Privacy-Preserving Performance Measurements

Matthew Roughan
School of Mathematical Science
University of Adelaide
SA 5005, Australia
matthew.roughan@adelaide.edu.au

Yin Zhang
Department of Computer Sciences
University of Texas at Austin
Austin, TX 78712, USA
yzhang@cs.utexas.edu

ABSTRACT

Internet performance is an issue of great interest, but it is not trivial to measure. A number of commercial companies try to measure this, as does RIPE, and many individual Internet Service Providers. However, all are hampered in their efforts by a fear of sharing such sensitive information. Customers make decision about "which provider" based on such measurements, and so service providers certainly do not want such data to be public (except in the case of the top provider), but at the same time, it is in everyones' interest to have good metrics in order to reduce the risk of large network problems, and to test the effect of proposed network improvements.

This paper shows that it is possible to have your cake, and eat it too. Providers (and other interested parties) can make such measurements, and compute Internet-wide metrics securely in the knowledge that their private data is never shared, and so cannot be abused.

Categories and Subject Descriptors

C.2.3 [**Computer-Communications Networks**]: Network Operations—*network monitoring, network management*; H.2.8 [**Database Management**]: Database Applications—*data mining*

Keywords

Secure distributed computation, privacy-preserving data-mining, network, management, performance, measurement.

1. INTRODUCTION

Imagine three Internet Service Providers (ISPs) who wish to make measurements of performance. They might wish to make such measurements to test improvements to their network, or detect major faults. ISPs can make internal measurements, but even for large providers the majority of traffic is "off-network" in the sense that it must traverse other networks besides their own. Additionally, it is (anecdotally) reported that many performance problems (congestion, and routing in particular) occur at the links between providers. Hence, inter-provider performance metrics are actually more important than the internal measures. Large-scale asymmetry in Internet (inter-domain) routing leads to the need for one-way performance measurements, but collection of such metrics requires some co-operation between providers. However, efforts to create co-operation are hampered by a fear of sharing sensitive information. Customers make decision about "which provider" based on such measurements, and network operators might also reveal more than they wish to their competitors.

It is sometimes possible to create bilateral agreements to make performance measurements, but these are highly inflexible, and multilateral agreements are very hard to reach. In one case an agreement was created through a trusted third party (RIPE) who now conduct many such measurements primarily in Europe [10], but other attempts have not been successful (for instance, for several years a group of providers met at NANOG to set up such measurements without ever reaching an agreement). In the United States and Asia no universally trusted third party has emerged. It seems then, that some work needs to be done to create such a measurement infrastructure. On the research side, several such exist (e.g., NIMI [14], PlanetLab [15],and PingER [6]), but they have well known limitations (primarily in that they focus on academic networks, rather than the commercial Internet). There have also been commercial attempts to build such infra-structure, but the technology behind these has not been subject to the scientific rigour that would be required, for instance, if one were measuring the performance of a new drug.

This paper is aimed at creating simple techniques that could be used to allow multilateral performance measurements without the need for a trusted third party, or to expose a provider's sensitive information. The paper is not intended to discount the prior work in the art of making measurements — in fact we rely heavily on all of the excellent work pioneering such measurements (in particular on synchronization of clocks (e.g., see [5,10]). This paper, in contrast, presents a method for avoiding the problems of trust implicit in making inter-provider measurements. We use techniques related to those described in [17], whereby one can perform a distributed calculation without sharing the inputs. Techniques to perform such computations are now well developed, though they have rarely been applied to communications networks (exceptions being [3,12,17]).

The methods stem from seminal work by Yao [20], who proposed a set of solutions to two-party problems such as the millionaires' problem: two millionaires wish to determine who is richer, but they do not wish to reveal their individual wealth to each other. Yao showed that one could solve a general class of such problems, and subsequent work has generalized this to multiple-parties of varying degrees of honesty. We use techniques from this research area, sometimes called secure-distributed computation, or Privacy-Preserving Data Mining (PPDM). The major differentiator of our work over prior work in PPDM is that in much of the preceding literature (for instance see our own work [17]), the aim has been

SIGCOMM'06 Workshops September 11-15, 2006, Pisa, Italy.

to perform data-mining on datasets collected by multiple parties. Each party holds one piece of data, and we wish to perform some algorithm (e.g., regression) on the collection of data. However, in our case, the measurements themselves are distributed. To clarify, in [17] the measurements to be combined were *intra-domain* measurements (for example of traffic volumes), which can be performed by a single network operator. The goal was to apply PPDM to combine each of these individual datasets to form a global picture (e.g., of the total Internet traffic). In this paper we consider *inter-domain* measurements, where the measurements themselves require the co-operation of multiple providers.

