# CapProbe: Inexpensive and Accurate Estimation of Narrow Link Capacity

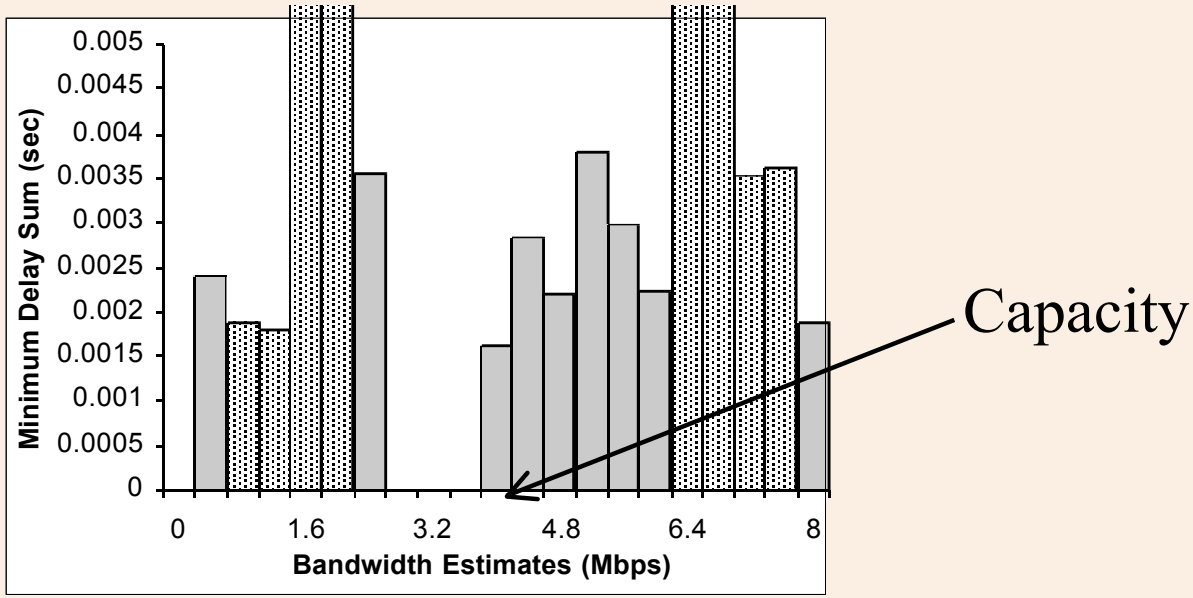
Rohit Kapoor
Ling-Jyh Chen
M. Y. “Medy” Sanadidi
Mario Gerla

# CapProbe: The Main Idea

- Observation: Both *expansion* and *compression* of dispersion involve *queuing due to cross traffic:*
  - Dispersion expansion => *second packet queued*
  - Dispersion compression => *first packet queued*
- Packet pair with *minimal end-to-end delay sum,* is likely to be dispersed corresponding to narrow link capacity
- Looking for packet pair with minimal delay sum is inexpensive
- CapProbe appears accurate in most of our experiments, simulations and measurements
- CapProbe fails under *heavy* (~>75%) utilization by *non-responsive* (UDP) traffic

