End-to-end Inference of Router Packet Forwarding Priority

Guohan Lu†, Yan Chen, Stefan Birrer, Fabián E. Bustamante, Chi Yin Cheung, and Xing Li†
Department of EECS, Northwestern University, Evanston IL, USA
†Department of EE, Tsinghua University, Beijing, China

***Abstract*—Packet forwarding prioritization (PFP) in routers is one of the mechanisms commonly available to network administrators. PFP can have a significant impact on the performance of applications, the accuracy of measurement tools' results and the effectiveness of network troubleshooting procedures. Despite their potential impact, no information on PFP settings is readily available to end users. In this paper, we present an end-to-end approach for packet forwarding priority inference and its associated tool, POPI. This is the $first$ attempt to infer router packet-forwarding priority through end-to-end measurement. Our POPI tool enables users to discover such network policies through the monitoring and rank classification of loss rates for different packet types. We validated our approach via statistical analysis, simulation, and wide-area experimentation in PlanetLab. As part of our wide-area experiments, we employed POPI to analyze 156 random paths across 162 PlanetLab nodes. We discovered 15 paths flagged with multiple priorities, 13 of which were further validated through hop-by-hop loss rates measurements. In addition, we surveyed all related network operators and received responses for about half of them confirming our inferences.**

I. Introduction

Packet forwarding prioritization (PFP) has been available in off-the-shelf routers for quite a while, and various models from popular brands, such as Cisco and Juniper Networks [1, 2] offer support for it. Network operators have come to rely on these mechanisms for managing their networks, for example as a way of rate limiting certain classes of applications (e.g. peer-to-peer) [3].

PFP can have a significant impact on the performance of applications, beyond those targeted by administrators. PFP can also severely impact the accuracy of measurement tools' output and the effectiveness of network troubleshooting procedures. For example, measuring network path characteristics is critical for the diagnosis, optimization and development of distributed services. PFP settings in routers (e.g. forwarding priority based on packets' protocols or port numbers), however, can potentially introduce a performance dissonance between the view portrayed by measurement tools and what is ultimately experienced by applications.

Despite its potential impact, users, developers and most other network administrators have no information of such settings nor ways to procure it. In this paper, we present an end-to-end approach for packet forwarding priority inference and its associated tool, POPI.

We addressed a couple of interesting challenges while designing and implementating POPI. First, end-to-end inference accuracy of router properties can be severely affected by background traffic fluctuations. *To overcome this challenge, POPI sends relatively large amount of traffic to temporarily saturate bottleneck traffic class capacity.* Secondly, while most existing inference methods assume certain independence (*i.e.*, i.i.d. processes) or strong correlation models (*e.g.*, back-to-back probe packets), probe traffic of multiple packet types are neither independent nor strongly correlated. Thus, none of the existing inference methods can help us here since our tool POPI needs to send relatively large amount of active probes, but cannot afford to have pair-wise measurements for every pair of packet types. Instead, *POPI employs a non-parametric method based on loss rate ranks (instead of pure loss rates) to infer priority settings.* Altogether, our approach to PFP inference gives POPI better resistance against background traffic fluctuations and allows it to cope with the characteristics of its measurement traffic.

We evaluate our approach via statistical analysis, NS-2 simulation and wide-area experiments in PlanetLab. We choose 32 packet types with various protocols (ICMP, TCP and UDP) and port numbers including some associated with traditional and non-traditional applications (e.g. P2P) and some security vulnerabilities. We run POPI over 156 directional paths on 162 random PlanetLab hosts. Our tool identified 15 paths flagged with multiple priorities, 13 of which were validated using a hop-by-bop measurement method similar to that used in [4]. After surveying all related network operators, we received response for seven of them all confirming our inferences.

The remainder of the paper is organized as follows. We review related work in § II before presenting the design and implementation of our tool POPI (§ III). Then we discuss our NS simulation experiments in § IV and the Internet experiments and validation in § V. We conclude in § VI.

II. Background

Available documentations [1, 2] indicate that there are three commonly available router mechanisms to enforce priority/link-sharing on traffic classes (usually defined by IP protocol and TCP/UDP port number): *Priority Queuing*, *Proportional Share Scheduling* and *Policing*. *Priority Queuing* (PQ) allows the assignment of absolute priority among queues. *Proportional Share Scheduling* (PSS), such as Weighted Fair Queuing (WFQ) and Weighted Round-Robin (WRR), enables the assignment of bandwidth limit to traffic classes. Lastly, *Policing* makes it possible to restrict the maximum rate of a traffic class. *Policing* differs from *PSS* in that, in the latter, the policed traffic class cannot borrow unused bandwidth from other classes.

Note that only the first mechanism, PQ, sets absolute priorities between traffic classes. The other two mechanisms do not impose such absolute model; i.e., the loss experienced by one class depends on its allocated bandwidth and its traffic rate.

A. Related Work

To the best of our knowledge, this is the first attempt to infer router packet-forwarding priority through end-to-end measurement.

Perhaps the efforts most closely related to this work are those identifying shared congestion [5–7]. Such efforts try to determine whether *two* congested flows are correlated and share a common congested queue along their paths. If we consider the flows of different packet types along a same path, our problem becomes to identify whether these flows do not share a common congested queue. While both problems are related clearly, we usually need to simultaneously consider a much larger number of packet types (*e.g.*, 32 packet types in the PlanetLab experiment). Note that the correlation based method used for shared congestion identification methods requires back-to-back probing which, in our case, translates into $O(n^2)$ pairs probing for n packet types. In addition, those efforts focused on flows which experience congestion (ignoring uncongested ones), so their probe traffic rate is low and not bursty [5–7]. To identify packet forwarding prioritization in routers, one must send relatively large amounts of traffic to temporarily force packet drops (by saturating the link). Thus, for better scalability and accuracy, our problem requires different measurement and statistical interference methods.

PFP inference also has some goals in common with efforts on network tomography [8–10]. However, unlike in network tomography where loss information and topology information are combined to infer link losses, we look to identify if different packet types (based on protocol or port numbers) experience different loss rates. In addition, while probes used for network tomography are always non-intrusive in order to get accurate link loss/delay, our problem requires that we saturate links in order to uncover the configuration of the routers.

Finally, there is a number of available tools for measuring hop-by-hop properties of a path [4, 11, 12]. By probing with different packet types, these tools can measure losses for such types in a hop-by-hop manner. POPI is complementary to these tools (indeed, we employ some of these tools' methods for validation). The statistical method used by POPI could be incorporated into some of these tools for PFP inference. In addition, POPI's lighter-weight end-to-end method could be used as a first step before applying any of the hop-by-hop methods implemented by such tools.

