The Case for Raising the Internet MTU

Matt Mathis
mathis@psc.edu

28 July 2003

http://www.psc.edu/~mathis/MTU

Slides:
http://www.psc.edu/~mathis/papers/Cisco200307/
Outline

- Background and Basic Position
- Path MTU Discovery
- Assorted technical arguments
- Positioning and Strategy
- Summary
Part 1

Background
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

  - 1 second: 1.25 Gigabytes
  - 1 minute: 75 Gigabytes
  - 1 hour: 4.5 Terabytes
  - 1 day: 108 Terabytes

(From prior presentation to DoE Networking)
Moore’s law

- For networks: a factor of 10 every 3 years

- Moore’s law applies to:
  - All computer, instrument, and network components
  - Raw end-to-end Internet path capacity
  - Aggregate traffic (even steeper!)

- Computational Microscope of Fractal Universe

- But not to single end-to-end TCP connections
Why not TCP?

- Why doesn’t Moore’s law apply to TCP?

- What are the causes of the wizard gap?
  
  - Tuning is really debugging
  - Autotuning
  - Congestion Control
  - No Remote Direct Data Placement
  - MTU
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
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, stack and application problems

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

- When there are problems, just ask TCP
Important Observation

- 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
  - Laboratory performance results often don’t carry over to the field

- We need diagnostic MIBs in all transport protocols
TCP Autotuning

- Dynamically adjust TCP buffer space
  - Sender must buffer all unacknowledged data
  - Receiver must buffer all partial or undelivered data
  - Ideal size is slightly larger than maximum actual window

- Required to balance end system memory use

- Included in current Web100 release
  - New memory model
  - Sender side directly follows cwnd
  - Receiver side has to infer cwnd
    - Still some rough edges
Observation

- The buffer tuning problem is common to many protocols
  - E.g. NFS read/write ahead
  - "pipe size" is data rate times total latency (net+disk)
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
Change Global Congestion Control?

- Need to address economics of the commons
  - Global fairness (not specific to TCP)

- 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

- Net100 Work Around Daemon
  - Explicit controls to adjust congestion control
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 abusive
Remote Direct Data Placement (RDDP)

- Also called Remote Direct Memory Access (RDMA)
  - or "zero copy"

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

- Not supported by standard Internet protocols (yet)
  - require extra copies through system memory

- Current IETF effort
What about MTU?

- Maximum Transmission Unit or "packet size"

- Predominant MTU is more than 2 orders of magnitude too small for 10 Gb/s

- See http://www.psc.edu/~mathis/MTU
Maximum Transmission Unit

- Revisit congestion control

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

- 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 (10 nines) 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 MTU
  - Approximately constant packet time
  - Approximately constant window (in 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
Another vantage point

- 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

- Why?
Path MTU Discovery
Path MTU Discovery is Busted

- Path MTU Discovery (RFC1191) does not work well

- It requires ICMP messages from routers
  - Many problems - some outlined in RFC2923

- When pMTUd 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

- Nobody noticed that FDDI is faster than Fast Ethernet

- Totally eliminated the demand for larger MTUs
  - because nobody noticed that they make a difference
A 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
Key feature

- Only one requirement of the lower layers:
  Over-sized packets must always be discarded
Reactivate IETF pmtud WG

- Met at IETF 57 in Vienna
- Already on 3rd Internet-Draft
  - Mostly suggestions and guidelines
  - Only tight specifications are for "no CC on loss"
- Two current unix implementations
  - We are encouraging more early prototypes
- Good exposure/input from affected communities
  - Especially from IPsec/VPN people
Expect easy deployment

- Deploy on just the sender
  - Helps all outbound flows
  - Does not require updated net or receivers

- Fixes major operational headaches in several areas
  - IPsec, etc (See RFC2401)
  - PPPOE and other layer 2 tunnels
  - Helps Internet2/DoE/NASA "jumbo clean" effort

- Can be adapted to other protocols
  - SCTP, etc

- Wide support may be triggered suddenly
  - But still likely to take several years to deploy
Likely long term impact

