

# POPI: A User-level Tool for Inferring Router Packet Forwarding Priority

Guohan Lu, Yan Chen, *Member, IEEE*, Stefan Birrer, *Student Member, IEEE*,  
Fabián E. Bustamante *Member, IEEE*, Xing Li

**Abstract**—Packet forwarding prioritization (PFP) in routers is one of the mechanisms commonly available to network operators. PFP can have a significant impact on the accuracy of network measurements, the performance of applications and the effectiveness of network troubleshooting procedures. Despite its potential impacts, no information on PFP settings is readily available to end users. In this paper, we present an end-to-end approach for PFP inference and its associated tool, POPI. This is the *first* attempt to infer router packet forwarding priority through end-to-end measurement. POPI enables users to discover such network policies through measurements of packet losses of different packet types.

We evaluated our approach via statistical analysis, simulation and wide-area experimentation in PlanetLab. We employed POPI to analyze 156 paths among 162 PlanetLab sites. POPI flagged 15 paths 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 all confirming our inferences.

Besides, we compared POPI with the inference mechanisms through other metrics such as packet reordering (called *out-of-order (OOO)*). OOO is unable to find many priority paths such as those implemented via traffic policing. On the other hand, interestingly, we found it can detect existence of the mechanisms which induce delay differences among packet types such as slow processing path in the router and port-based load sharing.

**Index Terms**—Packet forwarding priority, inference, packet loss, packet reordering

## I. INTRODUCTION

The Internet was designed with no gatekeepers over new content or services. A lightweight but enforceable neutrality rule is needed to ensure that the Internet continues to thrive.

— Vint Cerf

Packet forwarding prioritization has been available in off-the-shelf routers for quite a while, and various models from popular brands, such as Cisco and Juniper Networks offer support for it [1], [2]. 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, on the accuracy of measurement tools' output, and on the effectiveness of network troubleshooting procedures.

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 by measuring the loss rate difference of different packet types and its associated tool, POPI, which is the *first* such attempt to our best knowledge. This tool can be used by the enterprises or end-users to discover whether their traffic are treated differently by the ISPs, and whether the ISPs has fulfilled the contracts between them and the users. For example, after Comcast was found to treat BitTorrent with low priority in 2007, Comcast and BitTorrent reached an agreement to work together on network traffic in 2008 [4], [5].

There are a couple of challenges for designing and implementing POPI. First, background traffic fluctuations can severely affect the end-to-end inference accuracy of router properties. Secondly, probe traffic of a relatively large packet bursts are neither independent nor strong correlated. Most existing inference methods have to assume certain independence (i.e. i.i.d. processes) or strong correlation models for inference (e.g. back-to-back probe packets). However, as for the relatively large packet bursts sent by POPI, a good mathematical model is needed to determine whether the loss rates difference between two packet types is the consequence of a random effect or being treated really differently. Thirdly, we want to measure more than two packet types at the same time, so simply determining whether they are treated differently is not enough.

To overcome these challenges, POPI takes the following three steps to infer packet forwarding priority inference. First, it sends a relatively large amount of traffic to temporarily saturate the bottleneck traffic class capacity, which gives POPI better resistance against background traffic fluctuations. Secondly, we apply a robust *non-parametric* method based on the *ranks* instead of pure loss rates. Thirdly, we assign a rank-based metric to each packet type and use a hierarchical clustering method to group them when there are more than two packet types. We also believe some of these schemes can be applied in other Internet measurement applications.

We validated our approach via statistical analysis, ns2 simulation and wide-area experiments in the PlanetLab testbed. We chose 26 packet types on various protocols (ICMP, TCP and UDP) and various port numbers associated with traditional and P2P applications in addition to some random ports. Tested over 156 directional paths among 162 PlanetLab sites, POPI identified 15 paths flagged with multiple priorities, 13 of which were validated using a hop-by-hop measurement method. After surveying all related network operators, we received responses

Manuscript received January 20, 2002; revised November 18, 2002. This work was supported by the IEEE.

G. Lu and X. Li are with the Tsinghua University, P.R.China

Y. Chen, S. Birrer and F. Bustamante are with the Northwestern University

for seven of them all confirming our inferences.

As an extension of our earlier paper [6], we compared POPI with the inference mechanisms based on other metrics with less overhead such as packet reordering (called *out-of-order (OOO)*) in this paper. OOO is unable to find many priority paths such as those implemented via traffic policing. On the other hand, it can detect existence of the mechanisms which induce delay differences among packet types such as slow processing path in the router and port-based load sharing.

The rest of the paper is organized as follows. We review related work in § II before discussing the design space and challenges in § III. The design of POPI mechanisms is presented in § IV. We evaluate our POPI tool with both NS simulation (§ V) and the PlanetLab experiments (§ VI). Furthermore, we discuss the use of other metrics such as packet reordering and delay for inference in § VII. We conclude and summarize our findings in § VIII.

## II. 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 [7]–[9]. 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. 26 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 [7]–[9]. 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.

[10] proposed a framework for enabling network clients to measure a system’s multi-class mechanisms and parameters. The basic idea is similar to ours, i.e., to inject multi-class traffic into the system and use a statistical method to infer its scheduling types and parameters based on the output. However, the technique did not consider cross-traffic effects and only simulation results were presented.

PFPI inference also has some goals in common with efforts on network tomography [11]–[13]. 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, [14] is a content-based method to detect the middleboxes which modify a web page’s content. The method in this paper detect the middleboxes which generate packet loss differences but do not modify the contents of the packets.

## III. INFERRING PACKET-FORWARDING PRIORITY

### A. Background on Priority Mechanisms

Network administrators can enforce priority/link-sharing mechanisms in a router by defining a traffic class (usually IP protocol and TCP/UDP port number) and associating with it a particular queuing/scheduling mechanism [1], [15]. Some of the commonly available mechanisms are as follows.

- *Priority Queuing (PQ)*. This allows users to assign arbitrarily defined packet classes to queues with different priorities. Since queues are served based on their priority, this allows specified packet types to be always sent before other packet types.
- *Proportional Share Scheduling (PSS)*. With PSS each traffic class is given a weight. Bandwidth is allocated to classes in proportion to their respective weights. There is no strict priority difference between classes. There are different ways to implement this scheduling mechanism, e.g. Weighted Fair Queuing (WFQ), Weighted Round-Robin (WRR). In Cisco routers, the CBWFQ is Class-Based WFQ and the Custom Queuing is WRR based [1].
- *Policing*. This restricts the maximum rate of a traffic class. Traffic that exceeds the rate parameters is usually dropped. The traffic class cannot borrow unused bandwidth from others.

Only the first mechanism sets absolute priorities between traffic classes. There is no absolute priority difference between the other two classes, and the loss experienced by one class depends on whether its traffic rate exceeds its allocated bandwidth.

### B. Choosing Inference Metric

Three basic end-to-end performance metrics, loss, delay and out-of-order, can all be used as inference metrics. This is because these metrics of different packet types can become different when a router is configured to treat them differently. Consider a *PQ* of two priorities, where the high priority queue is always served first. Low priority packets will experience larger loss rates and longer queueing delays than the high priority packets. Besides, a low priority packet may arrive earlier than a high priority packet but leave after it while the contrary will never happen. The reordering events between them are *asymmetric*. Here, the loss, delay, and reordering can all be used as a metric to infer priority settings. In this paper, we’ll use the loss, delay and OOO (Out-Of-Order) based method to name the inference methods based on these metrics.

