

Packet Doppler: Network Monitoring using Packet Shift Detection

Tongqing Qiu
Georgia Inst. of Technology

Jian Ni
Yale University

Hao Wang
Yale University

Nan Hua
Georgia Inst. of Technology

Y. Richard Yang
Yale University

Jun (Jim) Xu^{*}
Georgia Inst. of Technology

ABSTRACT

Due to recent large-scale deployments of delay and loss-sensitive applications, there are increasingly stringent demands on the monitoring of service level agreement metrics. Although many end-to-end monitoring methods have been proposed, they are mainly based on active probing and thus inject measurement traffic into the network. In this paper, we propose a new scheme for monitoring service level agreement metrics, in particular, delay distribution. Our scheme is passive and therefore will not cause perturbation to real traffic. Using realistic delay and traffic demands, we show that our scheme achieves high accuracy and can detect burst events that will be missed by probing based methods.

1. INTRODUCTION

Recent large-scale deployments of delay and loss-sensitive applications have led to stringent demand on performance monitoring. Since delayed or lost packets in applications such as telecommuting, streaming media, gaming and conference applications can result in significant performance degradation, users nowadays demand strict guarantees from their network providers on key metrics related to delay and loss. These guarantees are typically summarized as service level agreements (SLAs). With well-defined SLAs and associated penalty in violating them, network providers have strong incentive to monitor these metrics more accurately and faster, so that they can provide more reasonable agreements and detect potential problems earlier.

Many SLA monitoring algorithms have been proposed in

the past few years (*e.g.*, [5, 19, 20, 30]). Since the key metrics in SLAs are defined end-to-end, the dominant type of monitoring approach is based on end-to-end probing. In a probing based approach, the network provider deploys multiple monitoring nodes across the network and measures the performance metrics of the probing packets. In particular, in [19], Sommers *et al.* proposed SLAm, and conducted a systematic study on probing based approaches to address issues including sampling intervals and bound on probing accuracy.

All existing probing based approaches share one common feature: they need to inject probing traffic into the network. There are several issues with these approaches. First, the probing traffic is overhead and thus reduces usable network bandwidth. For one source node, the bandwidth cost could be small to probe one destination. However, in order to measure end-to-end performance metrics for all destinations in a network of n edge nodes, the total bandwidth cost grows with $O(n)$. For example, the overhead traffic in [19] at a source is about hundreds of Kbps for probing a single destination, but when there are thousands of destinations in a large ISP, this overhead can grow to the Gbps level. Second, the probing traffic may cause perturbation to real traffic, as shown in many existing works [1, 2, 17]. Higher measurement precision requires higher probing rate and therefore causes more perturbation. The effect of the perturbation is more serious when the network load is high, the moment when it is important to get more accurate measurements. Third, active probing estimates the performance of the probing traffic instead of the real traffic. To minimize probing overhead, probing packets are usually much smaller (typically tens of bytes) compared with packets in real traffic. Thus, the treatment received by probing packets may be quite different from that received by packets in real traffic. Finally, performance anomalies between probing intervals will be missed. This can be serious given the fact that many routing failure events are short-lived: less than 10 seconds for transient BGP failures [22] and even shorter for many other cases [12].

In this paper, we propose Packet Doppler, a new technique for monitoring SLA performance metrics. More specifically, it is used to measure and monitor the delay distributions of

^{*}This work is supported in part by NSF grants CNS-0519745, CNS-0626979, CNS-0716423, CAREER Award ANI-0238315, and a gift from Cisco.

origin-destination (OD) node pairs in an ISP network, where a node could be an end-host, link, router, or Point of Presence (PoP). In astronomy, the Doppler effect (named after Christian Doppler), is the change in frequency and wavelength of a wave, known as the *spectrum shift*, as perceived by an observer moving relative to the source of the wave. Similarly, we can view the delays of the packets as the shifts of the packets in the time series perceived by the receiver relative to the source. We call such phenomenon *packet shift*. The estimation of the delay distributions can be viewed as the detection and recovery of the packet shift processes.

Unlike probing based approach, Packet Doppler is a passive measurement scheme. For each OD pair, it listens and records tiny digests of real traffic at both the origin and the destination so that the ISP can estimate the packet shift process. These digests, 3 to 4 orders of magnitude smaller than the real packet stream for current Internet packet size distribution, concisely *encode* the time series information of the packets passing by.

After that, Packet Doppler uses two innovative decoding techniques, namely, efficient Sequential Hypothesis Testing (SHT) with Viterbi Decoding, and Iterative Proportional Fitting (IPF), to fully take advantage of the structural constraints present in SLA measurements and extract all of the useful information out of the packet digests. These two techniques provide rigorous statistical guarantees (*e.g.*, Maximum Likelihood Estimation) and improve the decoding accuracy by more than one order of magnitude over just using a simple existing decoding procedure.

We evaluate our scheme using real delay and traffic volume traces. We show that compared with a probing based approach SLAm [19], our delay estimation scheme achieves better accuracy both for mean delay and delay variance. For example, Packet Doppler can maintain the error of mean delay within one millisecond, while SLAm can suffer from several milliseconds inaccuracy. The relative error of delay variance for SLAm can be tens of times larger than Packet Doppler. Moreover, Packet Doppler can effectively detect many burst events that will be missed by probing type methods. The proposed decoding techniques are the true heroes behind this success: Our evaluation results show that SHT improves the accuracy of the existing decoding scheme by more than one order of magnitude and the IPF procedure further cuts estimation errors by a significant percentage (*e.g.*, 50%).

Our key contributions can be summarized as follows:

- We propose a novel scheme to measure delay distribution in a passive way, as a viable alternative to current active probing methods. It can estimate the delay with high precision and low overhead. In addition, it can detect the burst of delay changes.
- We propose two decoding methods to effectively improve the decoding accuracy. These methods can explore more fruitful and accurate information from traf-

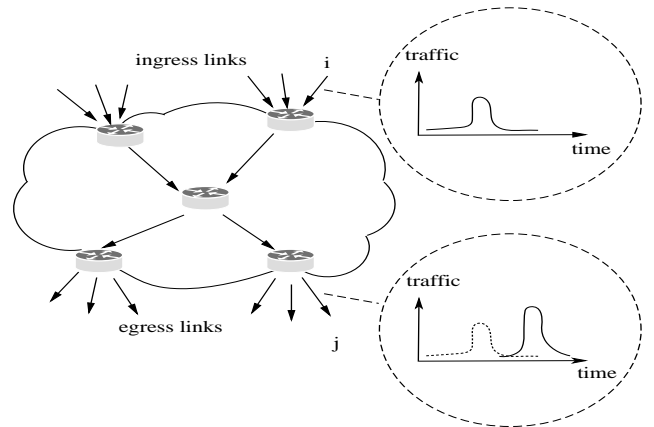


Figure 1: The phenomenon of packet shift.

fic digests, compared with existing methods for both network and database applications [23, 29].

The rest of the paper is organized as follows. Section 2 offers the overview of our scheme. Section 3 and 4 introduce two advanced decoding algorithms respectively. In Section 5, we present our evaluation results. We survey the related work in Section 6 and conclude the paper in Section 7.

2. OVERVIEW OF OUR APPROACH

In Figure 1, we illustrate the baseline idea of Packet Doppler for estimating the delay distributions of origin-destination (OD) link pairs in an ISP network¹. Consider an OD link pair i and j . At the ingress link i , the volume of traffic (packets) that is destined to egress link j over the time is shown in the chart on the upper right. After some delay, link j begins to receive the traffic stream, which is shown in the chart on the bottom right. Comparing these two time series, we can observe a shift and distortion of the shape of the traffic volume as a result of the delay between the OD pair. Our approach is to infer the delay distribution from the shifts of the packets over time.

Specifically, both i and j record and timestamp packet sets going through them during each measurement interval. The length of a measurement interval depends on the requirement on measurement accuracy. For now on, time is measured in the unit of measurement interval. Like other SLA measurement techniques, our technique assumes that the clock asynchronization at different routers is negligible compared with the measurement result. For example, we can achieve precision of a few microseconds or several hundred nanoseconds using off-the-shelf hardware based on LORAN-C [13], WWVB [14] or GPS [9, 11].

To find out the delay of the set of packets arriving at source i in time interval $[s, s + 1]$, we need to find out the intersections between the packet set at i in $[s, s + 1]$ and the packet

¹Our techniques, however, can be extended/adapted to other levels of granularity such as end-host, router, or PoP pairs.