III. Inferring Packet-Forwarding Priority

There may be several candidate metrics to infer packet forwarding priority, such as packet loss, delay or out-of-order events. In this paper, we only use packet loss as the inference metric because it is the most direct consequence of a priority configuration. We do not rely on packet delay measurements since they may fail to reveal the priorities experienced by packets, as low-priority packets may simply be dropped under congestion without having experienced significant increases in queueing delays. We do not use packet reorderings as the metric since certain priority setting mechanisms such as *Policing* may not generate out-of-order events at all. Since, for some other mechanisms, packet reorderings may occur before observing packet losses, as a part of our future work, we are exploring to use it as another inference metric.

PFP in routers are set in a per-interface basis. Prioritization of packets does not become evident until the associated link (or a sublink for a traffic class) is saturated, at which point the configured router will begin to drop packets based on its settings. This simple observation defines the basis of the approach used in POPI: *In order to reveal packet-forwarding priorities, one needs to saturate the path available bandwidth for a given class to produce loss rates difference among different classes.*

Assuming the existence of a PFP mechanism in routers such an approach will succeed at uncovering priority settings in routers along a path if the available bandwidth for the controlled class is lower than the bottleneck available bandwidth of the path. This is illustrated in Figure 1. In this figure, router A has the bottleneck link with 10Mbps of available bandwidth, while router B has a total of 91Mbps of available bandwidth. If, for a particular packet type, router B is configured to only allow 1Mbps using Policing, we can saturate the sublink and detect the priority setting on router B. On the other hand, if PQ or PSS is configured at router B, this approach could not detect the priority setting unless the whole link B is saturated.

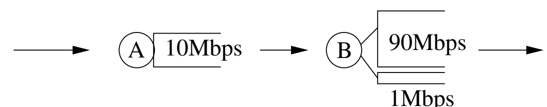

Fig. 1: Priority Inference

A. Challenges for POPI

In designing and implementing POPI we addressed a number of interesting challenges.

- **The accuracy of end-to-end inference of router properties can be severely affected by background traffic fluctuations.** Clearly, if one's probing introduces relatively small additional traffic, whether the link is saturated will depend on the amount of background traffic. To make our approach resistant to background traffic fluctuations we opt for sending relatively large amount of traffic to temporarily saturate bottleneck traffic class capacity.
- **Probe traffic of multiple packet types are neither independent nor strongly correlated.** While most existing inference methods assume certain independence (i.e., i.i.d. processes) or strong correlation models (*e.g.*, back-to-back probe packets), probe traffic of multiple packet types are neither independent nor strongly correlated. Thus, none of the existing methods can help us here since our tool needs to send relatively large amount of active probes, but cannot afford to have pair-wise measurements for every pair of packet types. To infer PFP settings we employ a *non-parametric* method based on *loss rate ranks* (instead of pure loss rates).

Thus, POPI adopts a two-step approach to PFP inference: (i) saturate the link with relatively large amount of traffic and (ii) cluster packet types based on their loss rate ranks. Such an approach gives POPI better resistance against background traffic fluctuations and allows it to cope with the inherent characteristics of its measurement traffic.

B. Link Probing Method

Fig. 2 illustrates our link probe method. We want to test k packet types. POPI sends a number of bursts (n_b) from

Notation	Meaning
n_b	number of bursts in a path measurement
n_r	number of rounds in one burst
Δ	time interval between bursts in a path measurement
k, k_j	number of all tested packet types, number of tested packet types for class j
J	number of queues/classes/groups.
ANR_i	the average normalized rank for packet type i over n_b bursts
θ	threshold used for comparing with ANR range

TABLE I: Key notations

a source to a destination. The interval between bursts is Δ. Each burst consists of n_r rounds, in which k full-length packets (1500Bytes), one for each packet type studied, are interleaved in random order. So, there are $n_r \times k$ back-to-back packets in each burst. There are three parameters for the probe method, Δ, n_b and n_r. Δ is set to tens of seconds in order to achieve independence between bursts, i.e., to ensure the router's queuing busy period caused by one burst does not interfere with the following one. We'll discuss n_b, n_r in § III-D and § IV.

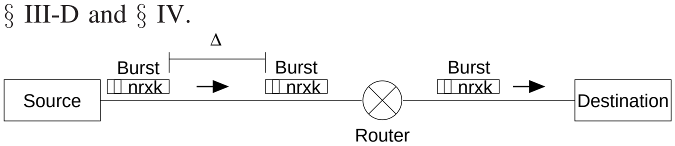

Fig. 2: A burst consists of $n_r \times k$ packets, one for each packet type.

C. *Inferring Priority Using Loss Rate Ranks*

We assume every burst can saturate the link and the background traffic is relatively stable. When this assumption holds and the link is configured with multiple queues, saturated queues will experience losses while non-saturated queues will not, or queues saturated to different degrees will experience different losses. As the background is relatively stable, such loss rates differences will remain consistent over all bursts.

However, if we were to infer using these absolute loss rates, the statistical method will be parametric. Parametric methods are usually applied when the data can be described with a good mathematical model, e.g. when the losses can be modelled as an independently and identically distributed (i.i.d) process. In our domain, a parametric method will estimate the model's parameters p_i, the loss rates of different packet types, and then determine if all p_i are the same for different packet types. However, our probing method makes the application of a parametric statistical method impossible as the packet losses of a packet type are neither identical nor independent, and are difficult to model.

Instead, we use a non-parametric statistical method based on loss rate ranks which relies on fewer assumption on the loss model of the data. Using loss rate ranks instead of absolute loss rate values has the additional advantage of being more robust in the face of measurement noise, as shown by our Internet experiments (§ V).

For every burst, loss rate ranks are computed by first sorting packet types in ascending order according to the number of loss packets in that burst and then assigning ranks in order, i.e. the packet type with the largest loss rate has rank 1, the one with the second largest loss rate has rank 2, etc.[1] Therefore, for packet types showing persistent loss rate difference over all bursts, their loss rate ranks will also show persistent difference over the bursts. On the other hand, for packet types of a unconfigured link or of a same priority group, their loss ranks will be like random arrangements among the n_b bursts. In sum, the loss rate ranks measured are well separated for different priority groups and mixed within the same priority group.[2]

The problem here is to identify whether the ranks are well separated or not, which is similar to the well-known statistical *problem of n rankings* [14]. There, n judges are asked to rank k objects in the order of their preferences to find out if there is any agreement among them with respect to their orders of preferences. In our problem, the k packet types are the k objects to rank and the n_b bursts are the n judges. Classic non-parametric solutions such as the Friedman test[13] can find whether there is agreed preference among packet types, but they do not tell what the agreed preference is, *i.e.*, they do not make partitions among packet types. Therefore, we proposed to use Average Normalized Ranks (ANR) which can be used to group packet types when there is agreed preference.

