To TCP or not to TCP?

Matt Mathis
mathis@psc.edu

13 April 2004
http://www.psc.edu/~mathis/papers/JET200404/

Is TCP the right answer?

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

The spoiler

- TCP is easy to blame, but I claim:
  - 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.
- In the short term, non-standard protocols do have an advantage

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?

"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 the Internet hourglass!

Web100 Instrumentation and Autotuning

- About 120 TCP "test points" to diagnose network problems
- Prototype TCP MIB on the standards track: draft-ietf-tsvwg-tcp-mib-extension-03.txt
- Adjust TCP bufferspace automatically
- 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
- Testbed results are overly optimistic

Reinventing Congestion Control

- Several good options:
  - Floyd's high speed TCP and Kelly's Scalable TCP
    - Preserve strict TCP fairness at small windows
    - More aggressive at larger windows
  - Caltech FAST and Katabi XCP
    - Use delay sensing as primary control (new headers?)
    - Must still respond to losses
- Beware: anybody can design an aggressive protocol that will win when hand tuned for an isolated network.
  - Claims of "TCP fairness" are not well defined

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 protocols, including TCP
- All bad algorithms are risky due potential misuse

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
  - Decode 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

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

Maximum Transmission Unit

- Revisit congestion control

$$Rate = \frac{MSS}{RTT} * \frac{0.7}{\sqrt{p}}$$

[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
  - Low, 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

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

- These are subject to change as we get better data

Another 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

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
- IETF Internet Draft: draft-ietf-pmtud-method-01.txt
  - Running code and commercial interest

User-mode vs kernel protocols

- Non-TCP protocols are implemented in user mode
- Pros:
  - Rapid prototyping and experimentation
  - Easy debugging w/ less exotic tools
  - Faster deployment
- Cons:
  - More total syscall overhead
- Double hit for TCP: it is also mission critical
- TCP looses

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 ROI working group,
- Push large Maximum Transmission Units

Which is more important to most users?

- Raise the "tent pole" by pushing highest end apps
  - Hundreds of systems that require >10 Gb/s
- Raise the "skirt" by pushing "everyday" systems
  - Millions of US R&E systems that might use >100 Mb/s
- Funders are more interested in glitzy prototypes
  - Not suitable for global deployment
  - Ignore the unglamorous real problems