Packet Probing: link capacity/available bandwidth

EL 933, Class4

Yong Liu

09/27/2005

Motivation

- Good to know how much bandwidth on a link
  - network operators
  - end users
- Limited access to detailed information
  - topology: link capacity
  - traffic load: SNMP summary (5 min.s)
- End-end probing with simple router support
  - sender, w./w.o. receiver cooperation
  - packet delay --> link bandwidth
  - end-end and location

Papers Today

- C. Dovrolis, P.Ramanathan, D.Moore, "What Do Packet Dispersion Techniques Measure?", Proc. IEEE/INFOCOM 2001.
  pathrate: www.pathrate.org/
- M. Jain, C. Dovrolis, "Pathload: A Measurement Tool for End-to-end Available Bandwidth", Proceedings of the 3rd Passive and Active Measurements (PAM) Workshop, March 2002.
  pathload:
  http://www.cc.gatech.edu/fac/Constantinos.Dovrolis/pathload.html
- N. Hu, L. Li, Z. Mao, P. Steenkiste, J. Wang, "Locating Internet Bottlenecks: Algorithms, Measurements, and Implications", Proc. ACM/SIGCOMM, 2004.
  pathneck: http://www.cs.cmu.edu/~hnn/pathneck/

slides modified from authors'

What do packet dispersion techniques measure?

C. Dovrolis, P. Ramanathan,

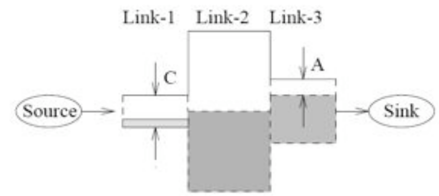
D. Moore

Overview

- Background: capacity and available bandwidth
- Dispersion of packet-pairs
- Dispersion of packet-trains
- A capacity estimation methodology: *pathrate*

Definition of capacity

- Maximum IP-layer throughput that a flow can get, without any cross traffic

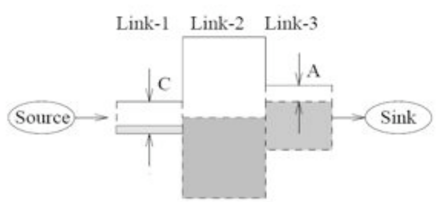


- C_i: capacity of link i (i = 1, … , H)
- Path capacity *C* is limited by *narrow link* n:

$$C = \min_{i=0 \ldots H} \{C_i\} = C_n$$

Definition of available bandwidth

- Maximum IP-layer throughput that a flow can get, given cross traffic

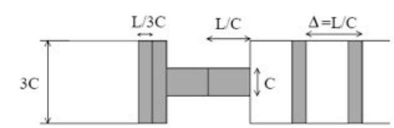


- u_i: utilization of link I
- Available bandwidth *A* limited by *tight link* t:

$$A = \min_{i=0 \ldots H} C_i(1 - u_i) = C_t(1 - u_t)$$

Packet-pair Dispersion: Basic Idea

- Packet transmission time: $\Delta = L/C$
- Sent two packets back-to-back
- Measure dispersion Δ at receiver
- Estimate *C* as $C = L/\Delta$



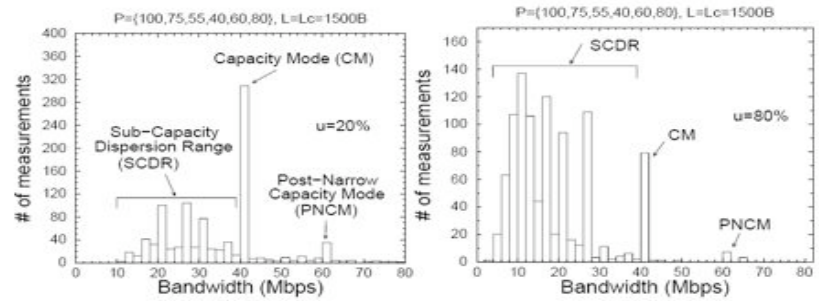
- But… cross traffic 'noise' can affect the packet dispersion

Creation of SCDR and PNCM modes

- Sub-Capacity Dispersion Range (SCDR)
  - is caused by cross traffic interfering with packet pair
- Post-Narrow Capacity Modes (PNCM)
  - are caused by back-to-back packet-pairs after narrow link (first packet is adequately delayed)

Effect of cross traffic

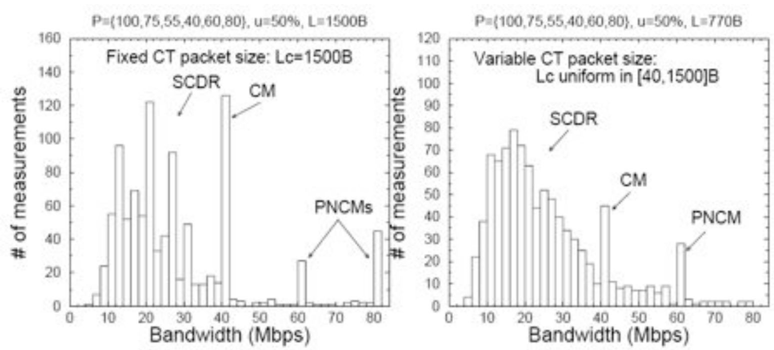
- Cross-traffic causes local modes below (SCDR) and above (PNCM) capacity mode (CM)