# Preliminary Simulation Results

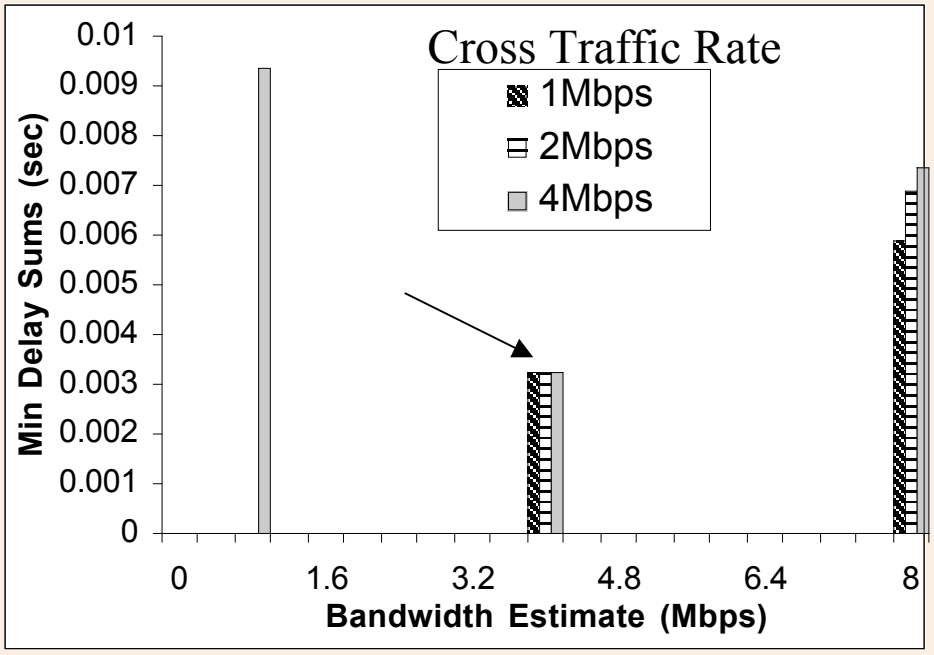
- A packet pair provides two pieces of information
  - *Dispersion* between the two packets
  - End-to-end/Round trip *delay* of each packet
- CapProbe combines both pieces of information
  - Calculate delay sum for each packet pair sample
  - *Dispersion* at *minimum delay sum* reflects capacity



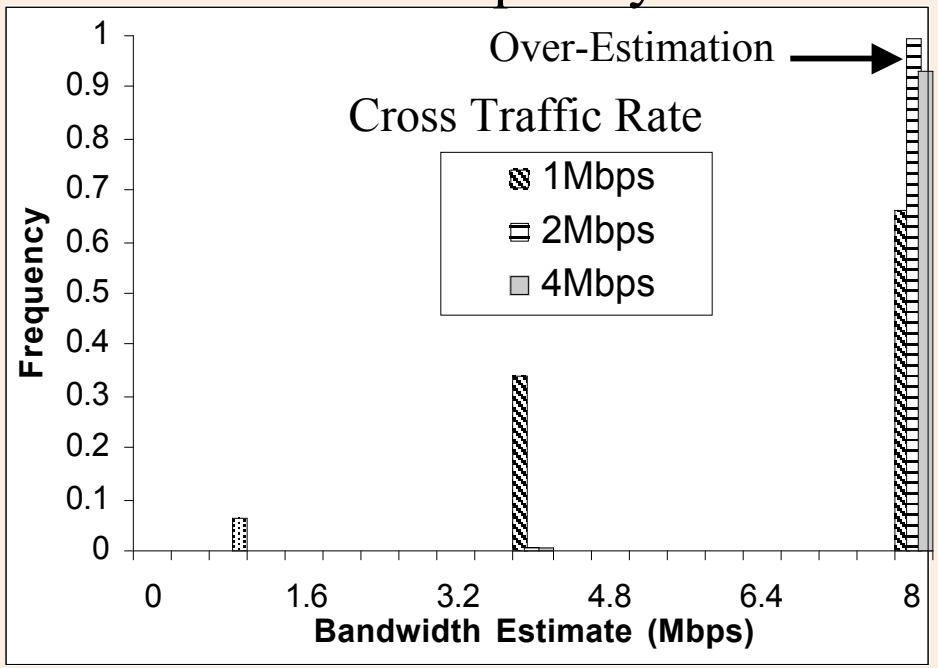
# CapProbe Filtered the Compression

- 6-hop path: capacities {10, 7.5, 5.5, 4, 6, 8} Mbps
- PP pkt size = 200 bytes, CT pkt size = 1000 bytes
- Path-Persistent TCP Cross-Traffic

