Modeling Persistent Congestion for Tail Drop Queue

Alex Marder & Jonathan M. Smith

University of Pennsylvania

Problem

- Can we determine the severity of persistent congestion?
  - 100mbit >> 1mbit
- Why?
  - How bad is interdomain congestion?
  - Is service degraded due to DDoS attack?
- What about TCP?

Can We Use TCP?

- Requires host on both sides of the link

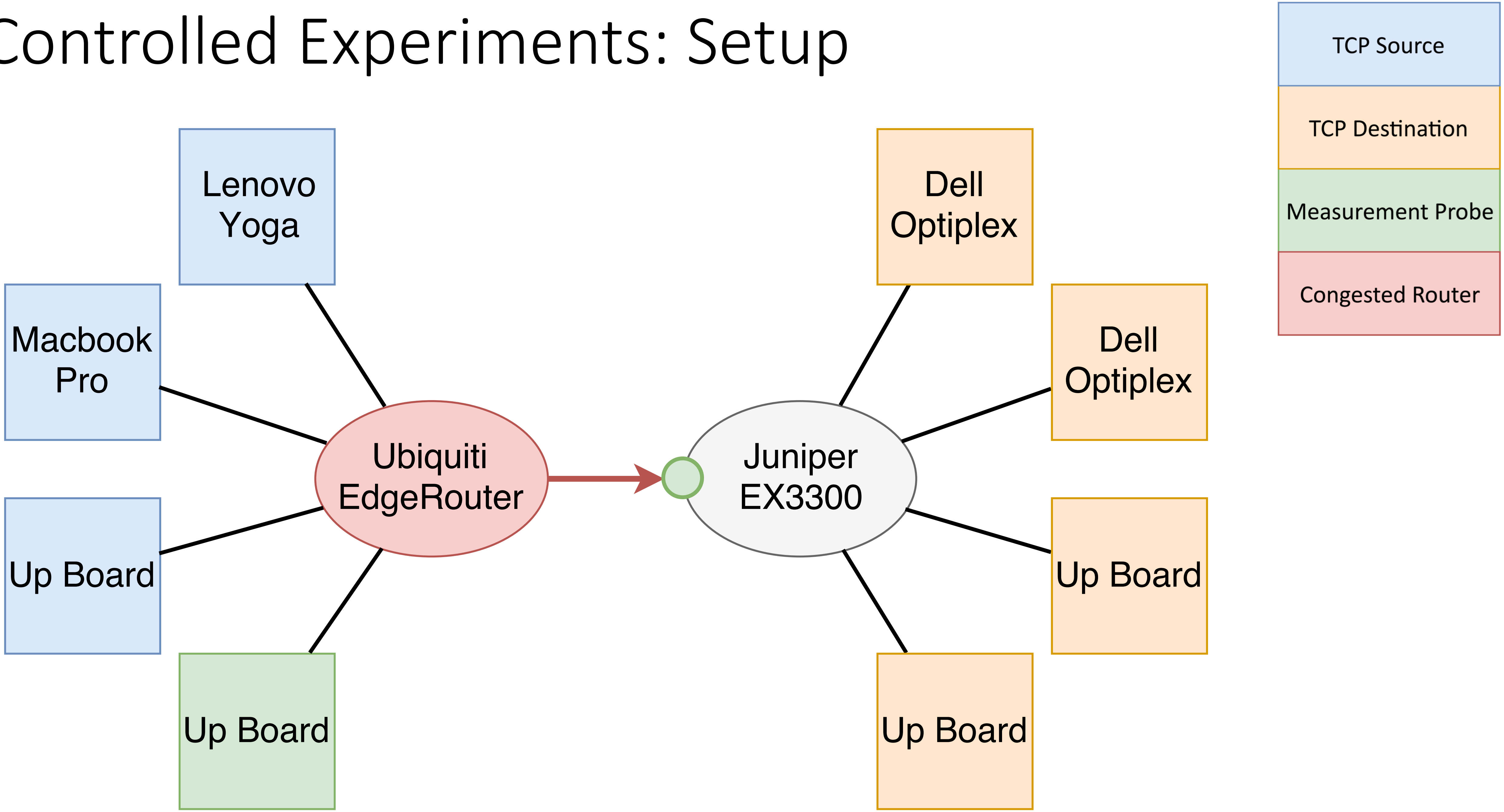
- Measures end-to-end throughput
  - Can be difficult to determine the bottleneck

- Smaller RTT gets more throughput

Goals

- Use edge probing to determine the average per flow throughput of TCP flows on persistently congested links

Controlled Experiments: Setup

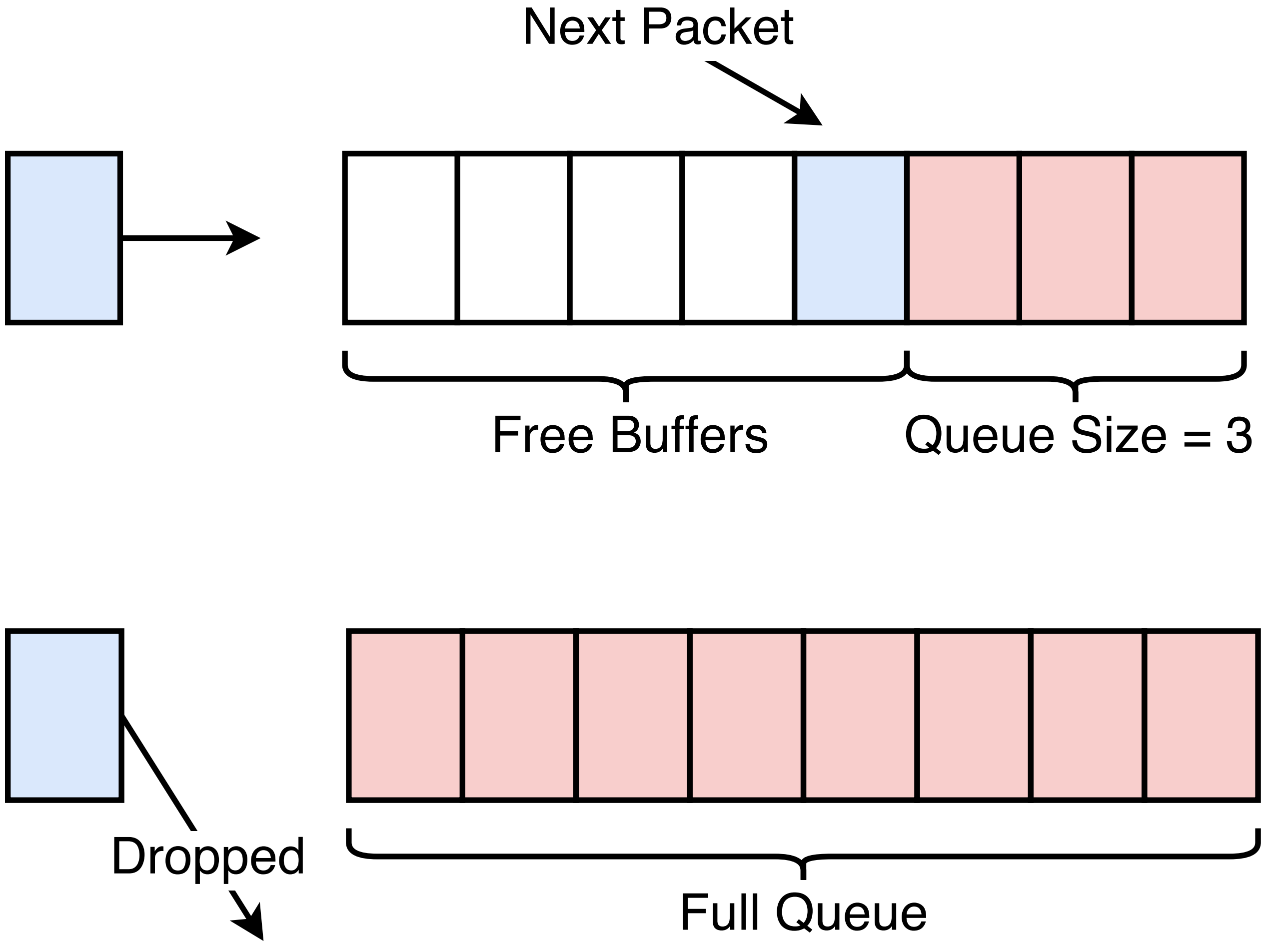


Controlled Experiments

- Use TCP flows to adjust per-flow throughput
    - 100 flows ≈ 10mbit, 1000 flows ≈ 1mbit
- Flows last [1, 5] seconds
    - Immediately replaced by new flow
- 1000 probes per measurement
    - 100ms intervals

FIFO Tail Drop Queue

- **Queue depth:** maximum number of packets in queue
- If Arrival Rate > Link Bandwidth
  - Queue size increases
- If Arrival Rate < Link Bandwidth
  - Queue size decreases
- Packets are dropped when queue is full



TCP Variants

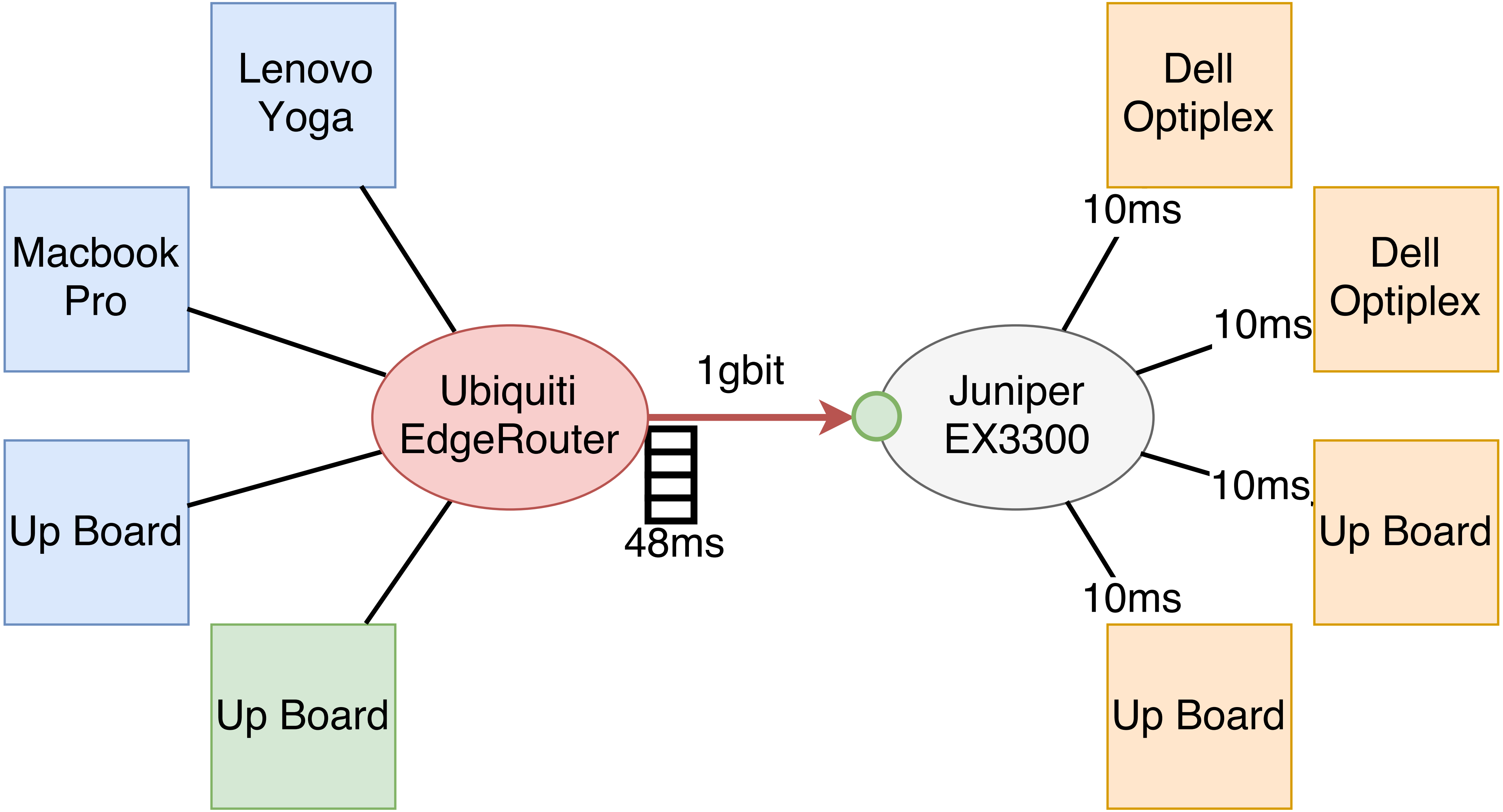
NewReno

- Additive Increase, Multiplicative Decrease
- Slow Start
- Fast Retransmit
- Fast Recovery with partial ACKs