Essentially, the delay and reordering metrics are equivalent because when a packet gets lagged behind another packet, its delay should be larger than the other. In the following, we discuss the *pros* and *cons* between loss metric and the other two metrics and the reason why we choose packet loss eventually.

**1. The probe overhead of packet loss metric is larger than the other two.** Obviously, loss rates difference will not

become evident until the associated link (or a sub-link for a traffic class) is saturated and begins to drop packets. This simple observation defines the basis of loss-based inference approach: *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.* On the other hand, packet reordering and delay differences can be observed as soon as queue begins to build up. We do not need to send as much traffic as the loss-based approach for the reordering and delay-based approach.

**2. Loss difference can be observed for all kinds of QoS mechanisms while the other two cannot.** Although using delay and reordering metrics can result in less probe overhead, they can not detect certain router QoS mechanisms simply because those mechanisms do not generate different delays at all. According to our test on a real Cisco router, *Policing* does not generate any packet reordering nor delay differences. However, any kinds of router QoS mechanisms will ultimately generate loss rates differences because that is the purpose of configuring such mechanisms. In § VII, we found that many multi-priority paths (MPPs) detected by the loss-based method could not be detected the other two methods.

**3. Packet delay difference can be caused by many other mechanisms than QoS.** As noted in [16], the root cause of packet reordering is the existence of parallel packet forwarding paths. Such paths can be in a router, parallel links between two routers, or different routes over a number of hops. When packets are split to these parallel paths according to their packet types and these paths have different delays, we'll observe asymmetric packet reordering and delay differences among different packet types. In § VII, we show that many paths showing differences in packet delay and reordering were not caused by router QoS mechanisms, but the various forms of parallel forwarding paths such as slow processing inside routers and port-based load sharing.

To sum up, although the loss-based method has larger probe overhead, it can detect all kinds of QoS mechanisms and is not likely to be affected by parallelism in the forwarding paths as the other two. Hence, we use packet loss as the metric for inference.

### C. Challenges for POPI

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

**I. 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 or not depends solely on the amount of background traffic. To make our approach more resistant to background traffic fluctuations we opt for sending relatively large amount of traffic to temporarily saturate bottleneck traffic class capacity, which increases the probability of observing loss rates difference. To note, the sender may not be able to saturate the bottleneck link due to limited resources, which is an inherit limitation of this method.

**II. Probe traffic of a relative large packet bursts are neither independent nor strongly correlated.** Once the loss rate for each packet type is obtained, we need to determine

whether the loss rates difference among them is large enough to conclude that they are treated differently. When packet losses can be described with a good mathematical model, e.g. independent and identical distribution (i.i.d) process, we can determine if the loss rates of different packet types were evidently different or not by comparing all  $p_i$ , the loss rate of packet type  $i$ , using parametric statistical methods. However, our probe packets are sent in large packet bursts. Packet losses in one burst are not independent but correlated. We are not aware of any well-known model for packet losses in a large packet burst in the Internet. Hence, we employ a *non-parametric* method based on *ranks*, which is independent to underlying packet loss model and insensitive to loss correlations.

**III. Grouping is needed for multiple packet types probing.** If we only probe two packet types at one time, simply determining whether they are treated differently is enough. However, we sometimes probe more than two packet types and need to group them based on their priorities. Here, we assign a rank-based metric to each packet type and use hierarchical clustering method to group them.

In summary, POPI saturates the link with relatively large amount of traffic and clusters packet types based on their loss ranks. Such an approach gives POPI better resistance against background traffic fluctuations, allows it to cope with the inherent characteristics of its measurement traffic, and enables it to measure more than two packet types at one time.

## IV. DESIGN OF POPI

In this section, we present the design of the POPI tool which has three steps: 1) probing the path; 2) deriving ranks; and 3) partitioning packets with different forwarding priorities to different groups based on their ranks.

### A. Probing the Path

Fig. 1 illustrates our link probe method. We want to test  $k$  packet types. POPI sends a number of bursts ( $n_b$ ) from a source to a destination. The interval between bursts is  $\Delta$ . Each burst consists of  $n_r$  rounds, in which  $k$  packets, 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,  $\Delta$ ,  $n_b$  and  $n_r$ . 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,  $\Delta$  should not be too small. On the other hand, in order not to experience large background traffic fluctuation duration the probe, we need to keep the whole probe duration within a relatively short period. In practice,  $\Delta$  is set to one to ten seconds to keep overall probe duration within several minutes. We'll discuss  $n_b$  in § IV-D and  $n_r$  in § V.

The probe overhead is comparable with current available bandwidth measurement tools such as Pathload. In our experiment,  $n_b = 32$ ,  $n_r = 40$  and  $\Delta = 10$ . Usually, the user only needs to compare two or three packet types. Hence when  $k = 2$ , 2560 packets are sent in one probe during 320 seconds. As for the Pathload, it usually sends  $K \times N = 100 \times 12 = 1200$  packets in one probe during 15 seconds [17].

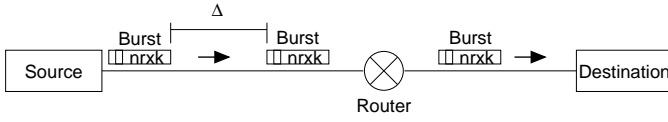


Fig. 1: A burst consists of  $n_r \times k$  packets

Notation	Meaning
$n_b$	number of bursts in a path measurement
$n_r$	number of rounds in one burst
$\Delta$	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
$\theta$	threshold used for comparing with $ANR$ range

TABLE I: Key notations

## B. Deriving Ranks

For every burst, loss rate ranks are computed by first sorting packet types in ascending order according to their packet loss rates 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, and etc on.<sup>1</sup>

Similar to packet loss rates, due to randomness of packet losses, the ranks of different packet types are like random arrangements over the all bursts when the packet types are treated equally. On the other hand, the ranks of certain packet types are always small when they are treated with low priority.<sup>2</sup> However, the advantage of using ranks is that we have a theory to bound the variance of loss ranks caused by the random effects whereas we do not have that bound for loss rates if the loss model is unknown.

## C. Partitioning Based on Ranks

Every packet burst can be regarded as an observation. Identifying whether there is consistent difference among  $k$  ranks over  $n$  observations is a well-known statistical problem called *problem of  $n$  rankings* [19]. Classic non-parametric solutions such as the Friedman test [18] can find whether there is consistent difference, but they do not make partitions among packet types. Therefore, we proposed to use Average Normalized Ranks ( $ANR$ ) to group packet types when there is consistent difference. The  $ANR$  is the average of the ranks for a packet type over all bursts. Our statistical method is as follows:

- 1) *Calculate ANR.* Let  $r_i^m = (1, 2, \dots)$  denote the rank for packet type  $i$  in  $m$ th 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

$$ANR_i = \left( \sum_{m=1}^{n_b} NR_i^m \right) / n_b. \quad (1)$$

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  $n_b$

<sup>1</sup>For tie breaks, we use the mid-ranks method [18] to distribute the total ranks equally among them.

<sup>2</sup>Our method is robust in the case when a fraction of bursts does not satisfy the assumption discussed in § IV-D.

bursts at confidence level  $1 - \alpha$  is

$$\theta_{1-\alpha, k_j, n_b, k} = Q_{1-\alpha, k_j} \times \sqrt{k_j^2 - 1/k\sqrt{12n_b}}, \quad (2)$$

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 Appendix. ■

According to this model, when  $R > \theta_{1-\alpha, k_j, n_b, k}$ , those packets should belong to multiple groups.