The ANR is the average of the ranks for a packet type over all bursts. According to above analysis, the $ANRs$ are roughly the same for packet types within the same priority group, but differ for those belonging to different priority groups. Our statistical method is as follows:

1) *Calculate ANR.* Let $r_i^m = (1, 2, ...)$ denote the rank for packet type i in mth burst. The Normalized Rank NR_i^m is r_i^m/k. The range of NR_i^m is between $1/k$ and 1. The ANR_i for packet type i is
$$ANR_i = (\sum_{m=1}^{n_b} NR_i^m)/n_b. \quad (1)$$
We developed a mathematical model for ANR using Central Limit Theorem:
Theorem 1: When k_j packets are in a same class j, the range of this class ($R = ANR_{\max} - ANR_{\min}$) for n_b bursts at confidence level $1 - \alpha$ is
$$\theta_{1-\alpha,k_j,n_b} = Q_{1-\alpha,k_j} \times \sqrt{k_j^2 - 1}/k\sqrt{12n_b}, \quad (2)$$
where $Q_{1-\alpha,k_j}$ is the $100(1-\alpha)\%$ percentile of the range (of k_j i.i.d. standard normals) distribution.
Proof: Please refer to our technical report [15]. ■
According to this model, when $R > \theta_{1-\alpha,k_j,n_b}$, those packets should belong to multiple groups.
2) *Partition priority groups based on ANR.* We use a hierarchical divisive partition approach to cluster ANRs. Initially, we assume all packet types belong to one group, then we use the above criteria to judge this assumption. If $R > \theta$, we partition them into two groups using the *k-means* clustering algorithm. This procedure is applied recursively to all newly partitioned groups until $R \leq \theta$ or there is only one packet type in the group.

D. *Performance Analysis on Priority Group Partitioning*

In this section, we first simulate many sets of random rank values (given the number of priority groups and packet types) that satisfy the following two conditions:

1) $\forall$ packet type $i \in$ priority group G_i, and $\forall$ packet type $j \in$ priority group G_j, the loss rate rank $r_i > r_j$ as long

[1] For tie breaks, we use the midranks method [13] to distribute the total ranks equally among them.

[2] Our method is robust in the case when a fraction of bursts does not satisfy the assumption discussed in § III-D.

α	type \ n_b	8	16	32	64	128
0.01	Over Partition	8.5	2.52	2.23	2.52	2.42
	Under Partition	0.20	0	0	0	0
	Sum	8.7	2.52	2.23	2.52	2.42
0.001	Over Partition	5.7	0.63	0.21	0.29	0.23
	Under Partition	43.5	0	0	0	0
	Sum	49	0.63	0.21	0.29	0.23

TABLE II: Average cluster error percentage (%) for $J = 2$

as group G_i has higher priority than group G_j.

2) For packet types within the same priority group, their ranks are randomly permutated in each burst in order to simulate the effects of random losses.

We then analyze the ANR group partition performance on the average of those sets of rank values. Note that the above conditions do not consider the worst case when the rank of i in higher priority group G_i can be smaller than the rank of some packet type j in a lower priority group G_j. However, we do consider an extreme case that violates those two constraints due to severe traffic fluctuations in the end of this section.

Generally speaking, our method gives three types of errors.

1) *over-partitioning*: The number of partitioned groups is more than that of the actual situation. It results from the type I error associated with the criteria $R > \theta$, when POPI thinks that k packet types are not in one group but actually they are. According to statistical theory, the percentage of these errors occuring is less than the significance level α we choose to calculate θ.
2) *under-partitioning*: The number of partitioned groups is less than that of the actual situation which results from the type II error of the criteria. We proved in [15] that when n_b is above certain value, i.e. larger than 12 for $k = 32$, the percentage of type II errors occuring is zero.
3) *mis-partitioning*: The number of partitioned groups is equal to that of the actual situation, but some packet types are partitioned to wrong groups.

As the basic operation of our cluster method is to split several packet types into two groups, we first analyze the case of two priority groups ($J = 2$). There are 256 (k_1, k_2) combinations where $k_1 + k_2 = k \leq 32$ and $k_1 \leq k_2$. We try 64 simulations for each of the combination, i.e. 16,384 simulations in total. Table II shows the percentage of the first two types of cluster errors for $n_b = 8, 16, 32, 64$, or 128 with $\alpha = 0.01$ or 0.001. The percentages for the third type of errors are all zero. When $n_b = 8$, both α have a large error percentage. Therefore, we should choose $n_b > 8$ for our experiments. For $n_b \geq 16$, both $\alpha = 0.01$ and $\alpha = 0.001$ have 0% *under-partitioning*, which agrees well with our theoretical analysis result stated just before. As $\alpha = 0.001$ has smaller percentage of *over-partitioning*, we will use $\alpha = 0.001$ for our further experiments. For $\alpha = 0.001$, its percentage of *over-partitioning* is 0.63% for $n_b = 16$ and decreases to 0.20% for n_b larger than 32. 0.20% is close to the confidence level we set. Further analysis shows that the cluster errors are essentially uniformly distributed over different (k_1, k_2) combinations.

When $J > 2$, the number of $(k_1, k_2, ..., k_J)$ combinations grows dramatically for $\sum k_j \leq 32$ and $k_j \leq k_{j+1}$. We tested all combinations, but with reduced number of simulations for

J \ n_b	8	16	32	64	128
3	50.0	0.30	0.31	0.35	0.46
4	69.0	0.44	0.39	0.54	0.53
5	82.1	0.93	0.60	0.52	0.48

TABLE III: Average cluster error percentage (%) for $J = 3, 4, 5$, $\alpha = 0.0001$

each combination to keep the total number of simulations about the same to that of $J = 2$. Table III shows the partitioning accuracy for $J = 3, 4, 5$. Our partitioning method consists of two basic operations, the threshold comparison and *k-means* partition. When J increase, the number of such operations for correct partitioning increases. As each operation may introduce error, the overall performance decreases as J increases, as shown in the table. However, the error percentages are all below 1% for $n_b \geq 16$.

