Why is BTC so hard?

Matt Mathis
mattmathis@google.com

Why is Bulk Transport Capacity so hard to measure?

Throughput maximization does not work for measurement

- Basics of congestion control
- Circular dependencies, Heisenberg and equilibrium behavior
- Examples of measurements that fail
- A better approach - application controlled traffic
- Aka "pseudo CBR"
- Further Opportunities

TCP is **throughput maximizing**

- **By design....**
- TCP fills any/every bottleneck by creating a queue
  - This raises the RTT
- The network "regulates" the queue by dropping packets
  - e.g. it raises the loss rate
  - Explicitly as part of "Automatic Queue Management" (AQM)
  - Implicitly for drop tail queues (perhaps with bufferbloat)
  - Which causes TCP to slow down
- Circular dependencies between data rate, loss rate and RTT
  - "Equilibrium" behavior
  - Any change in any component/parameter affects all others
- TCP causes **self inflicted congestion**
  - The Heisenberg effect: the measurement changes the thing measured

Traditional TCP bulk performance model

- Describes the TCP "Sawtooth" in steady state
- Three main variables: *Rate*, *RTT* and loss rate, *p*
  - C is a constant that depends on TCP implementation details, etc

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

- For bulk transport steady state, this is a statement of equilibrium
  - If you control any 2 parameters, TCP adjusts the third to agree
  - E.g. for a fixed path (fixed Rate and RTT) TCP "solves" for *p*
- This principle applies to all TCP models

Dissect the TCP model

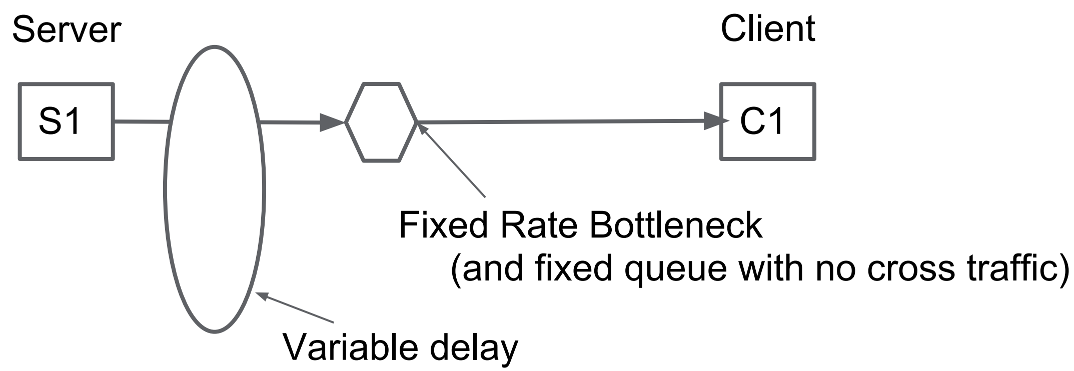
$$Rate = \left(\frac{MSS}{RTT}\right)\frac{C}{\sqrt{p}}$$

- All TCP models have the same general form
- The first term: (MSS/RTT)
  - Scales number of packets in flight to the data rate in bytes/second
  - Always has RTT in the denominator
  - Comes directly from "window behavior" in TCP (and other protocols)
- Second term estimates the number of packets in flight
  - Varies widely from model to model
    - The above model only applies to sustained bulk data
    - A direct consequence of sender side control algorithms
    - Not a solved problem in general
    - But all should have "similar forms"
  - Mostly depends on loss rate, sometimes RTT, etc
  - Other terms folded into constant C
    - Between 0.7 and 1.4 for most TCP's

Some ways in which TCP fails as a measurement protocol

- Counterintuitive RTT effects
- Meta Heisenberg at every shared bottleneck
- Performance is a system property
- Congested performance is a system property
- Local testing leads to incorrect blame and bad politics
- Not actionable by ISPs
- No model for concatenating paths