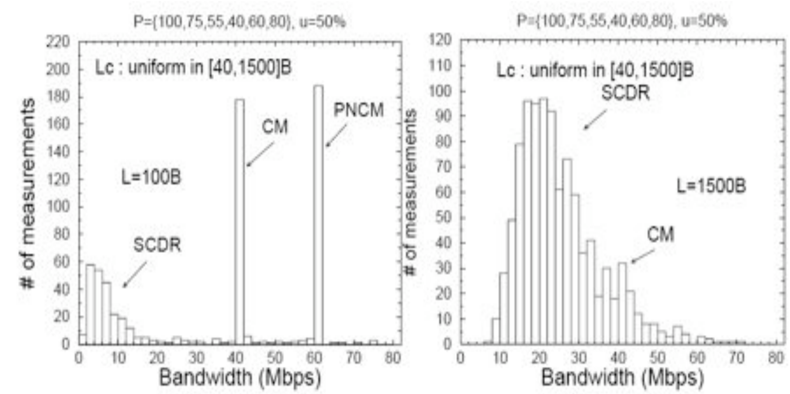
- Heavier cross traffic load makes CM weaker

Effect of cross traffic packet size

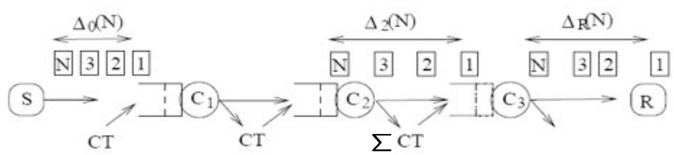
- Distinct cross traffic packet sizes cause SCDR local modes
- Common Internet traffic packet sizes: 40B, 550B, 1500B



Effect of packet-pair size



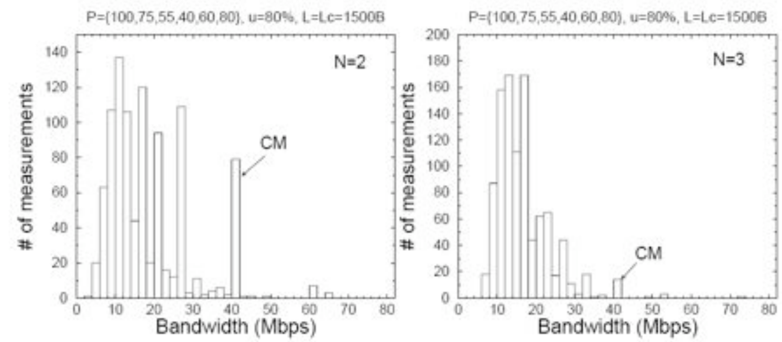
Packet-train dispersion



- Bandwidth estimate: $\frac{(N-1)L}{\Delta(N)}$

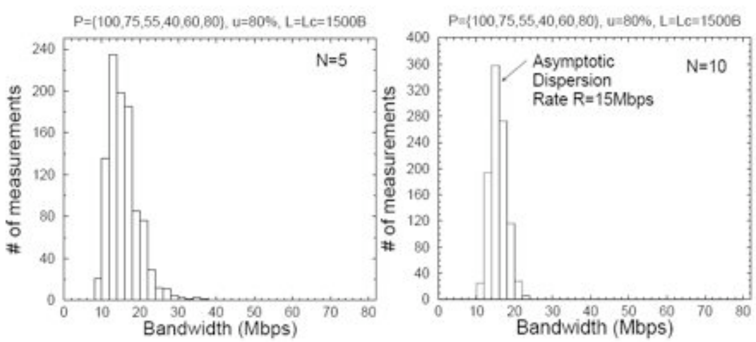
Packet-train experiments

- What happens as we increase the packet-train length N



Packet-train experiments

- Range of measurements decreases and becomes unimodal
- Measurements tend to Asymptotic Dispersion Rate (ADR) (less than C)



Pathrate: a capacity estimation methodology

Phase 1:

- Perform many (2000) packet-pair experiments to form distribution B
- Use packet sizes of about 800 bytes
- Determine local modes of distribution B
- Sequence of local modes in increasing order:

$$M = \{m_1, m_2, \cdots, m_K\}$$

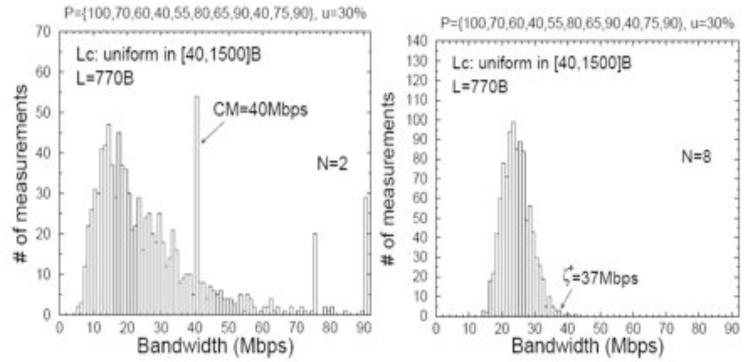
Pathrate: a capacity estimation methodology

Phase 2:

- Perform several packet-train experiments with certain N to get B(N)
- If bandwidth distribution not unimodal, increase N and repeat previous step
- Let N' be the minimum value of N such that B(N) is unimodal
- Let $[\zeta^-, \zeta^+]$ be the range of the unique mode in B(N)
- Estimate capacity as: $\hat{C} = \min\{m_i \in M | m_i > \zeta^+\}$

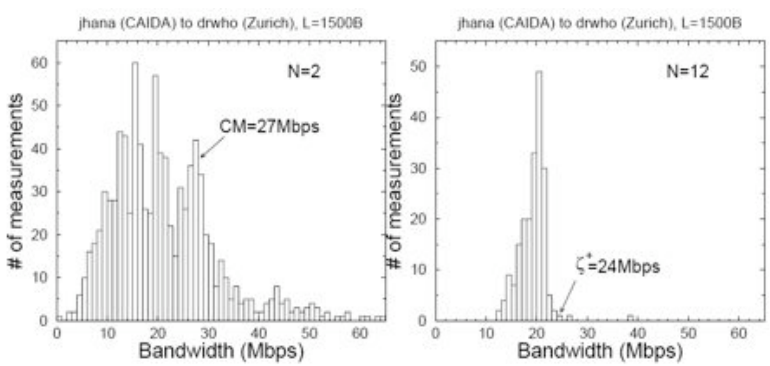
Example

- Packet-pair modes: M = {9,14,17,23,26,29,33,40,44,56,75,90}



Evaluation: CAIDA – ETH link

- Packet pair modes: M = {9,11,13,15.5,19.5,27.32,43}



Summary

- Examination of packet-pair and packet-train techniques taking cross traffic into account
  - Statistical filtering of packet-pair measurements does not work
  - Most common measurement range (mode) is not always the capacity
    - Interfering cross traffic packets cause local modes or SCDR
    - Loaded post-narrow links also cause local modes (PNCM)
  - Use of maximum size packets is not optimal
  - Packet-trains lead to ADR estimation
- Develop a capacity estimation technique

Pathload:

A measurement tool for end-to-end available bandwidth

Manish Jain, Univ-Delaware

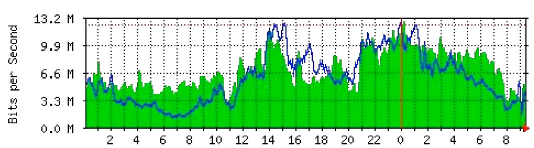
Constantinos Dovrolis, Univ-Delaware

Overview

- Self-Loading Periodic Streams (SLoPS) methodology
- Description of pathload
- Verification experiments

Measuring per-hop available bandwidth

- Network managers are *very* interested in available bandwidth
- Can be measured at each link from router utilization statistics
- MRTG graphs: 5-minute averages



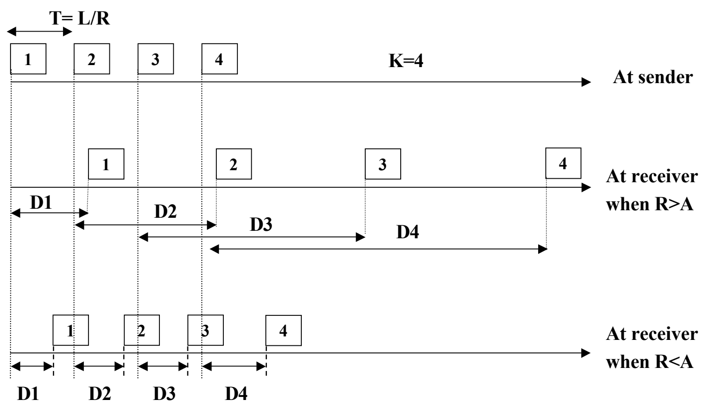
- BUT, users do not normally see this data and it is not end-to-end

Major Idea

- SLoPS analyzes *One-Way Delays (OWDs)* of packets from *sender S* to *receiver R*
- OWD: $D_i = T^R_{arrive} - T^S_{send} = T_{arrive} - T_{send} + Clock\_Offset(S,R)$
- Relative OWDs between successive packets: $D_i - D_{i+1}$
- *S* and *R* do not have synchronized clocks.