Minimum Delay Sums



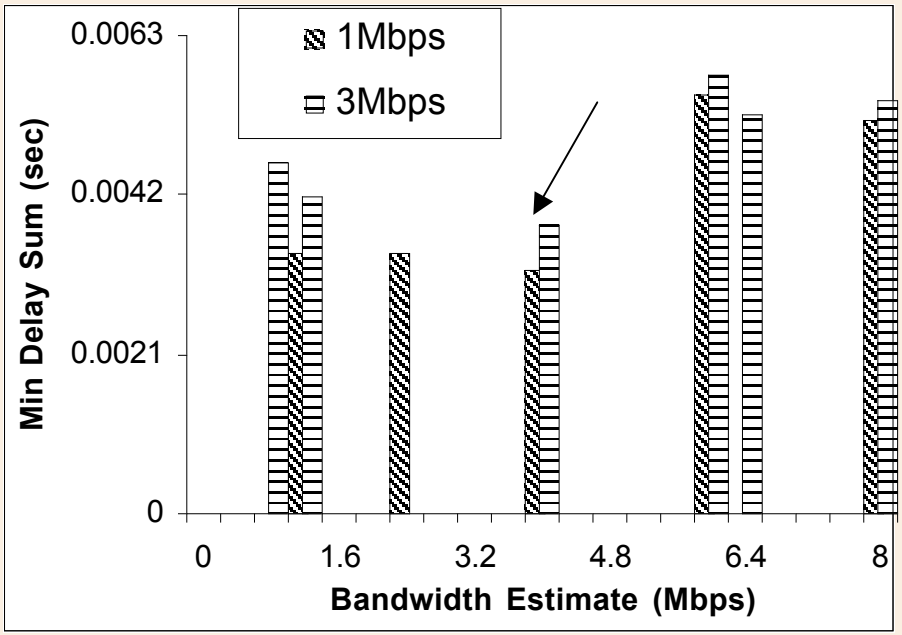

Bandwidth Estimate Frequency



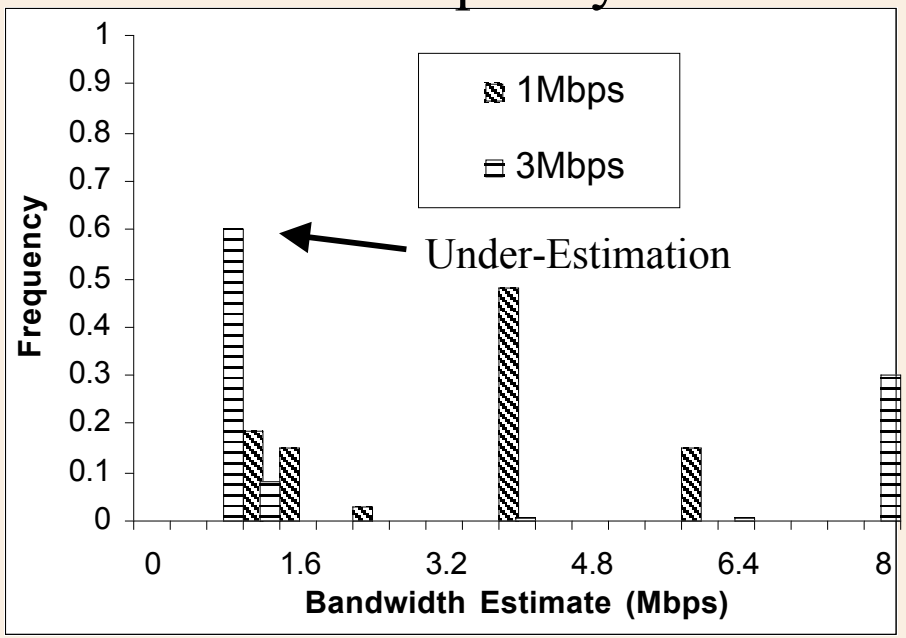
# CapProbe Filtered the Expansion

- PP pkt size = 500 bytes, CT pkt size = 500 bytes
- Non-Path-Persistent TCP Cross-Traffic

