Model Based Metrics

draft-mathis-ippm-model-based-metrics-01.txt

Matt Mathis
mattmathis@google.com
Al Morton
acmorton@att.com

IETF 86 IPPM
Pie Day-2013

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

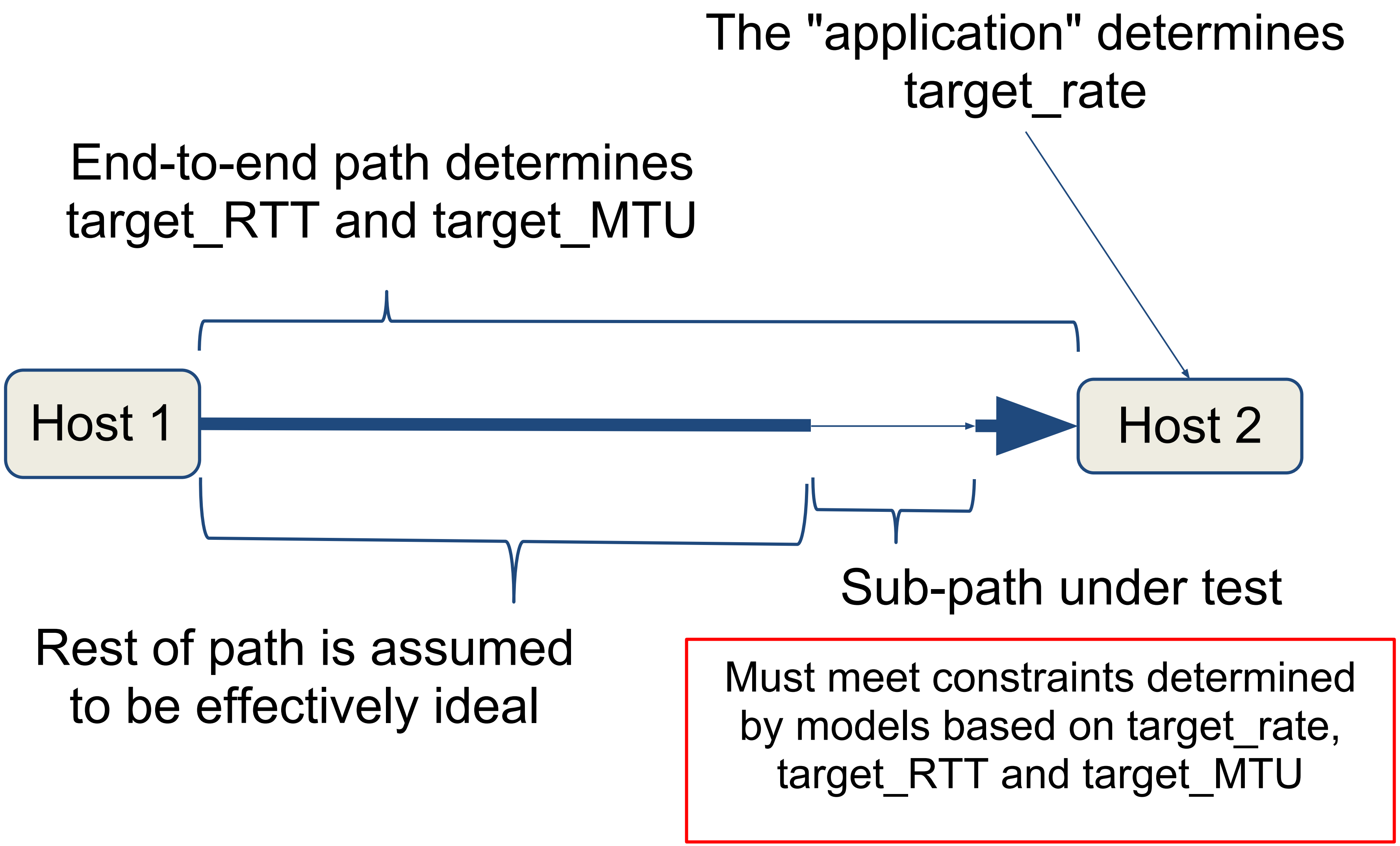
An example

- Goal: 1 MByte/s BTC over a path that is
  - 10 Mb/s raw capacity (~1.2 MByte/s)
  - 20 ms, 1500 Byte MTU, 64 byte headers
  - Invert TCP performance model [MSMO97]