- 2) *Partition groups based on ANR.* We use a hierarchical divisive partition approach to cluster ANRs. Initially, we assume all packet types belong to one group, and 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 [20]. This procedure is applied recursively to all newly partitioned groups until  $R \leq \theta$  or there is only one packet type in the group. The pseudo code of the partition procedure is shown as below.

---

### Procedure 1 $ksplit(anrs)$

---

- 1:  $k_j \leftarrow \text{len}(anrs)$
  - 2: **if**  $k_j = 1 \mid |(\max(anrs) - \min(anrs)) \leq \theta_{1-\alpha, k_j, n_b, k}|$  **then**
  - 3:     **return**  $anrs$
  - 4: **else**
  - 5:      $[anrs_1, anrs_2] = \text{kmeans}(anrs, 2)$
  - 6:     **return**  $[ksplit(anrs_1), ksplit(anrs_2)]$
  - 7: **end if**
- 

## D. Performance Analysis on Priority Group Partitioning

1) *Methodology:* 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  $G_u$ , and  $\forall$  packet type  $j \in$  priority group  $G_v$ , the loss rate rank  $r_i > r_j$  when  $G_u$  has higher priority than  $G_v$ .
- 2) For packet types within the same priority group, their ranks are randomly permuted in each burst in order to simulate the effects of random losses.

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

2) *Cluster error types:* Group partition 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 is less than the confidence level  $\alpha$  we choose to calculate  $\theta$ .
- 2) *under-partitioning:* The number of partitioned groups is less than that of the actual situation which results

$\alpha$	$n_b$					
	type	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  $J = 2$ 

$J$	$n_b$				
	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  $J = 3, 4, 5$ ,  $\alpha = 0.0001$ 

from the type II error of the criteria. Theorem 2 in the Appendix denotes that when  $n_b$  is above certain value, i.e. larger than 12 for  $k = 32$ , the percentage of type II errors 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.

3) *Average cluster error percentage*: 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 different  $n_b$  under  $\alpha = 0.01$  or  $0.001$ . To note, we do not show the percentages for the third type of errors since they are all zero.

As shown in this table, we should at least choose  $n_b > 8$  for our experiments since both  $\alpha$  have a large error percentage when  $n_b = 8$ . Besides, we will use  $\alpha = 0.001$  for our experiments since both  $\alpha = 0.01$  and  $\alpha = 0.001$  have 0% *under-partitioning* when  $n_b \geq 16$  but  $\alpha = 0.001$  has smaller percentage of *over-partitioning*. The results also agree well with our theoretical analysis, i.e. there is no *under-partitioning* when  $n_b \geq 16$  and the percentages of *over-partitioning* are close to the confidence level we set for  $n_b \geq 32$ .

When  $J > 2$ , the number of  $(k_1, k_2, \dots, k_J)$  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 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  $J$  increases, the number of such operations required 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  $n_b \geq 16$ .

4) *Effects of background traffic fluctuations*: The background traffic may not be stable during the probe. Consider an extreme case where the background traffic is ON/OFF traffic.

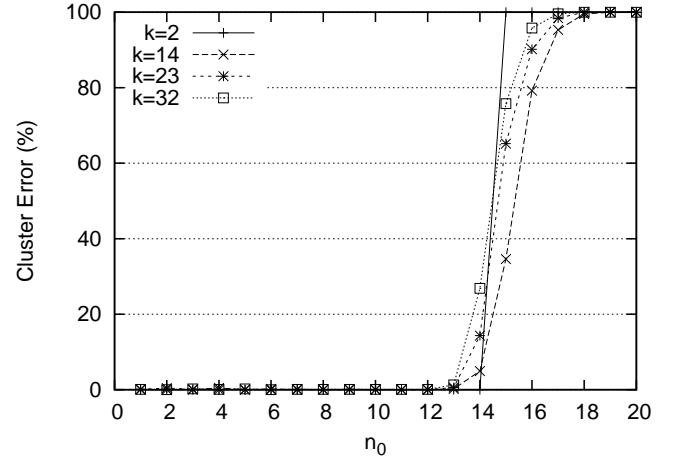
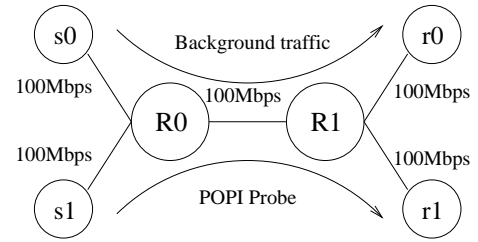
Fig. 2: Percentage of overall cluster errors for different  $n_0, k, n_b = 32$ .

Fig. 3: NS-2 topology

Suppose when a probe burst is sent during the ON period, the loss rates measured are well-separated. When a probe is sent during the OFF period, no loss rate difference is observed (as the link is not 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. 2 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.

## V. EVALUATION WITH NS-2 SIMULATION

We implemented POPI in NS-2. We use a dumbbell topology as shown in Fig. 3. The output queue of  $R0 \rightarrow R1$  is configured with *PQ* or *PSS* using CBQ (Class-Based Queueing) [21]. If not specified,  $R0$  is configured with *PQ* of two priority classes. Class 1 is the high priority class and class 2 is the low priority class. The queue length of high and low priority queues are 20 and 60, respectively. In the experiment, the size of both POPI packets and background packets are 1000 bytes. The probe traffic rate is 100Mbps. We always send 32 packet types and set  $n_b = 32$ .

As the POPI sends a packet burst in a very short period, it is the burstiness of background traffic over small time scales (less than 1s) rather than LRD over large time scales that matters. However, the traffic model over short time scales is still not clearly understood [22]. In our simulations, we use

*Gamma* traffic with shape parameter  $\gamma < 1$ , which has larger burstiness than Poisson traffic at small time scales in order to resemble realistic Internet traffic.

### A. The Effects of Probe Burst Size

The background traffic is 10Mbps *Gamma* with shape 0.5, which consists of equal share of high and low priority traffic. Fig. 4 shows how the partition results are affected as  $n_r$  increases for various  $(k_1, k_2)$  combinations. Every point in the figure denotes the result of a simulation probe. 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, 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 begin to experience losses, but since the losses are insufficient for POPI to properly classify the packets, the partition results are 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 the low priority group only experiences sporadic drops in some bursts. For most of the bursts, it shows no difference in terms of drops from the high-priority packet types, thus POPI still clusters them with high priority packet types, resulting in *under-partitioning*. As  $n_r$  increases, certain class 2 packet types have drops in almost every burst, and are clustered as the low-priority group, while some other class 2 packet types still do not have sufficient drops to be clustered as low-priority group, since their  $ANR_i$  are in the middle of the typical  $ANR$  of class 2 and the typical  $ANR$  of class 1. Those intermediates are clustered as a separate group when judged against the criteria, which results in *over-partitioning*.

Finally, POPI enters phase 3 in which the loss rates are well-separated, and POPI's accuracy solely depends on the performance of the cluster method, which has an error percentage below 1%, as shown in previous section. We also did simulations for the *PSS* queue configuration and observed the same three phases too.

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 to estimate the  $n_r$  for our PlanetLab experiments. In our PlanetLab experiments, we also probe at 100Mbps with a similar number of 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. We treat this result as a rule of thumb, and use  $n_r = 40$  in PlanetLab experiments.