The background traffic may not be stable during the probe. Consider an extreme case, where the background traffic is ON/OFF traffic. Suppose when a probe burst is sent during the ON period, the loss rates measured will be well-separated. When a probe is sent during the OFF period, no loss rate difference will be observed (as the link will not be saturated). The ranks in such bursts will be the same for all packet types. We fixed the total number of bursts to 32 and increased the number of bursts probed during the OFF period (n_0). Fig. 3 shows how n_0 affects the cluster results. The error is the average performance of all (k_1, k_2) combinations for $k_1 + k_2 = 2, 14, 23, 32$ and $k_1 \leq k_2$, where we run 64 simulations for every combination. The cluster error increase suddenly when n_0 becomes larger than 13, 40% of 32 bursts. Below that value, the error percentage remains zero. This shows our clustering method does not require every burst to have loss rate differences, and that it is robust against quite large fluctuations on background traffic.

IV. Evaluation with NS-2 Simulation

In this section, we use NS-2 simulations to investigate performance when the loss rates of different priority groups are possibly not well-separated. We implemented POPI in NS-2 and used CBQ (Class-Based Queueing) [16] for various queue configurations. We use a dumbbell topology as shown in Fig. 4. The router $R0$ is configured with the priority settings. The background traffic is constant because we have discussed the effects of ON/OFF background traffic in the previous section already. In the experiment, the size of both POPI packets and background packets are 1000 bytes long, and POPI is configured to send probe bursts at 100Mbps.

Here we show the result of Priority Queuing configuration. The result for PSS configuration is similiar and can be found in [15]. $R0$ is configured with two classes. Class 1 is the high priority class, with a queue length of 20 packets. Class 2 is the low priority class, with a queue length of 60 packets.

Fig. 5 shows how the partition results are affected as n_r increases for various (k_1, k_2) combinations and 10Mbps background traffic. Every point in the figure denotes the result of a simulation experiment in which POPI sends 32 packet types with $n_b = 32$. As shown in this figure, the partition results can be divided into three phases according to the value of n_r. In phase 1, $n_r \leq 18$, the probe does not saturate the link,

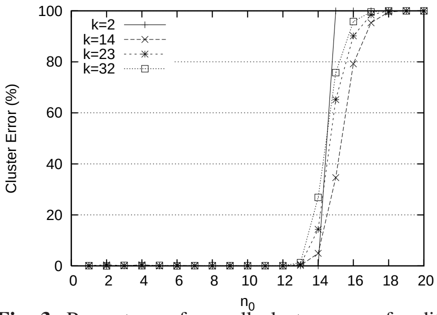

Fig. 3: Percentage of overall cluster errors for different $n_0, k, n_b = 32$.

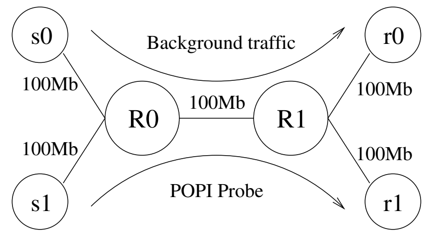

Fig. 4: NS-2 simulation topology

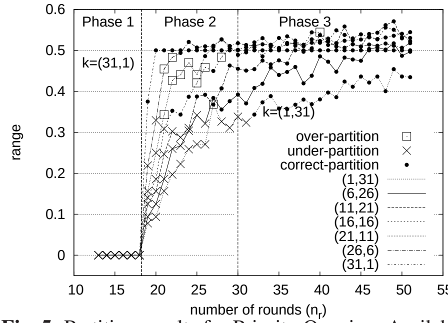

Fig. 5: Partition results for Priority Queuing. Available bandwidth = $90Mb$.

hence no loss is generated, and all packet types are partitioned as one group.

Phase 2 is a transitional phase. The low priority packet types began to experience losses, but since the losses are insufficient for POPI to properly classify the packets, the partition results were still incorrect. This phase can be further divided into two sub-phases based on the amount of losses generated. In the beginning, a packet type in low priority group only experienced sporadic drops in some bursts. For most of the bursts, it showed no difference in terms of drops from the high-priority packet types, thus POPI would still cluster them with high priority packet types, resulting in *under-partitioning*. As n_r increases, certain class 2 packet types would have drops in almost every burst, and would be clustered in the low-priority group, while some other class 2 packet types still did not have sufficient drops to be clustered as low-priority group, as their ANR_i were in the middle of the typical ANR of class 2 and the typical ANR of class 1. Those intermediates were clustered as a separate group when judged against the criteria, which resulted in *over-partitioning*.

Finally, POPI entered phase 3 in which the loss rates were well-separated, and POPI's accuracy solely depends on the performance of the cluster method, which has an error percentage below 1% shown in previous section.

The above simulations show that we can get very accurate results as long as we can generate enough losses. Besides, the above simulations help us estimate the n_r for our PlanetLab experiments. In our PlanetLab experiments, we also probe at 100Mbps with 32 packet types. We assume the queue length of the configured router is 60 (the default value for the *normal-limit* priority queue for Cisco Routers with recent IOS 12.2 [1]) and the available bandwidth is less than 90Mbps. According to this simulation, $n_r \geq 30$ is enough to get accurate result. Thus we use $n_r = 40$ in PlanetLab experiments.

V. PlanetLab Experiments

In this section, we first describe the experimental methodology, then present the results and its validation.

A. Experiment Methodology

While it may seem necessary to test all packet types of different protocol/port number combinations to validate our approach, in practice there is only a small number of packet types that network administrators may want to treat differently. For our evaluation, we selected 32 packet types as shown in Table IV to check:

- Whether ICMP, TCP and UDP packets are handled with equal priority.

PROT	Type/Port Number
ICMP	ICMP_ECHO
TCP	20, 21, 23, 110, 179, 443 (well-known app) 1214, 4661, 4662, 4663, 6346, 6347, 6881 (P2P applications) 161, 135, 137, 139, 445 (security-related) 1000, 12432, 25942, 38523, 43822, 57845 (random)
UDP	161 (SNMP) 1000 12432 25942 38523 43822 57845 (random)

TABLE IV: 32 packet types considered for PlanetLab experiments.

- Whether some well-known applications are granted higher priority. This set includes ftp (port 20, 21), telnet (port 23), POP3 (port 110), BGP (port 179), and HTTPs (port 443). Port 80 was not included because it was used by PlanetLab maintenance.
- Whether P2P traffic is treated with lower priority. The seven ports tested are used by four major P2P applications, Fasttrack, eDonkey, Gnutella, and BitTorrent.
- Whether vulnerable ports are treated with lower priority. These ports are used by worms such as Code Red and Nimda.