$$Rate = \left(\frac{MSS}{RTT}\right)\frac{C}{\sqrt{p}}$$

  - Yields loss probability budget less than 0.3%
  - Test each short section at 1 MByte/s
- Fails if total loss probability is more than 0.3%
  - This is a pass/fail test, not a measurement
  - But passing this test alone is not sufficient
    - Because the link can still fail in other ways

The pieces (simplified)



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 TCP Macroscopic Model
  - target_pipe_size
    - target_rate*target_RTT / (target_MTU-header_overhead)
    - 14 packets
  - reference_target_run_length (= 1/p)
    - (3/2)(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)
    - More than the target!
  - 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
    - bottlenenck_headway = target_MTU*8/effective_rate
    - bottlenenck_headway = 1.2 mS

Derating

- To some extent the models are 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

Deciding if a test passes

- Recursive run length measurement
  - Progressive testing
  - Accumulate counts of losses and delivered packets
  - When to:
    - Declare success
    - Declare failure
    - Give up (declare inconclusive)

- Inconclusive also covers other non-results such as:
  - Tester failed to generate prescribed traffic patterns
  - Link was determined to be non-idle
  - etc
- Beware: inclusive tests can introduce sampling bias
  - Must strive to eliminate them

Sequential Probability Ratio Test (SPRT) **

Help Determining Sample Size & Pass/Fail/Indeterminate:

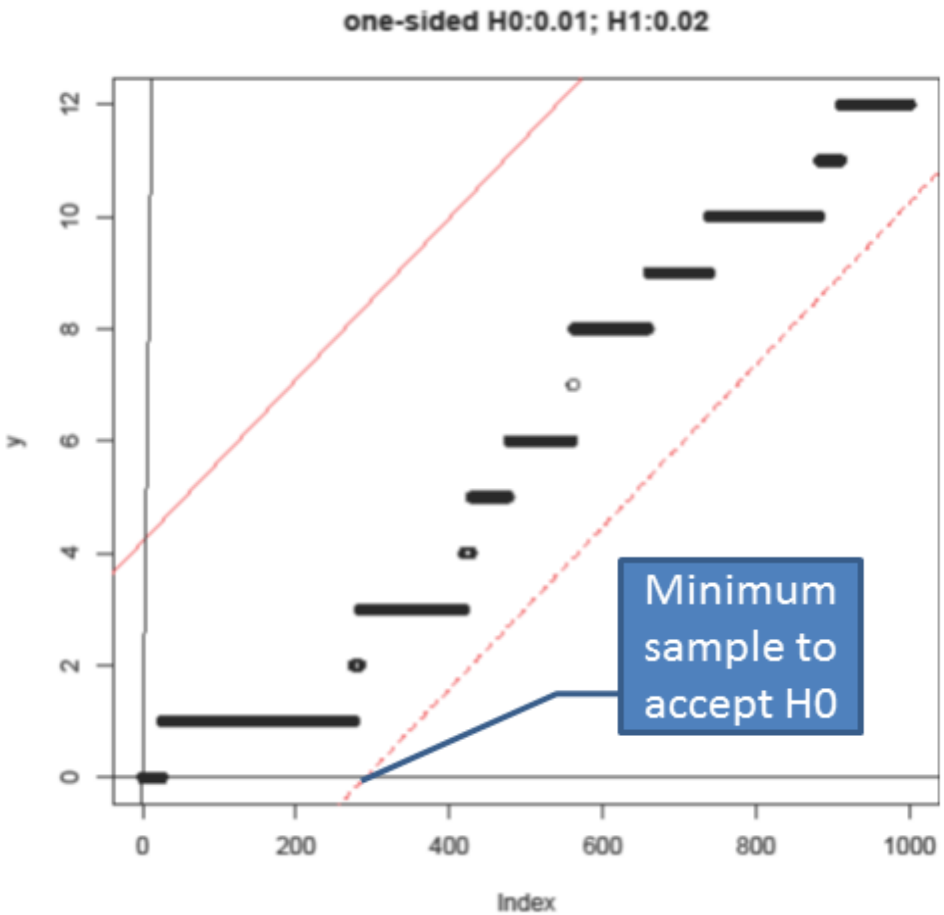
- In practice, can we compare the empirically estimated loss probabilities with the targets as the sample size grows?
- How large a sample is needed to say that the measurements of packet transfer/loss indicate a particular run-length is present (with desired error)?
- Lost packet or other impairment ~ Defect