Minimum Delay Sums



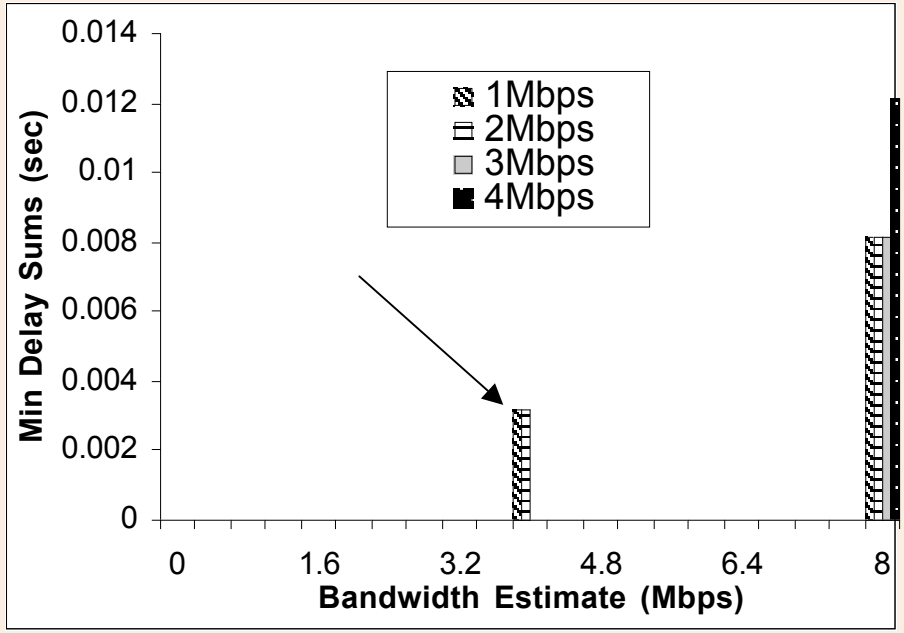

Bandwidth Estimate Frequency



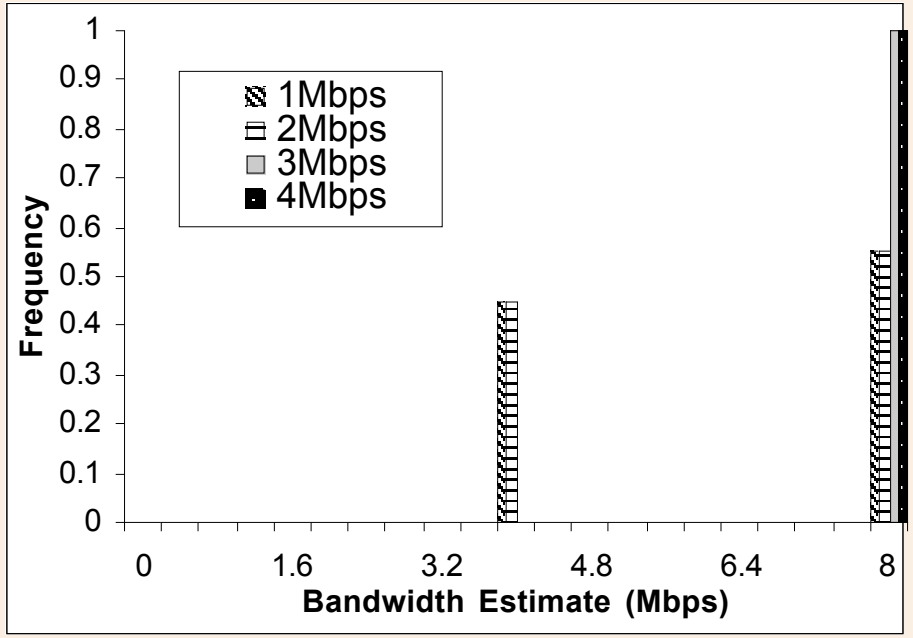
# CapProbe Failed Under Intensive Non-Responsive or Deterministic Cross Traffic

- Non-Path-Persistent UDP Cross Traffic

Minimum Delay Sums

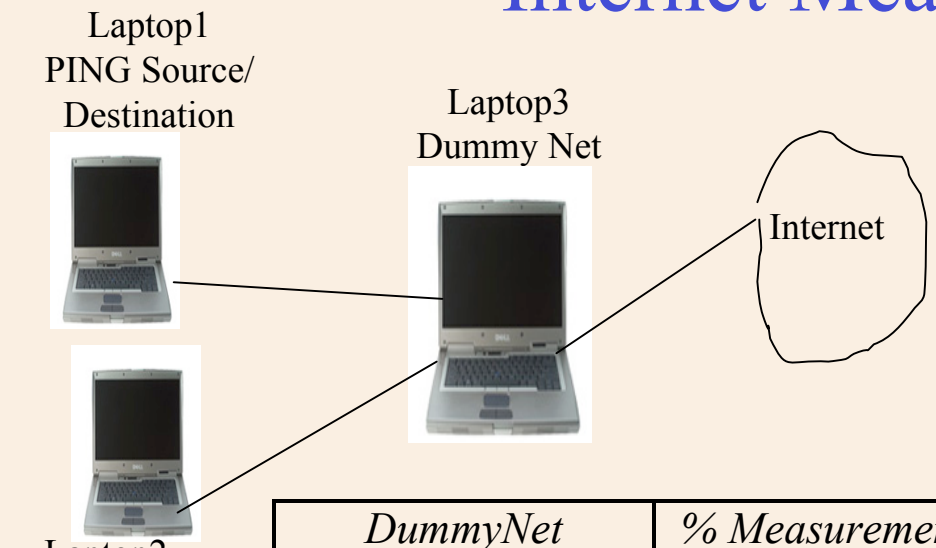



Estimate Frequency




- Only case where CapProbe does not work: Intensive UDP; No correct sample is obtained