We send TCP packets without any special flags because the firewalls may misinterpret our probes as a SYN-flooding DoS attack and perform SYN-limiting if we send many SYN-flagged packets.[3] Due to connection tracking of PlanetLab nodes, POPI sender and receiver first perform the TCP 3-way handshake. If it can establish a connection for a TCP port pair, POPI will probe with normal TCP packets for that port pair. Otherwise, we assume the port is banned and exclude it from the rest of measurement or priority inference. In fact, we have found that many ports related to security vulnerabilities are banned.

For UDP and TCP packets, we used the port numbers listed in Table IV as source ports to measure the source port based priority policy. 30002 is used as the destination port, because it is very unlikely that ISPs will set an explicit priority policy based on it. We can measure the destination port based priority policy in a similar manner.

We deployed POPI on 162 PlanetLab nodes distributed over the world. Each host belongs to a different site, and together they span over 100 autonomous systems. About 60% of the hosts are located in North America, while the others are distributed in Europe, Asia and South America.

We ran our measurement on May 12, 2006. Nodes were randomly paired to create 81 pairs, and we probed 162 paths in both directions for each pair. We ran POPI with $k = 32, n_b =$

[3] We measured on certain paths, and found out that SYN probe packets are periodically blocked if we probe with SYN floods.

$32, n_r = 40$ and $\Delta = 10$ seconds. Thus for each path, the measurement took 5 minutes, and 40,960 packets were sent (each with size 1500 B). Thus although the burst of probe traffic is quite intensive, the average bandwidth consumption is just 1.6 Mbps, well below the typical 100 Mbps capacity. The total traffic is 6.64 million packets, or 9,953 MB.

After completing our measurements, we gathered the packet dump files from the receiver nodes. We were only able to collect results from 156 of the 162 paths measured and we use those paths for the analysis. Among the 32 packet types measured, we usually have 27∼32 packet types for a path after excluding those banned packet types and those which failed to setup the TCP connection from priority group inference as mentioned before.

B. Data Analysis and Results

First, we check how packet drops are distributed among bursts of a path measurement to see if there are any effects caused by background traffic fluctuations, as we discussed in § III-D. Fig. 6 shows the distribution of number of path measurements in which a certain number of bursts experienced network packet drops. From this figure, we can see that for most of the path measurements, either all bursts experienced drops or no bursts experienced drops. This suggests that if the first burst in a path measurement can saturate the link, it is likely to be true for the rest of bursts, and vice versa. This is accordance with our notion that traffic remain relatively stable within a short period of time.

1) The Performance of Average Normalize Ranks: As we chose ANR as the metric to infer whether there is multiple priority groups and to cluster priority groups, it's crucial to understand how such a metric really captures packet forwarding priorities. In this section, we try to answer the following questions.

- Can ANR range clearly distinguish single priority vs. multiple priority settings?
- How does ANR perform when compared with other alternative metrics, *e.g.*, link loss rate range?
- With multiple priority settings, is ANR suitable to cluster the packet types into different groups?

First, we examine how accurately the ANR range metric can distinguish whether there is a packet forwarding preference in effect. Fig. 7 shows the cumulative percentage of the ratio between the ANR range measured and the threshold θ used for partitioning of the 156 paths. As discussed in § III-D, we use a confidence level $\alpha = 0.001$ to calculate the threshold θ. When the ratio is larger than one, packet types are partitioned into multiple priority groups. The curve shows that the ratios less than one and those larger than one are well separated. Among 141 paths whose ratios are less than one, the ratios of 140 paths are well below one (0.82). On the other hand, of the 15 paths with ratio larger than one, 13 paths have ratios larger than 1.20. In addition, we compare our method with the Friedman test as discussed in § III-C, using the same α. The results are almost exactly the same. Therefore, both evaluations suggest that the ANR range is a good indicator for whether there is an agreed preference among the packet types.

Secondly, we compare the ANR range metric with another possible candidate, the loss rates (LR) range metric. The LR range is the difference between the maximum and the minimum loss rate of different packet types. In Fig. 8 we plotted a path measurement as a point with its x-axis as the LR range and y-axis as the ANR range. Suppose we have designed another priority inference method based on the LR range and use it on these paths. Then, a loss rate based threshold θ' will be used to distinguish a priority path from a non-priority path in the same way as we do in ANR range method. For those points with both large ANR and large LR ranges, the two methods will both infer them as multi-priority paths, thus there is no difference between them. The points with both small ranges also make no difference. The points that make a difference are those with large ANR but small LR ranges, and those with small ANR but large LR ranges, because the two methods will make opposite conclusion for these points.

We first check the two typical points with large ANR ranges but small LR ranges, (0.04,0.40) and (0.04, 0.36). They correspond to the path 13 and 14 in Table V. The inference results are correct according to the feedback from the relevant network operators. Their ANR ranges are large, ranked as the 13th and 14th largest ANR ranges, whereas their LR ranges ranks are not as high as their ANR ranges ranks, ranked as the 26th and 29th largest LR ranges. Many non-priority paths have larger LR ranges than those two paths. Therefore, the LR based method creates two false negatives when it infers these two paths as non-priority paths, while it creates lots of false positives by inferring those non-priority paths that have larger LR ranges as multi-priority paths. The reason for them to have large ANR range but small LR range is that in most of the bursts the loss rate difference between their high priority group and their low priority group is just one or two packets. Although their loss rate differences are small, they are persistent over all bursts, which led to the large ANR range and suggests there exists a certain preference for certain packet types. In this case, the ANR range metric makes such persistent behavior obvious while the LR range metric tends to ignore it.

Then, we examine the point (0.09, 0.12), which has a small ANR range but a relatively larger LR range. When checking this path measurement logs, we found that the receiver received 11 bursts. All the loss rate differences stem from the first burst. In that burst, every TCP packet type received only two packets, while every ICMP and UDP packet type received 40 packets. Investigating further, we found that the TCP advertised window values in TCP headers, which were set to 32768 as we sent, were rewritten to 1460, 2920 when received for the two packets in the first burst, while remained unchanged for the rest of the bursts. As a normal TCP connection usually starts with congestion windows set to one or two, we suspect that our aggressive probing method has triggered a TCP congestion control mechanism related rule in a firewall, so that all packets not accommodated within the window size were discarded. We checked all 156 paths and found 11 others have the same phenomenon, *i.e.*, large losses for TCP packets with rewritten headers in the first burst. In the real Internet environment where there are lots of hidden middleboxes, there are many transient losses which may produce a large LR range for a certain packet type in some bursts. But the ANR range metric is much more robust to such burst errors than the LR range. Thus the ANR range is more suitable for discovering the priority settings.

