
  - Web100, IETF MIBs for all protocols

- Improved Congestion control algorithms and policy
  - Net100, Floyd, Kelly, Low, Katabi

- Develop Remote Direct Data Placement
  - IETF RDDP working group,

- Push large Maximum Transmission Units
  - My pet project, Internet 2, DoE?
Organizational problems

- Users are under represented at the IETF
  - Little input from high performance users
  - Little protocol review by theoreticians
  - Not enough long range engineering view

- Funders are more interested in glitzy prototypes
  - Less likely to have any real impact
  - Ignore the unglamorous real problems

- NSF is taking on longer range, higher risk research
  - Who is funding engineering for the common good?