We show that there are many problems one can solve in this domain: for example, our methods allow a provider to determine its average inter-domain delay without any provider learning inter-domain delays between specific providers. Thus one can measure one's own performance without learning any sensitive information about other networks. The methods applied are actually rather simple, requiring only basic mathematical and cryptographic operations, though measuring delay in the presence of packet loss (where we cannot reveal which packets are lost) is somewhat more challenging. We present a new PPDM algorithm that allows computation of average delays where some probe packets are dropped.

2. THE PROBLEM

We pose a simple inter-provider performance measurement problem. A group of N Internet Service Providers (ISPs) wish to be able to measure their inter-domain performance. They can do so by sending probe packets between servers on each network and measuring the delays these packets experience. However, they each have concerns about the other learning any information about their network. In particular, imagine that they do not wish anyone to learn the delays between their network, and any other network. It appears on the surface that this is an intractable problem. How can we use such measurements without automatically learning the implicit information in the measurements?

First, let us refine our notation to allow a precise description of the problem. Imagine that the N providers send probe packets between networks during measurement interval $[0, T]$, and record the transmission and arrival times of the packets[1,2]. Note however, that they will not share this information. Formally, we define $t_{ij}^{(k)}$ and $r_{ij}^{(k)}$ to be the transmission and receive times, respectively, of the kth probe packet between networks i and j. The delay (or latency) measurement is given by $d_{ij}^{(k)} = r_{ij}^{(k)} - t_{ij}^{(k)}$. The data is initially partitioned in the sense that

$$\begin{array}{ll} t_{ij}^{(k)} & \text{is known to node } i, \\ r_{ij}^{(k)} & \text{is known to node } j. \end{array} \tag{1}$$

For the moment we shall only consider the case where no packets are dropped, though in Section 4 we generalize to allow packet loss. The average delay between nodes is defined to be

$$\bar{D}_{ij} = \frac{1}{K_{ij}} \sum_{k=1}^{K_{ij}} r_{ij}^{(k)} - t_{ij}^{(k)} = \bar{R}_{ij} - \bar{T}_{ij}. \tag{2}$$

where there are K_{ij} probe packets between networks i and j, and $\bar{R}_{ij} = \frac{1}{K_{ij}} \sum_{k=1}^{K_{ij}} r_{ij}^{(k)}$ and $\bar{T}_{ij} = \frac{1}{K_{ij}} \sum_{k=1}^{K_{ij}} t_{ij}^{(k)}$. The $\bar{D}_{ij}$ are considered to be information that ISPs do not wish to allow to slip into the hands of their competitors. However, they do wish to know $\bar{D}_i^{\text{out}}$, and $\bar{D}_i^{\text{in}}$, the average delay of packets going out of, and coming into their network, as well as the average delay across all networks. Such data are useful for benchmarking themselves against the notional average provider, detecting network problems, and long-term planning. Formally, we wish keep the values $\bar{D}_{ij}$ secret from all participants, but to compute

$$\bar{D}_i^{\text{out}} = \frac{1}{N-1} \sum_{\substack{j=1 \\ j \neq i}}^{N} \bar{D}_{ij}, \tag{3}$$

$$\bar{D}_i^{\text{in}} = \frac{1}{N-1} \sum_{\substack{j=1 \\ j \neq i}}^{N} \bar{D}_{ji}, \tag{4}$$

$$\bar{D} = \frac{1}{N} \sum_{i=1}^{N} \bar{D}_i^{\text{out}} = \frac{1}{N} \sum_{i=1}^{N} \bar{D}_i^{\text{in}}. \tag{5}$$

The obvious approach to making such a calculation is for a trusted third party to collect the data, and share the results. In the absence of such a third party, there is another way to proceed, which we discuss in Section 3.

2.1 Assumptions

In any PPDM we must define the security model under consideration. We will assume here the commonly used "honest-but-curious" model. That is we assume that the ISPs in question are honest in the sense that they follow the algorithms correctly, but they are curious in the sense that they will perform additional operations in order to attempt to discover more information than intended. The honest-but-curious assumption has been widely used, and appears applicable here. ISPs will benefit from participating honestly in a such a scheme, and there is no downside in participating honestly, and so dishonest partners in computation (partners who do not follow the algorithm) will reduce their own benefits, without any notional gain.

It is noteworthy that while we assume that participants follow the algorithm correctly, we do allow collusion. Multiple partners are allowed to collude to attempt to learn more information than they otherwise could. The protocols we present can be made resistant to such collusion in the presence of a majority of non-colluding participants. Additionally, there is now a substantial literature on secure distributed computation and data mining (*e.g.,* see [1–4, 18, 20, 21] and the references therein), and this literature considers many variations on the type of assumptions considered here. It is likely that the assumption of honest-but-curious participants can therefore be substantially weakened.