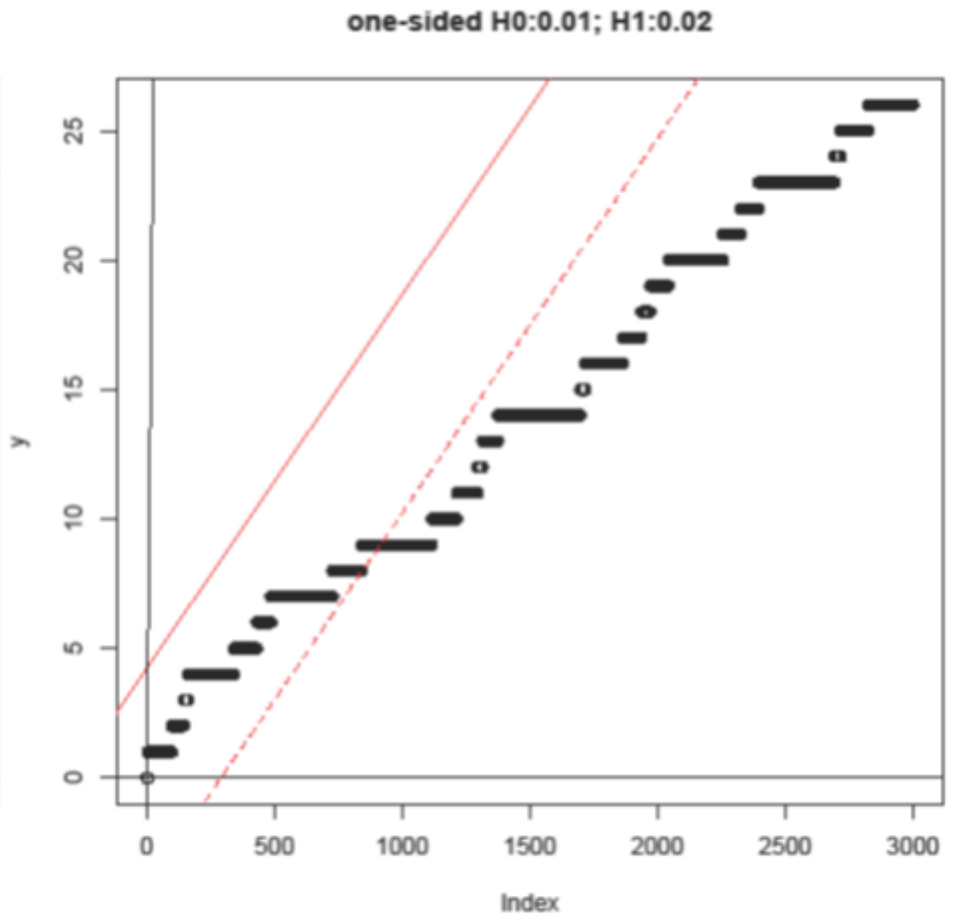
We set two probabilities, one using target_run_length (H0) and another target_run_length/2 (H1), and the Type I and II errors (0.05).