Basic Idea

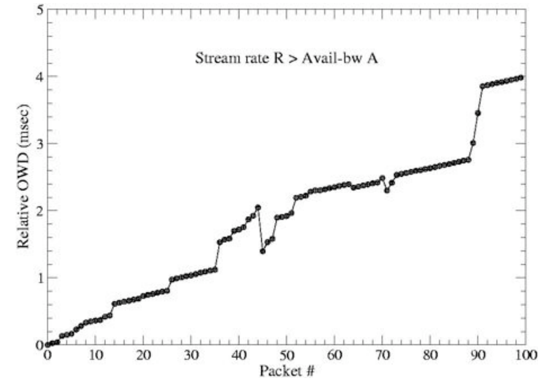
- Periodic Stream: *K* packets, size *L* bytes, rate *R* = *L*/*T*



- If *R>A*, OWDs gradually increase due to self-loading of stream

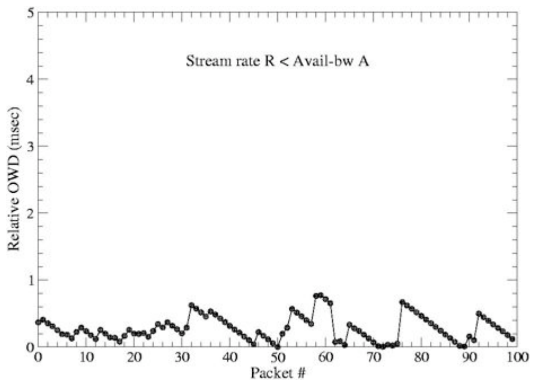
Experimental result: *R > A case*

- K = 100 packets, *A*= 74Mbps, R=96Mbps, T=100μs



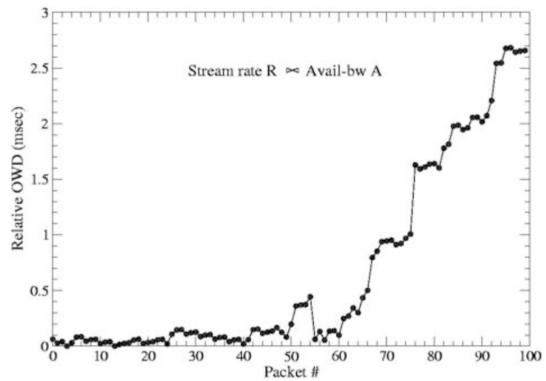
Experimental result: R < *A* case

- *K* = 100 packets, *A*= 74Mbps, *R*=37Mbps, *T*=100μs



Experimental result: R ∼ *A* case

- K = 100 packets, *A*= 74Mbps, R=82Mbps, T=100μs



Iterative algorithm in SLoPS

- At source: Send periodic stream *n* with rate *R(n)*
- At receiver: Measure OWDs D_i for *i=1…K*
- At receiver: Check for *increasing trend* in OWDs and notify source
- At source: if trend is :

  increasing (i.e. R(n)>*A*), →repeat with R(n+1) < R(n)

  non-increasing (i.e. R(n)<*A*),→ repeat with R(n+1)>R(n)
- Terminate if R(n+1) – R(n) < ω: resolution of final estimate

Selection of L, T and K

- *L* can not be less than certain number of bytes
- *L* should not be greater than path MTU, to avoid fragmentation
- *T* should be small to complete transmission of stream before context switch
- Large *K* may overflow the queue of the tight link when *R > A*
- Small *K* does not give enough samples to infer trend robustly

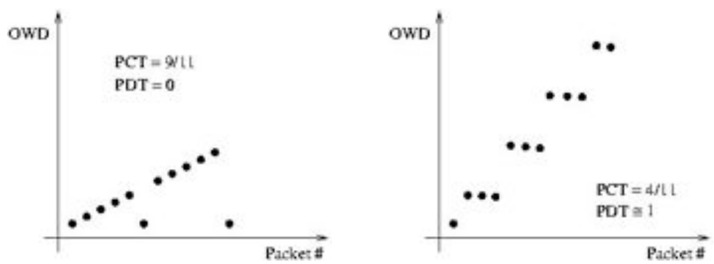
Use of Several Streams

- N streams allows us to examine N consecutive times whether R > *A* or not
- Multiple streams, separated by silence period allows queues in network to drain measurement traffic
- Duration of a fleet: $U = N \times (K \times T + \Delta)$

How do we detect an increasing trend?