On the practical side of the measurements, we assume that the measurements themselves are accurate. We do not include any measurement noise, or artifacts (e.g., such as might be introduced by imperfect clock synchronization). There is a large literature on performing such measurements (including IETF standards), and so we assume these problems are solved, though we realize that it may be non-trivial to do so, particular in the case of one-way measurements. We primarily consider one-way measurements here. Round-trip measurements can be performed (at least to a rough degree of accuracy) using mechanisms like 'ping' without any co-operation. However inter-domain routing is fundamentally asymmetric, and so one-way measurements are far more useful. For example, a large discrepancy in $\bar{D}_i^{\text{out}}$ and $\bar{D}_i^{\text{in}}$ might be a useful indication that routing problems exist.

There is an additional issue of importance here (as opposed to the majority of the PPDM literature). We consider inter-domain

[1] We assume that the clocks are carefully synchronized, which is non-trivial, but the issues have been well investigated [5, 10].

[2] A packet may arrive after the end of the measurement interval $[0, T]$ which we use to specify the transmission times of packets.

measurements here, that is, the measurements themselves are distributed (not just the datasets in question). As such, the measurements must be carried out in such a way that they don't reveal information in themselves. We must take steps to avoid a receiver being able to infer packet loss/delay from its measurements alone. Hence

- We avoid using a regular (uniform) sampling pattern for probe transmission times because this might allow the inference of delays (by the receiver), and also loss, through gaps in the pattern.
- We avoid putting sequence numbers in packets, because once again, out-of-order packets could then be used to infer loss.

The irregular sampling pattern we use is a Poisson Process (PP). Poisson probes have previously been proposed to avoid synchronization with network periodicities. Their advantage lies in the Poisson Arrivals See Time Averages (PASTA) property [19], and the fact that one cannot anticipate points in a PP [19]. Poisson sampling removes any ability to anticipate when samples "should have" happened, and hence one cannot determine their ordering from their arrival time. Furthermore, when events are removed (at random) from a PP the resulting stream is also a PP, and hence loss would be much harder to infer from such a sequence. We might infer loss by observing the difference between the observed number of probe packets, and the (user configurable) probe rate, but where loss rates are small, and the observation time not unduly larger, this strategy is unlikely to provide any reasonable degree of accuracy, given the natural variability in the number of Poisson probes.

3. THE SOLUTION

Let us delve slightly further into the calculation we wish to perform, in particular to compute $\bar{D}_i^{\text{out}}$. From (2) and (3) we get

$$\begin{aligned} \bar{D}_i^{\text{out}} &= \frac{1}{N-1} \sum_{\substack{j=1 \\ j \neq i}}^{N} \frac{1}{K_{ij}} \sum_{k=1}^{K_{ij}} \left[r_{ij}^{(k)} - t_{ij}^{(k)} \right] \\ &= \frac{1}{N-1} \left[\sum_{\substack{j=1 \\ j \neq i}}^{N} \bar{R}_{ij} - \sum_{\substack{j=1 \\ j \neq i}}^{N} \bar{T}_{ij} \right] \end{aligned} \quad (6)$$

The second summation involves only terms such as $t_{ij}^{(k)}$ and so is known to ISP i. However, ISP i does not know the receive times $r_{ij}^{(k)}$, so the problem (for ISP i) becomes that of calculating

$$\bar{R}_i^{\text{out}} = \frac{1}{N-1} \sum_{j=1, j \neq i}^{N} \bar{R}_{ij}, \quad (7)$$

the average receive time over the probes. Each of the terms $\bar{R}_{ij}$ is known to ISP j, but we cannot directly share these values with ISP i because this would reveal $\bar{D}_{ij}$ for each j.

The problem amounts to a distributed summation (a summation over a distributed set of datapoints), for which approaches are now well understood (see [4, 17] and the references therein). There are at least three approaches to the problem with varying degrees of efficiency and robustness to collusion. We describe the simplest here for the purpose of exposition. The problem is a special case of the problem of computing $V = \sum_{j=1}^{N} v_j$, where the individual values v_j are kept by ISP j, and considered to be secret. We can perform the distributed summation as follows. Assume the value V is known to lie in the interval $[0, n]$, where n may be large. Start from a particular ISP (we will denote this ISP 1), and list the other ISPs in some order with labels $2, 3, \ldots, N$. ISP 1 generates a random number R uniformly on the interval $[0, n]$, adds R to its value $v_1 \mod n$, and sends this to the next ISP, which adds its value, and repeats, until we get to the end of the sequence. The last ISP returns the total to ISP 1, who can then subtract the original random number R, and compute the total (and then provide this number to the other ISPs). Formally we specify the algorithm by

```
ISP 1:  randomly generate R ~ U(0,n)
ISP 1:  compute s_1 = v_1 + R  mod n
ISP 1:  pass s_1 to ISP 2
for i=2 to N
   ISP i:  compute s_i = s_{i-1} + v_i  mod n
   ISP i:  pass s_i to ISP i+1  mod N
endfor
ISP 1:  compute v_N = s_N - R  mod n
```