The SPRT test calculates cumulative limits to evaluate the defect ratio as the sample size grows.

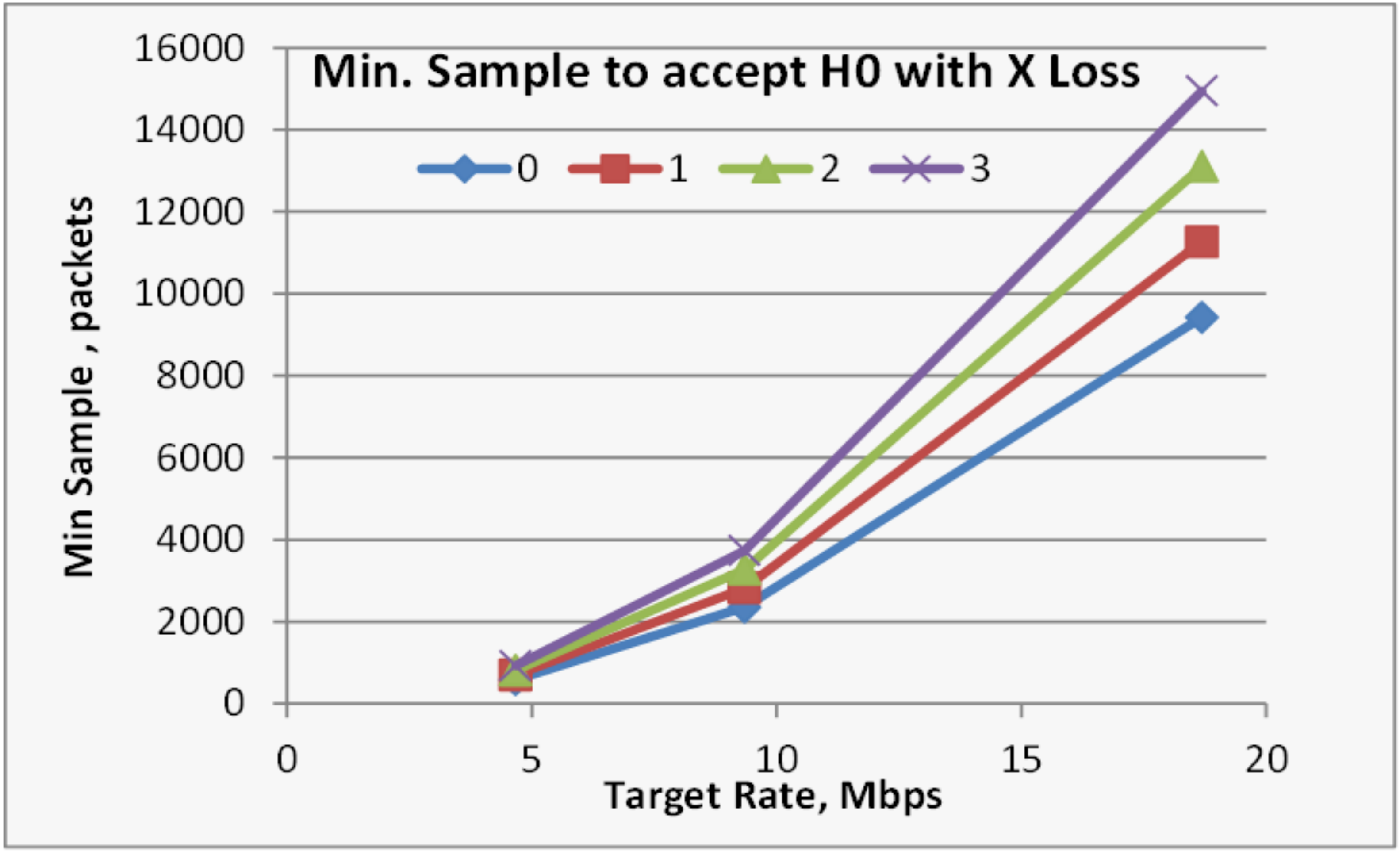
** suggested by Ganga Maguluri, AT&T Labs