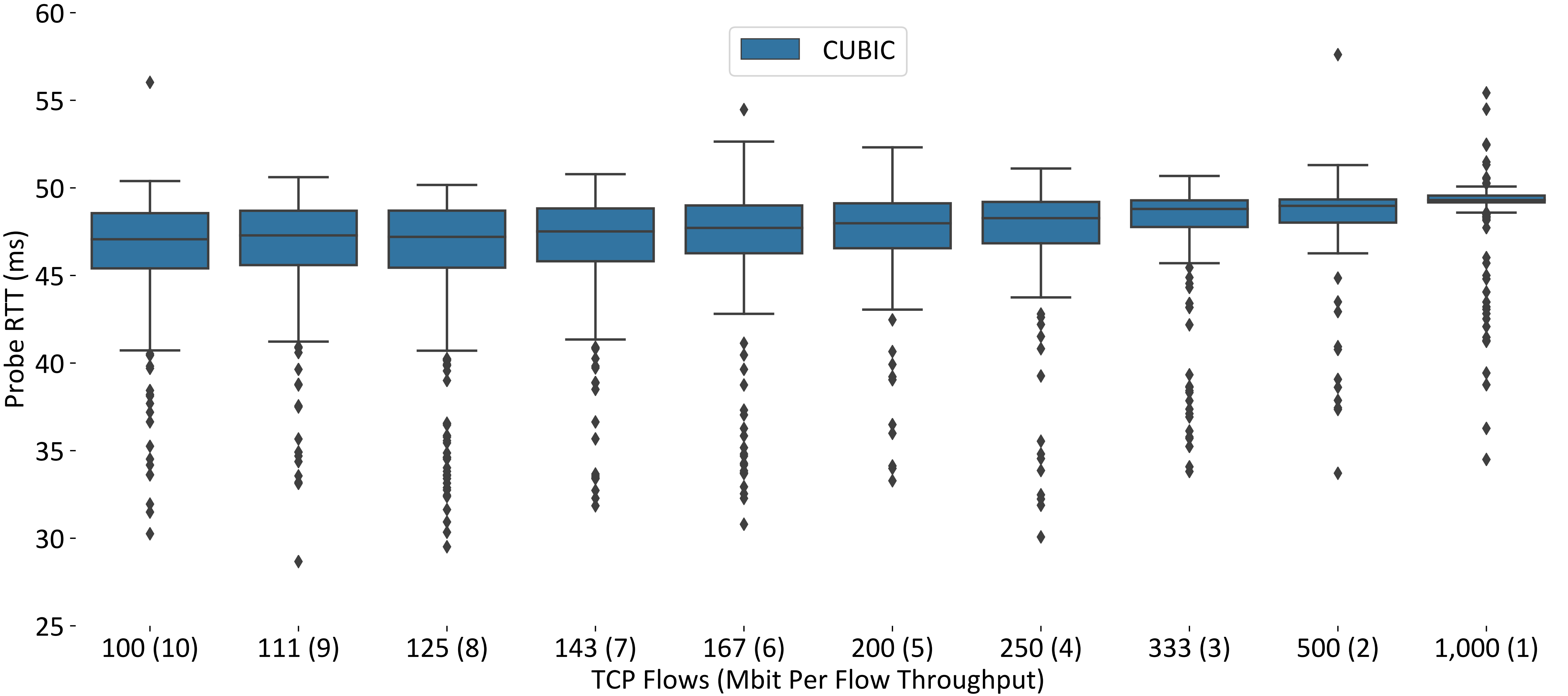
CUBIC

- Slow Start, Fast Retransmit, Fast Recovery
- Congestion Window increases follow a cubic function – quickly initially, but slows as it nears old window size
- Partially decouples window increases from RTT
- Default in current versions of Linux, MacOS, and Windows

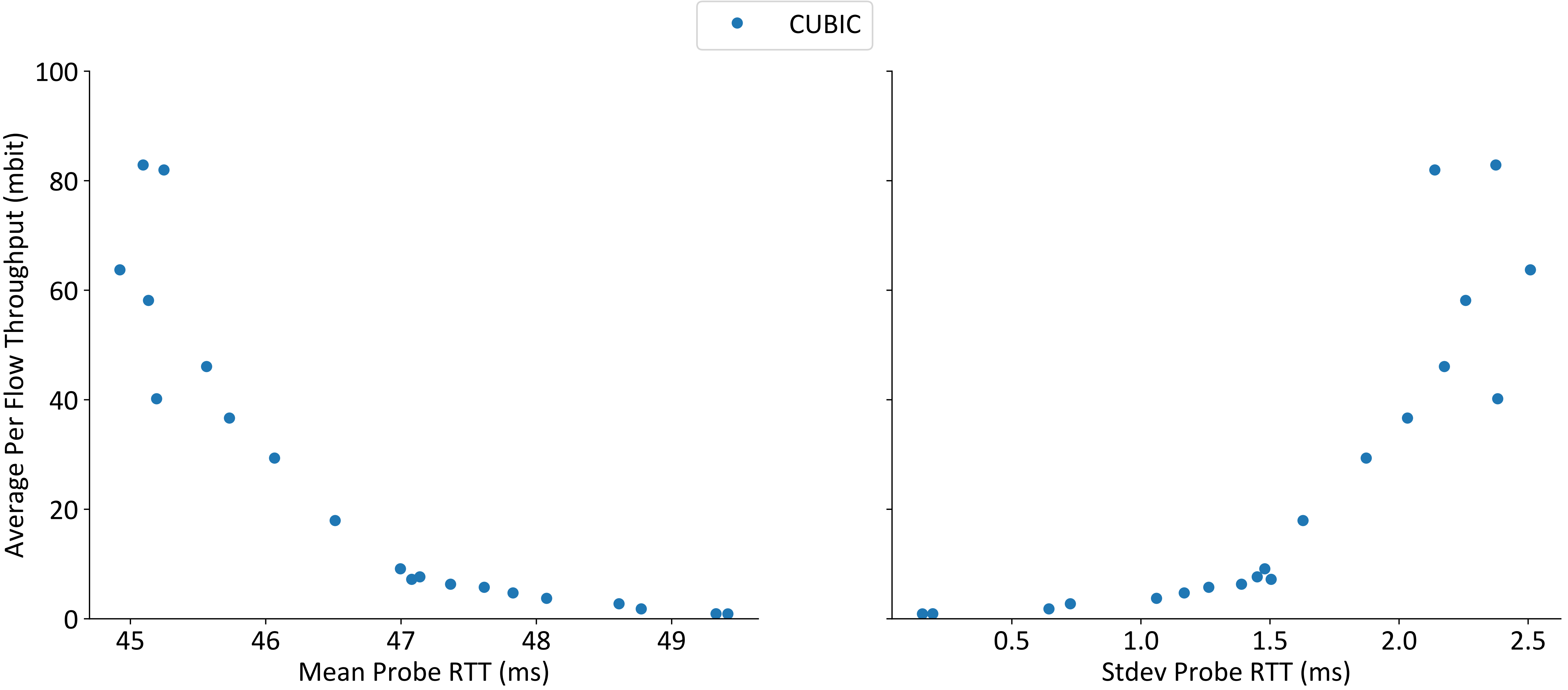
Initial Setup



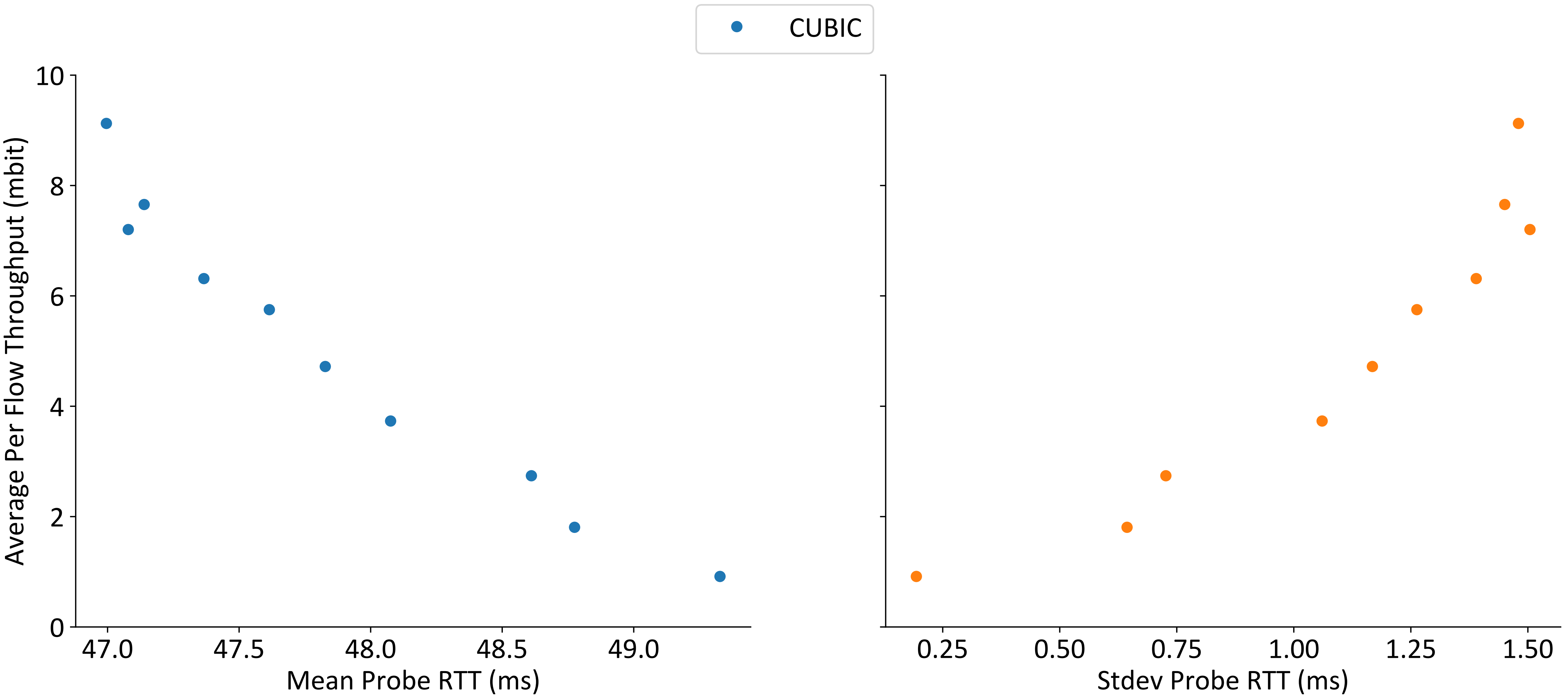
TCP CUBIC: Mean Probe RTT **Increases** and Spread **Decreases** as Per Flow Throughput **Decreases**