Each ISP $i = 2, \ldots, N$ has only the information v_i and s_{i-1}, which can be written in full as

$$s_i = R + \sum_{i=1}^{i} v_i \mod n. \quad (8)$$

Since this value is uniformly distributed across the interval $[0, n]$, ISP i learns nothing about the other values v_j, $j \neq i$. At the last step, ISP 1 has s_N, and when it subtracts R away it gets the V. Note that where we wish to allow computation of quantities that may be negative, the condition that $V \in [0, n]$ can be easily replaced by $V \in [-n/2, n/2]$. The algorithm above requires only that we adjust the range of the initial sum (*i.e.,* in the second step ISP 1 takes $s_1 = n/2 + v_1 + R \mod n$), and that in the last step ISP 1 reverses this addition (*i.e.,* $v_N = s_N - n/2 - R \mod n$).

Any ISP, given V can compute $V - v_i = \sum_{j \neq i} v_i$, and so this approach only works for $N > 2$, and in reality, where one could make meaningful guesses about some values, it is only really secure for reasonable values of N, and this is the case we consider here.

This incredibly simple process can, unfortunately, be corrupted if ISPs collude. If ISP $l-1$ and ISP $l+1$ share information, they can compute v_l by taking $s_l - s_{l-1}$ (s_l is received by ISP $l+1$, and s_{l-1} is sent by ISP $l-1$). There are various approaches to avoid this issue, as well as dealing with potentially unreliable partners in the summation, for instance see [17] for more discussion.

Once we compute $\bar{R}_i^{\text{out}}$ via a distributed summation, it is a simple matter for ISP i to compute $\bar{D}_i^{\text{out}}$ via (6). This approach provides a powerful, yet very simple way of combining transmit and receive times in such a way that we can compute the average performance without revealing the specifics. However, there are many generations, and we consider these below.

3.1 Other statistics

There are many other possible statistics of interest. In particular, percentiles of the distribution of delay (including the median, or 50th percentile), the minimum, and maximum delays, and higher-order statistics such as variance. All of these may be calculated from the distribution of delays. Note that the distribution may also reveal information that is sensitive even if the computation procedure is secure[3], and so we must take this into consideration in any future work.

Furthermore, note that the receive times may not be ordered, in the sense that we may have $r_{ij}^{(k)} > r_{ij}^{(k+1)}$, resulting in an ambiguity in packet measurements where the packets lack sequence

[3] Any multiparty computation (secure or otherwise) gives out the result, and hence cannot be secure against revealing whatever can be deduced from the input of a party and the ultimate output.

numbers. Figure 1 shows a set of measurement times and two possible interpretations of these. The interpretation doesn't matter when computing the averages, but is very important for higher-order statistics. Without some kind of packet ID we cannot resolve these ambiguities. We might choose some kind of cryptographic approach to create packet IDs, but we suggest a simpler approach.

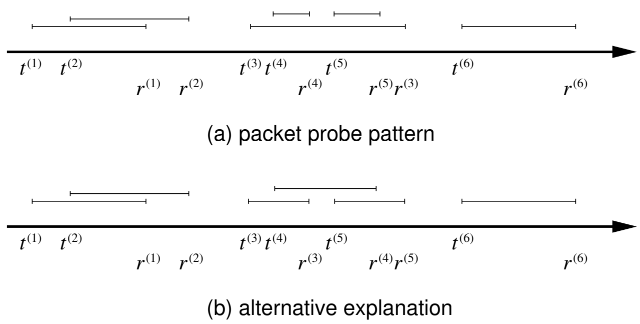

Figure 1: Example transmission and receive patterns.

Rather than try to preserve everything as secret, we will allow some information to leak, while keeping the critical information a secret. We do so by putting a timestamp into each probe packet, however, we do not put the transmission time into the packet. At the start of the experiment, each operator chooses a random number S_i uniformly over the interval $[0, S]$, where S is chosen so that it is larger than the longest reasonable delay (for instance, it might be chosen to take the value of the loss detection timeout), perhaps a value of the order of 10 seconds. When the experiment nominally starts, the actual transmission of the first packet from node i is delayed by S_i, and each packet is timestamped with the time $t_{ij}^{(k)} - S_i$. Therefore, the receiver j can determine

$$d_{ij}^{(k)\prime} = r_{ij}^{(k)} - t_{ij}^{(k)} + S_i, \tag{9}$$

from which j may compute a distribution for d'_{ij}. Note however, that j does not know S_i, so it cannot infer the true distribution – in effect j sees a shifted distribution. If j were to transmit this information back to i, then i would know the true distribution (which is not allowed), and so j cannot perform this step directly. However, j may approximate the distribution (via one of two approaches described below), and then perform a distributed summation to obtain a shifted distribution for the group of receivers $\{j | j \neq i\}$, which can then be transmitted to i, who removes S_i, and obtains an unshifted distribution for its delays to the group of other ISPs.