SPRT: alpha= beta=0.05





SPRT: alpha= beta=0.05, H0*2=H1



The MBM tests

- Baseline CBR performance
- Slowstart style burst tests
- Server interface rate burst tests
- Reordering tests
- Standing queue test

All tests have valuable properties

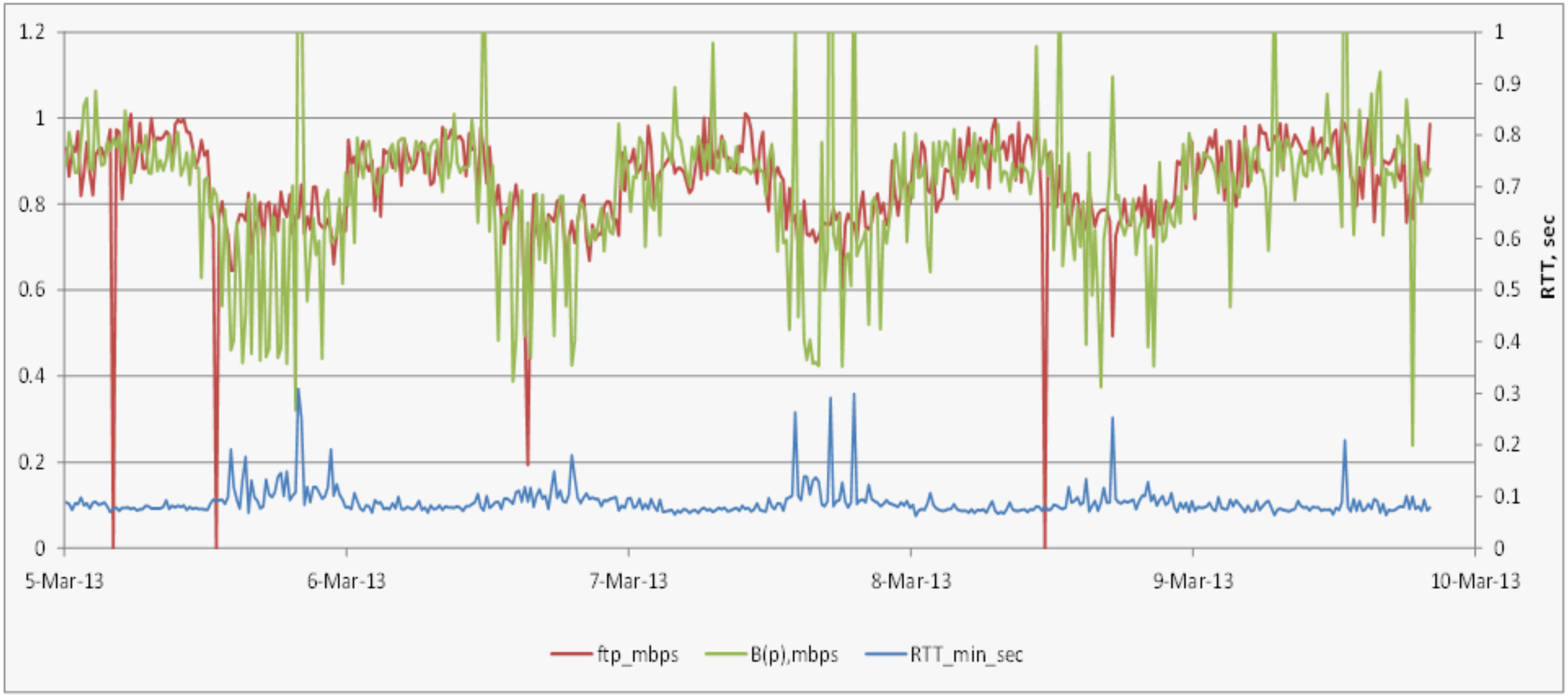
- Tests do not depend on sub-path RTT
  - (Except one detail)
- Tests do not depend on measurement vantage
  - As long as rest of path is good enough
- Tests should not depend on implementation
  - Different parties should get the same results
- There is an algebra on test result
  - Summing (or pre allocating) losses
  - Any failed test on any sub-path fails the path