### B. The Effects of Background Traffic Rate

In this experiment, we show the performance of using a fixed  $n_r$  under different background traffic rates, i.e., 10, 20, 40, 80, 90Mbps. For each rate, we let  $k_1 = 1, 2, 3, \dots, 31$ . Once the background traffic rate and  $(k_1, k_2)$  are fixed, we run 10 simulations. Altogether, 1550 simulations are performed and all of them are correctly partitioned. It shows that our algorithm works well under different available bandwidth. No

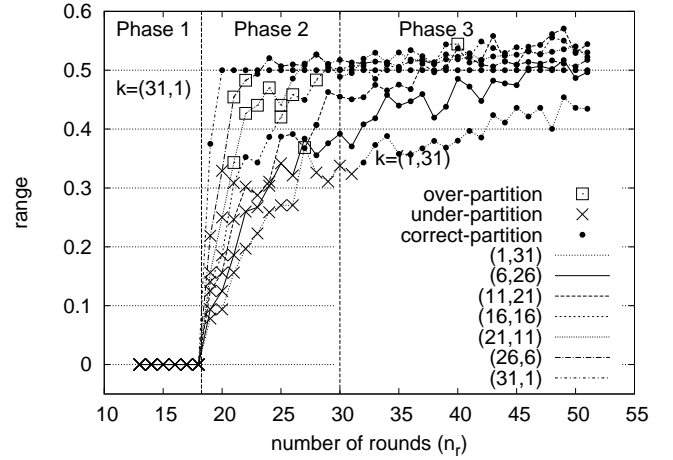


Fig. 4: Partition results for Priority Queuing. Available bandwidth = 90Mb.

matter how large is the background traffic, the low priority packets are always dropped prior to the high priority packets. Thus we can always observe the loss differences between low and high priority packets.

### C. Proportional Share Scheduling

We show how the different queuing disciplines affect the our algorithm in this experiment. We use *PSS* and define two traffic classes in *R0*. The bandwidth allocation ratios for the two classes are 0.2 and 0.8 respectively. The background traffic rates are 2 and 8Mbps. We let  $k_1 = 1, 2, 3, \dots, 31$ . For every  $(k_1, k_2)$  combination, we run 10 simulations. Altogether, 310 simulations are performed and 10 of them result in under-partitions. All of the under-partitions come from  $k_1 = 7$ . For *PSS*, whether a class experiences packet drops at the router depends on its input traffic rate and allocated bandwidth. Let  $T_i$  be the traffic rate of class  $i$  during the probe. Simple calculation reveals that when  $k_1 = 7$ ,  $T_1 = 24$ Mbps and  $T_2 = 86$ Mbps. Both two classes exceed their allocated bandwidth and experience packet drops. Thus, our algorithm can not differentiate them. As for  $k_1 \neq 7$ , only one of the two classes exceeds its allocated bandwidth and loss rates differences can be observed. However, only certain traffic pattern can trigger such behavior which should not be common in normal measurements.

## VI. PLANETLAB EXPERIMENTS

### A. POPI and its Two Probe Modes

POPI works in two probe modes, *End-to-End Probe (EEP)* and *Hop-by-Hop Probe (HHP)*. In both modes, the sender sends multiple packet types toward the receiver. The receiver feedbacks certain information of every received packet to the sender, which is used by the sender to measure the end-to-end losses and reordering events along the path. *EEP* mode works exactly as the way described in Fig. 1.

*HHP* mode is used to locate the configured router or device by measuring the losses and reordering events to every router on the path towards the destination (including the destination). It sends  $n_b$  cycles. In each cycle, it sends  $n_h$  bursts whose TTLs increase from one to  $n_h$ , which is the total hop count from sender to receiver. For  $TTL = n_h$ , POPI measures

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, 1000, 12432, 38523, 57845 (random)
UDP	110, 179, 161 (SNMP) 1000, 12432, 38523, 57845 (random)

TABLE IV: 26 packet types considered for PlanetLab experiments.

the end-to-end packet losses and reorderings based on the feedback from the receiver. For  $TTL < n_h$ , POPI calculates the losses and reorderings up to a hop by counting time-exceeded ICMP responses from that router. When packets do not traverse the configured box, we will not observe packet loss or reordering difference. After packets traverse the box, the loss or reordering difference will be similar to that observed at the receiver, and will exhibit over the remaining hops. Once we observe such phenomenon, the configured box should be around the *spot of difference*, the hop at which the difference begins to show.

To note, the losses and reorderings of ICMP responses actually include round-trip effects. However, as the response packets were all "ICMP time exceeded" packets of a same packet size, it is very unlikely that any router on the reverse path is going to treat them differently. Hence, even when there are losses or reorderings on the reverse link, the effects are unlikely to introduce bias against a specific packet type.

### B. Packet Types Tested

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. We selected 26 packet types as listed in Table IV. For UDP and TCP packets, 30002 is used as the destination port, because it is very unlikely that ISPs will set an explicit priority policy based on it. The port numbers listed in Table IV are used as source ports to measure the source port based priority policy. (Destination port based policy can be measured in a similar manner). These packet types are selected to check:

- Whether ICMP, TCP and UDP packets are handled with equal priority.
- 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 is not included because it is 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.

### C. Experiments and Data

We conducted the first run,  $\mathcal{N}_1$ , among 162 PlanetLab nodes on May 12, 2006 to test the loss-based method. Because the data set of  $\mathcal{N}_1$  only recorded loss information, we conducted the second run,  $\mathcal{N}_2$  on Jun 22, 2007 among 132 PlanetLab nodes in order to compare the loss-based method with the OOO and delay-based method. As the loss-based result of  $\mathcal{N}_2$

is very similar to  $\mathcal{N}_1$  and the result of  $\mathcal{N}_1$  had been carefully validated, we'll discuss the loss-based results for  $\mathcal{N}_1$  and only the comparison results for  $\mathcal{N}_2$ . Together, those nodes 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.

In each experiment we randomly paired the selected nodes and probed the paths in both directions for every pair of nodes. We ran POPI with  $k = 26, n_b = 32, n_r = 40$  and  $\Delta = 10$  seconds. For each path, the measurement took 5 minutes, and 33,280 packets were sent (each with size 1500 B). Thus although the burst of probe traffic is quite intensive, the average bandwidth consumption is just 1.04 Mbps, well below the typical 100 Mbps capacity.

After completing our measurements, we gathered the packet dump files from the receiver nodes. For  $\mathcal{N}_1$ , we were only able to collect results from 156 of the 162 paths measured and we use those paths for the analysis. Among the 26 packet types measured, some of them were (usually one or two packet types) banned along the path as no packet was received at the receiver during the probe. We excluded them from the priority group inference.

### D. Validation Method

Since it is very difficult to get the actual router configurations on the path, we use *HHP* method to locate the *spot of difference*. We find its corresponding organization using *whois* database, and then send emails to the related technical support for validation. To minimize the traffic sent to routers, we usually chose three to six packet types according to the group pattern and set  $n_r$  to 10.

To enquire the network operator, it is always better for us to ask the question as specific as possible. When we locate the spot of difference only along one direction, the configured box can be either at the spot or one hop before it depending on whether the rule is applied to the input queue or output queue.