We may approximate distributions in a number of ways. Most simply, given some set of data points $d_{ij}^{(k)\prime}$, we simply create a set a set of bins $[hn, h(n+1))$, and count the number of data points in each bin, i.e., we obtain

$$c_{ij}^{(n)} = \sum_k I\left[d_{ij}^{(k)\prime} \in [hn, h(n+1))\right], \tag{10}$$

where $I(\cdot)$ denotes an indicator function, i.e.,

$$I(A) = \begin{cases} 1, & \text{if } A \text{ is true,} \\ 0, & \text{otherwise.} \end{cases} \tag{11}$$

We then perform a distributed sum to compute

$$c_i^{(n)} = \sum_{j=1, j\neq i}^{N} c_{ij}^{(n)}, \tag{12}$$

which can be used to approximate the density of the delay distribution, and this value is transmitted to ISP i. The value of $c_i^{(n)}$ gives an approximation to the shifted distribution of delays of packets transmitted from ISP i, and so we can recover an approximation of the original delay distribution by reversing the shift by $-S_i$. Once one has a distribution, other statistics are easy to compute.

This approach, while useful, has the problem one often encounters when making empirical measurements of density functions, that the bin size is of some importance. Too small a bin size results in large errors in the probabilities (per bin), whereas too large a bin results in a coarse histogram of the distribution. Moreover, the smaller the bins, the higher the communications cost of computing the sum. An alternative is to use some other approximation technique for the distributions. For instance, one could approximate by a truncated Fourier series, and then sum the Fourier coefficients, and invert to obtain an approximation of the distribution. One could use any type of approximation basis that fits the distribution functions reasonably, as long as it has the property that sums of approximations are equal to the approximation of sums.

There is clearly information leakage in the above approach. Obviously, once one has the shifted distribution, one can compute the variance of the distribution, and so j, can compute $\text{Var}(d_{ij})$. However, neither i nor j can compute the mean of the delay between their networks, and so the critical information we wish to retain as a secret is preserved. The question here is how much information is needed about delays, and how much information do we need to keep secret? Only the participants in such a computation can answer such a question.

3.2 Traffic weighted statistics

An additional set of statistics one might be interested in are traffic-weighted performance metrics. An ISP is perhaps less worried about performance on paths that carry little traffic, and so might wish to give these less weight in an overall traffic performance metric. The simplest traffic-weighted performance metric would be

$$\begin{aligned} \hat{D}_i^{\text{out}} &= \frac{1}{N-1} \sum_{\substack{j=1 \\ j\neq i}}^{N} TM_{ij} \bar{D}_{ij}, \\ &= \frac{1}{N-1} \left[\sum_{\substack{j=1 \\ j\neq i}}^{N} TM_{ij} \bar{R}_{ij} - \sum_{\substack{j=1 \\ j\neq i}}^{N} TM_{ij} \bar{T}_{ij} \right], \end{aligned} \tag{13}$$

where TM denotes the origin-destination traffic matrix between the participating ISPs, i.e., TM_{ij} denotes the traffic volume from network i to j.

If all network operators measured their inter-domain traffic (for instance using flow-level collection on peering links), the multiplications $TM_{ij}\bar{R}_{ij}$ can be performed locally, and then secure distributed summation used to compute the results. However, the case might arise where a single provider wants this metric, and not all other providers can make the flow-level measurements required. In this case, operator i measures TM_{ij} for all j, and so can easily calculate $\sum_{j=1,j\neq i}^{N} TM_{ij}\bar{T}_{ij}$. However, ISP i may not wish to share TM_{ij} with other ISPs, so we need to use PPDM to perform the computation of $\sum_{j=1,j\neq i}^{N} TM_{ij}\bar{R}_{ij}$.

More precisely, we use a PPDM primitive [8] that allows two parties to compute an inner product $\mathbf{a} \cdot \mathbf{b}$ where one party knows a vector $\mathbf{a}$, the other knows $\mathbf{b}$. Obviously the result of this computation could reveal information, so the result is partitioned so that parties A and B learn v_a and v_b respectively, where

$$v_a + v_b = \mathbf{a} \cdot \mathbf{b} = \sum_i a_i b_i. \tag{14}$$

The v_a and v_b should be determined in such a way that no information is revealed about $\mathbf{a}$ and $\mathbf{b}$. For existing algorithms this may not be the case for arbitrary $\mathbf{a}$ and $\mathbf{b}$. We will discuss this further in Section 4 below, but note that for the case of traffic weighted performance metrics there should be no significant problem.