Keep these in mind

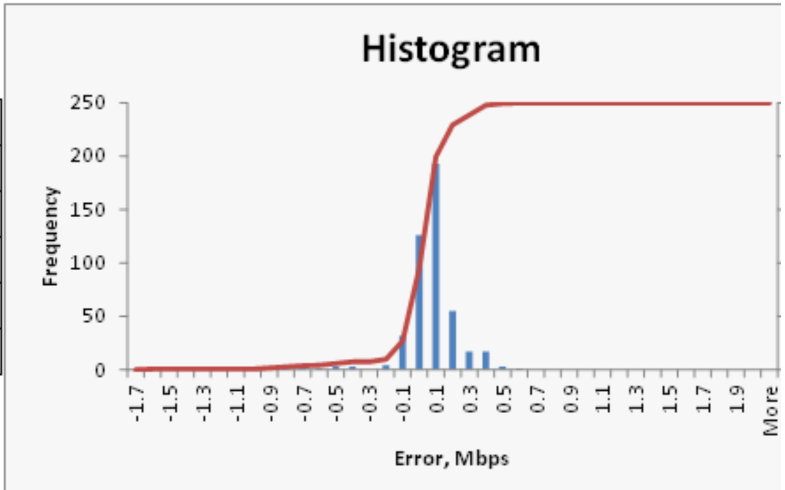
1) Baseline (CBR) performance test

- 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
- Derated or Intermittent testing
  - e.g. reduced data rate for stealth mode testing
  - No derating on target_run_length
    - (Use a different model instead)

Example Intermittent Testing



Date	Ave FTP	Ave B(p)	Ratio,%	dB increas	%<FTP
3/5/2013	0.830	0.807	97.2%	-0.12	2.8%
3/6/2013	0.831	0.815	98.1%	-0.08	1.9%
3/7/2013	0.839	0.825	98.4%	-0.07	1.6%
3/8/2013	0.838	0.828	98.8%	-0.05	1.2%
3/9/2013	0.905	0.899	99.3%	-0.03	0.7%



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_lenght
    - 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 server and application effects
  - 3 Packets: normal window increases, all states
  - 10 Packets: IW10
  - 44 Packets: TSO (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 IF rate 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