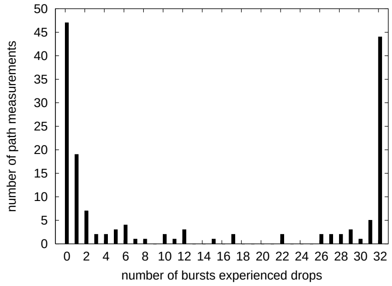

Fig. 6: Histogram of number of lossy bursts for 156 probes.

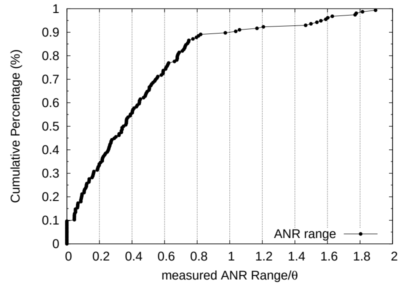

Fig. 7: The cumulative percentage of ANR range/θ.

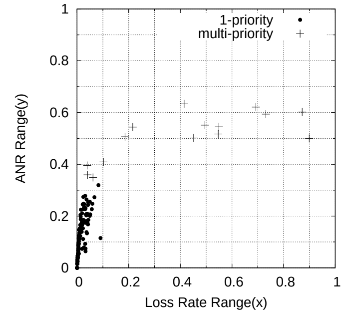

Fig. 8: Loss rate ranges v.s. ANR ranges

Finally, when the ANR range exceeds the threshold, we cluster packet types based on their ANR values. Therefore, we also want to know how the ANR values are distributed within the range. The 15 paths with largest ANR values were flagged with multiple priorities. We show them together with the next 15 paths (*i.e.*, 16th to 30th) with the largest ANR ranges in Fig. 9 and Fig. 10 to see how ANR values were distributed for both multi-priority paths and non-priority paths. The numbering of path in these figures will be used consistently throughout the rest of this paper, *e.g.*Table V. In the figures, higher priority number denotes smaller loss rates.

We define the $distance$ between two priority groups G_1 and G_2 as the minimal ANR difference between any pair of packet types i and j where $i \in G_1$ and $j \in G_2$. We define the $range$ of a priority group G as the maximal ANR difference for any pair of packet types in G. For most of the multi-priority paths, except for paths 1 and 15, the distance between any different priority group is always much larger than the maximal range of the priority groups. On the other hand, the fifteen non-priority paths in Fig. 10 not only have obviously smaller range than the top 15 paths, their ANRs are also concentrated within the range. Large distance between groups and a small range within a group are two good properties for clustering.

2) Priority Group Inference Result: Table V shows the source destination pairs, the ANR range, and the packet types information of priority groups for the 15 multi-priority paths identified. One path is partitioned to four priority groups, two paths to three groups, and all others to two groups. Except for the path 1 and 15, all other group partitions can be described concisely in the table. Four paths treated some P2P ports out of the seven P2P ports in Table IV as low priority. Although different paths set their policies based on different subsets of the seven ports, it is no surprise that all policies treated P2P ports as low priority. However, it is a little surprising to see that for the eight paths, more than half of the multi-priority paths identified are related to ICMP. Five of the paths treated ICMP as low priority, two treated it as high priority, and one treated it as medium priority. Although we have not found unanimous agreement on whether ICMP packets are treated with high or low priority, it does suggest that we have to be careful when using ICMP loss rates to estimate the network performance of TCP or UDP connections. Three paths 1 (See validation for path 1), 3 and 6 treat well-known TCP applications with high priority. When ISPs cannot over-provision their networks, it seems that giving bandwidth guarantee to well-known Internet applications is their usual solution.

Among the 15 paths, there are three pairs of paths (3,6), (5,8) and (13,14) that are worth extra attention. Each pair are bi-directional measurements between the same pair of nodes, and their priority group categorizations are the same. Given the router IP and the number of hops away from end hosts, as seen in the validation data (see Table VI), we believe that a single router on the path is responsible for the priority setting of each path pair (3,6) and (13,14), as confirmed with network operators. Take path (13,14) for example. They are both caused by the egress filtering on router `192.5.40.131` and thus the loss rate differences show up in the subsequent routers.

On the other hand, there are 9 other paths that do not have their reverse paths listed in the table, likely indicating that their priority configuration are asymmetric.

3) Effects of the number of rounds in one burst: In this section, we evaluate the number of rounds n_r needed to sent in one burst in order to infer the priority settings. Instead of probing the paths with new n_r values, we actually reuse our measurement data by only counting received packets up to a certain sending round j in one burst, ignoring the received packets sent after that round. Since POPI put the round number into the data payload for every packet, it's easy to obtain $n^m_{i(j)}$, the number of received packets for the packet type i up to the sending round j in the mth burst, and perform ANR analysis based on these values.

Fig. 11 shows that total partition errors decreases as n_r increases for the top 15 paths. The *under-partitions* are the main types of errors when n_r is small, and the number of them gradually drops from 14 at $n_r = 1$ to zero at $n_r = 38$. It suggests that we can have right partitions for some paths at small n_r, but we need to use large n_r for some other paths. This accords with our notion that for some paths with large available bandwidth, we need to send large bursts to make the priority settings show up.

C. Experiment Validation

In this section, we try to determine the accuracy of the inferred results in the previous section. Since it is very hard to get the actual router configurations on the path, we designed a hop-by-hop method to validate our inference. It measures the one-way loss rate differences to each router on the path towards the destination. For packet types from different priority groups, their loss rates should begin to differ when they reach a certain hop on the path (when packets traverse the configured router), and the difference should remain from that router onwards, in the same way as the end-to-end measurements. The bifurcation point and the persistent loss rates over several continuous hops afterward are very strong indicators to prove the existence of the priority groups. Besides, this method enables us to find the configured routers

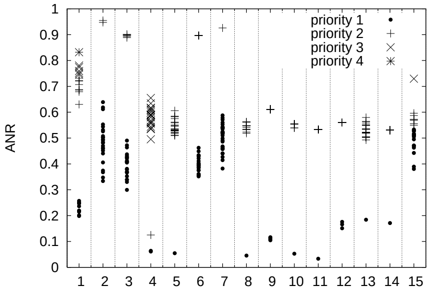

Fig. 9: Clustering results for the top 15 paths with the largest ANR range.