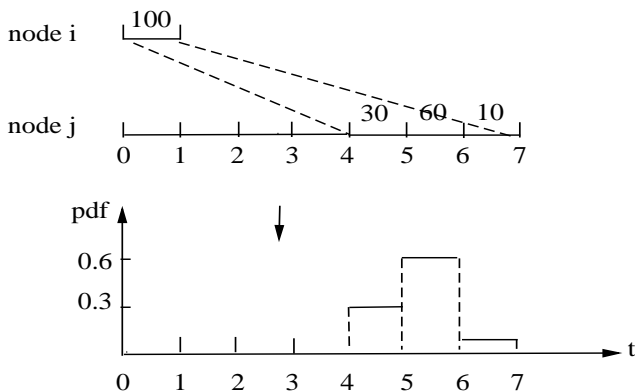


Figure 2: Estimation of delay distribution from packet shift process.

sets at destination j in time intervals $[t, t + 1]$, $[t + 1, t + 2]$, ..., and so on. We illustrate this procedure in Figure 2. Suppose 100 packets arrive at i in time interval $[0,1]$ and these packets are destined to j . By comparing the traffic digest at i in time interval $[0,1]$, which encodes not only these 100 packets but also other cross traffic, with the traffic digests at j (also with cross traffic encoded) in different time intervals, we find that 30, 60, and 10 packets, as estimated from the corresponding traffic digests, depart from j during time intervals $[4,5]$ $[5,6]$ $[6,7]$, respectively. We can then infer the delay distribution as shown in the bottom of Figure 2.

In the following two subsections, we describe the procedure that encodes the packet sets into traffic digests and the baseline procedure that infers the delay distributions from these digests. The notations for presenting the procedures are summarized in Table 1.

2.1 Encoding Procedure

Each monitored link continuously encodes the sets of packets passing by during every monitoring interval. The encoding procedure is simple enough to effectively process every packet in high speed networks. It generates extremely small digests for the packet sets. The generated digests will be shipped to one or several servers for decoding.

We base our encoding approach on [29] for traffic matrix estimation. It is important to note that traffic matrix estimation is itself a nontrivial problem [26, 27, 29, 28]. Our solution significantly extends the solution in [29]. To ensure high precision of delay estimation, the typical length of a monitoring interval is 1 ms². We will fix 1 ms as the interval length unless otherwise noted in the sequel.

The format of the digests is very simple: a (0-1) bitmap B indexed by a hash function h . Initially, all entries in the

²Note such a granularity would be an overkill for TM estimation (for which 5 minute interval is good enough), but is necessary for accurate delay estimation. However, there is little storage cost difference (in terms of gigabytes per hour) between different time granularities, as will be clear later.

Table 1: The Notation Table

Notation	Meanings
i	source node (ingress link, router, PoP)
j	destination node (egress link, router, PoP)
$n_i(s)$	number of packets that arrive at i in time interval $[s, s + 1]$
$m_j(t)$	number of packets that depart from j in time interval $[t, t + 1]$
$x_{ij}(s, t)$	number of packets that arrive at i in $[s, s + 1]$ and depart from j in $[t, t + 1]$

bitmap are set to 0. When a packet pkt arrives at ingress links or departs from egress links, the invariant portion of the packet³, denoted by $\phi(pkt)$, is extracted and hashed using h . The hashing result $h(\phi(pkt))$ is viewed as an index in B , and the bit at the corresponding index is set to 1. The encoding procedure uses the same hash function h with the same bitmap size b at every monitored link. Since each packet corresponds to one bit in the bitmap, the encoding module has extremely low storage cost. Since only a single hashing operation is involved, which is supported by the instruction sets of many modern processors, the computational cost is also very low.

2.2 Decoding Procedure

As we illustrate in Figure 2, we can recover the delay distribution of the packets from source i to destination j in any time interval if we know the intersections of the packet sets at i and j , denoted as the *packet shift process* between i and j .

For simplicity we use s to denote time interval $[s, s + 1]$. For OD pair i and j , let $x_{ij}(s, t)$ be the number of packets that arrive at source i in interval s and depart from destination j in interval t (hence the delay of those packets is $t - s$). The (empirical) delay distribution of the packets from i to j in interval s , p_{ij}^s , can be derived from the packet shift process $x_{ij}(s, t)$'s as follows:

$$p_{ij}^s(t - s) = \frac{x_{ij}(s, t)}{\sum_{t' \in T_{ij}^s} x_{ij}(s, t')}, \quad t \in T_{ij}^s, \quad (1)$$

where T_{ij}^s includes possible time intervals when packets (which arrive at i in s) depart from j . $\sum_{t' \in T_{ij}^s} x_{ij}(s, t')$ is the total number of packets arriving at i in s that are destined to j ; hence the right hand side of (1) is just the fraction (probability) of those packets that experience delay $t - s$.

³As described in [18], the invariant part of a packet should uniquely represent the packet and remain the same when the packet travels from one link to another. At the same time, it is desirable to make its size small to allow for fast hash processing. In our scheme, the invariant portion of a packet consists of the packet's IP header (where the variant fields like TTL, ToS, and checksum are marked as 0's) and the first 8 bytes of the payload if there is any. As shown in [18], these 28 bytes are sufficient to differentiate almost all non-identical packets.

More generally, if we want to estimate the (empirical) delay distribution of the packets from source i to destination j over a period of time, say from time interval 1 to T , it can be derived from the packet shift process as follows:

$$p_{ij}(d) = \frac{\sum_{1 \leq s \leq T} x_{ij}(s, s+d)}{\sum_{1 \leq s \leq T} \sum_{t' \in T_{ij}^s} x_{ij}(s, t')}, \quad d \in \Delta_{ij}, \quad (2)$$

where Δ_{ij} is the set of possible delays from i to j in that time period, e.g., $\Delta_{ij} = \{t_{ij}^{min}, \dots, t_{ij}^{max}\}$ where t_{ij}^{min} and t_{ij}^{max} are the minimum and maximum delays from i to j .

2.2.1 Estimating Intersections

Based on the preceding analysis, it remains to estimate the packet shift process (i.e., the intersections $x_{ij}(s, t)$'s). We first adapt the method proposed in [29]. Let the set of packets arriving at source i in s be $A_i(s)$, and the set of packets departing from destination j in t be $D_j(t)$. Let the corresponding bitmaps for $A_i(s)$ and $D_j(t)$ be $B_{A_i(s)}$ and $B_{D_j(t)}$.

Let $u_{A_i(s)}$ (resp., $u_{D_j(t)}$) denote the number of 0 bits in $B_{A_i(s)}$ (resp., $B_{D_j(t)}$). Recall that the size of the bitmap is b . The maximum likelihood estimators (MLEs) for $n_i(s) = |A_i(s)|$ and $m_j(t) = |D_j(t)|$, the numbers of packets in $A_i(s)$ and $D_j(t)$, are given in [23]:

$$\begin{aligned} \hat{n}_i(s) &= b \ln \frac{b}{u_{A_i(s)}}, \\ \hat{m}_j(t) &= b \ln \frac{b}{u_{D_j(t)}}. \end{aligned} \quad (3)$$

$x_{i,j}(s, t)$, the intersection between $A_i(s)$ and $D_j(t)$, can be described as $|A_i(s) \cap D_j(t)| = |A_i(s)| + |D_j(t)| - |A_i(s) \cup D_j(t)|$. An estimator for this quantity is given in [29]:

$$\hat{x}_{i,j}(s, t) = \hat{n}_i(s) + \hat{m}_j(t) - b \ln \frac{b}{u_{A_i(s) \cup D_j(t)}}, \quad (4)$$

where $u_{A_i(s) \cup D_j(t)}$ denotes the number of 0 bits in $B_{A_i(s) \cup D_j(t)}$ (the result of hashing the packets in $A_i(s) \cup D_j(t)$ to a single bitmap). $B_{A_i(s) \cup D_j(t)}$ can be computed as the bitwise-OR of $B_{A_i(s)}$ and $B_{D_j(t)}$.