When both directions of a path exhibit a same group pattern, *HHP* locates one spot of difference along each direction. As there is usually only one configured box along the path, the distance between the two spots can help us to point out the box and its type more precisely. As shown in Fig. 5(a), the two spots point to a same router when the rules are configured at the input queues of the router. The two spots are two hops away from each other when the rules are configured at the output queues as shown in Fig. 5(b). In the experiment, we observed many cases that the two spots were just one hop away. It could be that the rules were configured at the output queue of two adjacent routers as shown at the left of Fig. 5(c), or that there is a middlebox between two routers as shown at the right of the figure. We cannot tell exactly which type is from *HHP*, but the responses from network operators all indicated middleboxes.

When the configured router or routers around it are unresponsive, we may only locate a range for the spot of difference. However, as long as the range was not too large, we still wrote emails to the related network operator.

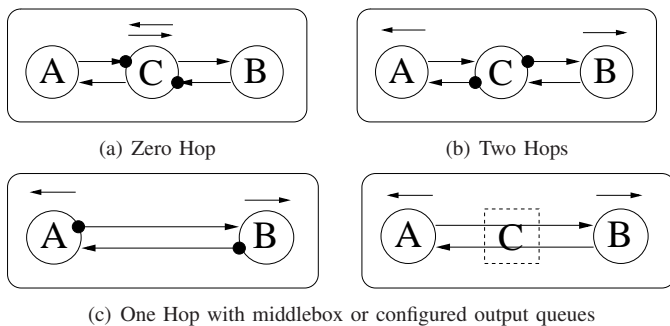


Fig. 5: Hops between two spots of difference in HHP probes. Small black circles are the places of configuration rules. Short arrows above denote the spots of difference and probe directions.

### E. Results of $\mathcal{N}_1$

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 § IV-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 is accordance with our notion that traffic remains 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  $\theta$  used for partitioning of the 156 paths. As discussed in § IV-D, we use a confidence level  $\alpha = 0.001$  to calculate the threshold  $\theta$ . 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 mentioned in § IV-C, using the same  $\alpha$ . As shown in Table V, the two methods are almost equivalent. 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

ANR Range		#G > 1	#G = 1
Friedman			
#Group > 1		14	2
#Group = 1		1	139

TABLE V: Comparison between ANR Range test and Friedman test

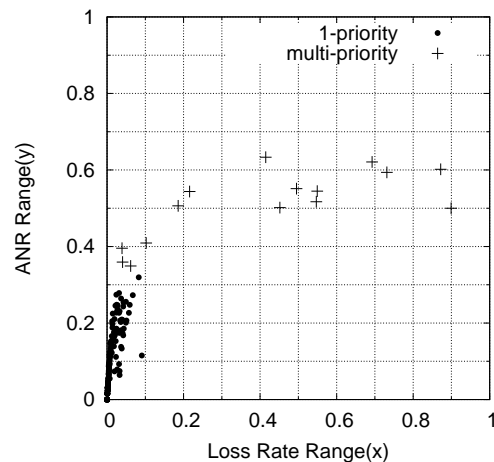


Fig. 8: Loss rate ranges v.s. ANR ranges

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  $\theta'$  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 MPPs, 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 VI. 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 MPPs. 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 leads 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

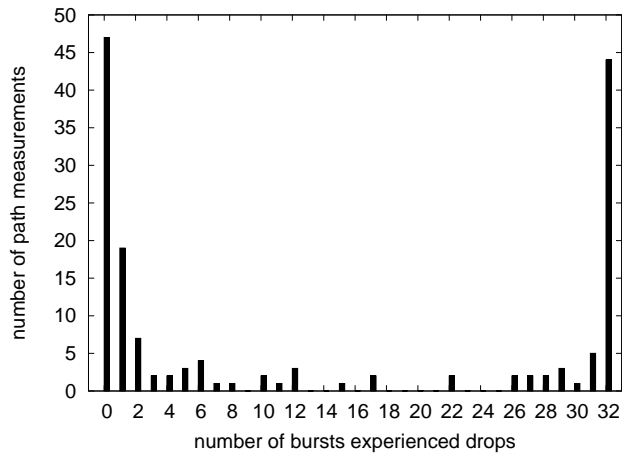


Fig. 6: Histogram of number of lossy bursts for 156 probes.

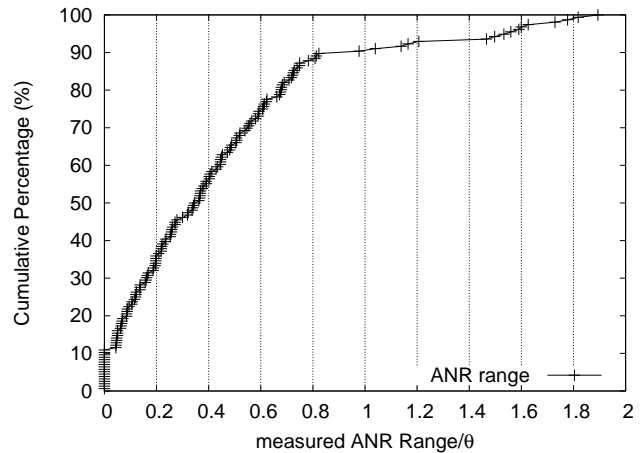


Fig. 7: The cumulative percentage of ANR range/ $\theta$ .

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.

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 MPPs and non-priority paths. The numbering of path in these figures will be used consistently throughout the rest of this paper, e.g. Table VI. 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 MPPs, 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 VI shows the source destination pairs, the ANR range, and the packet types of priority groups for the 15 MPPs identified. Three paths were partitioned to three priority 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's not a surprise to see 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 MPPs 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 one of their solutions.

Among the 15 paths, there are three pairs of paths (3,6), (5,8) and (13,14) that are worth extra attention. Each pair contains 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 VII), 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 configurations are asymmetric.

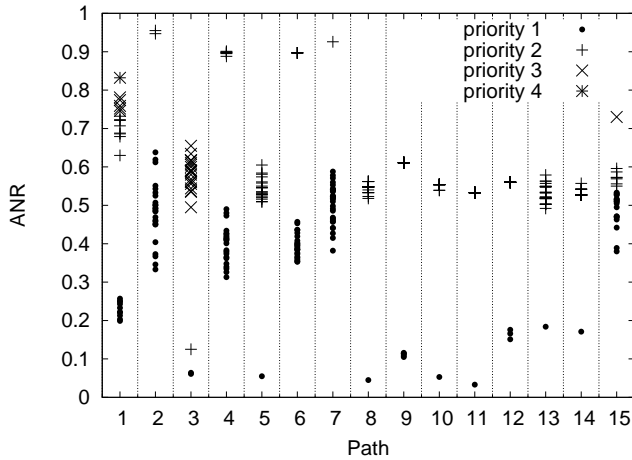


Fig. 9: Clustering results for the top 15 paths with the largest ANR range.

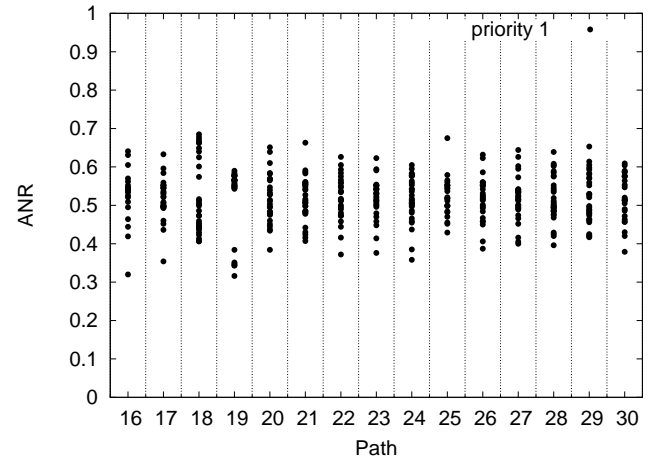


Fig. 10: Clustering results for the 16th - 30th paths.

