Drawing the line between transport and network requirements

Matt Mathis
mattmathis@google.com

IETF 86  ICCRG
6-Nov-2012
Goals of this presentation

- Ask some unsolved questions
- Stimulate thought
- Collect pointers to existing work
- Identify potential contributors
- Encourage interest in the revitalized IPPM WG

- Non-Goal: answering the questions at this point
IPPM Model Based Metrics

- Active IPPM Draft
  - draft-mathis-ippm-model-based-metrics-01.txt
  - Matt Mathis & Al Morton
  - A tractable way to characterize bulk performance

- It will implicitly divide responsibility
  - Bursts (e.g. IW10 and TSO)
    - TCP should send fewer or
    - The network should tolerate more
  - Reordering
    - TCP should tolerate more
    - Then network should cause fewer
  - Etc
Bulk Transport Capacity is hard for a reason

- TCP and all transports are complicated control systems
  - TCP causes self inflicted congestion
  - Governed by equilibrium behavior
  - Changes in one parameter are offset by others

- Every component affects performance
  - All sections of the path
  - End systems & middle boxes (TCP quality)
  - Routing anomalies and path length

- The Meta-Heisenberg problem
  - TCP "stiffness" depends on RTT
  - The effects of "shared congestion" depend on
    - Bottlenecks and RTT of the other cross traffic
  - Can't generally measure cross traffic with 1 stream
Model Based Metrics: A better way to do BTC

- Open Loop TCP congestion control
  - Prevent self inflicted congestion
  - Prevent circular dependencies between parameters
    - Data rate, loss rate, RTT

- Independently control traffic patterns
  - Defeat congestion control (generally slow down)
  - Mimic all typical TCP traffic (bursts, etc)

- Measure path properties section by section
  - Mostly losses
  - Compare to properties required per models
  - E2E path passes only if all sections pass all tests
The pieces (simplified)

End-to-end path determines target_RTT and target_MTU

Rest of path is assumed to be effectively ideal

The "application" determines target_rate

Sub-path under test

Must meet constraints determined by models based on target_rate, target_RTT and target_MTU
A common context for all examples

- **Target parameters:**
  - 1 MByte/s bulk data over a path that is
  - 10 Mb/s raw capacity (~1.2 MByte/s)
    - More than the target!
  - 20 ms, 1500 Byte MTU, 64 byte headers

- **Compute from Macroscopic Model**
  - **target_pipe_size**
    - \[\text{target_rate} \times \text{target_RTT} / (\text{target_MTU} - \text{header_overhead})\]
    - 14 packets
  - **reference_target_run_length** (= 1/p)
    - \[(3/2)(\text{target_pipe_size}^2)\]
    - 274 packets
    - Same as \(p < 0.365\%\)
A common context for all examples

● Target parameters:
  ○ 1 MByte/s bulk data over a path that is
  ○ 10 Mb/s raw capacity (~1.2 MByte/s)
  ○ 20 ms, 1500 Byte MTU, 64 byte headers

● Compute two additional (new) parameters:
  ○ Headway at target rate
    ■ target_headway = target_MTU*8/target_rate
    ■ target_headway = 1.5 mS
  ○ Headway at bottleneck rate
    ■ bottleneck_headway = target_MTU*8/effective_rate
    ■ bottleneck_headway = 1.2 mS
1) Baseline (CBR) performance test

- Measures basic data and loss rates
- Send one 1500 byte packet every 1.5 mS
  - 1 MByte/s target rate
  - Losses MUST be more than 274 packets apart
    - Otherwise "standard" Reno TCP can't fill the link
- Note that this is pass/fail
- Cool properties
  - Does not depend on sub-path RTT
  - Does not depend on measurement vantage
    - As long as rest of path is good enough
  - Run length bounds loss rate for entire path
Derating

- To some extent the model is subjective
  - And too conservative
  - What if TCP isn't standard Reno?
- Must permit some flexibility in the details
  - As TCP evolves
  - As the network evolves
  - The ID permits "derating"
- Actual test parameters must be documented
  - and justified relative to the targets
  - and proven to be sufficient
    - Meet the target goal over a derated network
- (ID will have) text about calibration and testing
2) Slowstart style burst test

- Mimic last RTT of a conventional TCP slowstart
  - Measure queue properties at the "constrained link"

- Send 4 packets every 2*bottleneck_headway (2.4 mS)
  - Builds a queue at bottleneck
  - Burst of 2*target_pipe_size (28 packets)
    - Peak queue will be target_pipe_size (14 packets)
    - (Test inconclusive if ACK are too early ->no queue)
  - Repeated bursts on 2*target_RTT headway
    - Below 14 packets, MUST meet target_run_length
    - Beyond 14 packets MAY derate
    - Beyond 28 packets (more?) loss rate SHOULD rise
      - To prevent excess queueing (bufferbloat)
      - THEORY or MODELS NEEDED
3a) Interface rate bursts caused by the server

• Full rate (e.g. 10 Gb/s) bursts from a server/tester
  ○ Note that these mostly stress the "front path"
    ■ Server up to the primary bottleneck
  ○ Typically not the same queue as the SS tests
    ■ Smaller bursts
    ■ Higher rate

• Caused by various application effects
  ○ 3 Packets: normal window increases, all states
  ○ 10 Packets: IW10
  ○ 44 Packets: TSO (only if cwnd is large enough)
  ○ Application or scheduler stalls
    ■ Any fraction of 2*target_pipe_size possible
    ■ Statistics scale: (target_rate) * (sched_quanta)
3b) Interface rate bursts caused elsewhere

- TCP sender reflects ACK bursts into the data
- Caused by:
  - ACK compression due to other traffic
  - Thinning/merging ACKs (network or receiver)
  - Compression due to channel allocation
    - E.g. Half duplex
  - Reordering etc of cumulative ACKs

- Clearly if the network caused the problem
  - TCP isn't likely to fix it
  - Even if it was a different network section
What burst tolerance should be ok?

● General pattern:
  ○ No runlength derating for small burst sizes
  ○ Progressively more RL derating at larger burst sizes
  ○ It is a tradeoff between TCP and the network
    ■ Small bursts must be tolerated by the network
    ■ Network must tolerate network induced bursts
    ■ TCP should not cause large bursts

● NEED A MODEL

● Quick answer
  ○ We have been underestimating the impact of TSO
4) Tolerance to reordering

- OPEN QUESTION My specualtion
- Strict sequential switching costs Internet scale
  - Forced sequential processing
  - Less concurrency within chassis
  - No ECMP routing, even at the fabric level
  - Extra interlocks, controls or hashes
  - Mostly motivated by non-TCP applications
  - But TCP has it's limitations too

- How would it change net if reordering was common?
  - What is the opportunity cost of the current state?
5) Standing queue test

- Run approximately fixed window transport
- Gradually increment window
- Collect statistics on the onset of loss
Queuing example

From "Windowed Ping" - INET 94
Standing queue test

- Collect statistics on first loss
- Must not be before target_run_length
  - Otherwise TCP will not fill the link
- Must not happen at too large of queue
  - Direct measure of bufferbloat
  - How big is too big?
  - NO MODEL or THEORY
Open Questions

- During SS, how large of queue is too big?
  - Should there be an upper bound on the queue size?
- How large server rate bursts should:
  - the network tolerate?
  - TCP avoid?
- How much reordering should be ok?
- How long/much standing queue is ok?
  - Should there be an upper bound on triggering AQM?
• The end