Given a privacy preserving protocol for performing this calculation, we can build a protocol as follows: assume ISP i wishes to compute its outgoing traffic-weighted delay $\hat{D}_i^{\text{out}}$.

(1) for each $j \neq i$, use PPDM to compute

$$v_{ij}^{(a)} + v_{ij}^{(b)} = \sum_{j=1, j\neq i}^{N} TM_{ij}\bar{R}_{ij}$$

where ISP x learns $v_{ij}^{(x)}$.

(2) The ISPs perform a distributed sum

$$V_i^{(b)} = \frac{1}{N-1}\sum_{j=1, j\neq i}^{N} v_{ij}^{(b)}$$

which they transmit to i.

(3) ISP i computes

$$V_i^{(a)} = \frac{1}{N-1}\sum_{j=1, j\neq i}^{N} v_{ij}^{(a)}$$

and thence

$$\hat{D}_i^{\text{out}} = -V_i^{(a)} - V_i^{(b)} + \frac{1}{N-1}\sum_{\substack{j=1\\ j\neq i}}^{N} TM_{ij}\bar{R}_{ij}.$$

Thus the desired results is obtained by i. Step (1) uses the PPDM inner product operation, and so reveals nothing about the inputs to the participants. In step (2), ISP i learns only an aggregated version of the $v_{ij}^{(b)}$, and so cannot separate out the individual terms to determine the value of the individual inner products. Hence at no stage does any other ISP hold enough data to determine sensitive data. A similar approach could be performed for computing $\hat{D}_i^{\text{in}}$.

4. LOSS

Until now, we have assumed that all probe packets are received, however, in the Internet packets are sometimes dropped. This introduces the problem of measuring loss rates, but it also complicates the problem of measuring delays as we must censor the lost packets from our measurement data. We consider both problems below.

4.1 Estimating loss rates

Let us first consider the case where all data is shared. In this case, one can set a *timeout* S beyond which we believe the packet could not survive "in the wild" of the Internet. If a packet is delayed by more than this time, we assume it has been lost. Usually this timeout is several orders of magnitude larger than typical lifetimes, resulting in minimal false positive packet-loss detection. If the total measurement time interval is $T \gg S$ and the number of probes is also large then it is a reasonable approximation to count the number of packets which were transmitted K_{ij} and received M_{ij} in the time-interval and compute to get the loss rate

$$p_{ij} \simeq 1 - M_{ij}/K_{ij}. \tag{15}$$

There is an error in this calculation at the edges, of the order of S/T, resulting from packets which we transmitted, but which don't arrived during the measurement interval $[0, T]$. Likewise, we can compute average loss rates such as

$$p_i^{\text{out}} \simeq 1 - M_i/K_i, \tag{16}$$

where $K_i = \sum_{j\neq i} K_{ij}$, and $M_i = \sum_{j\neq i} M_{ij}$.

It is simple to generalize this approach to PPDM. The counts K_{ij} and M_{ij} are known to ISP i and j respectively. Hence ISP i can directly compute K_i, and M_i can be computed (as in Section 3) using a secure distributed sum, and we can perform similar operations to obtain p_i^{in}.

4.2 Estimating delay in the presence of loss

A much trickier problem is estimating delay in the presence of loss. The above approach tells us how many packets were lost, but not which packets, and so we cannot appropriately censor the data being used to calculate the averaged delay (for received packets).

One approach would be to simply use the shifted timestamps described in Section 3.1 in order to compute the distribution, from which we may calculate the mean. However, as noted above, this approach results in information leakage. We can do better.

We will add a packet ID to each packet, but note that these are not sequential. We choose the packet IDs in some random fashion from the numbers $\{1, 2, \ldots, L\}$, where $L \geq K_{ij}$ for all i and j. The transmitter knows K_{ij}, but the receiver does not (remember we use Poisson sampling intervals for measurements). The receiver only knows L. The receiver can form a series of indicator vectors

$$I_{ij}^{(k)} = \begin{cases} 1, & \text{if the packet with ID } k \text{ from } i \text{ to } j \text{ is received,} \\ 0, & \text{otherwise.} \end{cases}$$

As before, the receivers can use distributed summation to compute the sum of receive times (for packets which are not lost). The receivers may also (as noted above) perform a standard distributed summation to compute $M_i = \sum_{j\neq i} M_{ij} = \sum_{j\neq i}\sum_{k=1}^{L} I_{ij}^{(k)}$. Hence we need only provide a distributed method for computing the sum of transmit times for the packets which are not dropped. We denote this by

$$s_{ij} = \sum_{k=1}^{L} I_{ij}^{(k)} t_{ij}^{(k)}, \tag{17}$$

and we note that the value is not dependent on the value of L, i.e., we can have as many "dummy" probes in the sequence as required. One way to choose L would be maximize the entropy of the indicator functions by assuring that $p\{I_{ij}^{(k)} = 1\} = 0.5$. Given small loss rates, we can do so approximately by taking $L = 2E[K_{ij}]$.