Like in [23], for bitmap size b , we define the *load factor* for a set A with cardinality $|A|$ as $\lambda_A = \frac{|A|}{b}$. The variance of the estimator $\hat{x}_{i,j}(s, t)$ in (4) is ([29]):

$$\begin{aligned} \text{var}(\hat{x}_{i,j}(s, t)) &= b(2e^{\lambda_{A_i(s) \cap D_j(t)}} + e^{\lambda_{A_i(s) \cup D_j(t)}} - e^{\lambda_{A_i(s)}} \\ &\quad - e^{\lambda_{D_j(t)}} - \lambda_{A_i(s) \cap D_j(t)} - 1). \end{aligned} \quad (5)$$

The selection of the bitmap size will keep the load factor small (less than 1). Hence the (relative) variances of the estimators are small when $x_{i,j}(s, t)$'s are large. We can directly use $\hat{x}_{i,j}(s, t)$'s in (4) to estimate the delay distribution in (2). We call it the *naive decoding scheme* hereafter.

2.2.2 Limitation of the Naive Decoding Scheme

However, our experiments show that directly applying the naive decoding scheme to estimate the intersections and the

delay distributions does not produce accurate results. The main reason is that when the signal noise ratio (SNR)⁴ is low, i.e., the amount of traffic between i and j is small relative to the cross traffic at i and/or j , the naive decoding scheme may report incorrect delay values since the intersection estimation can be drowned out by noise.

In order to address this problem and improve the delay estimation accuracy, we propose two advanced decoding techniques, namely, efficient Sequential Hypothesis Testing (SHT) with Viterbi Decoding and Iterative Proportional Fitting (IPF), in the following two sections.

3. PACKET SHIFT DETECTION USING SEQUENTIAL HYPOTHESIS TESTING

Consider OD pair i and j . Suppose node i sends a set of packets to node j in time interval s . Remember T_{ij}^s is the set of possible time intervals when j receives those packets, e.g., $T_{ij}^s = \{s + t_{ij}^{min}, \dots, s + t_{ij}^{max}\}$, where t_{ij}^{min} and t_{ij}^{max} are the minimum and maximum delays from i to j . Let $\hat{T}_{ij}^s \subseteq T_{ij}^s$ be the exact set of time intervals when j receives those packets. The value of $|\hat{T}_{ij}^s|$ (cardinality of \hat{T}_{ij}^s) is much smaller than $|T_{ij}^s|$ because delay normally does not change dramatically within a short interval like 1ms. In this section we apply hypothesis testing to detect the set of time intervals \hat{T}_{ij}^s that the packets are most likely falling into.

3.1 Problem Formulation

We formalize the packet shift detection problem as a hypothesis testing problem. Every possible shift can be viewed a hypothesis H . Based on the bitmap observation, we test all the hypotheses and select the most likely hypothesis as the detection result. The advantage of hypothesis testing lies in its capability to *combine* different shift intervals into the hypothesis space. In other words, it can utilize the implicit relationship among different intervals.

Assume that $|T_{ij}^s| = K$. We consider K hypotheses H_1, H_2, \dots, H_K , where H_t means that the packets fall into (up to) $|\hat{T}_{ij}^s|$ consecutive intervals beginning at interval t , more formally, marked as $H_t : \theta_t \in \Theta_t$. Applying likelihood ratio test, we compute the likelihood ratio $\lambda_t(\hat{x})$ for each hypothesis

$$\lambda_t(\hat{x}) = \frac{\sup_{\Theta_t} L(\theta_t | \hat{x})}{\sup_{\Theta} L(\theta_t | \hat{x})}. \quad (6)$$

Here θ_t is the exact intersection between the OD pair and \hat{x} is the observed intersection from bitmap estimation. The likelihood function of one hypothesis H_t is given by:

$$L(\theta_t | \hat{x}) = \Pr[\hat{x} | \theta_t]. \quad (7)$$

$\Pr[\hat{x} | \theta_t]$ represents the probability that we observe \hat{x} from bitmap estimation given the exact intersection between the OD pair is θ_t . The detailed computation of the probability is

⁴Here the *signal* is the real intersection, and the *noise* is caused by the hash collision and the estimation methodology itself.

given in Appendix A. The most likely hypothesis is the one with the maximum likelihood ratio λ_t :

$$\hat{t} = \operatorname{argmax}_{t \in T_{ij}^s} \lambda_t(\hat{x}). \quad (8)$$

3.2 Maximum Count Detection

When applying hypothesis testing for each monitoring interval separately (we call it HT(1)), it reduces to the simple *maximum count* (MC) detection scheme: for any monitoring interval s , the packets from source i to destination j will fall into (up to) $|\hat{T}_{ij}^s|$ consecutive intervals beginning from interval \hat{t} at j , where \hat{t} is chosen to maximize the sum of observed intersections during these intervals:

$$\hat{t} = \operatorname{argmax}_{t \in T_{ij}^s} \sum_{t'=t}^{t+|\hat{T}_{ij}^s|-1} \hat{x}_{ij}(s, t'). \quad (9)$$

If the estimated intersections $\hat{x}_{i,j}(s, t)$'s from direct bitmap computation (see (4)) are very accurate in millisecond level, then the MC detection is good enough. However, according to the variance equation (5), the noise of estimation can dominate when the OD traffic volume $x_{i,j}(s, t)$ is extremely small (say two orders of magnitude smaller than either total ingress or egress traffic), in other words, the cross traffic rate is very large. In this case, the MC detection may choose wrong delay intervals because a large noise can be mistaken for a strong signal, i.e., the estimation of an intersection $|A_i(s) \cap D_j(t)|$ could be large even if its actual value is near 0. In the next section, we describe how to solve this problem using sequential hypothesis testing (SHT).

3.3 SHT Through Viterbi Decoding

The idea of sequential hypothesis testing, in the context of this work, is as follows. Instead of performing hypothesis testing for each and every source interval individually using equation (8), we bundle 2 or more consecutive source intervals together and perform a joint hypothesis testing on them. We refer a bundling of 2 intervals as $HT(2)$, and $HT(3)$, $HT(4)$, \dots are defined similarly. SHT is the full-blown case where all intervals from logical time 0 to the current time are bundled together. SHT can significantly enhance the estimation accuracy because incorrect individual hypothesis testings can be detected and corrected by “surrounding” correct individual hypothesis testings.

However, since the hypothesis space grows exponentially with the number of intervals bundled together, the computation complexity will become prohibitively high when this number becomes larger. Fortunately the following reasonable assumption allows us to significantly reduce the hypothesis space. We assume that, for any time s , the vast majority (say more than 95%) of packets in $A_i(s)$ will be delivered no later than those of $A_i(s+1)$ by more than one unit of time. Note this is a very weak assumption which only requires no more than a small percentage of packets are *significantly re-ordered*. This is a very reasonable assumption with

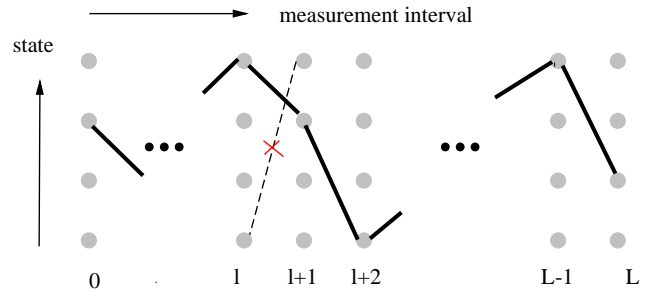


Figure 3: An example of Viterbi algorithm. There are four states ($K = 4$). The solid line represents the active path, and dash line represents an example of invalid path.

today’s Internet, where most routers employ FCFS queuing. Although routing changes or multi-path events may happen to cause packet re-ordering, the majority of packets in fact do arrive in order, as reported by a fairly recent study, which shows only 0.02% to 0.5% of packets are out of order [10]. We refer to the above assumption as “near-FCFS”, although our assumption is indeed much weaker (and more reasonable) than this name might suggest.

This near-FCFS assumption allows us to eliminate many invalid hypotheses in the hypothesis space. It also makes our SHT problem structurally similar to that of decoding the convolutional code in the coding theory, and therefore solvable by the Viterbi algorithm.