Path	Source→Destination	Range	Group Partition
1	pku2.6planetlab.edu.cn→planet2.att.nodes.planet-lab.org	0.63	6,8,10
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 VI: 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.

3) *Effects of the number of rounds in one burst*: In this section, we evaluate the number of rounds  $n_r$  needed to send 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  $n_{i(j)}^m$ , the number of received packets for the packet type  $i$  up to the sending round  $j$  in the  $m$ th 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. We use the validation results in the following section as the baseline to compare. 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.

## F. 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 VII 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 VII, we checked its corresponding organization using *whois* database, and sent email to the tech support for validation. We received responses 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

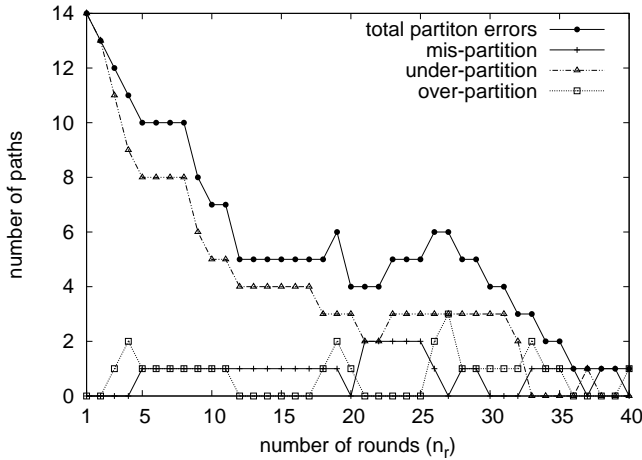
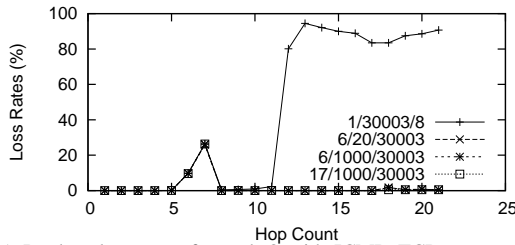
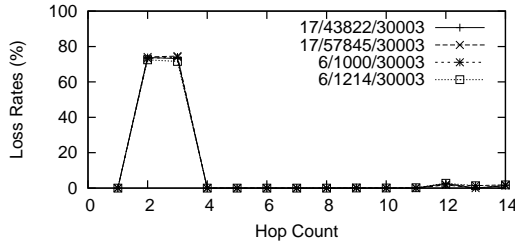


Fig. 11: The ANR partition errors v.s. ANR  $n_r$ . The *mis-partition* is wrong group partition with correct number of groups.



(a) Per-hop loss rates for path 8 with ICMP, TCP source port 20,1000, UDP source port 1000



(b) Per-hop loss rates for path 24 (194.80.38.243 → 194.70.143.51) with TCP source port 1000,1214, two random UDP source ports

Fig. 12: Typical path loss rates

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(a) gives the typical per hop loss rates for a successfully proved path: path 8. Other MPPs are similar. 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. 12(b) 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 two

Path	#G	Router w/ OP	HC	NH	Location	CFM
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	2	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	–
13	2	192.5.40.53	11	15	Indiana	Yes
14	2	192.5.40.134	5	15	Indiana	Yes

TABLE VII: Summary of hop-by-hop validation results for the 12 successfully validated paths. The *Path* numbers are the same as those in Table VI. *#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. *NH* is number of hops of the path. "Yes" in column *CFM* means positive confirmation received, "–" means no reply.

high priority groups (6,8) 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  $\theta$  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.

## VII. COMPARING PRIORITY INFERENCE WITH DIFFERENT METRICS

In § III-B, we the discussed general principles of choosing the inference metric. In this section, we compare the inference results of the loss-based and OOO-based method using PlanetLab experiments.

We do not use packet delay because the reordering metric is more robust than the delay metric although they both reflect the packet delay differences. When the delay variation generated by the non-configured devices is large, a packet with a shorter delay at the configured box can have a larger end-to-end delay than a packet with a larger delay at the configured box. Hence, the delay differences between different packet types introduced by the configured box are overwhelmed by the large delay variation introduced by the non-configured devices along the path. Large delay variation can often be observed for congested routers. However, routers usually do no reorder packets. Hence the reordering events introduced by the configured box are usually observed by the receiver without any distortion. OOO-based method is generally more accurate than the delay-based method.

### A. Methodology

As discussed in § VI-C, we performed the loss-based and OOO-based inference for every probe in  $\mathcal{N}_2$ . We also ran *HHP* method for every OMGP (OOO-based Multi-Group Path) and

	Total	P2P	ICMP	T179	Load Share	Unknown
# OMGP	56	3	15	11	17	12
# HHP	36	2	12	10	12	0
Validation	11/0	2/0	4/0	4/0	1/0+8	0
# LMGP	19	5	6	3	3	4
# HHP	10	4	4	2	0	0
Validation	7/0	4/0	1/0	2/0	0	0
# Overlap	7	0	1	2	3	1

**TABLE VIII:** Group patterns given by the OOO-based and loss-based methods for  $\mathcal{N}_2$ . The validation result is (the number of positive confirmed paths)/(the number of negative confirmed paths). For load sharing pattern, please refer to the related text for details.

LMGP (Loss-based Multi-Group Path), and sent emails to the administrators of related networks for confirmation.

The OOO-based method is almost the same as the loss-based method except that we use reordering ranks instead of loss ranks to perform group partition. The reordering rank of packet type  $i$  is derived from its *Packet Reordering Ratio* ( $PRR_i$ ), which is the fraction of reordered packets in all received packets of this packet type. Here, we define a reordered packet as the one that is outpaced by a behind packet.

In OOO-based method, we recalculate  $PRR$ s and their ranks for every subgroup before partitioning it during the hierarchical process. The packet reordering event is not just a binary indicator. It indicates which two packets are out-of-ordered. As the reordering events between two packet types are already enough to determine whether they are treated equally or not, we calculate  $PRR$ s only based on the out-of-order events among packet types within a same group. Once a group is divided into two new subgroups,  $PRR$  within each subgroup is then recalculated.

## B. Results

Table VIII shows the number of MGPs (Multi-Group Paths), the number of paths that had their spots of difference identified by *HHP* and the number of validated paths. We sent more than 20 emails and got 10 replies. All of them are positive confirmations. To note, one reply can confirm several paths, e.g. two uni-directional paths between a pair of nodes. Therefore, the number of validated paths is slightly larger than the number of replies received.

There are 56 OMGPs and only 19 LMGPs in  $\mathcal{N}_2$ . The overlap between the OMGPs and LMGPs is small, i.e. only seven paths are both flagged as OMGP and LMGP. In  $\mathcal{N}_1$ , P2P and ICMP are two main priority patterns configured by the ISPs. However, only one of 11 LMGPs of these two patterns in  $\mathcal{N}_2$  is identified by the OOO-based method. On the other hand, OMGPs mainly concentrate on the ICMP, T179 and load sharing patterns whereas very few LMGPs show the latter two patterns. Such findings substantiate our analysis in § III-B that the OOO-based method may fail to discover many multi-priority paths but to flag many paths caused by the mechanisms other than QoS.