In order to compute this value we exploit methods designed to allow the computation of inner products. Given such a protocol our task is relatively easy: the inner product s_{ij} is divided into two components, $s_{ij}^{(a)}$, and $s_{ij}^{(b)}$ which are learnt by the sender i and the receiver j, respectively. Then the sender performs a simple sum over his values, and the receivers perform a secure-distributed summation over their values, and provide this value to the transmitter i. The transmitter then calculates

$$\bar{D}_i^{\text{out}} = \frac{1}{M_i}\left[\sum_{\substack{j=1\\ j\neq i}}^{N}\sum_{k=1}^{L} I_{ij}^{(k)} r_{ij}^{(k)} - \sum_{\substack{j=1\\ j\neq i}}^{N} s_{ij}^{(a)} - \sum_{\substack{j=1\\ j\neq i}}^{N} s_{ij}^{(b)}\right].$$

There are several existing approaches for computing the inner product (without a trusted third party). However, most of them may result in information leakage in the case where we know that one of the vectors in the inner product is a $\{0, 1\}$ vector (*e.g.*,, [8]). This issue was not a problem when computing traffic-weighted performance measures (as in Section 3.2), but is a problem here. We present here a novel approach to solving the distributed inner product which avoids this pitfall. We use the inner product protocol described in [7], which is based on another primitive from PPDM referred to as 1-in-n Oblivious Transfer (OT) (for original work on 1-in-2 oblivious transfer see [9], and for many relevant papers see [11, 13, 16]). OT is an often used primitive in PPDM, which provides the following operation. Assume B knows n values $b_1, \ldots, b_n$, and A holds a value $\alpha \in \{1, 2, \ldots, n\}$. OT provides a mechanism that allows A to learn b_α, but none of the values of b_n for $n \neq \alpha$, and B does not learn the value of α. Given 1-in-n oblivious transfer the inner product algorithm works as follows.

(1) A and B agree on two numbers m and n
(2) A finds m random vectors $\mathbf{t}_i$ such that

$$\mathbf{a}_1 + \mathbf{a}_2 + ... + \mathbf{a}_m = \mathbf{a}$$

B finds m random numbers $r_1, r_2, \ldots, r_m$.
(3) for $i = 1$ to m
(3a) A sends B n different vectors:

$$\{\mathbf{a}_i^{(1)}, \mathbf{a}_i^{(2)}, ..., \mathbf{a}_i^{(n)}\}$$

where exactly one $\mathbf{a}_i^{(q)} = \mathbf{a}_i$, the other $n-1$ vectors are random
(3b) B computes $\mathbf{a}_i^{(j)} \cdot \mathbf{b} - r_i$
(3c) A uses 1-in-n OT to retrieve

$$v_i = \mathbf{a}_i^{(q)} \cdot \mathbf{b} - r_i = \mathbf{a}_i \cdot \mathbf{b} - r_i.$$

(4) B computes $V_b = \sum_{i=1}^{m} r_i$
(5) A computes

$$V_a = \sum_{i=1}^{m} v_i = \sum_{i=1}^{m} \mathbf{a}_i \cdot \mathbf{b} - r_i = \mathbf{a} \cdot \mathbf{b} - V_b.$$

Then we have $V_a + V_b = \mathbf{a} \cdot \mathbf{b}$. A learns nothing about $\mathbf{b}$ (due to r_i), and B learns something about $\mathbf{a}$, but the probability of correctly guessing q for $i = 1, \ldots, m$ is around $1/n^m$. With sufficiently large n and m, this is close to 0. Even if B has some idea of the properties of $\mathbf{a}$ (e.g., that it is a $\{0, 1\}$ vector), it still does not help, because we split $\mathbf{a}$ into fragments $\mathbf{a}_i$ which loose these properties before performing the computation. We apply this here by taking A to be the transmitter, and B the receiver, with $\mathbf{a}$ being a vector of the transmission times $t_{ij}^{(k)}$ for $k = 1, \ldots, L$, and $\mathbf{a}$ being a vector of the indicator functions $I_{ij}^{(k)}$. As before, once we have performed this operation for each $j \neq i$, a secure distributed summation is used to add up the values of V_b, which can then be passed back to the transmitter i.

One can tradeoff efficiency in this protocol for security. We require $O(m)$ 1-in-n OTs, which are not a cheap operation (in terms of communications cost). Larger values of m and n retain a greater level of secrecy, but result in a larger overhead.