The Viterbi algorithm [21], discovered by Viterbi in 1967, reduces the complexity of maximum likelihood decoding by systematically removing paths that cannot achieve the optimal path metric from consideration. Mapped to our problem, the *decoding* process is to find the packet shift interval sequence. Recall that \hat{t} is the estimation result of the packet shift from node i at time interval s . Now we consider a sequence of these results $\hat{t}_0, \hat{t}_1, \hat{t}_2, \dots$ which corresponds to the time interval $s, s+1, s+2, \dots$ at source node i . As mentioned above, for each monitoring interval at the source node there are K possible choices of consecutive intervals at the destination node. We call each choice a *state*. The state can change from one monitoring interval to another, and the sequence of changes can be described by a *path*. Viterbi algorithm can remove the *invalid* paths which do not follow near-FCFS assumption and eventually return an *active* path which has the maximum likelihood. Figure 3 gives an example of Viterbi algorithm.

The general steps are described as follows, and the pseudo-code of the algorithm is illustrated in Algorithm 1.

Initialize: Initially, all K states are labeled as 0.

Update: For each time interval l and each state k we memorize (and later retrieve and reuse) the *active* path $P(k, l)$. An active path is a valid path (which obeys the near-FCFS assumption) whose distance is maximal (defined next). We

Algorithm 1: The pseudo-code of Viterbi algorithm

```

1 Initialize
2    $D(k, 0) := 0, k = 1, 2, \dots, K$ 

3 Update
4   for  $l=1$  to  $L$ 
5     for  $k=1$  to  $K$ 
6       for  $m=1$  to  $K$ 
7         if  $(m - k > -2)$  /* near-FCFS assumption */
8           if  $(D(k, l) < L(k, l) + D(m, l - 1))$ 
9              $D(k, l) = L(k, l) + D(m, l - 1);$ 
10             $P(k, l) = m;$ 

11 Terminate
12    $t(L) = \operatorname{argmax}_k D(k, L);$ 
13   for  $l=L-1$  to  $1$ 
14      $t(l) = P(t(l+1), l);$ 

```

also label each state k at time l using the distance of its active path, $D(k, l)$. Here the distance of a path $D(k, l)$ equals to the length of the last edge $L(k, l)$ plus the distance of the last-but-one state at time $l - 1$, while the length of an edge $L(k, l)$ is defined as the log likelihood $\log L(\theta_k | \hat{x})$, where $L(\theta_k | \hat{x})$ is the likelihood defined in equation (7).

Terminate: The algorithm proceeds in $L + 1$ steps, and recover the packet shift interval sequence based on the active path. L is a pre-selected window size. And the final active path also describes the delay sequence.

Viterbi algorithm is essentially a dynamic programming algorithm, which memorizes, retrieves, and reuse already-computed active paths $P(k, l)$. It can perform a joint hypothesis testing of L consecutive intervals and identify their joint Maximum Likelihood Estimator (MLE) with only $O(K^2)$ computational complexity⁵. In an ISP network environment, the value K can be safely limited to a couple of hundreds. The computational cost per SHT (no more than tens of thousands of instructions) is therefore very reasonable. In the evaluation section, we will show that Viterbi algorithm significantly improves the accuracy of our estimation.

4. NOISE REDUCTION USING IPF

During a period of time when there is no or extremely small amount of packet losses across the ISP network, as can be witnessed by SNMP loss counts, there will be a number of *conservation laws* concerning ingress and egress packet sets in the sense that all packets coming into the network have to exit somewhere at some point of time. In this section we discuss how to fully utilize these conservation laws to reduce the noises of direct Bitmap estimation using an optimization procedure: *iterative proportional fitting* (IPF).

Consider a time window from (interval) 1 to (interval) T . We can construct two types of conservation equations. A source-based conservation equation says that all the packets arriving at an ingress link in each time interval will eventually leave one of the egress links in the future. Similarly,

⁵In Algorithm 1, we perform this procedure L times (line 4) for L source intervals with total complexity $O(LK^2)$.

a destination-based conservation equation says that all the packets departing from an egress link in each time interval came from one of the ingress links in the past. Formally,

$$\begin{aligned}
 n_i(s) &= \sum_j \sum_{t \in T_{ij}(s)} x_{ij}(s, t), \quad \forall i, s = 1, 2, \dots, T, \\
 m_j(t) &= \sum_i \sum_{s \in S_{ij}(t)} x_{ij}(s, t), \quad \forall j, t = 1, 2, \dots, T,
 \end{aligned} \tag{10}$$

where $T_{ij}(s)$ is the set of possible time intervals when packets (which arrive at source i in interval s) depart from destination j , e.g., $T_{ij}(s) = \{s + t_{ij}^{min}, \dots, s + t_{ij}^{max}\}$. Similarly, $S_{ij}(t)$ is the set of possible time intervals when packets (which depart from destination j in interval t) arrive at source i , e.g., $S_{ij}(t) = \{t - t_{ij}^{max}, \dots, t - t_{ij}^{min}\}$.

Let $\hat{n}_i(s)$, $\hat{m}_j(t)$, $\hat{x}_{ij}(s, t)$ be the estimated packet counts for $n_i(s)$, $m_j(t)$, $x_{ij}(s, t)$, respectively, which are obtained from direct Bitmap calculation (3) and (4). Let $\epsilon_i(s)$, $\epsilon_j(t)$, $\epsilon_{ij}(s, t)$ be the corresponding Bitmap estimation errors, i.e.,

$$\begin{aligned}
 n_i(s) &= \hat{n}_i(s) + \epsilon_i(s), \\
 m_j(t) &= \hat{m}_j(t) + \epsilon_j(t), \\
 x_{ij}(s, t) &= \hat{x}_{ij}(s, t) + \epsilon_{ij}(s, t).
 \end{aligned} \tag{11}$$

It was shown in [23] that the estimators $\hat{n}_i(s)$, $\hat{m}_j(t)$ are asymptotically Gaussian and unbiased. The same argument can be established for $\hat{x}_{ij}(s, t)$. Therefore, the estimation noises are asymptotically Gaussian noises.

PROPOSITION 1. $\epsilon_i(s)$, $\epsilon_j(t)$, $\epsilon_{ij}(s, t)$ are asymptotically zero mean Gaussian noises with variances $v_i(s)$, $v_j(t)$, $v_{ij}(s, t)$, given by (suppose the bitmap size is b) [23], [29]:

$$\begin{aligned}
 v_i(s) &= b \left(e^{\frac{n_i(s)}{b}} - \frac{n_i(s)}{b} - 1 \right), \\
 v_j(t) &= b \left(e^{\frac{m_j(t)}{b}} - \frac{m_j(t)}{b} - 1 \right), \\
 v_{ij}(s, t) &= b \left(2e^{\frac{x_{ij}(s, t)}{b}} + e^{\frac{n_i(s) + m_j(t) - x_{ij}(s, t)}{b}} - e^{\frac{n_i(s)}{b}} \right. \\
 &\quad \left. - e^{\frac{m_j(t)}{b}} - \frac{x_{ij}(s, t)}{b} - 1 \right).
 \end{aligned} \tag{12}$$

Substituting (11) into the conservation equation (10) we have:

$$\hat{n}_i(s) + \epsilon_i(s) = \sum_j \sum_{t \in T_{ij}(s)} (\hat{x}_{ij}(s, t) + \epsilon_{ij}(s, t)), \quad \forall i, s, \tag{13}$$

$$\hat{m}_j(t) + \epsilon_j(t) = \sum_i \sum_{s \in S_{ij}(t)} (\hat{x}_{ij}(s, t) + \epsilon_{ij}(s, t)), \quad \forall j, t. \tag{14}$$

Based on these conservation equations we want to calculate the most likely values of the Gaussian noises, and use them to refine the estimators $\hat{n}_i(s)$, $\hat{m}_j(t)$, $\hat{x}_{ij}(s, t)$. However, directly computing the maximum likelihood estimators (MLEs) for $\epsilon_i(s)$, $\epsilon_j(t)$, $\epsilon_{ij}(s, t)$ is extremely difficult if we

consider all the conservation equations simultaneously, because it involves a very large matrix. We propose an iterative proportional fitting (IPF) procedure which considers the conservation equations one by one. Since the noise terms are asymptotically Gaussian distributed, we can in fact prove that the IPF procedure will converge to the joint MLEs!

Before we describe our IPF algorithm, we state a proposition which shows that each step of the IPF algorithm tries to “locally” maximize the overall likelihood function subject to a source-based conservation equation. Its proof is straightforward based on minimizing the negative quadratic exponent in the joint Gaussian likelihood function of these uncorrelated terms.