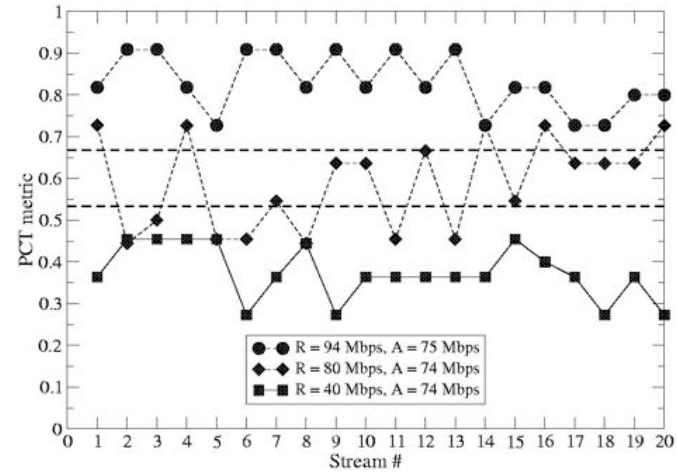
- Pairwise Comparison Test (PCT):
  - $R_{pct} = \frac{\sum_{j=2}^{K} I(D_j > D_{j-1})}{K-1}, \quad 0 \leq R_{pct} \leq 1$
  - *E[PCT]=0.5* , independent OWDs,
  - PCT -> 1, when increasing trend
- Pairwise Difference Test (PDT):
  - $R_{pdt} = \frac{\sum_{j=2}^{K}(D_j - D_{j-1})}{\sum_{j=2}^{K}|D_j - D_{j-1}|} = \frac{D_K - D_1}{\sum_{j=2}^{K}|D_j - D_{j-1}|}$
  - E[PDT]=0 for independent OWDs
  - PDT -> 1 when increasing trend

Illustration of PCT and PDT metrics

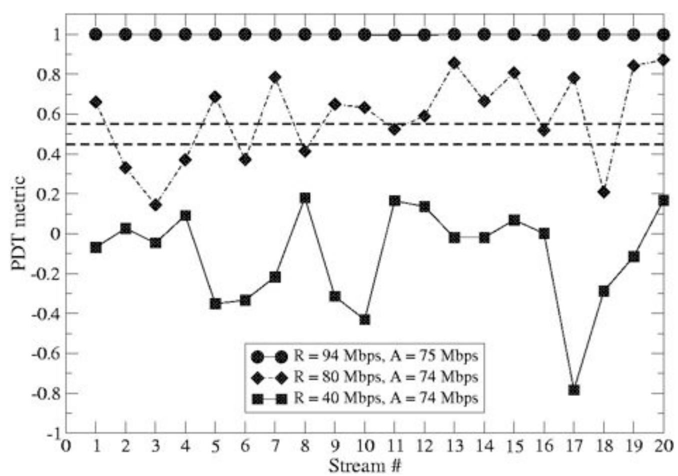


- Infer increasing trend when PCT or PDT trend ≈ 1.0

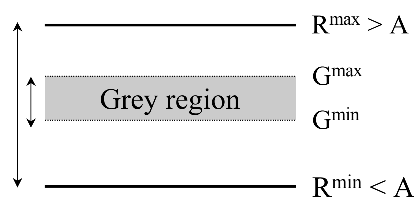
PCT variation for 3 fleets



PDT variation for 3 fleets



Rate adjustment algorithm



Terminate if:

$R^{max} - R^{min} < \omega$

or

$G^{max} - G^{min} < \chi$

Increasing trend :

$R^{max} = R(n)$

$R(n+1) = (G^{max} + R^{max})/2$

Non-increasing trend:

$R^{min} = R(n)$

$R(n+1) = (G^{max} + R^{min})/2$

Grey region & $R(n) > G^{max}$:

$G^{max} = R(n)$

$R(n+1) = (G^{max} + R^{max})/2$

Grey region & $R(n) < G^{min}$:

$G^{min} = R(n)$

$R(n+1) = (G^{min} + R^{min})/2$

Other *pathload* features

- Clock skew between sender and receiver can distort the relative OWD.
- Clock skew not an issue in *pathload* due to small stream duration.
- *Pathload* aborts the fleet if :
  - stream encounters excessive loss (>10 %)
  - a fraction of streams encounter moderate loss
- For default tool parameters, and avail-bw ≈ 10 Mbps, *pathload* takes 12 seconds

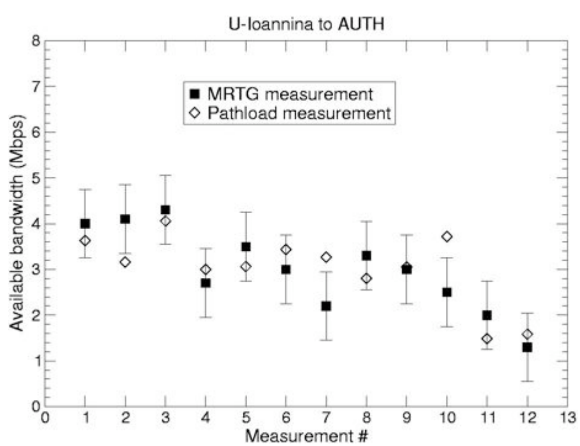
Verification Approach

- Use paths from U-Delaware to Greek universities and U-Oregon.
- Routes through UDel, Abilene, Dante, GRnet
- MRTG graphs for all links in path report 5-min averages for avail-bw
- In 5-min interval, *pathload* runs *W* times, each for q_i *secs*
- 5-min average avail-bw *R* reported by pathload:

$$R = \sum_{i=1}^{W} \frac{q_i}{300} \frac{R_i^{max} + R_i^{min}}{2}$$

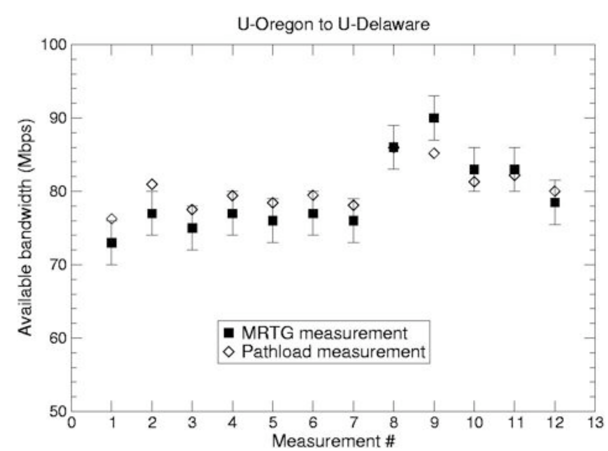
Verification I

- Tight link: U-Ioannina to AUTH(C=8.2Mbps), w=1Mbps



Verification II

- Tight link: U-Oregon gigapop-Abilene(C=155Mbps), w=1 Mbps



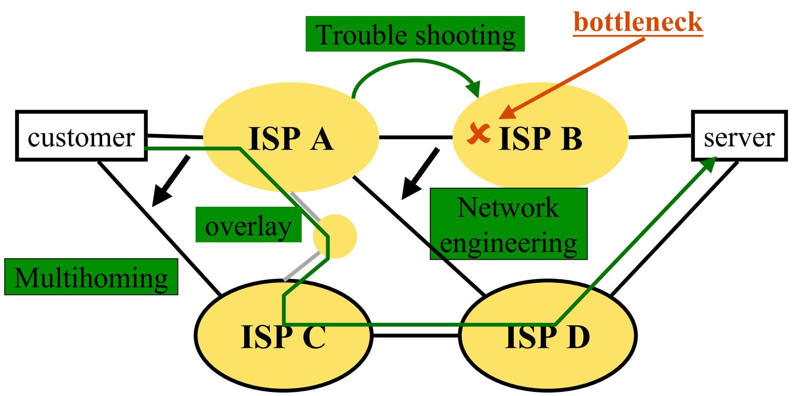
Summary

- Avail-bw has estimation numerous application
- SLoPS: fast, accurate and non-intrusive measurement
- First release of *pathload* in Spring'02
- Examined avail-bw variability using *pathload*, and results published in a technical report,
- Future work: incorporate avail-bw estimation in transport,QOS and routing

Locating Internet Bottlenecks

Ningning Hu (CMU)
Li Erran Li (Bell Lab)
Zhuoqing Morley Mao (U. Mich)
Peter Steenkiste (CMU)
Jia Wang (AT&T)

Motivation



- Location is critical for intelligent networking

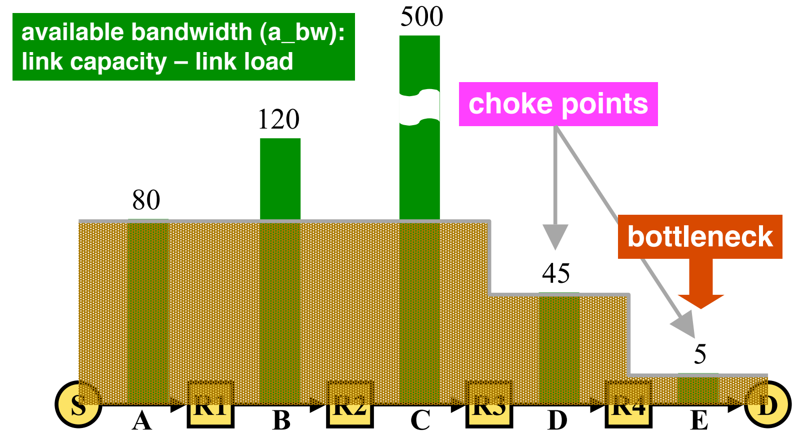
State of Art

- SNMP load data
  - Directly calculate the available bandwidth on each link
- Tomography
  - Congestion sharing among partially overlapped network paths
- Active probing tools
  - Pathchar, pipechar, Cartouche, BFind, STAB, DRPS
  - Measure each link or amplify the bottleneck
  - Large overhead/time or two-end control

Proposed Approach: Pathneck

- Pathneck is also an active probing tool, but with the goal of being easy to use
  - Low overhead (i.e., in order of 10s-100s KB)
  - Fast (i.e., in order of seconds)
  - Single-end control
  - High accuracy