- Random users notice that MTU makes a difference
- MTU becomes a marketable parameter
- Provoke MTU creep across the Internet
Technical Arguments
The general scaling argument

- For most bottlenecks, performance scales with MTU
  - End system CPU limits
  - Constant number of packets in flight
    - Including constant per packet loss rate
  - Most common alternative is constant bit/byte rate

- Every individual bottleneck can be treated
  - Or the entire problem can be made proportionally easier by raising the MTU

- Many current schemes can be viewed as workarounds
  - TCP Offloading
  - End-2-end optical switching
Performance scaling

- Network performance with constant packet time naturally follows Moore’s law for the lower layers
  - Constant packet time ->
  - Constant window in packets ->
  - Unchanging protocol dynamics

- Constant sized packets fundamentally disadvantage upper layers and end systems
Performance conjectures

- Large MTU + RDDP
  - Any reasonable pair of hosts can outperform any LAN

- Large MTU + autotuning
  - Any reasonable pair of hosts can outperform any WAN

- Large MTU + RDDP + autotuning
  - Any reasonable pair of hosts can outperform any network
No clear measurement data (Yet)

- Very few long 9k jumbo clean paths in the Internet
  - Known paths exhibit "interesting" properties
  - PSC has a proposal currently in review (NSF EIN)

- Good linear data below 1500 bytes
  - Both host limits and network limits

- Frequent demo lore:
  - Can’t meet performance goal at 1500
  - But at 9kB the problems are solvable

- All Land Speed Records are at 9kB MTU
Ethernet must be standard

- Ethernet is one standard
  - Common address and packet formats, etc
  - Spanning 4 orders of magnitude in data rate

- Claims:
  - Common MTU is central to Ethernet interoperability
  - Non-standard MTUs lead to interoperability problems
An alternative view:

- Does 10GigE <-> FastE <-> 10GigE really work?
  - Yes, if you just flick packets at it
  - No, if you care about data rate
  - No, if you care about jitter
  - No, if you care about headway

- There is already an implicit "convexity" assumption for some parameters

- MTU would not be any different
  - New pMTUd eliminates the need for explicit MTU negotiation
The CRC Argument

- Some claims that CRC-32 is only good for < 12 kB

- The worry is that there will be too many "false passes", corrupted packets with zero CRC syndromes

Observations
- CRC-32 is only error detecting
- Length of correct data does not affect CRC properties
- Random syndromes cause 1 false pass in 4 Billion (32bits)
Invert the Problem

- Start with a fixed data set and errors

- How does the probability of a false pass change with packet size?

- If the errors cause random syndromes, then
  - larger packets $\rightarrow$ more likely to have errors
  - larger packets $\rightarrow$ fewer packets to check
  - Exactly offsetting effects

- The total risk is constant across reasonable sizes
  - (improves at very large sizes due to packets with double errors)
  - This is due to assumed "random syndromes"
What about bit errors?

- Substitute for $p$ in the model and rearrange

$$p = BER \times 8 \times MSS$$

$$Rate = \left( \frac{0.25}{RTT} \right) \times \sqrt{\frac{MSS}{BER}}$$

- Raising MTU still increases performance
  - but more gradually
It doesn’t fit into current HW designs

- Most basic architectures have specific hard limits

- E.g. Simple bridges using rate adaption FIFO to compensates for clock drift
  
  - Implicit MTU limit from:
    - tolerances on inter-frame gap
    - FIFO size
    - and worst case clock drift
  
  - About a few kBytes for Ethernet

- To use larger frames you MUST store and forward
The current common limit

- One or two static packet buffers on the ASIC
- Buffer size limited by current fabrication

Alternative architecture
- External DRAM, possibly in cells
- Stream (DMA) the data from the framer into memory
- Already in use by some interfaces in some products
What - no DRAM?

- Many current designs have no large memory
  - Theses are generally vastly under buffered

- For reliable TCP self clock:
  - All network devices must be able to accept an entire window of data at line rate
  - At better than 6 nines reliability!