PROPOSITION 2. *For any source node i and time interval s , the maximum likelihood estimators for $\epsilon_i(s)$, $\epsilon_{ij}(s, t)$ subject to the conservation equation (13) are:*

$$\begin{aligned}\epsilon_i^*(s) &= \frac{v_i(s)}{v_i(s) + \sum_{j,t} v_{ij}(s, t)} \left(\sum_{j,t} \hat{x}_{ij}(s, t) - \hat{n}_i(s) \right), \\ \epsilon_{ij}^*(s, t) &= \frac{v_{ij}(s, t)}{v_i(s) + \sum_{j,t} v_{ij}(s, t)} \left(\hat{n}_i(s) - \sum_{j,t} \hat{x}_{ij}(s, t) \right).\end{aligned}$$

PROOF. We know that $\epsilon_i(s)$ and $\epsilon_{ij}(s, t)$ are asymptotically zero mean Gaussian random variables with variances $v_i(s)$ and $v_{ij}(s, t)$, hence their joint probability density function is given by:

$$f(\epsilon_i(s); \epsilon_{ij}(s, t) : j, t) \propto \exp \left\{ - \left(\frac{\epsilon_i^2(s)}{2v_i(s)} + \sum_{j,t} \frac{\epsilon_{ij}^2(s, t)}{2v_{ij}(s, t)} \right) \right\}.$$

To find the maximum likelihood estimators (MLEs) of $\epsilon_i(s)$ and $\epsilon_{ij}(s, t)$ that maximize the above joint probability density function subject to the conservation equation (13). It is equivalent to solve the following convex optimization problem:

$$\begin{aligned}\text{minimize} \quad & \frac{\epsilon_i^2(s)}{2v_i(s)} + \sum_{j,t} \frac{\epsilon_{ij}^2(s, t)}{2v_{ij}(s, t)} \\ \text{subject to} \quad & \hat{n}_i(s) + \epsilon_i(s) = \sum_{j,t} (\hat{x}_{ij}(s, t) + \epsilon_{ij}(s, t)).\end{aligned}$$

Define the Lagrangian function

$$\begin{aligned}L(\lambda) &= \frac{\epsilon_i^2(s)}{2v_i(s)} + \sum_{j,t} \frac{\epsilon_{ij}^2(s, t)}{2v_{ij}(s, t)} \\ &\quad - \lambda \left(\hat{n}_i(s) + \epsilon_i(s) - \sum_{j,t} (\hat{x}_{ij}(s, t) + \epsilon_{ij}(s, t)) \right),\end{aligned}$$

where λ is the Lagrangian multiplier. The first order conditions $\frac{\partial L}{\partial \epsilon_i(s)} = 0$ and $\frac{\partial L}{\partial \epsilon_{ij}(s, t)} = 0$ imply

$$\begin{aligned}\epsilon_i(s) &= \lambda v_i(s), \\ \epsilon_{ij}(s, t) &= -\lambda v_{ij}(s, t).\end{aligned}$$

Using the conservation equation (13), we know that the MLEs of $\epsilon_i(s)$ and $\epsilon_{ij}(s, t)$ are given by

$$\begin{aligned}\epsilon_i^*(s) &= \frac{v_i(s)}{v_i(s) + \sum_{j,t} v_{ij}(s, t)} \left(\sum_{j,t} \hat{x}_{ij}(s, t) - \hat{n}_i(s) \right), \\ \epsilon_{ij}^*(s, t) &= \frac{v_{ij}(s, t)}{v_i(s) + \sum_{j,t} v_{ij}(s, t)} \left(\hat{n}_i(s) - \sum_{j,t} \hat{x}_{ij}(s, t) \right).\end{aligned}$$

□

The MLEs for the noise terms based on the destination-based conservation equations can be similarly derived. Now we are ready to present the iterative proportional fitting (IPF) algorithm.

1. Use $\hat{n}_i(s)$'s, $\hat{m}_j(t)$'s, and $\hat{x}_{ij}(s, t)$'s as the initial values of $n_i(s)$'s, $m_j(t)$'s, and $x_{ij}(s, t)$'s.
2. For source node i and time interval $s = 1, \dots, T$:
For all destination node j , and $t \in \hat{T}_{ij}(s)$, let

$$x_{ij}(s, t) = \max(x_{ij}(s, t), 0).$$

For positive $x_{ij}(s, t)$, estimate its variance $v_{ij}(s, t)$ (using (12) based on current estimation of $n_i(s)$, $m_j(t)$, $x_{ij}(s, t)$). Compute:

$$A = \sum_j \sum_{t \in \hat{T}_{ij}(s)} x_{ij}(s, t).$$

Then for positive $x_{ij}(s, t)$, proportional fitting:

$$x_{ij}(s, t) = x_{ij}(s, t) + \frac{v_{ij}(s, t)}{\sum_{j,t} v_{ij}(s, t) - v_i(s)} (n_i(s) - A),$$

$$n_i(s) = n_i(s) + \frac{v_i(s)}{\sum_{j,t} v_{ij}(s, t) - v_i(s)} (A - n_i(s)).$$

3. For destination node j and time interval $t = 1, \dots, T$:
For all source node i , and $s \in \hat{S}_{ij}(t)$, let

$$x_{ij}(s, t) = \max(x_{ij}(s, t), 0).$$

For positive $x_{ij}(s, t)$, estimate its variance $v_{ij}(s, t)$ (using (12) based on current estimation of $n_i(s)$, $m_j(t)$, $x_{ij}(s, t)$). Compute

$$B = \sum_i \sum_{s \in \hat{S}_{ij}(t)} x_{ij}(s, t).$$

Then for positive $x_{ij}(s, t)$, proportional fitting:

$$x_{ij}(s, t) = x_{ij}(s, t) + \frac{v_{ij}(s, t)}{\sum_{i,s} v_{ij}(s, t) - v_j(t)} ((m_j(t) - B),$$

$$m_j(t) = m_j(t) + \frac{v_j(t)}{\sum_{i,s} v_{ij}(s, t) - v_j(t)} (B - m_j(t)).$$

Steps 2 and 3 are repeated until a certain convergence criterion is satisfied.

Since the noise terms are asymptotically Gaussian and uncorrelated ([23], [29]), their likelihood function is concave.

This means the likelihood function has a unique stationary point which provides the maximum likelihood estimators (MLEs) of the noise terms. Therefore, the IPF algorithm, which converges to a stationary point of the likelihood function, will guarantee to return the MLEs.

In conservation equations (10) we assume there is no packet loss. Our framework can be extended to handle packet losses as well. For example, we can add additional variable: $x_{ij}(s, \infty)$, which is the number of packets sent from node i (in interval s) to node j that are lost. Then the conservation equations become:

$$n_i(s) = \sum_j \left\{ \sum_{t \in T_{ij}(s)} x_{ij}(s, t) + x_{ij}(s, \infty) \right\}, \forall i, s = 1, \dots, T,$$

$$m_j(t) = \sum_i \sum_{s \in S_{ij}(t)} x_{ij}(s, t), \forall j, t = 1, \dots, T.$$

The iterative proportional fitting algorithm can be applied to the above conservation equations with minor modifications.

5. EVALUATIONS

We first introduce our experimental methodology and the data sets. We then present the evaluation results as follows. First, we analyze the overhead of our scheme. Second, we show the accuracy of our scheme, including the estimation of mean delay, delay variance, and whole delay distribution. Third, we illustrate the ability of our scheme to detect delay bursts. Finally, we illustrate the robustness of our scheme.

5.1 Evaluation Methodology

We build an event-driven platform to simulate a high-speed network environment. One event occurs when there is a packet arrives at an ingress link or departs from an egress link. A bitmap is generated on each monitored link for each monitoring interval, the default value is 1 ms. The default size of each bitmap is 2 KB⁶.

Delay: We use both measured delay and simulated delay traces. Our measured delays are collected from one source node to five different PlanetLab⁷ nodes (illustrated Table 2) which are geographically distributed. The source node sends a probing packet every 0.1 ms. We understand that the delays measured from PlanetLab nodes may be inaccurate because these nodes are heavily used by many users, which may negatively impact the measurement accuracy. However we use these measurements only as constant parameters in our experiments to evaluate the accuracy of our decoding procedures, which is orthogonal to the inaccuracy of these measurements per se. Moreover, the delay traces from Planetlab, in some sense, reflect the case when the network has non-negligible congestion, for which SLA monitoring becomes more important.

⁶It is enough to handle the Abilene traffic trace.