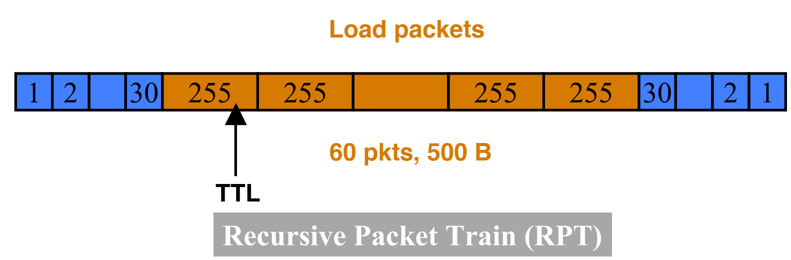
Bottleneck & Available Bandwidth



Available Bandwidth Estimation

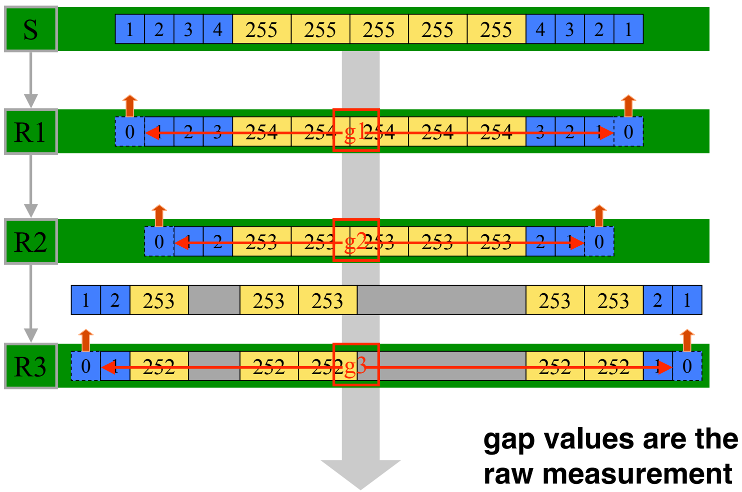
- Packet train probing
  - train_rate > a_bw ➔ train_length increases
  - train_rate ≤ a_bw ➔ train_length keeps same
- Current tools measure the train rate/length at the end nodes ➔ end-to-end available bandwidth
- Locating bottlenecks needs the packet train length info from each link

Probing Packet Train in Pathneck

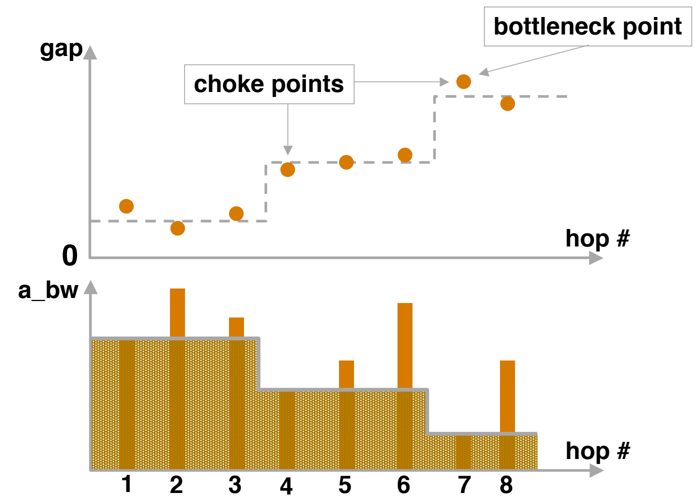


- Load packets are used to measure available bandwidth
- Measurement packets are used to obtain location information

Transmission of RPT



Choke Point Detection



Configuration Parameters

- Confidence Threshold (conf)
  - Set the minimum step change in the step function
  - To filter out the gap measurement noise
  - Default: conf ≥ 10% available bandwidth change
- Detection Rate (d_rate)
  - N probings for each destination
  - A hop must appear as a choke point for at least M times (d_rate ≥ M/N)
  - To select the most frequent choke point
  - Default: d_rate ≥ 5/10 = 50%

Patheneck: the Algorithm

Probe the same destination 10 times

- conf ≥ 10% filtering

  For each probing, only pick the choke points which satisfy conf ≥ 10% threshold

- d_rate ≥ 50% filtering

  A hop must appear as a choke point in at least 5 times to be selected

- The last choke point is the bottleneck

Output from Pathneck

- Bottleneck location (choke point locations)
- Upper or lower bound for the link available bandwidth
  - Gap value increase: probing rate is upper bound
  - Gap value unchanges: probing rate is lower bound
- IP level route
- RTT to each router along the path

Accuracy Evaluation

- Location measurement accuracy
  - Abilene experiments
  - Testbed experiments on Emulab (U. of Utah)
    - Construct different types of bottleneck scenarios using real traffic trace
- Bandwidth estimation accuracy
  - Internet experiments on RON (MIT)
    - Compare with IGI/PTR/Pathload