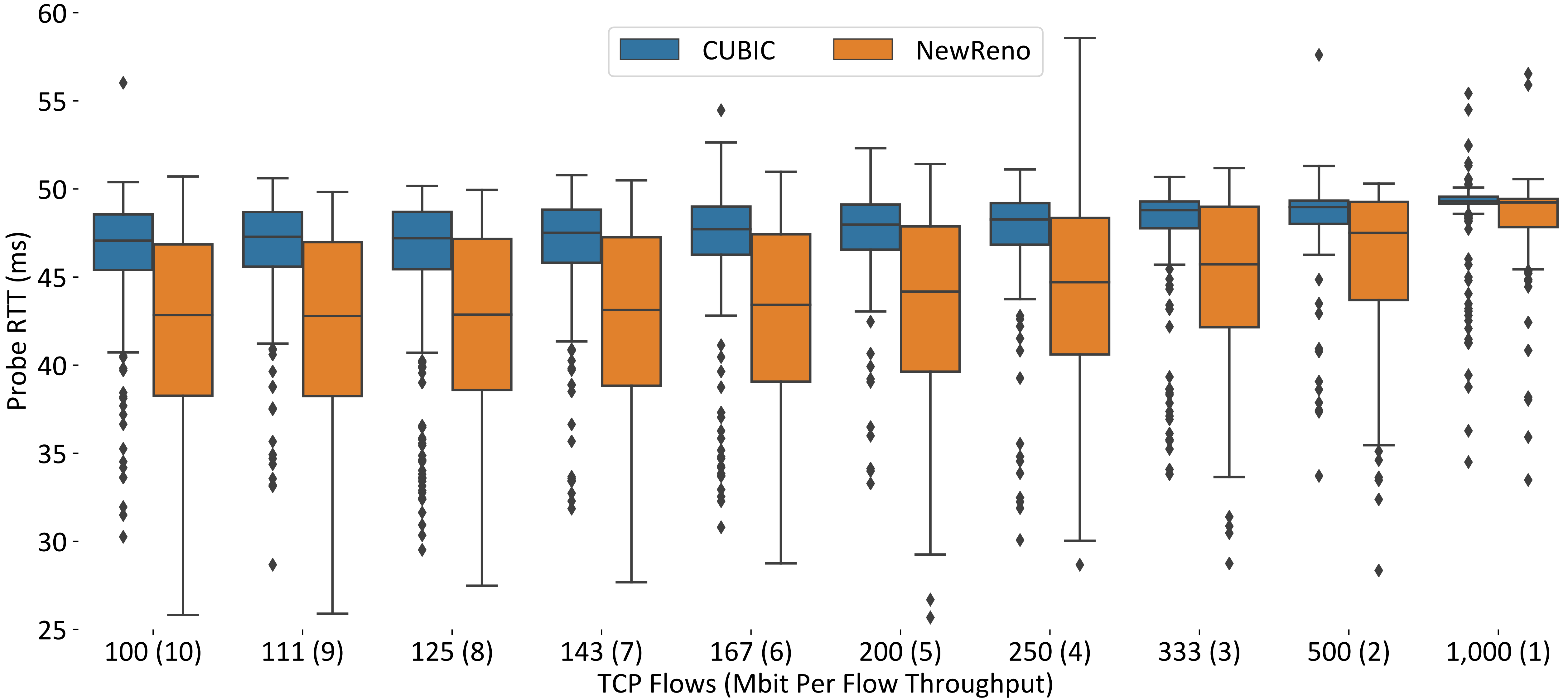
TCP CUBIC: **100mbit** (10 Flows) – **1mbit** (1000 Flows)



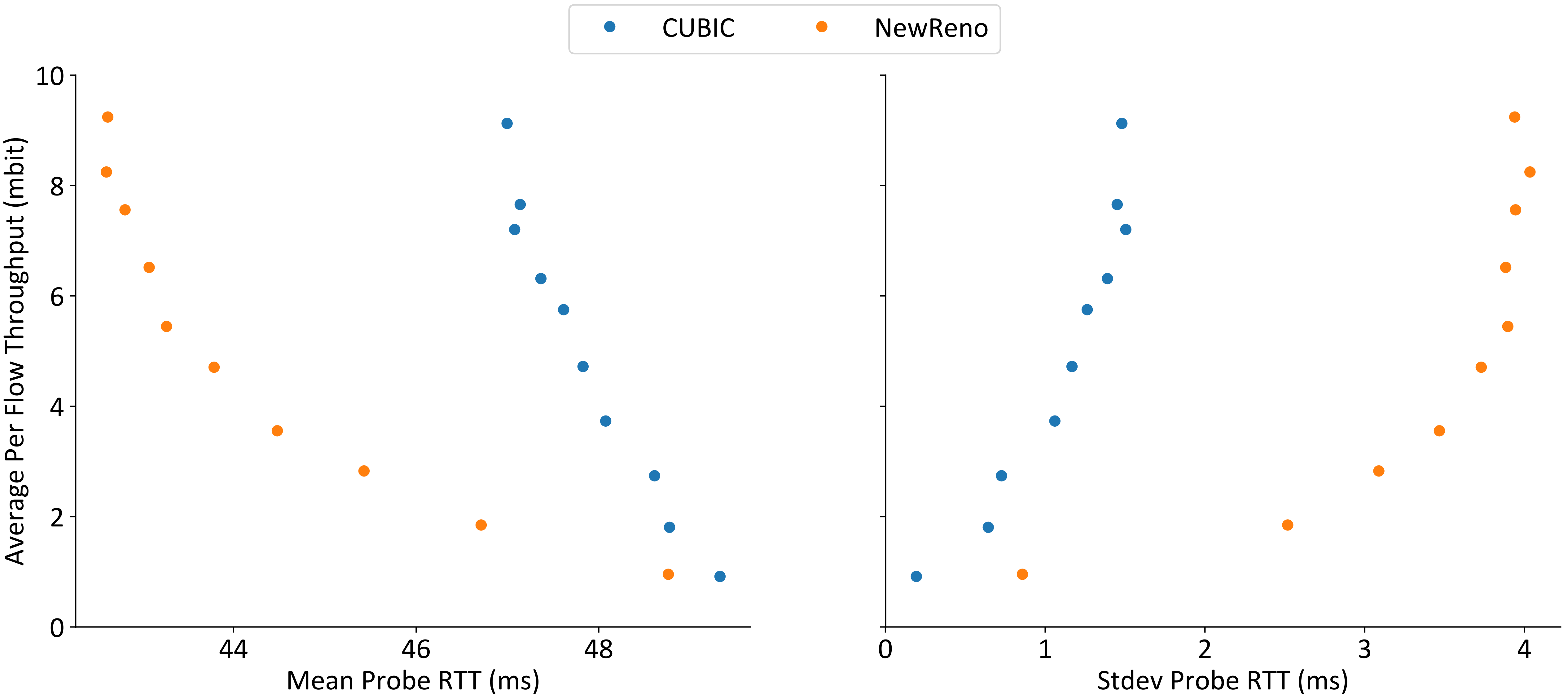
TCP CUBIC: **10mbit** (100 Flows) – **1mbit** (1000 Flows)



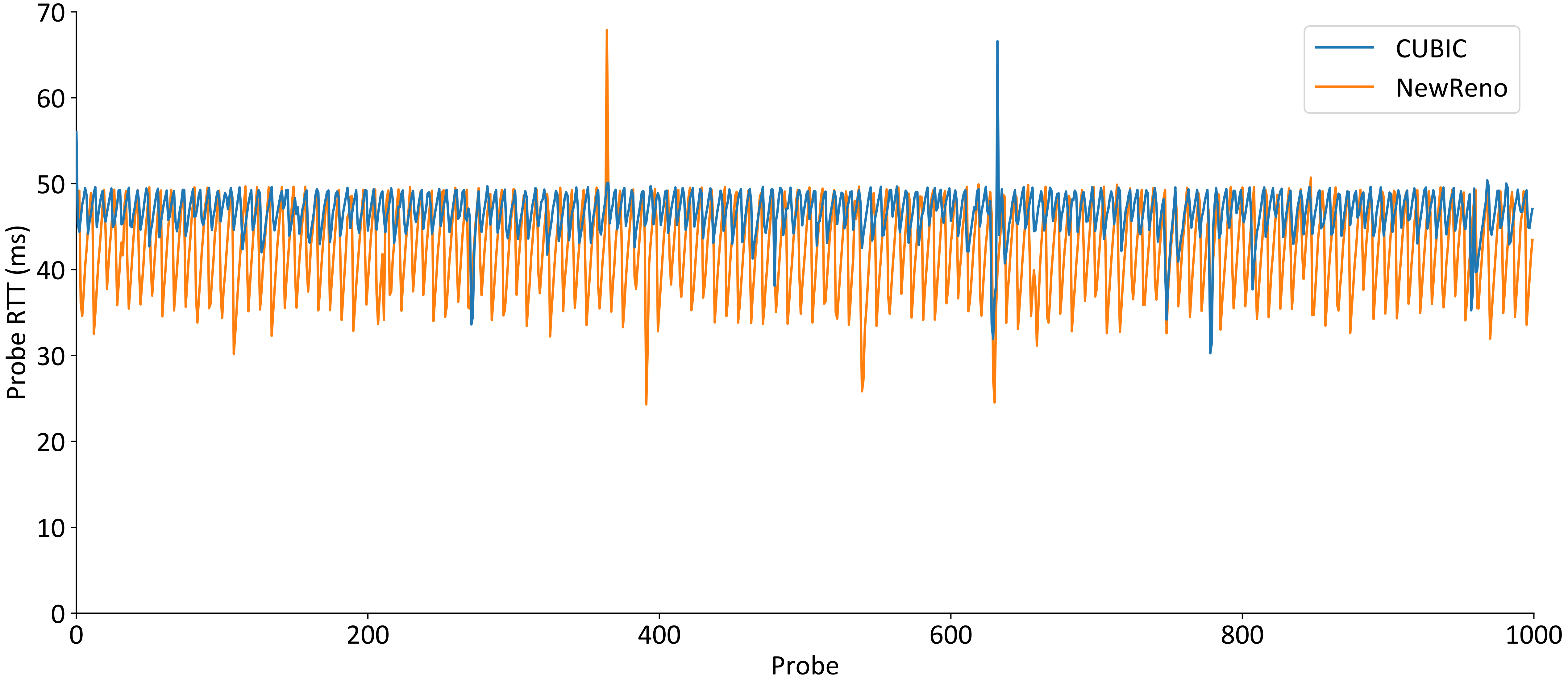
CUBIC vs NewReno: Mean and Spread are Different



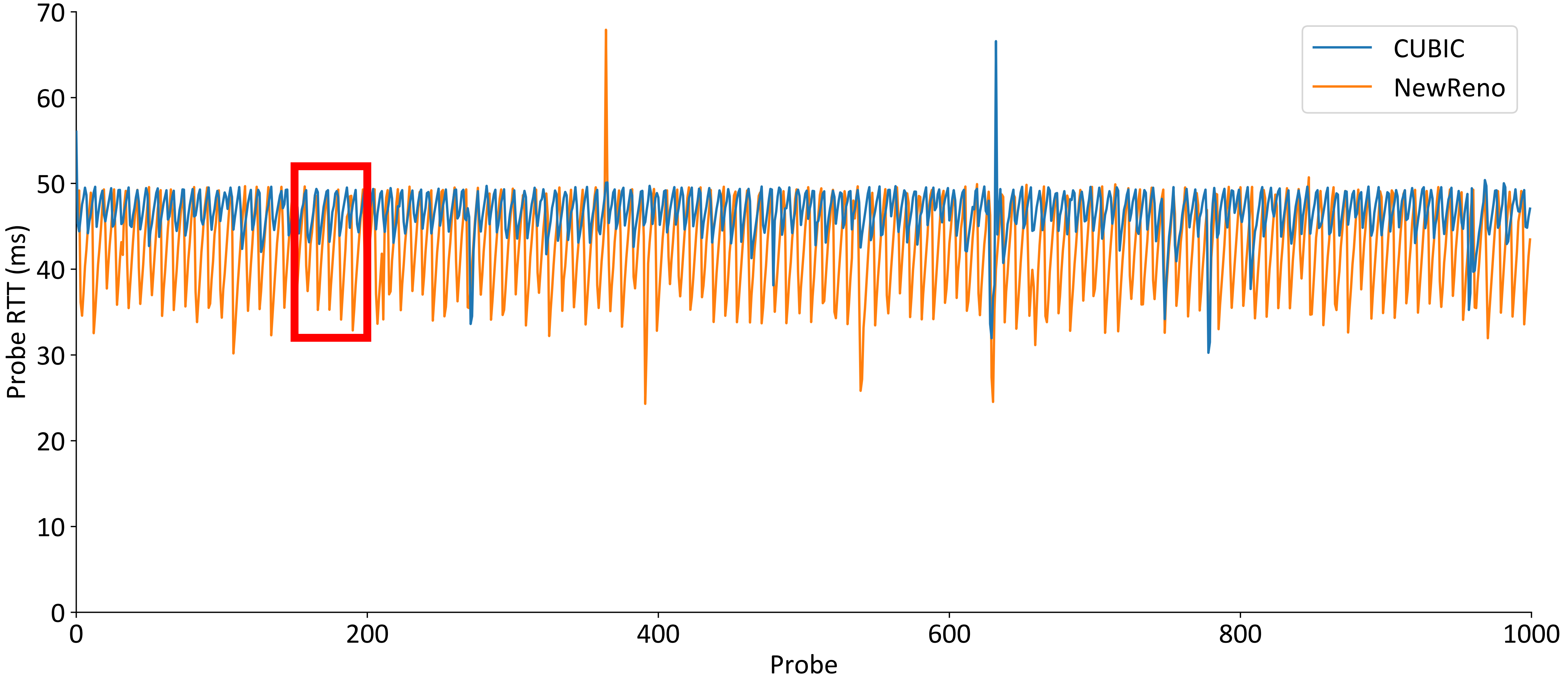
CUBIC vs NewReno: Model for CUBIC is Unusable for NewReno