⁷<http://www.planet-lab.org>.

Notation	Destination
Node0	planetlab-4.cs.princeton.edu
Node1	ricepl-2.cs.rice.edu
Node2	planetlab02.cs.washington.edu
Node3	planetlab2.cs.uiuc.edu
Node4	planet2.berkeley.intel-research.net

Table 2: The destination nodes from Planetlab.

The simulated delays are generated based on the observation of Choi et al., 99% of delay concentrate into a small range [7]. We consider three different simulated delay sets, in which the delays are: (1) always stationary; (2) usually stationary but with some short interval bursts; (3) usually stationary but can shift from one stationary state to another.

Traffic demand: We also use both real traffic demand traces and simulated traffic demands. Our real traffic demand traces were collected from Abilene on September 4th, 2004 [27]. When generating simulated traffic demands, we consider three parameters: the total amount of traffic from ingress links, the total amount traffic to egress links, and their intersections. For the first two parameters, we choose from one of the three categories of link speed: low (OC-48: 2.488 Gbps), medium (OC-192: 10 Gbps) or high (OC-768: 40 Gbps). For the third parameter, we use a *traffic ratio*, which is the percentage of the traffic sent from the ingress links. In fact, traffic ratio reflects the impact of cross traffic on estimation we mentioned before.

Evaluated algorithms: Our scheme includes different levels of implementation: the naive one (Naive) from direct bitmap estimation, the one using maximum count (MC) detection, the one using sequential hypothesis testing (SHT), and the one using both sequential hypothesis testing and iterative proportional fitting (SHT+IPF). The default scheme, Packet Doppler, is SHT+IPF. We also choose SLAM as a typical example of end-to-end probing methods. The setting of the parameters of SLA is consistent with the paper [19], 200 ms interval for mean delay estimation, 30 ms interval for delay variance estimation.

Performance metrics: In our evaluations, we use the mean, variance and distribution (PDF) of the delays as the performance metrics. In order to “zoom in” and investigate more details, we use another metric named “miss rate”. For each monitoring interval, if the delay estimation is consistent with the actual delay, we say that this estimation “hits” the actual delay. Otherwise, it “misses” the actual delay. The miss rate is defined as the number of missing intervals divided by total monitoring intervals during the whole monitoring period.

5.2 Overhead

For each packet our encoding procedure only produces a little more than one bit in the digest, which is three orders of magnitude reduction compared to the original traffic volume. Since our default size of bitmap is 2 KB, 2 MB of

Destination	Actual	SLAm	Naive	MC	SHT	SHT+IPF
Node0	16.49	0.24	0.21	0.13	0.11	0.09
Node1	53.29	0.39	0.31	0.21	0.10	0.03
Node2	87.83	0.53	0.43	0.32	0.03	0.01
Node3	33.64	1.39	1.04	0.92	0.07	0.05
Node4	78.28	2.56	1.35	0.81	0.15	0.13

Table 3: Mean delay estimation (milliseconds), using absolute error as the metric.

Destination	Actual	SLAm	Naive	MC	SHT	SHT+IPF
Node0	0.32	20.3%	18.2%	12.6%	7.5%	4.9%
Node1	0.44	54.3%	43.4%	39.3%	14.6%	9.2%
Node2	0.63	50.4%	50.3%	49.1%	8.3%	5.1%
Node3	0.52	36.5%	35.1%	32.0%	3.8%	1.9%
Node4	3.21	70.1%	52.2%	43.0%	5.6%	2.6%

Table 4: Delay variance accuracy, using relative error as the metric.

digests will be generated and stored every second. Accordingly, if the SLA monitoring process is performed online, the transmission bandwidth cost is also 2 MB per second. For SLAm, the bandwidth of one pair probing is 80Kb/s. However, given that our scheme can use n link to estimate n^2 pairs. The bandwidth is much less than SLAm when link number n is large (e.g. hundreds or thousands).

In section 5.5, we also evaluate our scheme for different link speed (based on different load factor). Typically, for an OC-192 link, about 1~2 MB of digests will be generated and stored every second. For an OC-768 link, the number is about 10 MB. Moreover, the storage and bandwidth overhead can be further reduced using data compression.

5.3 Accuracy

In this part, we use measured delay traces from Planetlab nodes and the real traffic demands from Abilene to perform the evaluation.

Mean Delay: Table 3 shows the results of mean delay estimation. Our algorithms (except the naive one) can achieve precision within one millisecond of the actual mean delay. The improvement is significant by applying SHT and IPF. For example, SHT can improve the accuracy of the naive scheme by more than one order magnitude. By contrast, SLAm may suffer from larger errors, e.g., 2.56 ms. Given that most SLA defines the mean delay on millisecond level [7], the precision of SLAm may fail to satisfy the requirement in some cases.

Delay Variance: Table 4 shows the estimation results for delay variance of different schemes. In general, the relative errors of SLAm are several or even tens of times larger than the relative errors of our scheme. The advantage of our scheme is more clear when the delay variance is getting larger. The reason is that with a fixed probing rate, probing-based methods will miss more changes when the delay process is becoming more unstable. Moreover, a large variance

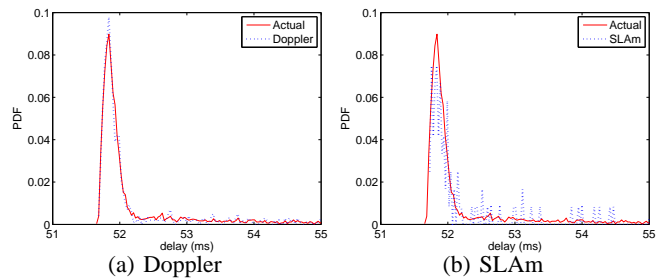


Figure 4: Delay estimation of Node 1.

of the delays is usually caused by congestion in the network. Under this situation probing-based methods need to increase the probing rate (hence injecting more probing traffic) in order to improve the accuracy of delay estimation, while more probing traffic will lead to more congestion and perturbation to the network. So eventually the increase of probing rate will result in larger bias of delay estimation. From Table 4, we also find that both SHT and IPF can significantly reduce the estimation noises and improve the accuracy of estimation. Typically, SHT can reduce the relative errors from tens of percents to several percents, and IPF can further cut the estimate errors by nearly 50%.

Distribution Estimation: Our scheme can not only estimate mean delay and delay variance but also estimate the whole delay distribution. Figure 4 illustrates the delay distribution (PDF) estimation of both Packet Doppler (SHT + IPF) and SLAm in Node1. The estimation of Packet Doppler matches the actual delay distribution very well (the subgraph on the left side). The cases in other nodes are very similar.

5.4 Delay Burst Detection

In a large ISP network, the delays between the OD pairs may experience unexpectedly changes. Moreover, many studies illustrate that these burst events are short-lived. Wang *et al.* found that 90% transient BGP routing failures last less than 10 seconds. According to the study of Markopoulou *et al.*, there are at least 10% of link failures last less 1 second in Sprint backbone network.

The similar condition has been found in our delay traces. We randomly select two intervals [4000,4300] and [8300,8600] from Node 0 trace. There are many delay bursts in these two short intervals, as shown in Figure 5. Our scheme can effective detect the most of burst events (just miss the first small burst from [4000,4300] and miss 0 burst from [8300,8600]). Its estimated value matches the real trace so well that it is difficult to distinguish the solid and lotted lines. As a comparison, SLAm will miss many burst events (only capture one burst in each condition), and estimation value is far away from the actual value when burst happens.