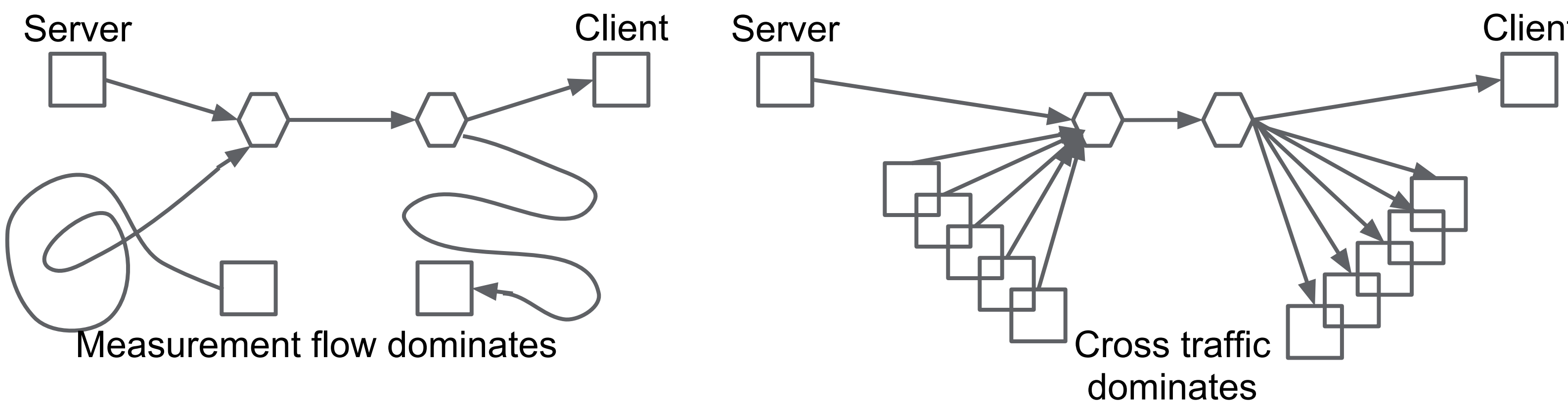
Counterintuitive RTT effects



- A better (shorter) path reduces the RTT
- But the data rate stays the same
- So the average quantity of data in flight must be smaller
- So the losses must happen sooner or more frequently
- So the loss probability must be higher

- Shorter RTT also has shorter request Response Time (RT)
- So the user with the better experience has higher losses!
- **Raw loss statistics do not imply network quality**

Meta Heisenberg at every shared bottleneck



- Heisenberg knew he was measuring electrons with photons
  - For networks, the relative "stiffness" is unknown
  - Measurement stream vs the cross traffic
- Things that increase the stiffness of the cross traffic:
  - Short RTT
  - Many flows
  - Additional bottlenecks stabilizing the cross traffic
- Stiffness can vary by orders of magnitude in either direction
  - A single measurement tells you very little

Congested TCP performance is a system property

- TCP congestion control is a complicated control system
- Every component contributes to the overall performance
  - TCP implementation details and quality
  - Application behavior
  - Network link properties
  - Other portions of the network (e.g. the home net)
  - **End-to-end RTT**
- Since the system has circular dependencies
  - Every metric depends on every component
- Calibration is (essentially) impossible
  - See RFC 3148 "A Framework for ... Bulk Transfer ... Metrics"
  - RTT dependence is the big killer
    - The NPAD tool (Measurement-Lab) attempted to address RTT