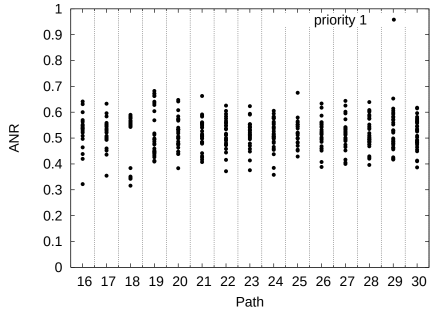

Fig. 10: Clustering results for the 16th - 30th paths.

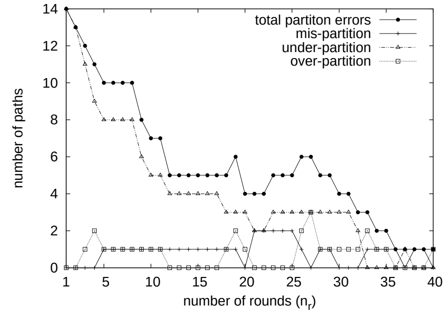

Fig. 11: The ANR partition errors v.s. n_r.

Path	Source→Destination	Range	Group Partition
1	pku2.6planetlab.edu.cn→planet2.att.nodes.planet-lab.org	0.63	1,5,8,10
2	planetlab01.erin.utoronto.ca→planetlab-3.amst.nodes.planet-lab.org	0.62	2(ICMP+Bittorrent),24
3	planetlab1.nycm.internet2.planet-lab.org→soccf-planet-001.comp.nus.edu.sg	0.60	7(APPS),22
4	lzu1.6planetlab.edu.cn→planetlab1.ls.fi.upm.es	0.59	21,1(ICMP),2(P2P)
5	planetlab2.iii.u-tokyo.ac.jp→planetlab5.upc.es	0.55	22,1(ICMP)
6	soccf-planet-001.comp.nus.edu.sg→planetlab1.nycm.internet2.planet-lab.org	0.54	7(APPS),22
7	planetlab1.ukc.ac.uk→planet2.ics.forth.gr	0.54	1(ICMP),30
8	planetlab5.upc.es→planetlab2.iii.u-tokyo.ac.jp	0.52	21,1(ICMP)
9	planetlab1.cs.colorado.edu→plab1.nec-labs.com	0.51	22,5(P2P)
10	scratchy.cs.uga.edu→planetlab1.georgetown.edu	0.50	18,5(P2P)
11	planet1.scs.cs.nyu.edu→planetlab2.net-research.org.uk	0.50	29,1(ICMP)
12	planetlab2.lsd.ufcg.edu.br→planetlab1.mnlab.cti.depaul.edu	0.41	25,3(P2P)
13	planetlab2.postel.org→planetlab1.cs.purdue.edu	0.40	27,1(ICMP)
14	planetlab1.cs.purdue.edu→planetlab2.postel.org	0.36	27,1(ICMP)
15	planetlab1.informatik.uni-erlangen.de→planetlab2.ece.ucdavis.edu	0.35	1(ICMP),7,18

TABLE V: 15 paths with multiple priorities. In the *Group Partition* column, each number represent a group and its size (*i.e.*, the number of packet types in that group). The groups are ordered by priorities with the highest priority on the left. The description for each group is enclosed in the parentheses. Except paths 1 and 15, the groups without description contain the rest of packet types probed. APP denotes well-known TCP applications.

located at the bifurcation points. We then sent email to the tech support of the configured routers for validation.

1) Hop-by-Hop Method: We designed our hop-by-hop method as follows: For each path, we send different packet types with TTL ranging from one to its number of hops (nh). For TTL in the range of [1, nh - 1], we will receive "ICMP time exceeded" packets. We then use the original probe packet information contained in an "ICMP time exceeded" packet to pair it with the original packet that triggered it.

For each hop, the difference between the numbers of the received ICMP packets for different original packet types reflects the loss rate difference on the forwarding path to that hop. This is because for all the different types of probe packets sent towards a router, the triggered "ICMP time exceeded" return packets all have the same size and will have the same loss rates on the return path. Thus the loss rate difference observed are mostly likely caused on the forward path.

For each path, we sent 1000 packets for each packet type per router. To avoid the ICMP rate limiting on many routers, we probed each hop once every second. Thus the validation of each path takes 1000 seconds. Since this method would generate a significant amount of probe traffic to the routers, for path with multiple priority groups, we selected two packet types from every priority group, unless there was only one packet type in that group. If the path did not show loss rate differences, it would be a false positive. For path with only one priority group, we selected two packet types of the largest two ANRs and two packet types of the smallest two ANRs. If there were loss rate differences, it would be a false negative.

2) Validation Results: The validation experiment took place on May 17 2006. We validated the 30 paths in Fig. 9 and 10 in order to search for both false positives and false negatives. Among the 30 paths, four paths could not be checked, and one of them is in top 15. Among the 14 paths in top 15, 13 paths are validated to have multiple priorities as shown in Table VI and 12 of them were correctly partitioned. For the 12 of the bottom 15 paths, we did not find loss rates differences for any of them. Therefore, no false negatives were found for any of the paths. The four unchecked paths, one over-partitioned path 1 and one unproven path 15 will be explained later.

For each priority router inferred in Table VI, we checked its corresponding organization using $whois$ database, and sent email to the tech support for validation. We received response for seven paths and they all confirmed our inference results. Paths 1, 3 and 6 are confirmed by their network operator as setting separate high bandwidths for typical applications. In addition, the network operator of Path 10 confirmed that they use a traffic shaper (we consider it as part of a router for forwarding functionality) close to the campus edge routers to limit P2P traffic. The other three MPPs (5, 13 and 14) confirmed by their network operators are all caused by severe ICMP traffic rate limiting on their routers

Fig. 12 gives the typical per hop loss rates for a successfully proved path: path 8. Other multi-priority paths are similar.