5.5 Robustness

In this subsection, we illustrate the robustness of our

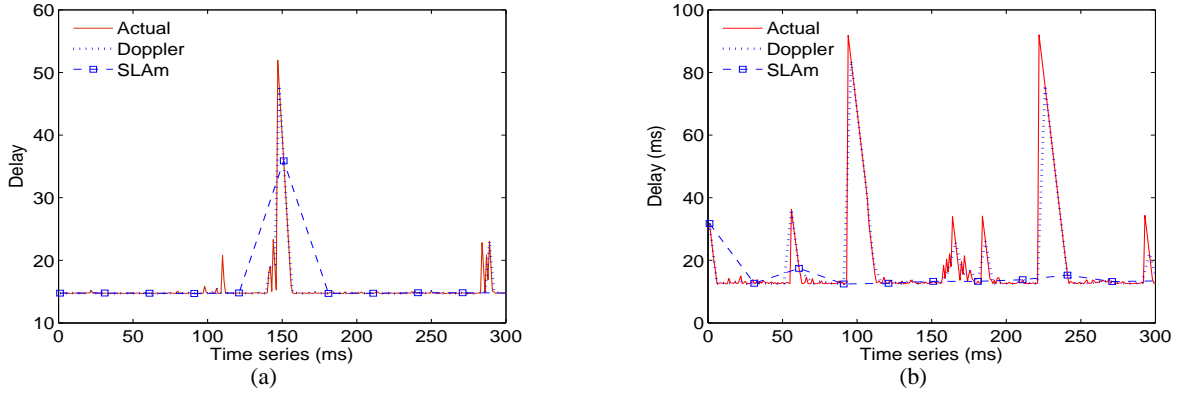


Figure 5: The detection of burst events. (a) The delay trace of the interval [4000,4300], from Node 0. (b) The delay trace of the interval [8300,8600], from Node 0.

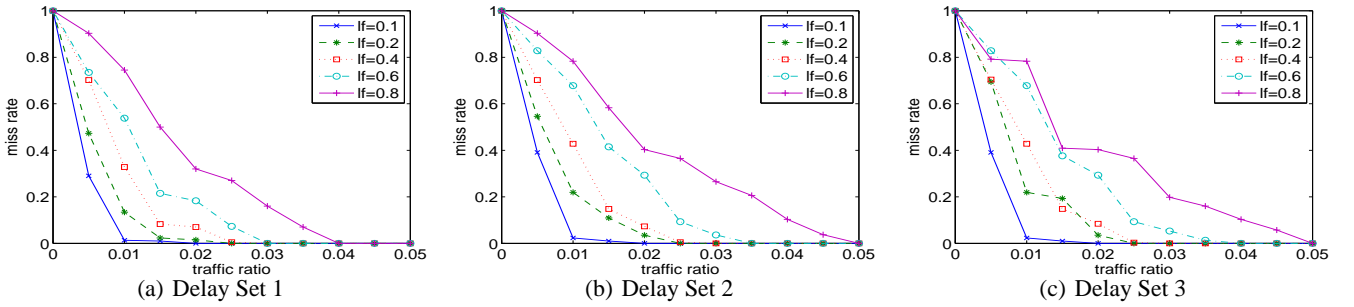


Figure 6: The change of miss rate when varying traffic ratio.

scheme: choosing different values for the parameters and testing different algorithms, the estimating results can match the actual delays well in most cases. In order to control the network settings, we use the simulated delay sets and traffic demands, as we discussed in Section 5.1.

Load factor: The bitmap size is an important parameter for the algorithm. Basically, when the traffic volume is large, we need large bitmap to encode the information. We use load factor to describe the bitmap size. Figure 6 shows the estimation results of the MC scheme under three different delay sets when varying the traffic ratio. In general, there is a tradeoff between the load factor (bitmap size) and the estimation precision. With a fixed traffic ratio, when the load factor is smaller, the bitmap size is larger. For example, in order to estimate the delays accurately for a traffic ratio of 3%, the load factor should be less than 0.6. In other words, the bitmap size should be more than 1.6 times of the number of packets arriving during a interval. From Figure 6 we also find that with a load factor of 0.1, we can always estimate the delays with very small miss rate.

Different algorithms: In this part, we investigate the effectiveness of different algorithms to handle small flows. From Figure 6 we found that the simple MC algorithm is very ef-

fective to detect fine-grained delay metrics under large flow conditions (*e.g.*, larger than 5%), *i.e.*, the miss rate is nearly zero when the traffic ratio is large enough, regardless of the load factor, but it may not produce accurate results for small flows. Now we will show that SHT can improve the accuracy for small flows. Table 5 shows the improved results. We simulate three link speed: OC-48 (2.488 Gbps), OC-192 (10 Gbps), and OC-768 (40 Gbps). The typical values of shared traffic ratio is chosen to show the capability of our estimation methods under different source and destination link speed conditions. If the traffic ratio is larger than typical values we choose, our schemes can produce even more accurate estimation (near zero miss rate).

From Table 5 we can see that the improvement of sequential hypothesis testing is obvious. Compared with MC implementation and HT(5), SHT can significantly reduce the miss rate. In this way, the capacity of bitmap estimation can be increased to estimate delays for small flows with 1% or less traffic.

Another observation from Table 5 is that normally a high traffic rate can help us to predict the delays more accurately, which implies that our scheme is adaptive to the keeping growth of backbone link speed. People may have concern that the bitmap size will also grow accordingly. But in fact,

Source	Destination	traffic ratio	miss rate		
			MC	HT(5)	SHT
OC-48	OC-48	2.0%	34.8%	14.3%	6.5%
		3.0%	13.9%	7.2%	2.0%
		4.0%	1.8%	1.3%	0.0%
OC-192	OC-192	1.0%	54.8%	44.4%	14.0%
		2.0%	20.8%	10.0%	4.0%
		3.0%	3.6%	1.2%	0.0%
OC-768	OC-768	0.3%	70.3%	56.4%	27.0%
		0.5%	43.2%	20.3%	4.3%
		1.0%	28.4%	10.20%	0.00%
OC-48	OC-192	1.0%	56.30%	40.6%	16.0%
		2.0%	32.4%	18.4%	5.0%
		3.0%	7.30%	5.2%	0.0%
OC-192	OC-768	0.5%	37.7%	26.4%	11.0%
		1.0%	10.3%	8.0%	2.0%
		2.0%	2.6%	1.5%	0.0%
OC-48	OC-768	1.0%	45.3%	40.4%	21.0%
		2.0%	21.0%	16.3%	3.2%
		3.0%	5.3%	4.1%	0.0%

Table 5: Applying hypothesis testing to handle small flows

the relative storage/bandwidth consumption will not grow since the bitmap size is linear to traffic rate.

6. RELATED WORK

Measuring the characteristics of Internet traffic (*e.g.*, traffic volume, average delay, delay distribution, packet loss rate) is critical for diagnosing current performance problems as well as for designing future distributed services. Unfortunately, the Internet architecture was not designed with performance measurement as a primary goal and therefore has few “built-in” services that support this need [8]. Consequently, measurement tools must either “make do” with the services provided by the Internet, or deploy substantial new infrastructures geared towards measurement. Accordingly, we can divide the mainstream network measurement methodology into two categories: probing based (active) and router based (passive).

There has been a great deal of work on active measurements of end-to-end delay, delay variation, and loss (*e.g.*, [4] [15] [16] [24] [25]). The major problems of probing based methods are discussed in Section 1. Another problem of end-to-end probing is scalability. In order to get the information of the whole network for one ISP, the end-to-end probing will be sent on every pair of end hosts or ingress/egress links. For instance, there are currently thousands (n) of links in AT&T network. In order to estimate the delay for each OD pair, there will be millions (n^2) probing pairs. Although recent research shows that the cost can be reduced to $O(n \log n)$ by using network tomography ideas [6], the scalability is still an issue, given the fact that network size is keeping growing.

Compared with probing based methods, router based methods passively record the traffic information in the network. A standard approach of passive monitoring is to use

management information base (MIB) counters available on the routers via the Simple Network Management Protocol (SNMP) or sampled NetFlow records. For example, Barford *et al.*[3] compared the probing based and router based methods to measure packet loss and illustrated that SNMP method is generally better. However, SNMP provides only coarse-grain information, which cannot be used to estimate delay accurately. Moreover, current research illustrates that both SNMP and NetFlow will introduce some dirty data [28].

7. CONCLUSIONS

In this paper, we propose Packet Doppler as a scheme to monitor service level agreement metrics, in particular, delay distribution. Packet Doppler is passive and thus will not cause perturbation to real traffic. It is also scalable with low storage and bandwidth overhead. Our evaluations using realistic delay and traffic demands show that Packet Doppler achieves high accuracy and can detect burst events that will be missed by probing based methods.

There are many avenues for future work. In particular, by exploring the structural constraints present in SLA measurements, Packet Doppler makes significant improvement in decoding accuracy. It is an interesting question to further explore structural constraints in the SLA monitoring domain. Moreover, in this paper, we focus on the estimation of delay distribution. Another important performance metric in network monitoring is packet loss. Our packet shift framework can be extended to detect packet loss events, since loss means that the shift length of the lost packets is infinite. Assume that the traffic is relatively stable within a short time interval, then a dramatic change of traffic volume on time series between continuous measurement intervals can be viewed as an evidence of packet loss event.