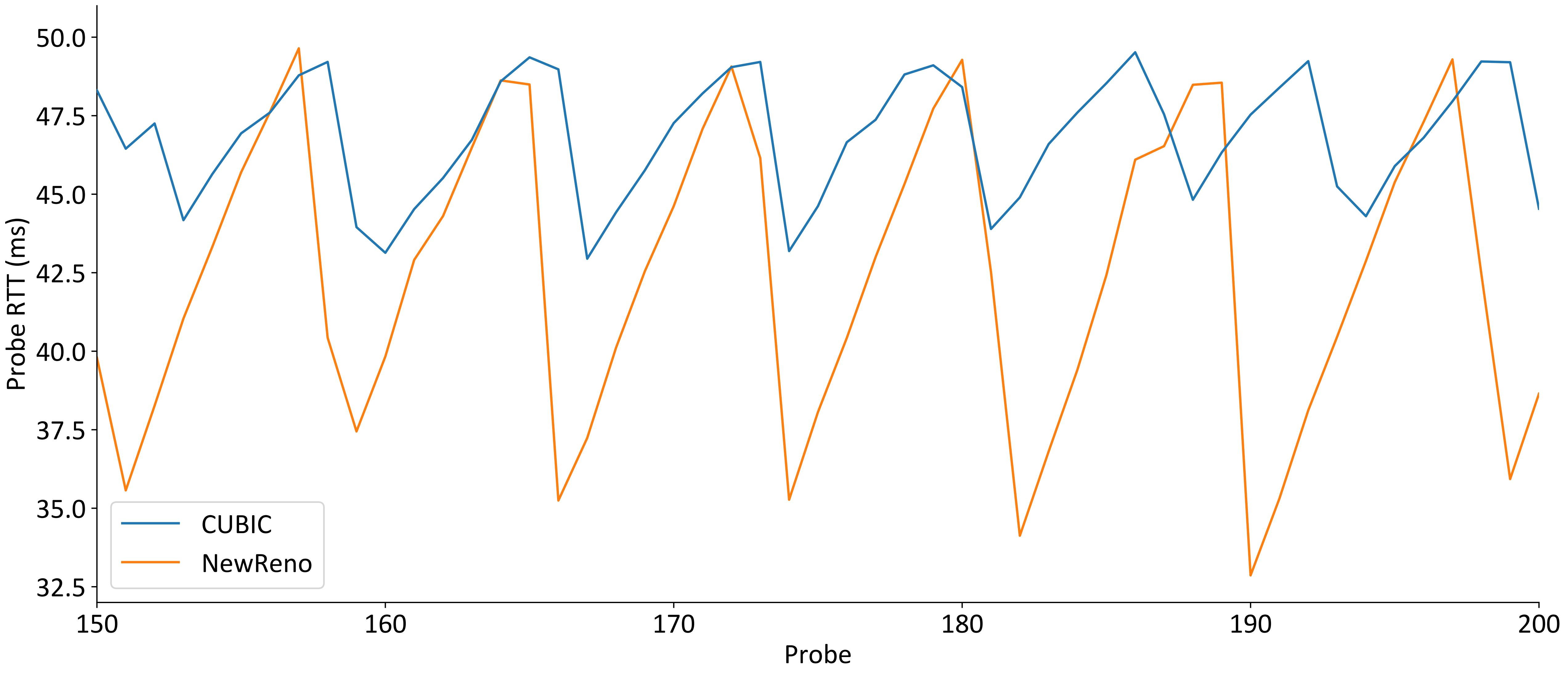
CUBIC vs NewReno: 1000 Probe RTTs Every 100ms



CUBIC vs NewReno: 1000 Probe RTTs Every 100ms



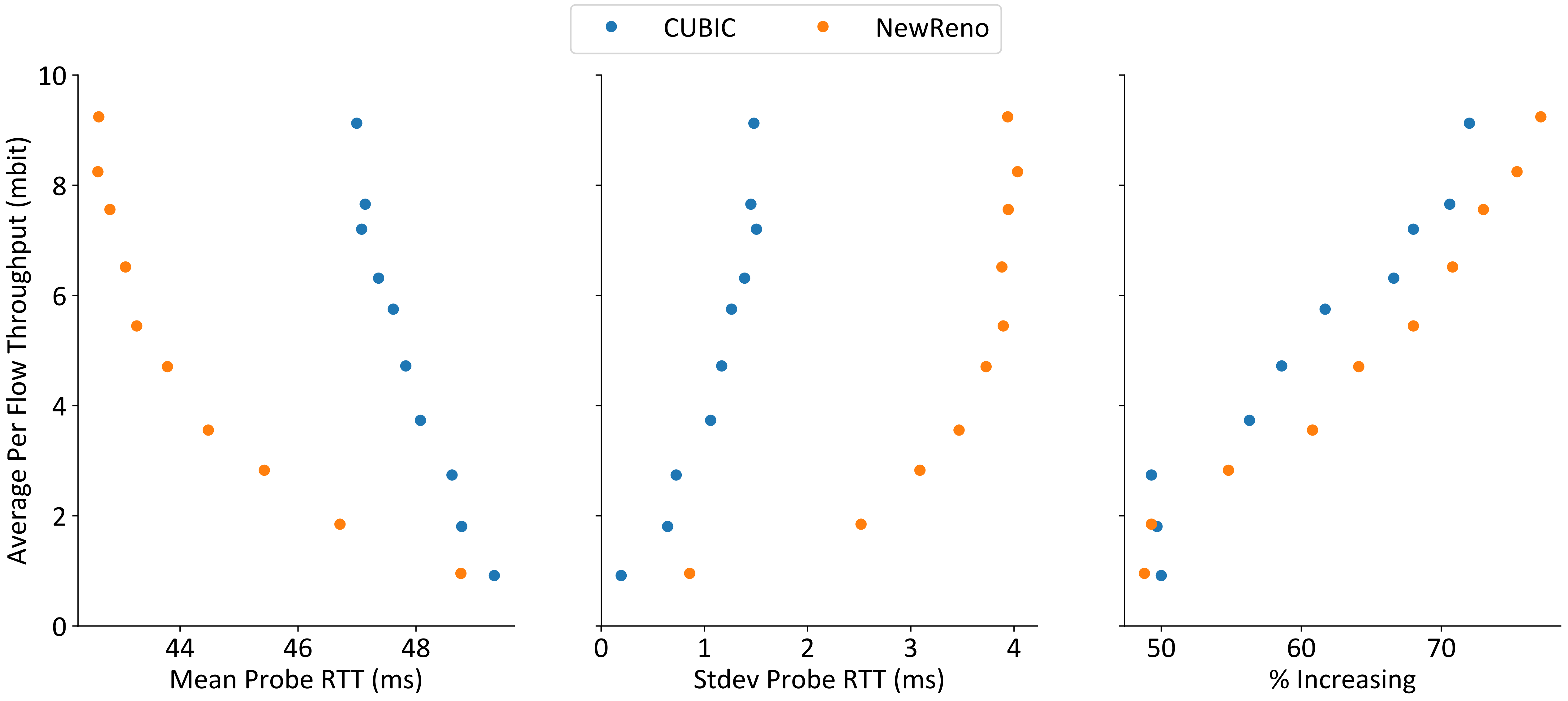
CUBIC vs NewReno: Probe RTTs Increase Slower Than Decrease



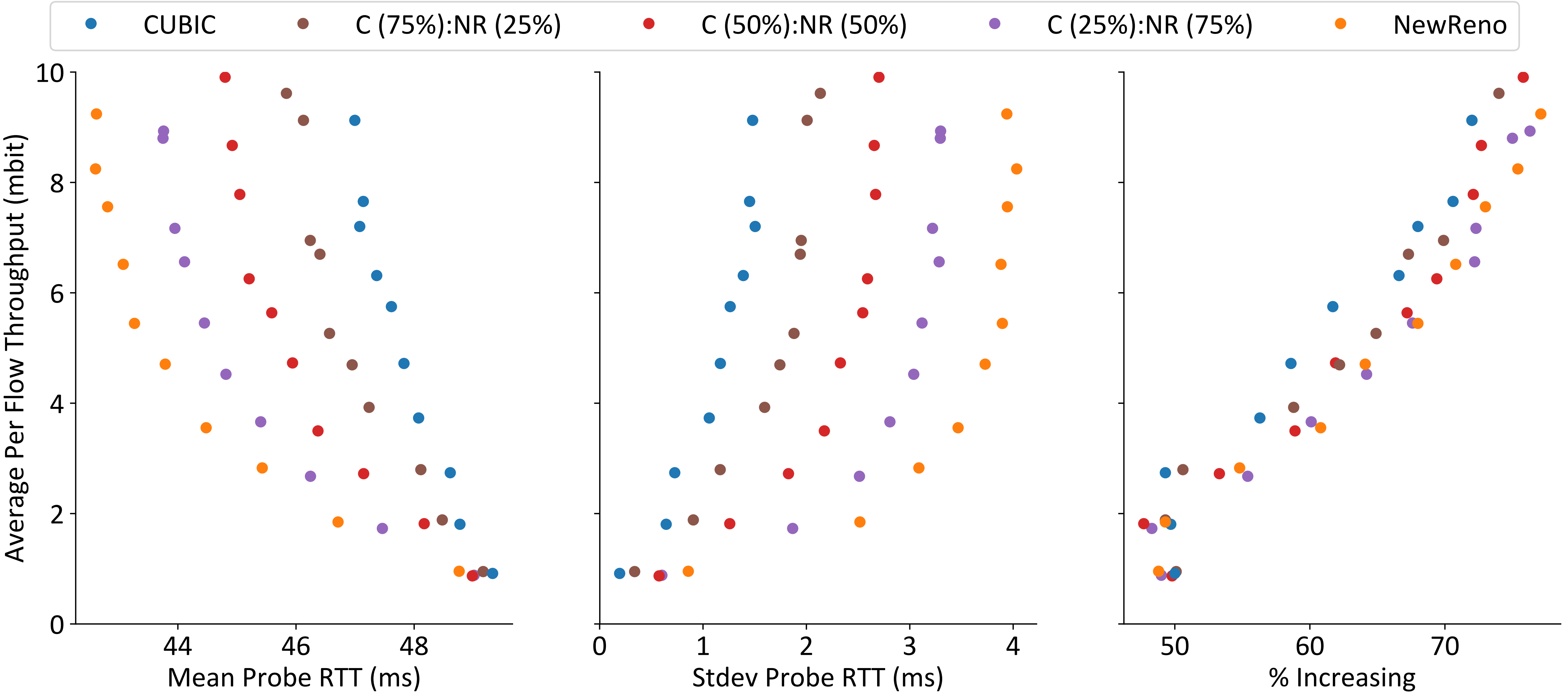
Percent Increasing Metric

- Percentage of Probe RTTs where $\mathbf{RTT_i > RTT_{i-1}}$
- Attempt to capture rate of queue increases vs decreases
- Example:
    - 10 RTTs = [44, **46**, **48**, 43, **45**, 44, **47**, 42, **45**, **48**]
    - 6 RTTs are greater than previous RTT
    - Percent Increasing = 60%