Local testing leads to incorrect blame and bad politics

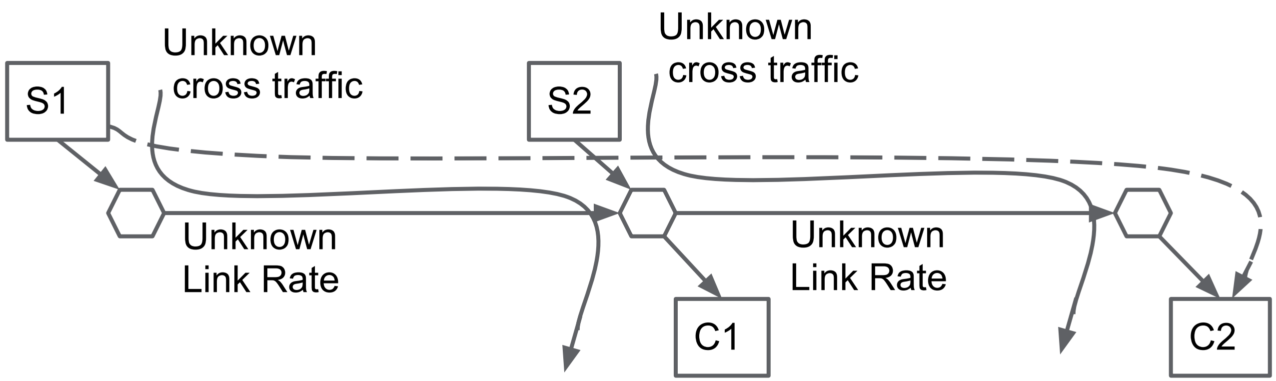
- The users tests the ISP and get one result
- The ISP owns just one of many elements of the test
  - The ISP does their own tests
- User measurements **NEVER** agree with the ISP's measurements
  - (NOTE: vantage sensitivity is a serious problem)
- ISP's logical conclusion: the fault must lie elsewhere
  - The ISP is being blamed for other people's problems
- User's logical conclusion: the ISP is cooking the test results
  - Anything hidden or proprietary is probably corrupt
- But both conclusions are probably wrong

Vantage sensitivity poisons sane conversations about policy

Not actionable by ISPs

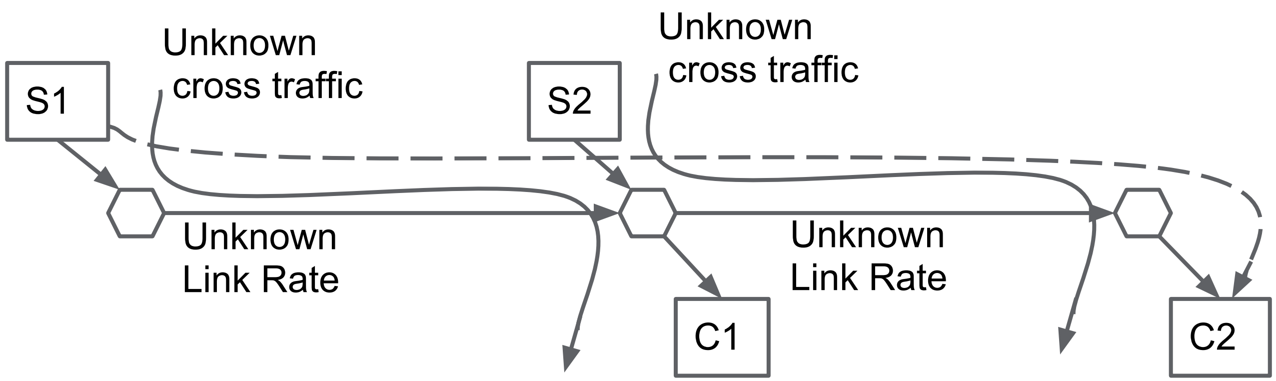
- Note that the ISPs wants to sell layer 2 (link) or 3 (IP) services
  - User wants to buy end-to-end layer 4 (TCP services)
- Since TCP performance is a system property
  - It can't be replicated by others
    - Vantage point matters
  - The ISP can't create the same path or system as a user
  - Testing an alternate path may not have the same symptoms
  - Fixing an alternate path may not help the user
- It would be foolish to include non-actionable items in a SLA
  - Never see real SLA language about application performance, ever
- Failing workaround.....
  - Define SLAs in terms of private, ISP based measurements
  - But they don't agree with user's measurements
  - Users assume that unverifiable measures are crooked......
- Unverifiable measurement has bad karma
  - This is why Measurement Lab is so focused on open measurement