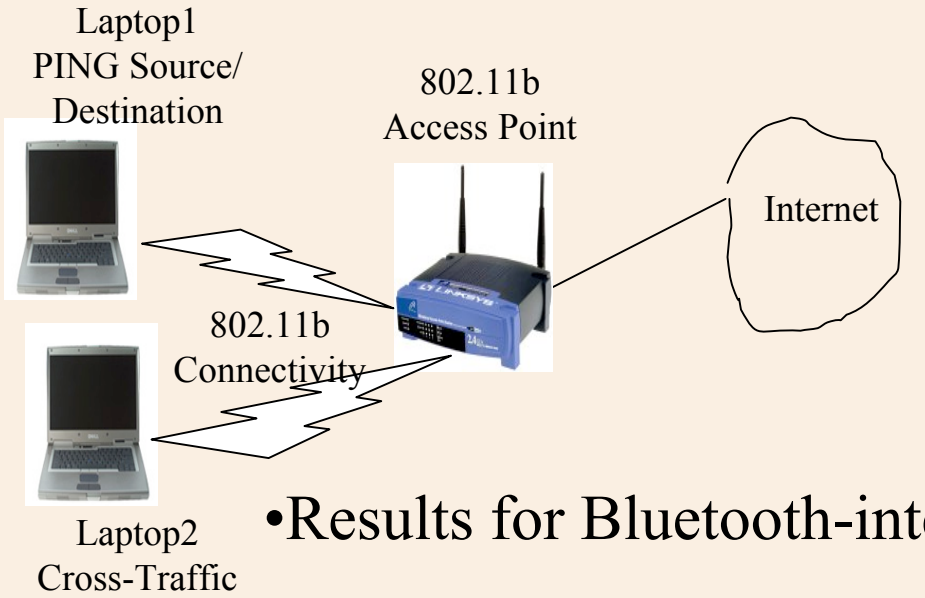
# Internet Measurements




- Each experiment: 500 PP at 0.5s intervals
- 100 experiments for each {Internet path, nature of CT, narrow link capacity}
- OS also induces inaccuracy: being fixed!

| *DummyNet Capacity* | *% Measurements Within 5% of Capacity* | *% Measurements Within 10% of Capacity* | *% Measurements Within 20% of Capacity* |
|---|---|---|---|
| *500 kbps Yahoo* | *100* | *100* | *100* |
| *1 mbps Yahoo* | *95* | *95* | *100* |
| *5 mbps Yahoo* | *100* | *100* | *100* |
| *10 mbps Yahoo* | *60* | *100* | *100* |
| *20 mbps Yahoo* | *75* | *100* | *100* |
| *500 kbps Google* | *100* | *100* | *100* |
| *1 mbps Google* | *100* | *100* | *100* |
| *5 mbps Google* | *95* | *100* | *100* |
| *10 mbps Google* | *80* | *95* | *100* |
| *20 mbps Google* | *65* | *100* | *100* |

# Wireless Measurements




- *Bad channel → retransmission → larger dispersions → lower estimated capacity*

•Results for Bluetooth-interfered 802.11b, TCP cross-traffic

| *Experiment No.* | *Capacity Estimated by CapProbe (kbps)* | *Capacity Estimated by strongest mode (kbps)* |
|---|---|---|
| *1* | *5526.68* | *4955.02* |
| *2* | *5364.46* | *462.8* |
| *3* | *5522.26* | *4631.76* |
| *4* | *5369.15* | *5046.62* |
| *5* | *5409.85* | *449.73* |

# CapProbe: Work in Progress

- Enhancements for *confidence level*, and *adaptive probing*
- Probabilistic analysis to determine number of samples required to get min sample, that is *probe length*
- Implementation in OS Kernel to reduce host inaccuracy
- Extensive Internet and Abilene (and NLANR testbed?) measurements experiments
- Use within TCP: estimating capacity accurately may help