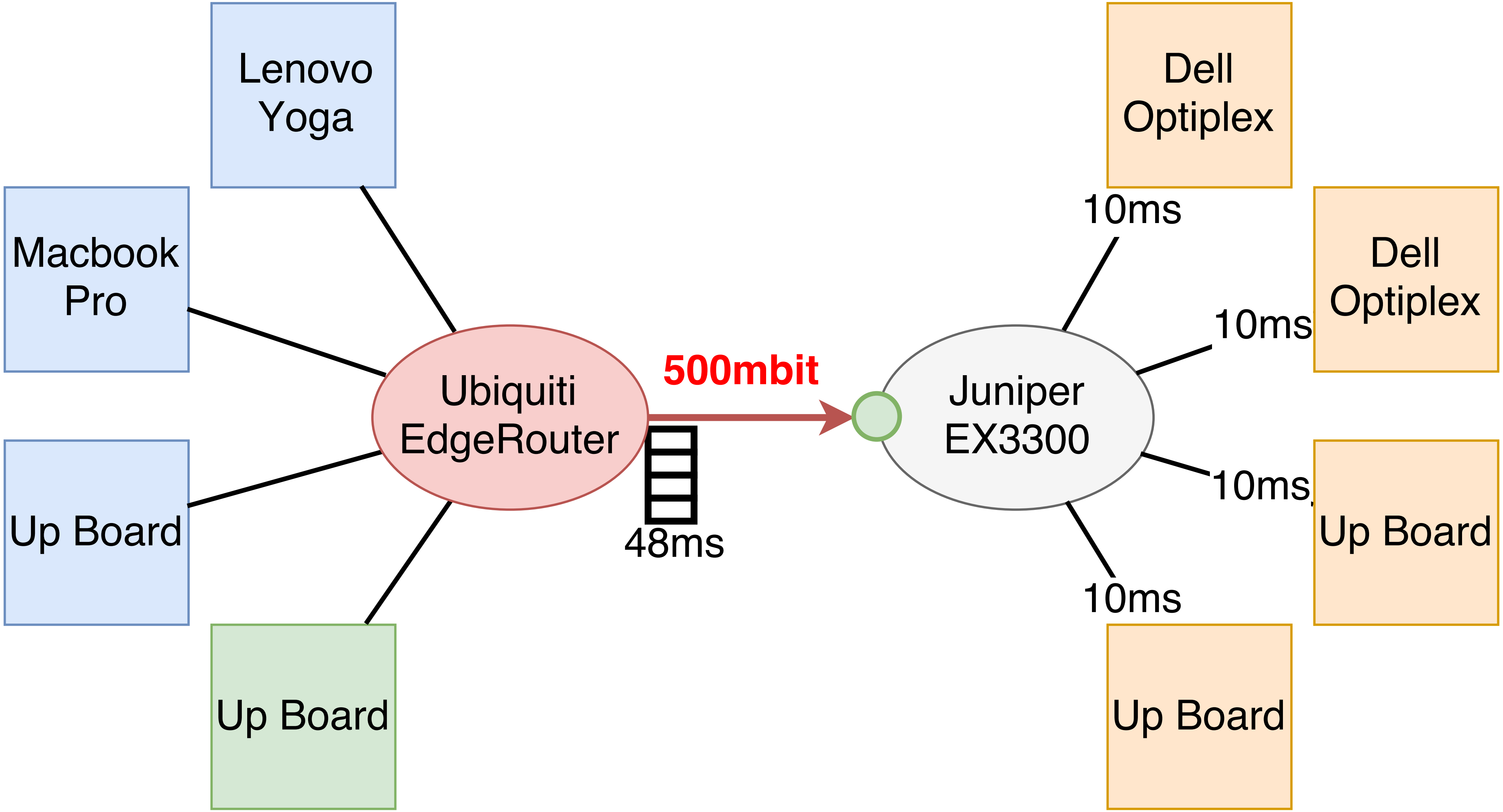
CUBIC vs NewReno: Percent Increasing Metric Reduces Potential Estimation Error (≈ 2Mbit)



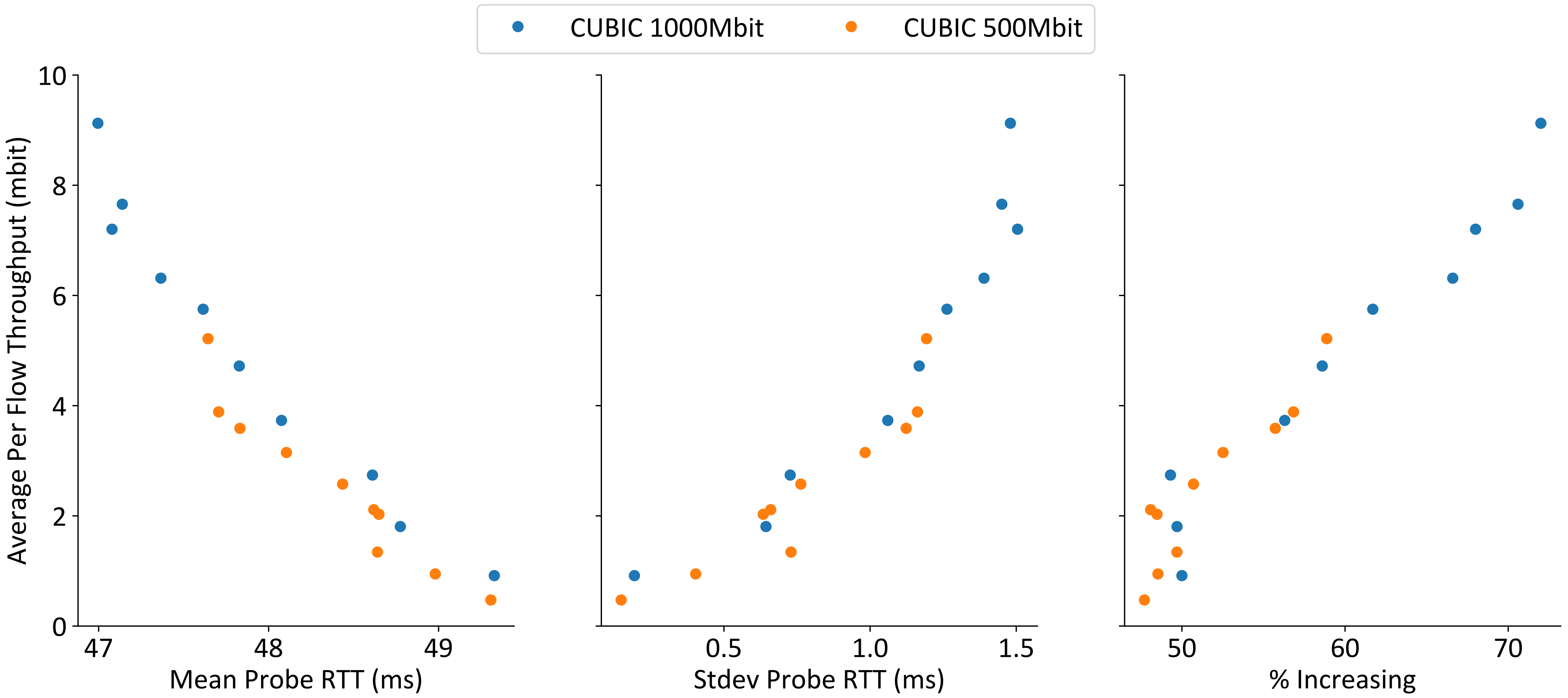
CUBIC & NewReno Mixes: All Fall Between CUBIC and NewReno Curves