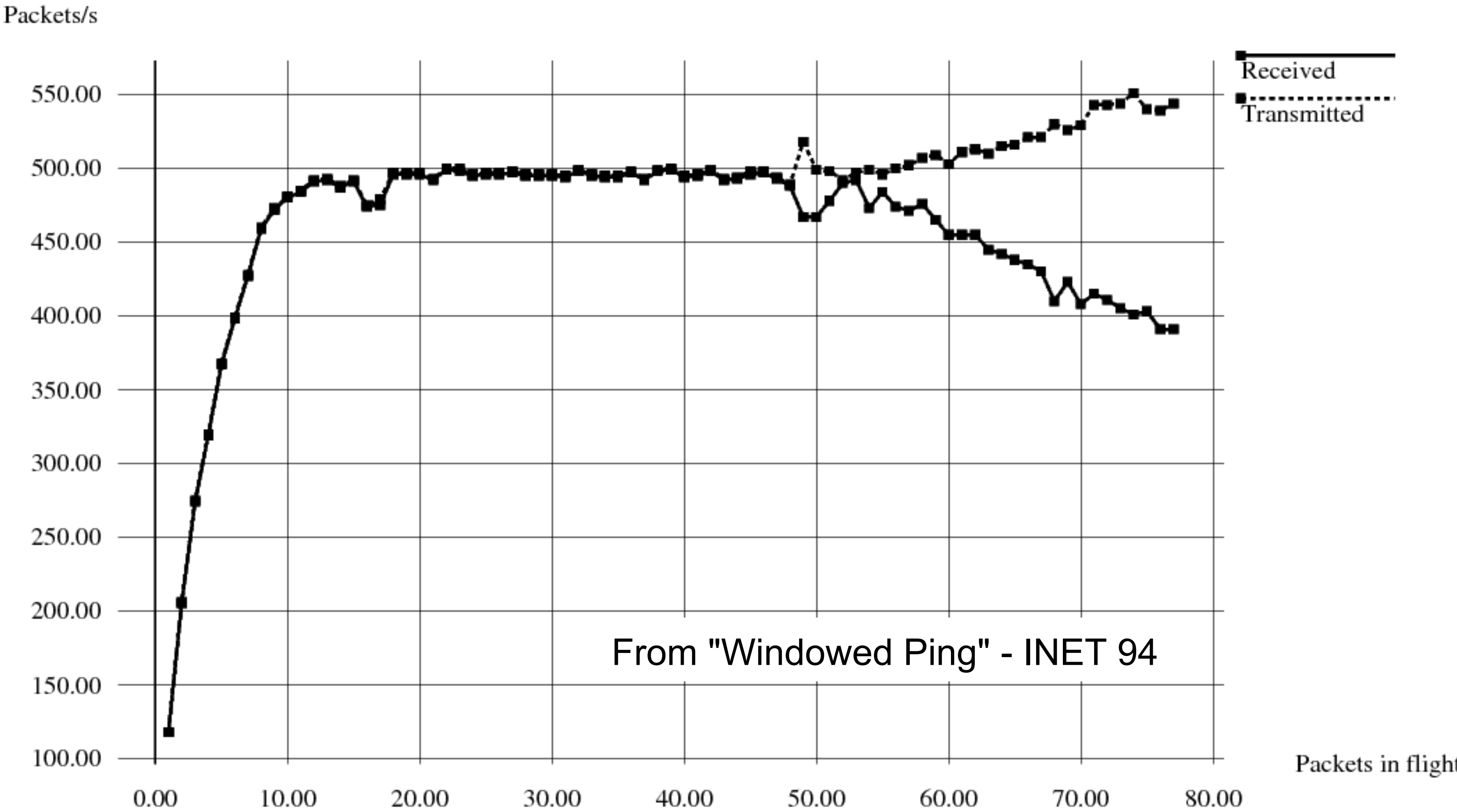
4) Tolerance to reordering

- OPEN QUESTION   My speculation
- 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?
- Seeking compelling stories about reordering tolearnce

5) Standing queue test

- Run approximately fixed window transport
  - Start slightly below **test_RTT*bottleneck_Rate**
- Increment window once per 2*target_RTT
  - Mimic congestion avoidance at the target RTT
- Collect statistics on the onset of loss
  - Count from MAX(Rate/RTT) to first loss
- Must not be before target_run_length (in average)
  - 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

5) Standing queue test



Conclusion

- Must defeat:
  - Throughput maximizing
  - Congestion control
  - Equilibrium behavior
  - Heisenberg effects
- The trick:
  - open-loop congestion control
  - application determines data rate and bursts
- Compute pass/fail/inconclusive outcome
  - application accurating controlling the traffic
  - Loss statistics consistent with required pattern

Adopt as a WG work item?

- Also seeking contributors
  - Can give shared write to anybody with @gmail login