Accuracy Evaluation Results

- Location measurement accuracy (on Emulab)
  - 100% accuracy for capacity determined bottlenecks
  - 90% accuracy for load determined bottlenecks, mainly due to the dynamics of competing load
  - At most 30% error with reverse path congestion
- Bandwidth estimation accuracy (on RON)
  - Pathneck returns upper bound for the bottleneck available bandwidth
  - On RON: consistent with available bandwidth estimation tools

Please refer to the paper for more details

Properties

- ✓ Low overhead
  - 33.6KB each probing
- ✓ Fast
  - 5 seconds for each probing
  - (1-2 seconds if RTT is known)
- ✓ Single end control
- ✓ Over 70% of accuracy

Limitations

- Can not measure the last hop
  - ✓ Fixed recently (use ICMP ECHO packets for the last hop)
- ICMP packet generation time and reverse path congestion can introduce measurement error
  - They directly change the gap values
  - Considered as measurement noise
- Packet loss and route change will disable the measurements
  - Multiple probings can help
- Can not pass firewalls
  - Similar to most other tools

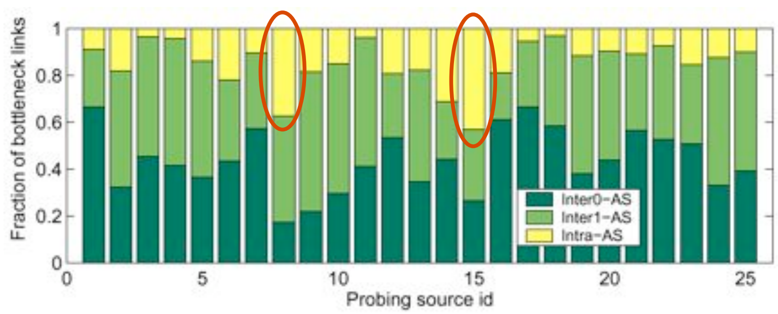
Measurement Methodology

- Probing sources
  - 58 probing sources (from PlanetLab & RON)
- Probing destinations
  - Over 3,000 destinations from each source
  - Covers as many distinct AS paths as possible
- 10 probings for each destination
  - conf ≥ 10%, d_rate ≥ 50%

1. Bottleneck Distribution

- Common Assumption: bottlenecks are most likely to appear on the peering and access links, i.e., on Inter-AS links
- Identifying Inter/Intra-AS links
  - Only use AS# is not enough (Mao et al [SIGCOMM03])
  - We define Intra-AS links as links at least one hop away from links where AS# changes
  - Two types of Inter-AS links: Inter0-AS & Inter1-AS links
  - We identify a subset of the real intra-AS links

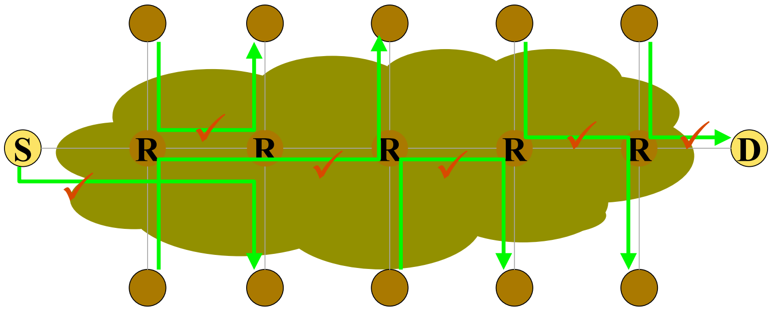
1. Bottleneck Distribution (cont.)



- Up to 40% of bottleneck links are Intra-AS
  - Consistent with earlier results [Akella et al IMC03]

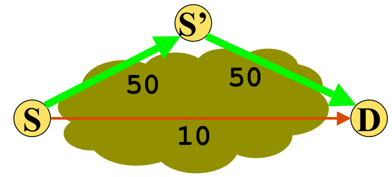
2. Inference

- Help to reduce the measurement overhead



- 54% of inferences are successful for 12,212 paths with “enough information”

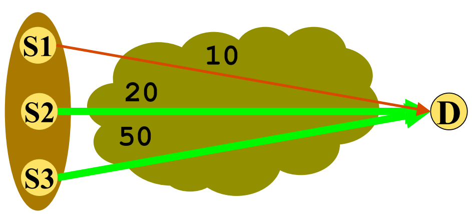
3. Avoidance: Overlay Routing



- Useful metric: the estimated bandwidth on S-S'-D is larger than those on S-D

- 53% of 63,440 overlay attempts are useful

3. Avoidance: Multihoming



- Method
  - Use multiple sources in the same region to simulate multihoming
  - Useful metric: if the bandwidth on the worst path can be improved by at least 50% by all other sources

- 78% of 42,285 multihoming attempts are useful

Conclusion

- Pathneck is effective and efficient in locating bottlenecks

  Up to 40% of bottleneck links are Intra-AS

  54% of the bottlenecks can be inferred correctly

  Overlay and multihoming can significantly improve the bandwidth performance

- Source code is available at http://www.cs.cmu.edu/~hnn/pathneck