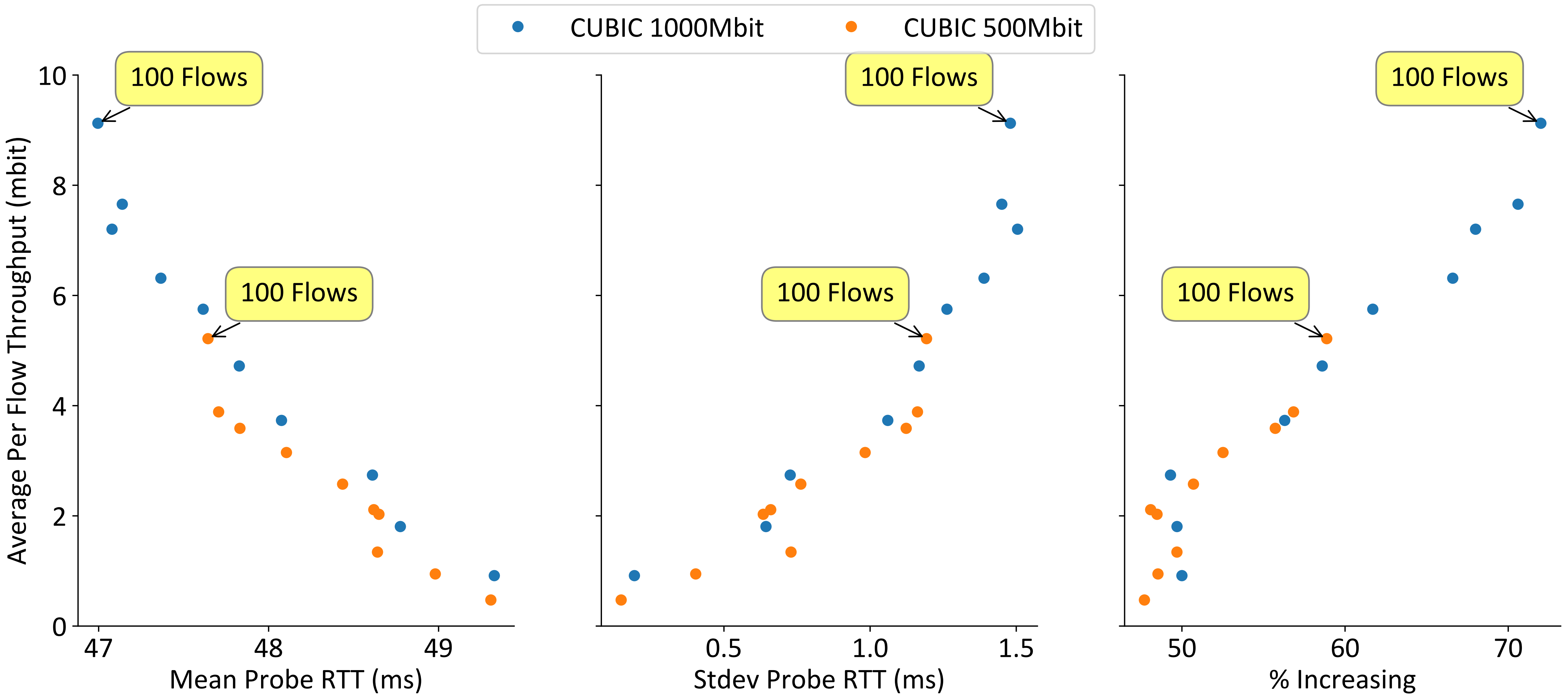
Bandwidth: Reduce Bandwidth to 500Mbit



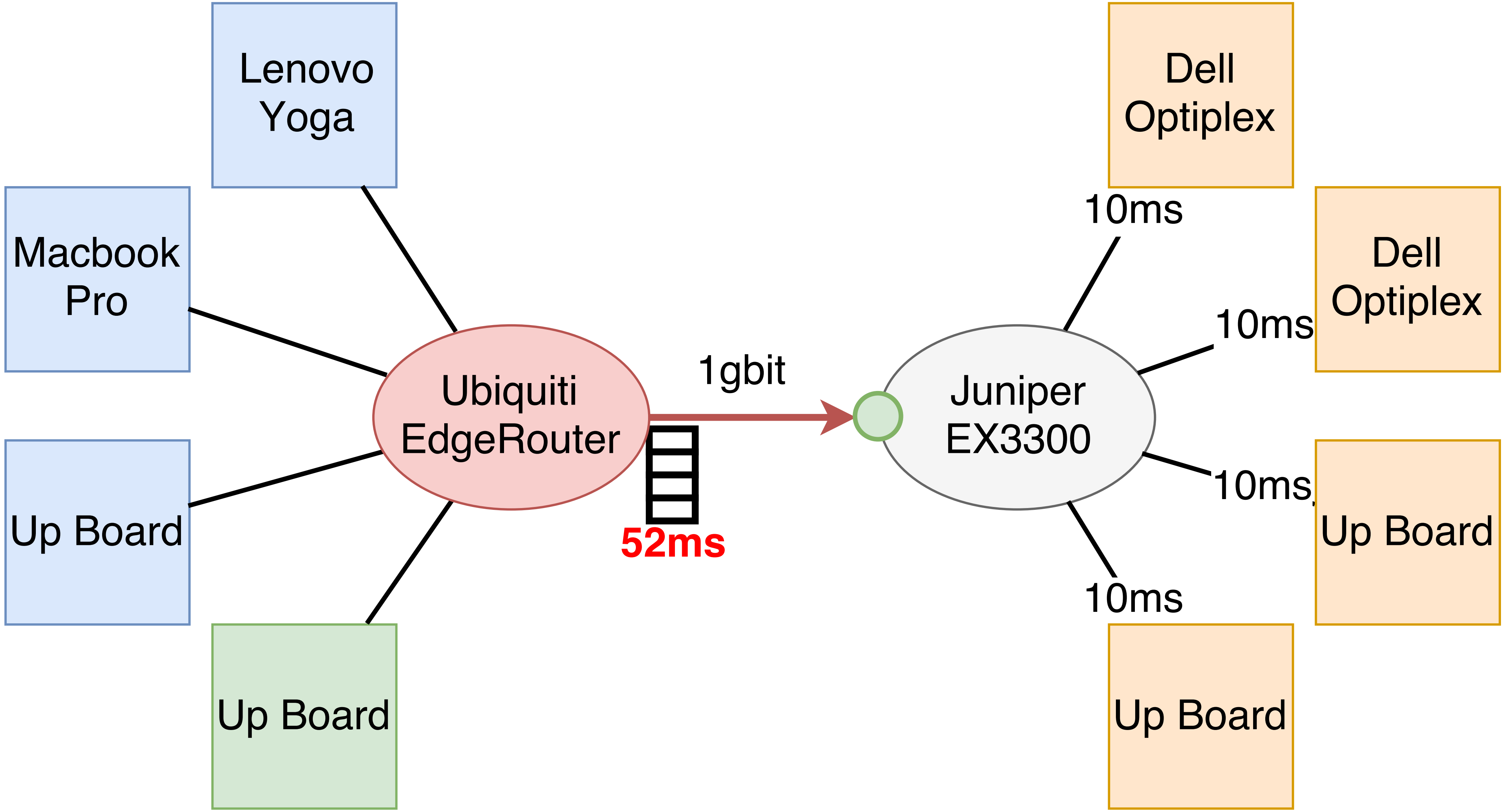
Bandwidth: Measuring Raw Average Throughput



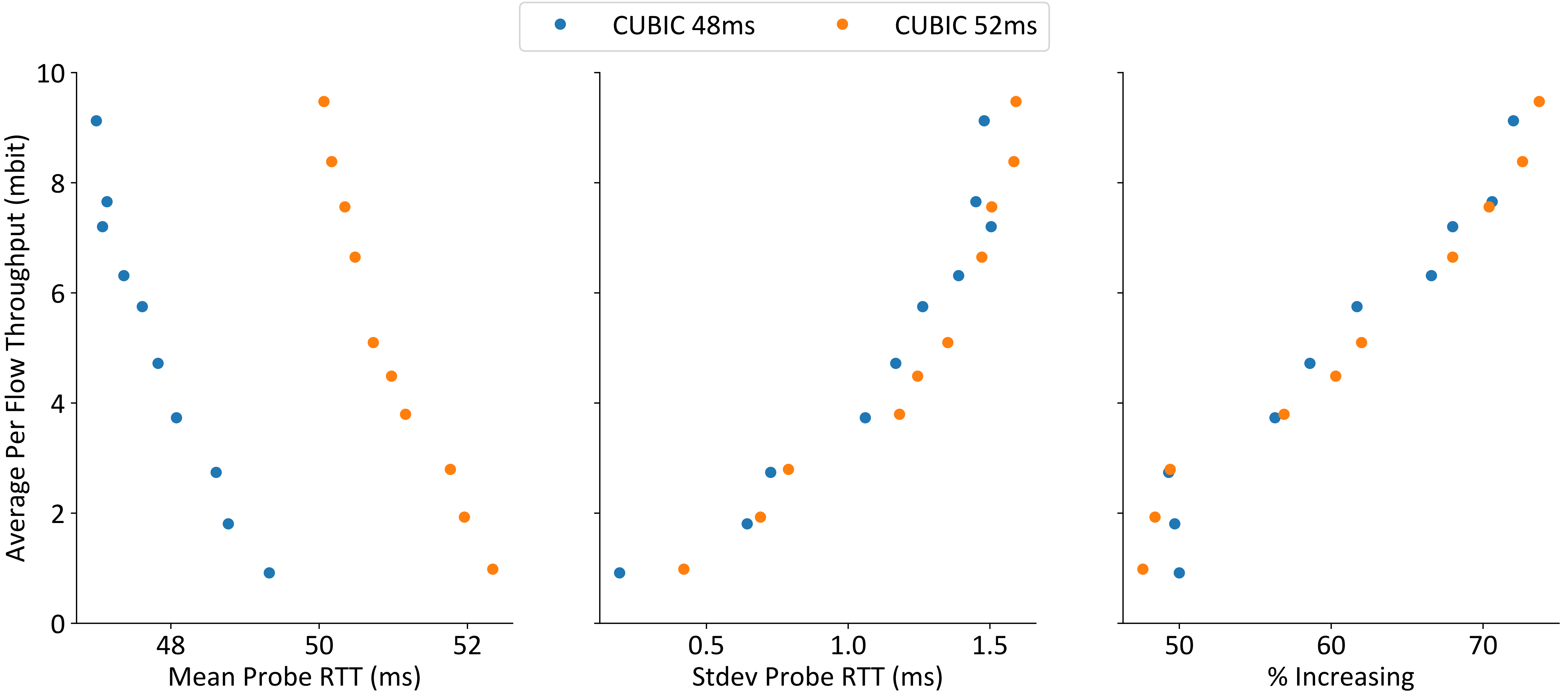
Measurements Are Independent of the Number of TCP Flows



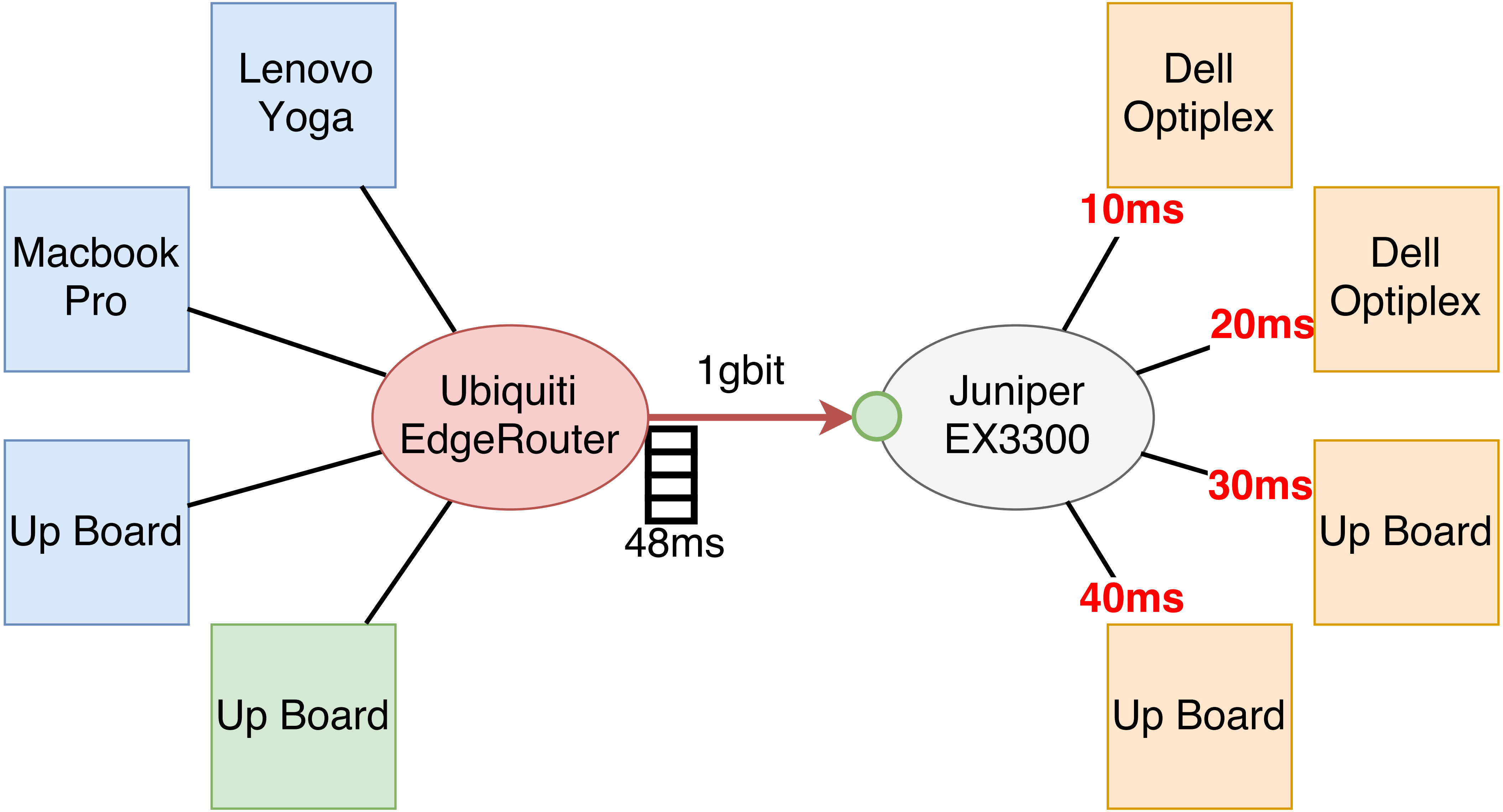
Queue Depth: Increase By 4ms (From 48ms to 52ms)