5. CONCLUSION

This paper has shown that there are techniques for solving performance measurement problems that do not require sharing of detailed information. Such techniques have the capability to enable much larger scale Internet performance measurements than are currently available to the research community by removing security and privacy concerns that currently restrict the number of willing participants.

This paper presents a variety of techniques, with different privacy tradeoffs. This work does not stop here, however. There are many variations of the techniques described here, and these may be applicable to other problems such as computing jitter, or other performance metrics. Additionally, it has been shown how to apply such distributed computation to sketches [17], and such may find utility in cases where the volume of measurements becomes very large. There are approaches to apply such methods to time series. For example, Atallah *et al.* [1] show that time-series algorithms (such as detecting a linear trend) can be performed, without revealing intermediate values. In the case of detecting a trend, the algorithm will reveal the slope to all parties, without revealing the absolute values. Likewise, detection of anomalies could prove very useful in improving Internet health.

6. REFERENCES

[1] M. Atallah, M. Bykova, J. Li, K. Frikken, and M. Topkara. Private collaborative forecasting and benchmarking. In *Proc. of the ACM Workshop on Privacy in the Electronic Society (WPES'04)*, Washington, DC, USA, October 2004.

[2] J. Benaloh. Secret sharing homomorphisms: Keeping shares of a secret secret. In *Proc. Advances in Cryptology (CRYPTO '86)*, pages 251–260, 1987.

[3] J. Brickell and V. Shmatikov. Privacy-preserving graph algorithms in the semi-honest model. In *ASIACRYPT, LNCS*, pages 236–252, 2005.

[4] C. Clifton, M. Kantarcioglu, J. Vaidya, X. Lin, and M. Zhu. Tools for privacy preserving data mining. *SIGKDD Explorations*, 4(2), December 2002.

[5] E. Corell, P. Saxholm, and D. Veitch. A user friendly TSC clock. In *Passive and Active Measurement Conference*, Adelaide, Australia, March 2005.

[6] L. Cottrell, W. Matthews, and C. Logg. Tutorial on internet monitoring & PingER at SLAC. `http://www.slac.stanford.edu/comp/net/wan-mon/tutorial.html`.

[7] W. Du and M. J. Atallah. Privacy-preserving cooperative statistical analysis. In *Proc. of the Annual Computer Security Applications Conference (ACSAC '2001)*, 2001.

[8] W. Du, Y. S. Han, and S. Chen. Privacy-preserving multivariate statistical analysis: linear regression and classification. In *Proc. 4th SIAM Int. Conf. on Data Mining*, pages 222–233, Lake Buena Vista, Florida, April 2004.

[9] S. Even, O. Goldreich, and A. Lempel. A randomized protocol for signing contracts. *Commun. ACM*, 28(6):637–647, 1985.

[10] F. Georgatos, F. Gruber, D. Karrenberg, M. Santcroos, H. Uijterwaal, and R. Wilhelm. Providing active measurements as a regular service for ISPs. In *In Proceedings of the Passive and Active Measurements Workshop (PAM)*, Amsterdam, April 2001.

[11] H. Lipmaa. Oblivious transfer or private information retrieval. `http://www.cs.ut.ee/~lipmaa/crypto/link/protocols/oblivious.php`.

[12] S. Machiraju and R. H. Katz. Reconciling cooperation with confidentiality in multi-provider distributed systems. Technical Report UCB/CSD-04-1345, EECS Department, University of California, Berkeley, 2004.

[13] M. Naor and B. Pinkas. Efficient oblivious transfer protocols. In *Proc. of the SIAM Symposium on Discrete Algorithms (SODA '01)*, Washington, DC, USA, January 2001.

[14] V. Paxson, A. Adams, and M. Mathis. Experiences with NIMI. In *in Proceedings of PAM*, 2000.

[15] L. Peterson, T. Anderson, D. Culler, and T. Roscoe. A blueprint for introducing disruptive technology into the internet. In *First Workshop on Hot Topics in Networking (HotNets-I)*, October 2002.

[16] B. Pinkas. Oblivious transfer. `http://www.pinkas.net/ot.html`.

[17] M. Roughan and Y. Zhang. Secure distributed data-mining and its application to large-scale network measurements. *SIGCOMM Comput. Commun. Rev.*, 36(1):7–14, 2006.

[18] V. S. Verykios, E. Bertino, I. N. Fovino, L. P. Provenza, Y. Saygin, and Y. Theodoridis. State-of-the-art in privacy preserving data mining. *SIGMOD Record*, 33(1):50–57, 2004.

[19] R. Wolff. Poisson arrivals see time averages. *Operations Research*, 30:223–231, 1982.

[20] A. Yao. How to generate and exchange secrets. In *Proc. of the 27th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 162–167, 1986.

[21] A. Yao. Protocols for secure computations. In *Proc. of the 23th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 160–164, 1986.