Next, we discuss the mechanisms found by the OOO-based method for every group pattern. The mechanisms mainly cause packet types to experience different delays, such as fast/slow forwarding paths inside a router or parallel load sharing links.

**P2P** As shown in Table VIII, there is no overlap between the three OMGPs and five LMGPs. Because four of five LMGPs are confirmed by their operators, we can see the OOO-based method is not good at discovering those QoS mechanisms. As for the three OMGPs, two of them are rate-limited by the Packeteer’s standalone traffic shapers (confirmed by the related network operators). The shaper uses a more intelligent technique to do rate-limiting without dropping packets, which is the reason why the loss-based method did not discover them. Both of them show extremely large delays for TCP port 6881 (Bittorrent) packets. One of the two shows the pattern in two directions and its *spots of difference* matched Fig. 5(c). The shaper limits TCP connection throughput by rewriting its TCP advertised window using a mechanism named TCR [23]. We found that the advertised window was rewritten at the receiver.

**ICMP** Similar to the P2P pattern, the overlap between the OMGPs and LMGPs is small too. For the LMGPs, we speculate that the ISP used *policing* for rate-limiting, so we did not observe any reordering event. As for the OMGPs, we found two mechanisms for them and neither of them is likely to generate a difference in loss rate. For two of the OMGPs, *HHP* indicates that the ICMP packets take a different one or two-hop route along the path, i.e. the router IP addresses returned for the ICMP and other packets are different for one or two hops. It seems to us that the network operators route the ICMP packets for some purpose. For the other 12 OMGPs, eight of them are from four bi-directional paths. Three of the four bi-directional paths show one-hop pattern as Fig. 5(c), and another one shows two-hop pattern as Fig. 5(b). We received two confirmations saying that the reason is because the ICMP packets are CPU processed and thus are slower than other packet types. One of the boxes is a router, another is an IPS device (McAfee’s InstruShield 4000).

**T179** The 11 OMGPs all have the TCP port 179 packets lagged behind other packet types. We received two replies from the network operators and both indicated that they used the CISCO 3750 switches as routers at the places we located. We then tested that switch in our testbed and found when it is configured as a router and its BGP routing module is enabled, all BGP packets are sent to the switch CPU for software forwarding without even checking if the destination IP of the packet is to the router or not. Therefore, they lag behind all packets of other types because all other packets are fast switched without intervening of the switch CPU. As processing delay only shows after the packets traverse the router, the hop distances between their spots of difference are all larger than one.

Although two paths (two directions between a pair of nodes) are flagged as both LMGPs and OMGPs, we found that their loss rate differences and reordering events are actually caused by two different devices along the paths. The reordering events are generated by the above mechanism whereas the loss rate difference is caused by a *policing* rule in another router along the path. Both of these findings are confirmed by their corresponding network operators.

**Load Sharing** The port-based load sharing can be detected by both methods. The OOO-based method detects it when the delays of the parallel links are different, whereas the loss-based method detects it when the loss rates of the parallel links

are different. However, the loss based method is less likely to detect this mechanism.

Nowadays, port-based load sharing is supported by both Cisco and Juniper routers [24], [25]. It splits the traffic based on IP protocol, source and destination port in the packet to achieve finer load sharing granularity over parallel links. Since the hash function used to determine the output link of a packet is lack of application-level meaning, e.g., XOR of the source and destination of TCP/UDP port number in Cisco router, each of the partitioned groups contains a mixture of TCP and UDP packet types. Usually, there are two parallel links so that the packet types are partitioned to two groups.

In the 17 OMGPs, we identified *spots of difference* for 12 of them. Eight of them are directly validated by the *HHP* method, which is similar to the method proposed in [26] to identify such load-balanced paths. Multiple router IP addresses are returned at certain hops for them, which shows that packets of different types reach different router interfaces at those hops. For the other four paths without returning multiple IP addresses, we received one response from the related network operators and it confirmed that they assigned the load-sharing links to a same IP. We believe the other three are also caused by this mechanism as they all give two groups and each of the groups contains a mixture of TCP and UDP packet types.

We did not observe loss rate differences for the three LGMPs during *HHP* probes. This is likely to happen as the link utilizations of parallel links should usually be roughly the same due to the finer granularity of this load sharing mechanism.

**Unknown** Both methods produce a number of MGPs not included in previous group patterns. For example, packets of FTP control port (T21) lag behind other packet types for four OMGPs, and one of them experiences smaller loss rates than all other packet types. Because the port pattern is not as typical as previous patterns, we did not perform *HHP* along these paths. As the fraction of these paths is small (less than 25% of all MGPs), we believe the conclusion drawn is still valid, i.e. the loss-based method is more suitable than the OOO-based method in detecting packet forwarding priorities.

## VIII. CONCLUSIONS

In this paper, we have demonstrated that POPI, an end-to-end priority inference tool, is able to accurately infer the router's packet forwarding priority. The contributions of this work are the findings over Internet as well as the methodology.

In the PlanetLab experiments, the loss-based method detected several multi-priority paths in the Internet. Further validation showed the low false-positive rate of this method. In searching for a method with less probe overhead than the loss-based method, we used packet reordering and delay as the inference metrics and found they were not as effective as loss in detecting packet forwarding priorities. They failed to flag many multi-priority paths that were discovered by the loss-based method, and the paths flagged by them were mainly caused by the mechanisms which induce different delays among packets types. Hence, as for the packet forwarding priority, we believe packet loss metric is more suitable than the other two.

In the PlanetLab experiments, we have identified several common priority settings (e.g. low-priority treatment for P2P, high-priority treatment for well-known TCP applications and different treatment for ICMP echo requests). Although OOO-based method is not suitable for detecting forwarding priorities, it discovers several types of middleboxes such as a traffic shaper which shapes traffic by rewritten TCP header, and the IPS devices which examine ICMP packets in its slow path. To our best knowledge, no other existing network measurement techniques are able to detect such middleboxes.

## ACKNOWLEDGEMENT

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. We thank the network administrators who kindly provided us with the ground truth. Finally, we thank Chi Yin Cheung, 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.

## APPENDIX

Proof for Theorem. 1.

*Proof:* Let  $k$  be the total number of packet types. When  $k_j$  packet types are in same class  $j$ , the  $NR_i$  of a packet type within this class has the discrete uniform distribution at  $i_0/k, i_0 + 1/k, \dots, (i_0 + k_j - 1)/k$ , where  $i_0$  is the minimum  $NR$  for this class. Then,  $\sigma^2(NR_i) = (k_j^2 - 1)/12k^2$ . According to the central limit theorem,  $ANR_i = \frac{1}{n_b} \sum_{i=1}^{n_b} NR_i$  approximates to the normal distribution with mean  $\mu = \mu(NR_i)$  and variance  $\sigma^2(ANR_i) = \sigma^2(NR_i)/n_b$ . From [27], the range of  $ANR$ s is  $\theta_{1-\alpha, k_j, n_b, k} = Q_{1-\alpha, k_j} \times \sigma(ANR_i)$ . ■

*Theorem 2:* The percentage of type II error is 0 when

$$n_b > \left( \frac{2Q_{1-\alpha, k_j + k_{j-1}} \sqrt{(k_j + k_{j-1})^2 - 1}}{\sqrt{12}(k_j + k_{j-1})} \right)^2. \quad (3)$$

*Proof:* Let  $j-1, j$  be two classes. Assume without loss of generality that class  $j$  has higher priority than class  $j-1$ . Then, define  $D$  is the difference between the maximal  $ANR_i$  in class  $j$  and the minimal  $ANR_i$  in class  $j-1$ .  $D = \max(ANR_i^{(j)}) - \min(ANR_i^{(j-1)})$ .

$$\begin{aligned} D &= \max(ANR_i^{(j)}) - \min(ANR_i^{(j-1)}) \\ &\geq \text{mean}(ANR_i^{(j)}) - \text{mean}(ANR_i^{(j-1)}) \\ &= (k_j + k_{j-1})/2k \end{aligned}$$

The percentage of type II error is  $P(D < \theta_{1-\alpha, k_j + k_{j-1}, n_b, k})$ . Thus, when

$$\theta_{1-\alpha, k_j + k_{j-1}, n_b, k} < (k_j + k_{j-1})/2k, \quad (4)$$

the percentage will be zero. Combine Eq. (2) and (4), we can get Eq. (3). ■

For  $k_j + k_{j-1} = 32$ , the minimum  $n_b$  satisfy Eq. (3) is 12.