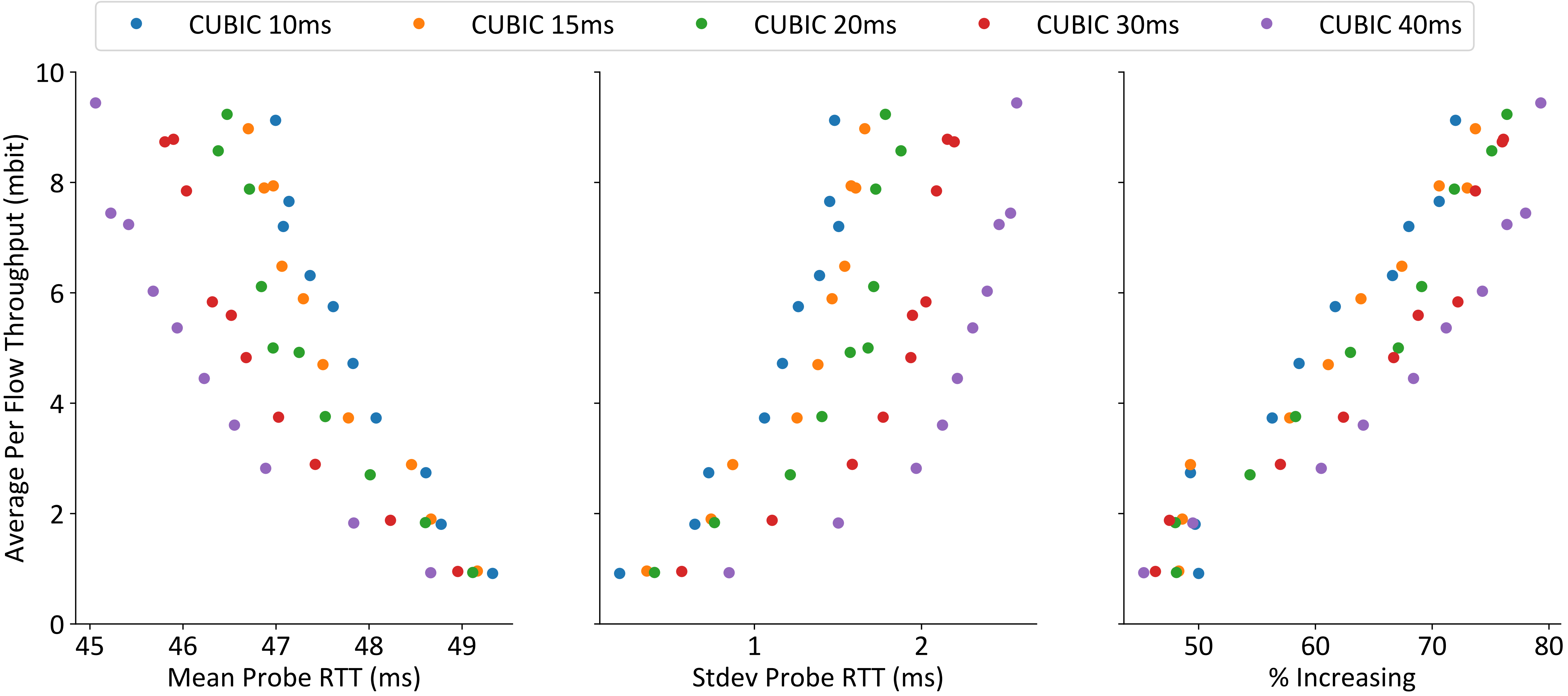
Queue Depth: Stdev and % Increasing Are Resilient to Small Differences, Mean is Not



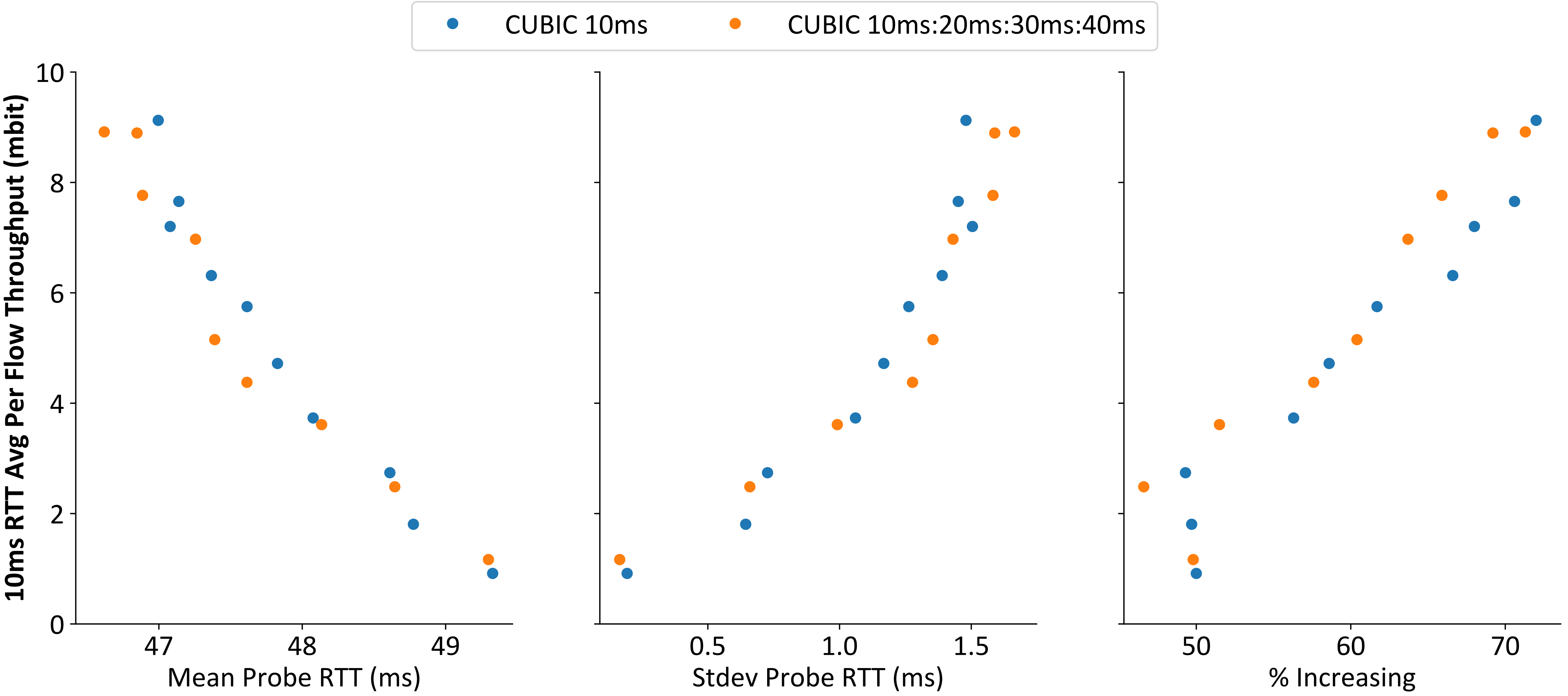
TCP RTT: Impact of Different RTTs



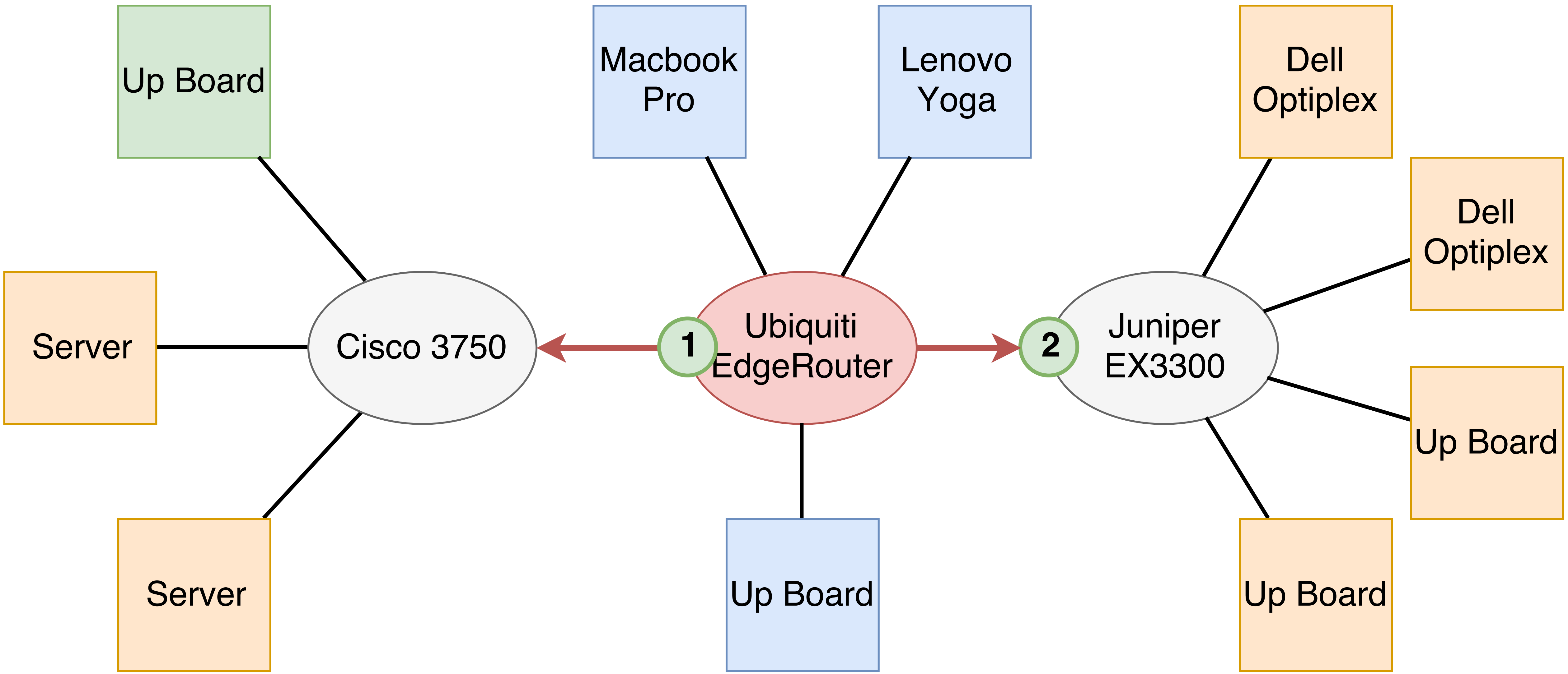
TCP RTT: Percent Increasing Estimation Error Based on RTT Assumption



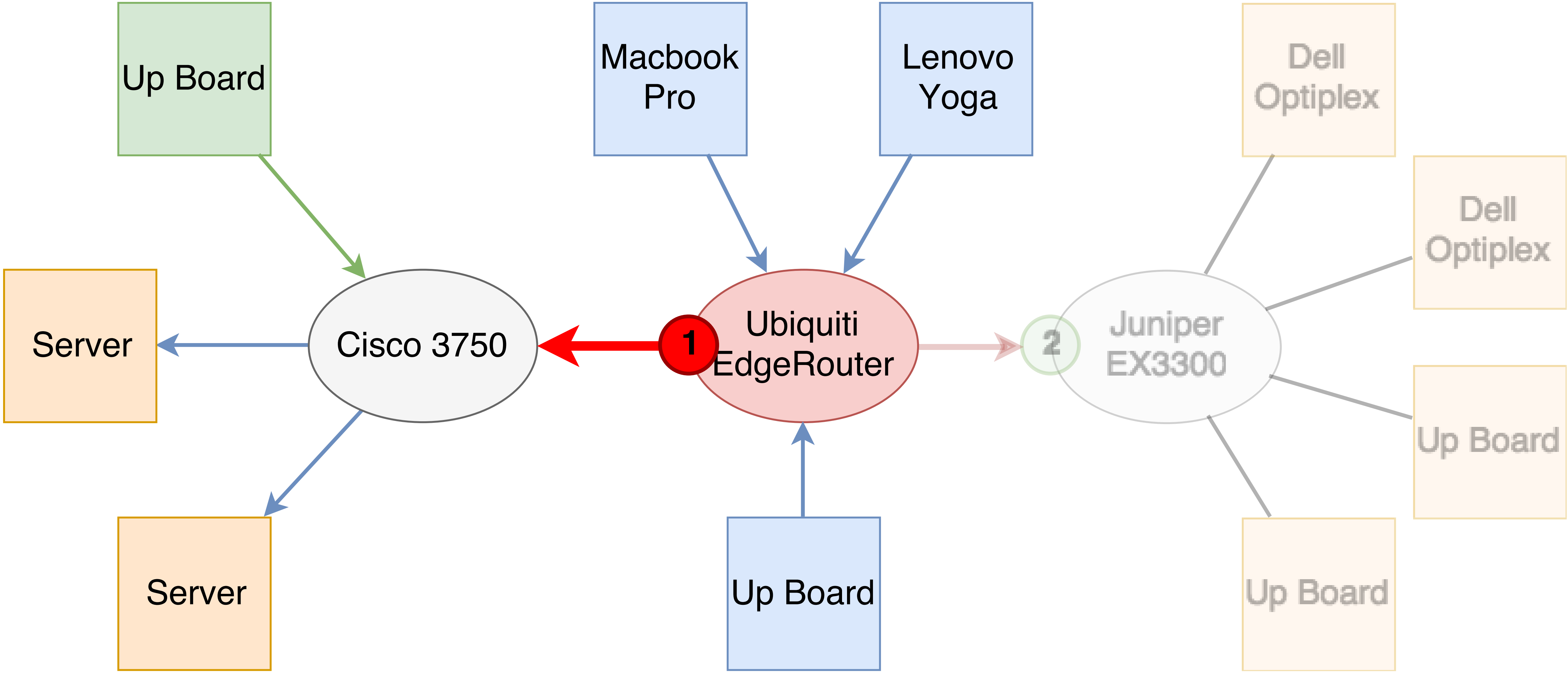
TCP RTT: Probe RTTs Measure Throughput of Smallest TCP RTT Flows