- Short queues are ok if the egress never blocks
  - no cross traffic
  - no head of queue blocking with other ports

- Failures are extremely hard to detect
  - Lossless in simple environments
  - Lossy in complex situations involving bursty traffic
New project at PSC

- Diagnostic methodologies for long Internet paths
  - Explicit tests for burst capacity
  - In situ tests for suspect gear
  - Bench tests to test gear in the lab

- Successor to Web100
  - (Passed NSF review, but no funding letter yet)
Conjecture

Devices that overly depend on buffering in ASICs will prove to be generally problematic for high performance flows over wide area paths

- Easy to identify: buffer drain time is a few ms or less
- These architectures generally don’t scale with MTU
- Insufficient buffering to sustain TCP self clock
  - But tiny MTU forces the bottleneck to be elsewhere
  - So the buffering problem remains hidden
How much more cost?

- Architectures that scales to large MTUs
- Assume "cellified" memory
- Roughly double costs
  - Two levels of buffer chaining
  - Worst case needs twice the memory bandwidth
    - (1 cell + 1 byte)
Comparison to ATM

- 1500 Bytes over 10 GigE takes 1.2 us
- 53 Bytes over OC-3 takes 2.7 us

ATM was largely killed by the cost of SAR
  - Explicit cost reflected in the NIC HW
  - But now the cost is hidden in SW overheads (or TOE cost)
Comparison to Interconnects

- Major performance problem for clusters

- PSC: 3000 HP (DIGITAL/Compaq) alpha chips
  - $40M in CPUs + $5M in Quadrix switch
  - Huge MTU (hidden cells) -> low overhead
  - Otherwise burn 100% of the CPUs to fill the switch at 1500 Bytes

- Quadric, GSN (HIPPI-6400), Myrinet, etc
  - Exist due to Ethernet MTU and no standard RDMA
  - But these all have hidden cells inside!
IP fragmentation is not sufficient

- Consider sequence numbers at 10 Gb/s
  - TCP wraps sequence number (32 bits) every 3.5 s
  - Use timestamps and PAWS to extend sequence space
  - Effective sequence space is at least 40 bits
  - Protect against segments 180 seconds late

- IP fragment numbers are 16 bits
  - Wraps every 80 mS
  - Reordered fragments are prone to miss-association
A new protocol

- Need to consider interactions with congestion control
- FEC to recover some fragments (adaptive?)
- Large IDs on fragments (64 bits)

Requirements:

- Use IP in IP encapsulation to emulate large MTU
Deployment issues

- Extra box or in the NIC?
  - Extra box would require a large MTU LAN interface
    - May as well make it a router interface....

- Within the NIC
  - Introduces negotiation problems
    - How can you determine if the other end is capable?
  - Clash with TOE, etc

- Feels like a temporary workaround
  - May shed light on the ideal MTU
Part 4

Positioning and Strategy
The Pitch to Large Users

- Slides from my recent presentation to DoE networking
  - Introduction from above
Two intertwined subtasks

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

- 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
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
LAN vendor resistance

- The LAN vendors want tiny jumbograms (9k)

- Vendors are focused on the mass market
  - Current NIC + port costs are below $50
  - We are at best only a small share 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
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
Pitch to Manufacturers

- Today users have little incentive to upgrade NICs
  - Well understood that typical links are under filled
  - TCP gets blamed for poor performance
    - Still way below 100 Mb/s
  - No wildfire to deploy 1 Gb/s or 10 Gb/s

- My claim: for most R&E users raising MTU will make more difference than raising the clock rate

- (Yea, I know I need data)
Selling gear

- Which will get better performance?
  - 1 GigE at 64k bytes
  - 10 GigE at 9k bytes

- Easy story to sell

- Even users stuck at 100 Mb/s (E.g. old wiring)
  - Will get more performance w/ 12 kB packets

- Turn over entire R&E installed base

- What do enterprise networks need?
Longer Term Impact

- Users will redefine "large files" by two order of magnitude
- The lasers are going to be a major problem
- QoS will be required (yea right!)
My goal is diversity in deployed Internet MTUs
  • Raise the high end

Many users are happy with 1500 bytes (and short queues)

R&E users are very unhappy with current performance

One product may greatly alter the market
  • A router interface for a large MTU LAN
http://www.psc.edu/~mathis/MTU