## 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/ios122/122cgr/fqos.c/>
- [2] Juniper Networks, "Filter-based forwarding," 2001. [Online]. Available: <http://www.juniper.net>
- [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] J. Cheng. (2007) Evidence mounts that comcast is targeting bittorrent traffic. [Online]. Available: <http://arstechnica.com/news.ars/post/20071019-evidence-mounts-that-comcast-is-targeting-bittorrent-traffic.html>
- [5] V. Kumar. (2008) Comcast, bittorrent to work together on network traffic. [Online]. Available: <http://online.wsj.com/article/SB120658178504567453.html>
- [6] G. Lu, Y. Chen, S. Birrer, F. E. Bustamante, C. Y. Cheung, and X. Li, "End-to-end inference of router packet forwarding priority," in *Proc. IEEE INFOCOM*, 2007.
- [7] K. Harfoush, A. Bestavros, and J. Byers, "Robust identification of shared losses using end-to-end unicast probes," in *Proc. IEEE ICNP*, 2000.
- [8] D. Rubenstein, J. Kurose, and D. Towsley, "Detecting shared congestion of flows via end-to-end measurement," *IEEE/ACM Transactions on Networking*, vol. 10, no. 3, pp. 381–395, 2002.
- [9] M. S. Kim, T. Kim, Y. Shin, S. S. Lam, and E. J. Powers, "A wavelet-based approach to detect shared congestion," in *Proc. ACM SIGCOMM*, 2004.
- [10] A. Kuzmanovic and E. W. Knightly, "Measuring service in multi-class networks," in *Proc. IEEE INFOCOM*, 2001.
- [11] A. Coates, A. Hero III, R. Nowak, and B. Yu, "Internet tomography," *IEEE Signal Processing Magazine*, vol. 19, no. 3, pp. 47–65, 2002.
- [12] T. Bu, N. Duffield, F. Presti, and D. Towsley, "Network tomography on general topologies," in *Proc. ACM SIGMETRICS*, 2002.
- [13] M. Rabbat, R. Nowak, and M. Coates, "Multiple source, multiple destination network tomography," in *Proc. IEEE INFOCOM*, 2004.
- [14] C. Reis, S. D. Gribble, T. Kohno, and N. C. Weaver, "Detecting in-flight page changes with web tripwires," in *Proc. NSDI*, 2008.
- [15] 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>
- [16] J. C. R. Bennett, C. Partridge, and N. Shectman, "Packet reordering is not pathological network behavior," *IEEE/ACM Transactions on Networking*, vol. 7, no. 6, 1999.
- [17] M. Jain and C. Dovrolis, "End-to-end available bandwidth: Measurement methodology, dynamics, and relation with tcp throughput," *IEEE/ACM Transactions on Networking*, vol. 11, pp. 537–549, 2003.
- [18] J. D. Gibbons, *Nonparametric Statistical Inference*. Marcel Dekker, Inc, 1985.
- [19] G. E. Noether, *Introduction to Statistics: A Nonparametric approach*. Houghton Mifflin Company, 1976.
- [20] B. S. Everitt, S. Landau, and M. Leese, *Cluster Analysis*. Oxford University Press Inc., New York, 2001.
- [21] S. Floyd and V. Jacobson, "Link-sharing and resource management models for packet networks," *IEEE/ACM Transactions on Networking*, vol. 3, no. 4, pp. 365–386, 1995.
- [22] H. Jiang and C. Dovrolis, "Why is the internet traffic bursty in short time scales?" in *Proc. ACM SIGMETRICS*, 2005.
- [23] S. Karandikar, S. Kalyanaraman, P. Bagal, and B. Packer, "Tcp rate control," *ACM Computer Communication Review*, vol. 30, no. 1, Jan 2000.
- [24] Juniper Networks, "Junos configuration guides – policy framework." [Online]. Available: <http://www.juniper.net/techpubs/software/junos/junos70/swconfig70-policy/download/swconfig70-policy.pdf>
- [25] Cisco System, "Cisco IOS IP Switching Configuration Guide, Release 12.4." [Online]. Available: <http://www.cisco.com/univercd/cc/td/doc/product/software/ios124/124cg/hisw.c/index.htm>
- [26] B. Augustin, T. Friedman, and R. Teixeira, "Measuring load-balanced paths in the internet," in *Proc. IMC*, 2007.
- [27] Y. Hochberg and A. C. Tamhane, *Multiple Comparison Procedures*. Wiley Series in Probability and Mathematical Statistics, 1987.



**Guohan Lu** received the B.S. degree in mechanical engineering and the M.S. degree in electronic engineering, both from Tsinghua University, China. He is currently pursuing the Ph.D. degree in the Department of Electrical Engineering, Tsinghua University, Beijing, China.

His research interests are on network measurement and monitoring, network security, and P2P.



with his colleagues.

**Yan Chen** is an Assistant Professor in the Department of Electrical Engineering and Computer Science at Northwestern University, Evanston, IL. He got his Ph.D. in Computer Science at the University of California at Berkeley in 2003. His research interests include network measurement, monitoring and security, and P2P systems. He won the Department of Energy (DOE) Early CAREER award in 2005, the Air Force of Scientific Research (AFOSR) Young Investigator Award in 2007, and the Microsoft Trustworthy Computing Awards in 2004 and 2005



**Stefan Birrer** is a Ph.D. candidate in the Department of Electrical Engineering and Computer Science at Northwestern. His research interests span the intersection between distributed systems and networking. Birrer has a M.S. in computer science from Northwestern University and a B.S. in computer science from FH Aargau, Switzerland. He is the recipient of a Neokast Fellowship (2005-2007).



Society, the ACM, and USENIX.

**Fabián E. Bustamante** is an assistant professor of computer science in the Department of Electrical Engineering and Computer Science at Northwestern. His research interests include distributed systems and networks, and operating systems support for massively distributed systems, including Internet and vehicular network services. Bustamante has an M.S. and Ph.D. in computer science from the Georgia Institute of Technology. Bustamante is the recipient of a National Science Foundation CAREER Award (2007). He is a member of the IEEE Computer



ter.

**Xing Li** received the B. S. degree in radio electronics from Tsinghua University, Beijing, China, in 1982, and the M. S. and Ph.D. degrees in electrical engineering from Drexel University, Philadelphia, PA, in 1985 and 1989, respectively.

He is currently a Professor in the Electronic Engineering Department, Tsinghua University. His research activities and interests include statistical signal processing, multimedia communication and compute networks. He is Deputy Director of China Education and Research Network (CERNET) Cen-