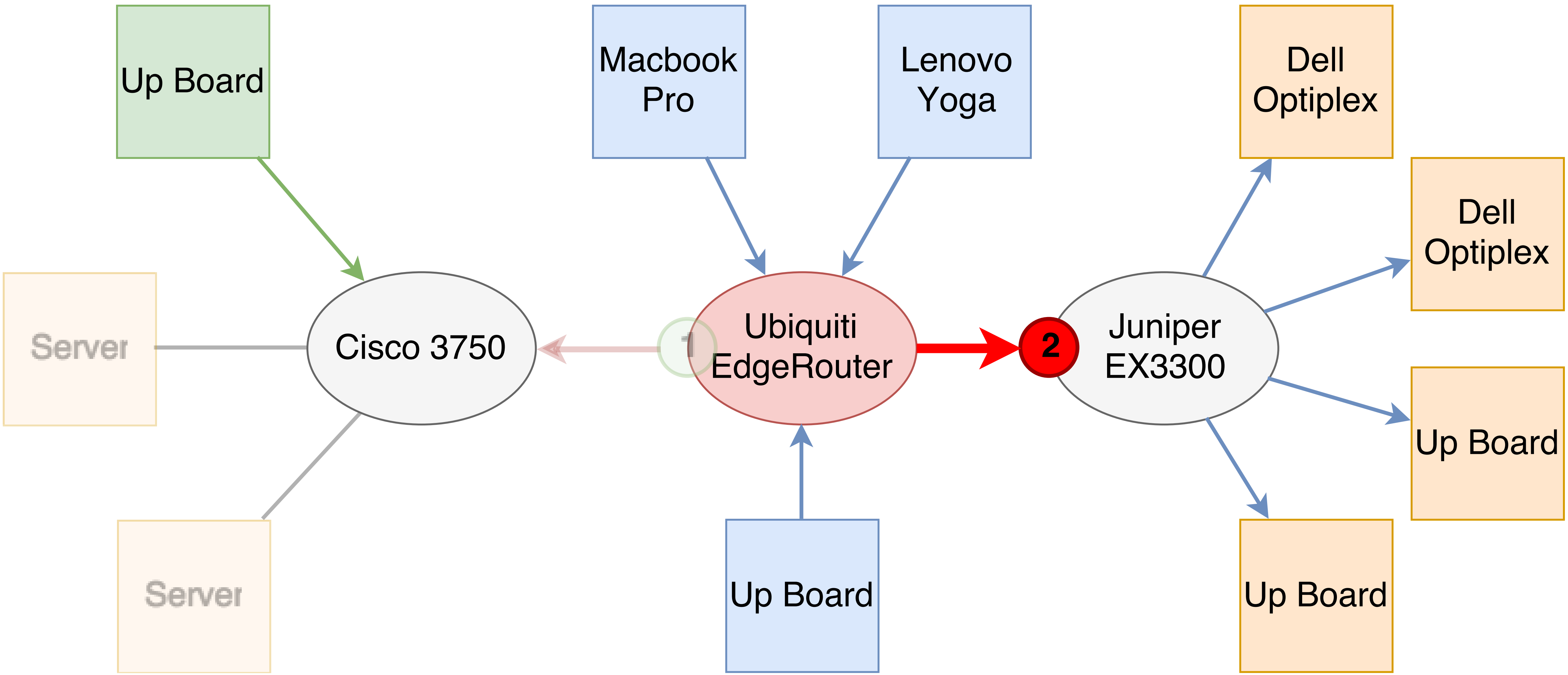
Probing Through Congestion



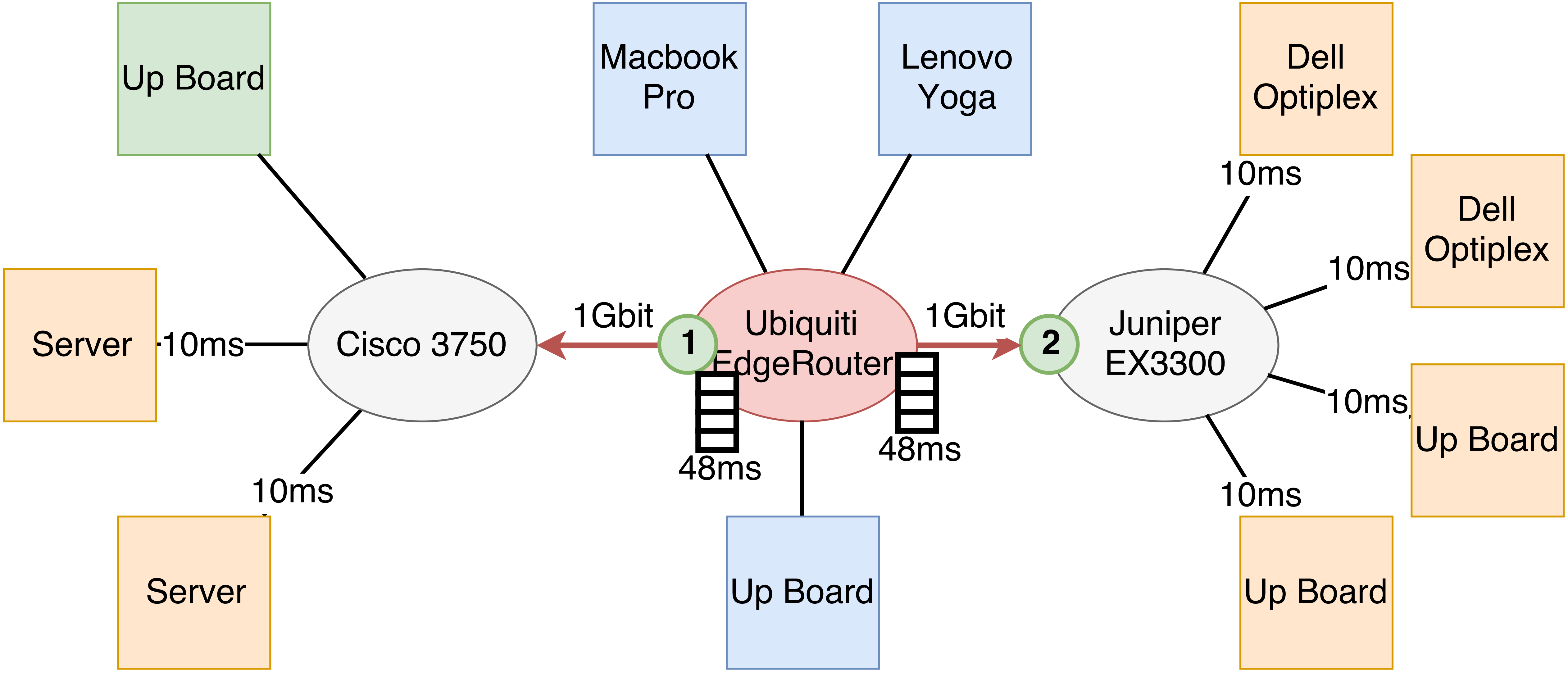
1st Link: Reverse Path Congestion



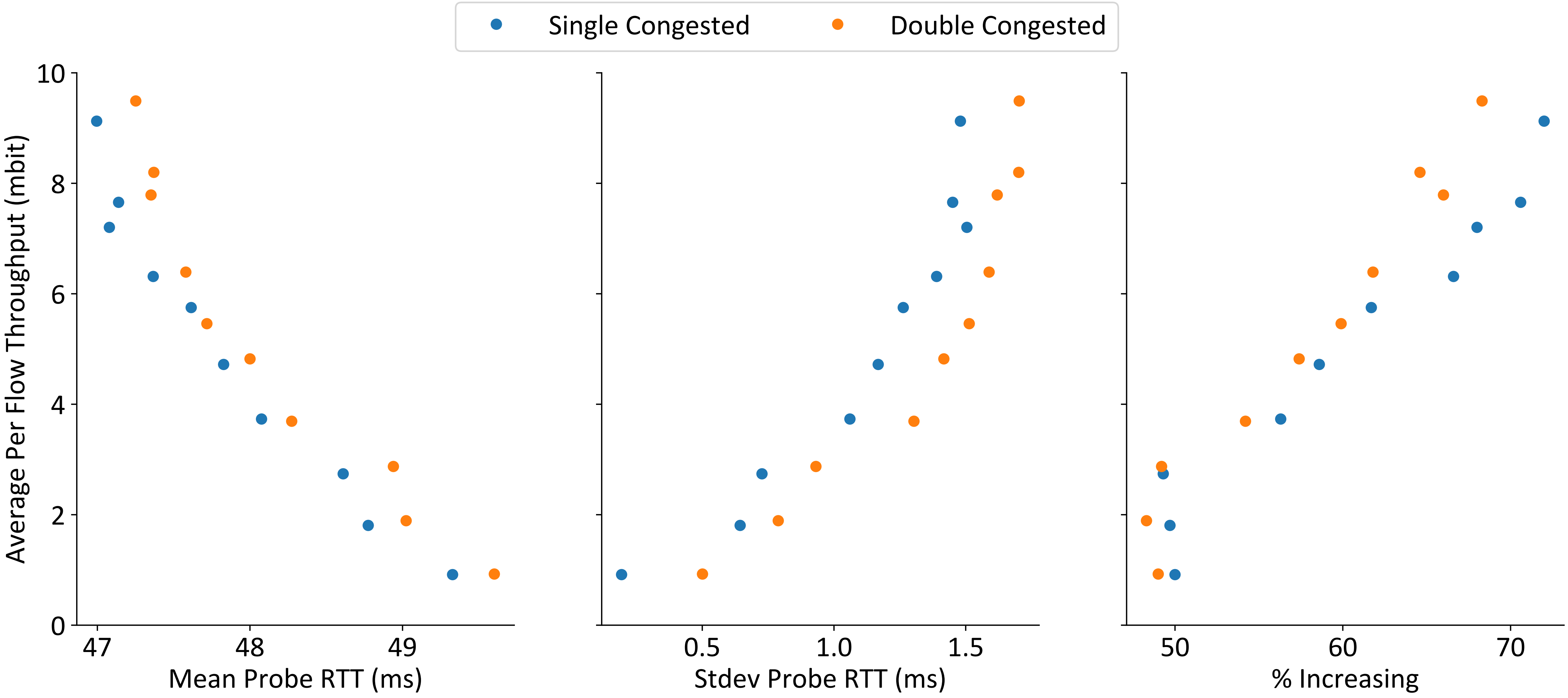
2nd Link: Forward Path Congestion



Probing Through Congestion



Probing Through Congestion: Looks Possible



Conclusions & Future Work

- Where it works:
  - CUBIC, NewReno, mixed
  - Bandwidth
  - Queue depth
  - Assumed TCP RTT distribution
- Hopefully soon:
  - Reduce error due to TCP RTT
  - Probing through congestion
- New experiments:
  - BBR
  - Higher bandwidths (10+ Gbit)
  - Throughput fluctuations