8. REFERENCES

- [1] F. Baccelli, S. Machiraju, D. Veitch, and J. C. Bolot. The role of PASTA in network measurement. *SIGCOMM Comput. Commun. Rev.*, 36(4):231–242, 2006.
- [2] F. Baccelli, S. Machiraju, D. Veitch, and J. C. Bolot. On optimal probing for delay and loss measurement. In *IMC '07: Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pages 291–302, New York, NY, USA, 2007. ACM.
- [3] P. Barford and J. Sommers. Comparing probe- and router-based packet loss measurements. In *IEEE Internet Computing*, 2004.
- [4] J. Bolot. End-to-end packet delay and loss behavior in the Internet. In *ACM SIGCOMM*, Sept. 1993.
- [5] M. C. Chan, Y.-J. Lin, and X. Wang. A scalable monitoring approach for service level agreements validation. In *ICNP*, 2000.
- [6] Y. Chen, D. Bindel, H. Song, and R. H. Katz. An algebraic approach to practical and scalable overlay network monitoring. *SIGCOMM '04.*, pages 55–66, 2004.
- [7] B.-Y. Choi, S. Moon, Z.-L. Zhang, K. Papagiannaki, and C. Diot. Analysis of point-to-point packet delay in an operational network. *Elsevier Journal of Computer Networks*, 51(13):3812–3827, Sept. 2007.
- [8] D. Clark. The design philosophy of the DARPA Internet protocols. In *ACM SIGCOMM*, Sept. 1988.
- [9] T. Clark. Tom clark’s totally accurate clock ftp site. [ftp://aleph.gsfc.nasa.gov/GPS/totally.accurate.clock/.](ftp://aleph.gsfc.nasa.gov/GPS/totally.accurate.clock/)

- [10] S. Jaiswal, G. Iannaccone, C. Diot, J. Kurose, and D. Towsley. Measurement and classification of out-of-sequence packets in a Tier-1 IP backbone. In *ACM SIGCOMM Internet Measurement workshop*, 2002.
- [11] T. N. Limited. Data sheet and specifications for trimble thunderbolt GPS disciplined clock. Available at <http://www.trimble.com/thunderbolt.html>.
- [12] A. Markopoulou, G. Iannaccone, S. Bhattacharyya, C.-N. Chuah, and C. Diot. Characterization of failures in an IP backbone. In *INFOCOM*, 2004.
- [13] D. L. Mills. A computer-controlled LORAN-C receiver for precision timekeeping. In *Technical Report, Department of Electrical and Computer Engineering, University of Delaware*, 1992.
- [14] D. L. Mills. A precision radio clock for WWV transmissions. In *Technical Report, Department of Electrical and Computer Engineering, University of Delaware*, 1997.
- [15] A. Pasztor and D. Veitch. A precision infrastructure for active probing. In *ACM Passive and Active Measurement Workshop*, Sept. 2001.
- [16] V. Paxson. Measurements and analysis of end-to-end Internet dynamics. In *PhD thesis, University of California Berkeley*, 1997.
- [17] M. Roughan. Fundamental bounds on the accuracy of network performance measurements. In *SIGMETRICS '05*, pages 253–264, New York, NY, USA, 2005. ACM.
- [18] A. Snoeren, C. Partridge, L. Sanchez, C. Jones, F. Tchakountio, S. Kent, and W. Strayer. Hash-based IP traceback. In *ACM SIGCOMM*, Aug. 2001.
- [19] J. Sommers, P. Barford, N. Duffield, and A. Ron. Accurate and efficient SLA compliance monitoring. In *SIGCOMM '07*, pages 109–120, New York, NY, USA, 2007. ACM.
- [20] X. Ta and G. Mao. Online end-to-end quality of service monitoring for service level agreement management. In *International Journal of Communication Systems*, 2007.
- [21] A. J. Viterbi. Error bounds for convolutional codes and asymptotically optimum decoding algorithm. *IEEE Trans. Inform. Theory*, pages 260–269, 1967.
- [22] F. Wang, L. Gao, J. Wang, and J. Qiu. On understanding of transient Interdomain routing failures. In *ICNP*, 2005.
- [23] K. Whang, B. Vander-zanden, and H. Taylor. A linear-time probabilistic counting algorithm for database applications. *IEEE transaction of Database Systems*, pages 208–229, June 1990.
- [24] M. Yajnik, S. Moon, J. Kurose, and D. Towsley. Measurement and modeling of temporal dependence in packet loss. In *Proceedings of IEEE INFOCOM 99*, 1999.
- [25] Y. Zhang, N. Duffield, V. Paxson, and S. Shenker. On the constancy of internet path properties. In *Proceedings of ACM SIGCOMM Internet Measurement Workshop 01*, 2001.
- [26] Y. Zhang, M. Roughan, C. Lund, and D. Donoho. An information-theoretic approach to traffic matrix estimation. In *ACM SIGCOMM*, Aug. 2003.
- [27] Y. Zhang, M. Roughan, C. Lund, and D. L. Donoho. Estimating point-to-point and point-to-multipoint traffic matrices: an information-theoretic approach. *IEEE/ACM Trans. Netw.*, 13(5):947–960, 2005.
- [28] Q. Zhao, Z. Ge, J. Wang, and J. Xu. Robust traffic matrix estimation with imperfect information: making use of multiple data sources. *ACM SIGMETRICS*, 34(1):133–144, 2006.
- [29] Q. Zhao, A. Kumar, J. Wang, and J. Xu. Data streaming algorithms for accurate and efficient measurement of traffic and flow matrices. In *ACM SIGMETRICS*, June 2005.
- [30] T. Zseby. Deployment of sampling methods for SLA validation with non-intrusive measurements. In *Passive and Active Measurement Workshop*, 2001.

APPENDIX

A. COMPUTE $\Pr(\hat{x}|\theta)$

LEMMA 1. *Given two non-overlapping sets A and D , we hash these two sets into bitmaps B_A and B_D with bitmap*

size b . Let n and m be the number of bits with value 1 in B_A and B_D , respectively. Let X denote the intersection between B_A and B_D , i.e., the number of bits with value 1 in $B_A \wedge B_D$. The probability distribution of X is given as follows:

$$\begin{aligned} \Pr(X = x) &= \frac{\binom{n}{x} \binom{b-n}{m-x}}{\binom{b}{m}} \\ &= \frac{n!m!(b-n)!(b-m)!}{(n-x)!(m-x)!(b-n-m+x)!b!x!} \\ &\triangleq PF(b, n, m, x), \quad x \leq \min(n, m). \end{aligned} \quad (15)$$

PROOF. For the two bitmaps with size b , given that we (randomly) set n bits in the first bitmap to 1, there exist $\binom{b}{m}$ different choices to set m bits in the second bitmap to 1. Among these choices, we can compute the number of choices where the intersection between the two bitmaps is exactly x , as follows: suppose the set of bits with value 1 in the first bitmap is I ($|I| = n$), we first randomly set x bits in the set I of the second bitmap to 1, which we have $\binom{n}{x}$ choices; and then randomly set $m - x$ bits in the set I^C of the second bitmap to 1, which we have $\binom{b-n}{m-x}$ choices. This will guarantee that the intersection between the two bitmaps is x and the second bitmap has m bits with value 1. \square

PROPOSITION 3. *Given two sets A and D with the corresponding bitmaps B_A and B_D which have n and m bits with value 1, respectively, and θ common bits with value 1 between B_A and B_D are caused by the actual overlap between A and D while others are caused by hash collisions, then the probability of observing \hat{x} common bits with value 1 in $B_A \wedge B_D$ is given by:*

$$\Pr(\hat{x}|\theta) = PF(b - \theta, n - \theta, m - \theta, \hat{x} - \theta), \quad \hat{x} \geq \theta.$$

PROOF. We can “eliminate” the θ common number of bits with value 1 between B_A and B_D that are caused by the actual overlap of A and D from consideration. This is equivalent to considering two bitmaps with size $b - \theta$ that are generated by two non-overlapping sets, where the first bitmap has $n - \theta$ bits with value 1, the second bitmap has $m - \theta$ bits with value 1, and their intersection is $\hat{x} - \theta$. Therefore we can apply Lemma 1 to compute $\Pr(\hat{x}|\theta)$. \square