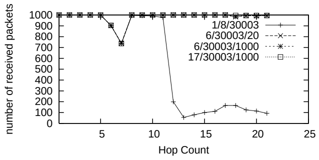

Fig. 12: Per-hop loss rates for path 8 with ICMP, TCP source port 20,1000, and UDP source port 1000

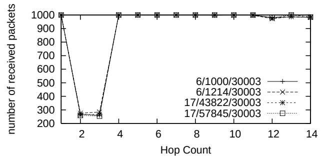

Fig. 13: Per-hop loss rates for path 24 (194.80.38.243 → 194.70.143.51) with TCP source port 1000, 1214, and UDP source port 43822, 57845

Path	#G	Router w/ OP	HC	PL	Location	Reply
1	2	202.112.61.197	6	18	China	Yes
2	2	128.100.200.97	5	17	Toronto	–
3	2	137.132.80.104	11	15	Singapore	Yes
4	3	138.100.254.18	15	16	Spain	–
5	2	84.88.18.18	23	27	Spain	Yes
6	2	137.132.3.131	5	15	Singapore	Yes
8	2	62.40.96.169	12	21	UK	–
9	2	128.138.81.134	5	15	Colorado	–
10	2	128.192.166.1	4	12	Georgia	Yes
11	2	193.63.94.6	12	13	UK	–
12	2	200.143.252.21	7	21	Brazil	–
13	2	192.5.40.53	11	15	Indiana	Yes
14	2	192.5.40.134	5	15	Indiana	Yes

TABLE VI: Summary of hop-by-hop validation results for the 13 successfully validated paths. The *Path* numbers are the same as those in Table V. *#G* is the number of priority groups shown in the hop-by-hop loss rates. *Router w/ OP* is the router where the multiple priorities are observed. Note that they may not be the routers with priority configured. *HC* is the hop count from the source to that router. *PL* is the length of the path. "–" means no reply.

There are persistent loss rate differences between ICMP and the TCP/UDP packets beginning at hop 12 with similar differences all the way to the destination, while there is no such difference before hop 12. Therefore, the configured router should be at hop 11 or 12, depending on the router configuration (ingress filtering or egress filtering). Fig. 13 shows the per-hop loss rates for a typical non-priority path, path 24. The large loss rates for hops 2 and 3 are probably due to ICMP rate-limiting.

The unchecked paths are path 7 and three others in the bottom 15. Path 7 was not checked because traceroute did not return any result beyond hop two, and the other three were not checked because the source hosts were down, and there were no other available hosts at those institutes either.

For the over-partitioned path 1, packet types from the three high priority groups (1,5,8) show no loss rate differences. It has been confirmed by its operator that all these ports (UDP, ICMP and the well-known TCP application ports such as 20, 21, 110, 179, 443 and etc.) are set to one high priority. This is because the ANR method is a statistical method and cannot guarantee 100% correctness. As shown in Fig. 9, their ANRs range from 0.6 to 0.9 and are very close to each other.

For the unproven path 15, we did not observe loss rates difference for all routers on the path. In the experiment, its ANR was just above the threshold θ when detected as a MPP. Besides, when we measured it again with POPI for validation, its ANR is less than the threshold. Thus we believe it is a false positive in our detection.

VI. Conclusions and Future Work

In this paper, we introduced POPI for end-to-end inference of packet forwarding priority on routers. The tool is available at [17]. We evaluated POPI through both simulations and PlanetLab experiments, and discovered several multi-priority paths in the Internet. Further hop-by-hop validation and survey of network operators confirm our inferences. We believe that some of POPI design principles can be applied to other Internet measurement applications.

As our work is the first step to discover router packet forwarding priorities in the Internet, POPI currently only uses packet losses as the inference metric. For future work, we plan to use other metrics for inference such as packet order. In addition, we also plan to have senders dynamically adjust burst sizes, based on the loss rates feedback from the receivers, in order to reduce the amount of probe traffic.

Acknowledgments

We would like to thank Vern Paxson for his discussion and encouragement at early phase of this work. We are also grateful to Steve Muir and Mark Huang who kindly helped us in the early development of POPI on PlanetLab. In addition, we thank Junxiu Lu, Lanjia Wang and the anonymous reviewers for their helpful comments. This work was partly supported by China 863 Program under grant 2006AA01Z201130 and the DOE CAREER Award DE-FG02-05ER25692.

References

[1] Cisco System, "Cisco ios quality of service solutions configuration guide release 12.2." [Online]. Available: http://www.cisco.com/univercd/cc/td/doc/product/software/ios122/122cgcr/fqos_c/

[2] Juniper Networks, "Junose 7.0.x policy and qos configuration guide." [Online]. Available: http://www.juniper.net/techpubs/software/erx/junose701/bookpdfs/swconfig-policy-qos.pdf

[3] P. Grant and J. Drucker. (2005) Phone, cable firms rein in consumers' internet use. [Online]. Available: http://online.wsj.com/article/SB112985651806475197.html

[4] R. Mahajan, N. Spring, D. Wetherall, and T. Anderson, "User-level internet path diagnosis," in *ACM SOSP*, 2003.

[5] K. Harfoush, A. Bestavros, and J. Byers, "Robust identification of shared losses using end-to-end unicast probes," in *ICNP*, 2000.

[6] D. Rubenstein, J. Kurose, and D. Towsley, "Detecting shared congestion of flows via end-to-end measurement," *Networking, IEEE/ACM Transactions on*, vol. 10, no. 3, pp. 381–395, 2002.

[7] M. S. Kim, T. Kim, Y. Shin, S. S. Lam, and E. J. Powers, "A wavelet-based approach to detect shared congestion," in *SIGCOMM*, 2004.

[8] A. Coates, A. Hero III, R. Nowak, and B. Yu, "Internet tomography," *Signal Processing Magazine, IEEE*, vol. 19, no. 3, pp. 47–65, 2002.

[9] T. Bu, N. Duffield, F. Presti, and D. Towsley, "Network tomography on general topologies," in *ACM SIGMETRICS*, 2002.

[10] M. Rabbat, R. Nowak, and M. Coates, "Multiple source, multiple destination network tomography," in *INFOCOM*, 2004.

[11] S. Savage, "Sting: a tcp-based network measurement tool," in *Proc. of the 3rd USENIX Symposium on Internet Technologies & Systems (USITS)*, June 1999.

[12] K. Anagnostakis, M. Greenwald, and R. Ryger, "cing: Measuring network-internal delays using only existing infrastructure," in *IEEE INFOCOM*, 2003.

[13] J. D. Gibbons, *Nonparametric Statistical Inference*. Marcel Dekker, Inc, 1985.

[14] G. E. Noether, *Introduction to Statistics: A Nonparametric approach*. Houghton Mifflin Company, 1976.

[15] G. Lu, Y. Chen, S. Birrer, F. E. Bustamante, C. Y. Cheung, and X. Li, "End-to-end inference of router packet forwarding priority," Northwestern University, Tech. Rep. CUCIS-2006-07-001, July 2006.

[16] S. Floyd and V. Jacobson, "Link-sharing and resource management models for packet networks," *Networking, IEEE/ACM Transactions on*, vol. 3, no. 4, pp. 365–386, 1995.

[17] http://list.cs.northwestern.edu/popi/.