No model for concatenating paths



- Want to predict properties S1->C2
  - From measuring S1->C1 and S2->C2
- This does work for one case
  - When there is zero cross traffic then
    - rate(S1->C2) = MIN(rate(S1->C1), rate(S2->C2))
    - Loss rate if you can invert a suitable model
  - But you may not be able to tell if you have zero cross traffic

No model for concatenating paths 2

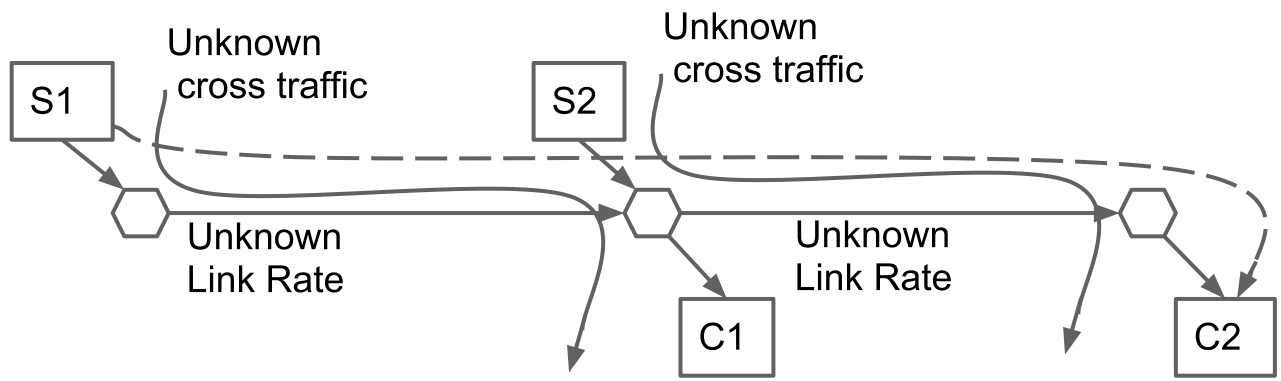


- Want to predict properties S1->C2
  - From measuring S1->C1 and S2->C2
- With unknown cross traffic - There is no hope....
  - Data rate always worse than either path alone
    - And sometimes very much worse due to multiplicative cross terms
  - Loss rate can be better than either path alone
    - Due to RTT effects, if the cross traffic is small
  - Loss rate can be loss(S1->C1)+loss(S2->C2)
  - or anywhere in between
- TCP has zero predictive value due to its equilibrium behavior!

Application controlled avoids equilibrium behavior

- Control the data rate by a non-network element such as a codec
  - TCP chronically runs out of data
    - Must avoid "startup" bursts too
  - Or a real time process controls UDP transmissions
- The measurement traffic should not cause queues or losses
  - Any queues or losses should be caused by cross traffic
- "Open loop" all congestion control algorithms
  - Rate (or traffic pattern) is determined only by the application (tester)
  - Losses and RTT are determined only by the network and cross traffic
- This suppresses all circular dependencies
  - Can measure the "open loop response" of each component

Model concatenation using application controlled traffic



- Want to predict properties S1->C2 from S1->C1 and S2->C2
  - Measure both sub-paths with fixed rate traffic
- Trivial to predict the loss rate
  - Losses determined solely by the network and are statistically independent
  - losses(S1->C2) = losses(S1->C1)+losses(S2->C2) // small probability assumption
- Supports algebra and inference on loss rate
  - Loss rate can be treated as a linear property!
  - Can predict S2->C2 by subtracting loss(S1->C1) from loss(S1->C2)
  - We can do tomography!

Model Based Metrics

- Use performance targets to precompute
  - Traffic Patterns
  - Success Critera
- Perform open loop testing
  - Details of the network behavior do not affect the traffic
  - Details of the testing topology do not affect the traffic
  - Loss measurements are independent per section
  - For low rates, losses can be treated as linear
- Solves ALL of the problems above with throughput maximizing
  - Especially "vantage sensitivity"