Towards Real-Time Performance Monitoring for Encrypted Traffic

Mehdi Kharrazi, Subhabrata Sen, and Oliver Spatscheck
AT&T Labs-Research, Florham Park, NJ, 07932
{mkharrazi,sen,spatsch}@research.att.com

Abstract

IP networks are increasingly carrying mission-critical applications with robust end-to-end network performance and reliability requirements. Network performance monitoring forms an essential component of critical IP network management functions such as troubleshooting, anomaly detection, and Service-Level-Agreement (SLA) compliance monitoring. However, privacy and security considerations are fueling the use of IP-level encryption techniques such as IPsec, which obscure important transport layer features that existing performance measurement techniques need. New techniques are therefore needed for monitoring performance of encrypted traffic. Towards this goal, in this paper we present a new technique for monitoring round-trip times (RTT) for IP-level encrypted communications. Our approach involves using network-level features like packet size and inter-packet timing to infer specific timing events, and aggregating measurements across short time intervals and related connections to derive final RTT estimates for network paths of interest. Extensive evaluations using traces from an enterprise and a broadband access network, demonstrate that the resulting RTT estimates are quite accurate.

Categories and Subject Descriptors

C.2.0 [**Computer-Communication Networks**]: General—*Security and Protection*; C.2.3 [**Computer-Communication Networks**]: Network Operations—*Network Monitoring*; C.4 [**Computer Systems Organization**]: Performance of Systems—*Measurement Techniques*

General Terms

Measurement, Performance, Security

Keywords

Performance Monitoring, Encrypted Traffic, IPsec Traffic, Traffic Analysis

INM'07, August 27–31, 2007, Kyoto, Japan.

1. INTRODUCTION

Five nines of availability has been an elusive goal for IP network operators for many years. Lately this goal has become even harder to achieve in that IP network operators not only need to ensure network availability but also application availability. This has lead to more sophisticated service level agreements (SLAs) guaranteeing, in addition to basic network uptime, also various network performance metrics such as loss, latency and jitter. In addition, mission critical communication, for which application level SLAs are crucial, is typically also highly sensitive and therefore encrypted. Unfortunately, encryption decreases the availability of information in the traffic path that can be used by network operators to passively monitor these more complex SLAs. In this paper we examine the issues posed and present a technique to passively monitor network path performance for encrypted communications.

In general, performance monitoring can be either *active* where the monitoring system launches probe traffic and makes inferences about performance based on the observed responses, or *passive* where the inferences are based on passive observation of traffic flows in both directions. Some of the challenges active probing faces are that it can only approximate the performance of the real traffic and that the probing traffic itself adds to the network load, in particular if the performance needs to be monitored frequently between a large number of end points. But more importantly, in the presence of multipath routing the active probe may traverse a path that is different from that of the traffic being monitored. The main drawback of passive measurements is that it only measures network paths which carry active traffic. Fortunately, this is typically not a problem in that the performance of network paths which carry no active traffic are rarely of importance.

Passive monitoring could be performed at different locations in the network. In particular the end systems or application servers in an overlay network are in a perfect position to collect performance data passively. Unfortunately for the network operator, these systems are rarely under its control or are rarely willing to provide such information. Therefore, operators rely on in-network monitoring devices to collect performance data. Such devices are commercially available and have also been described in the research literature (e.g Gigascope [4]). These devices typically rely on information obtained from observed TCP transactions to derive the desired performance metrics [12, 16, 9]. The two metrics which are typically derived from TCP and which are of interest to us in this work are loss and latency.

The packet loss properties of a network path are frequently computed by tracking the gaps or repetition in the TCP sequence number space [3, 6, 2]. As for latency measurements, a common approach involves analysing TCP handshakes [1, 10]. These techniques rely on information in the packet stream to identify higher level timing events. However, such information is hidden by IP level encryption techniques used by security protocols like IPsec [11].

This paper investigates the potential for real-time, in-network passive performance monitoring of IP layer encrypted traffic. Our aim is to develop a tool which can assist operators in managing and troubleshooting ISP and Enterprise networks. Such operators are particularly interested in actionable performance events. For example, if the performance changes only for a few seconds or for one TCP connection, operators are generally not in a position to react to the performance change and consequently, reporting such events has little value to the operator. However, network operators do care if customer(s) performance is impacted by loss or high latency for a sustained period of time which would violate application level SLAs and therefore impact application level availability.

We develop a measurement methodology for lightweight online measurement of RTTs for IP-layer encrypted traffic. Given that transport-level timing information is lacking, our technique uses simple and robust network-level features such as packet size and inter-packet timing information to infer specific timing events and estimate performance. We evaluate our techniques assuming IPsec as the example encryption/security protocol, but expect the proposed technique and forthcoming analysis to generalize to other IP layer encryption/security protocols with similar properties. Extensive evaluations using large traffic traces from an enterprise and a broadband access network, demonstrate the feasibility and potential of the approach. For example, for the enterprise data set, for the End-to-End IPSec scenario, more than 90% of the RTT estimates are within 1 msec of the actual values.

To the best of our knowledge, this paper presents the first proposed solution to the IP-level encrypted traffic performance monitoring problem, an important area that has received surprisingly little prior attention. [13] uses certain signal processing techniques for exploring network characteristics in the context of a wireless network consisting of 4 nodes and 3 flows. The authors indicate that the technique does not work when applied to larger networks. In contrast our work focuses on identifying potential events and inferring specific performance measurements in a real network environment. Another related work is [15] which measures the "client-perceived page view response time" in SSL/TLS encrypted traffic. Our work differs substantially from that paper, in that they focus on a single application, and do not address network layer encryption which hides TCP information. In contrast our work is application-agnostic, focusing on basic network characteristics such as RTT, and measures performance based on all TCP traffic. Because of the very nature of our problem, we have to deal with a inherently far noisier, more heterogeneous, and less informative environment.

The remainder of the paper is organized as follows. We discuss the challenges faced by network operators when monitoring links carrying encrypted traffic in Section 2. We present our RTT estimation algorithm in Section 3 and evaluation results in Section 4. Finally, we conclude the paper in Section 5.

2. THE IMPACT OF ENCRYPTION

Network security protocols differ in the amount of information they make unavailable to a network monitor, and traditional performance measurement tools may still be applicable in some cases. For example, application layer encryption such as SSL or TLS [7, 5] preserve network and transport level headers and, therefore, allow the use of traditional tools to estimate RTT, loss, and other traffic characteristics from TCP. However, with IP layer encryption techniques such as IPsec , all headers with the exception of some IP header information are encrypted, and encapsulated with new headers. Therefore, traditional sources of information used for tracking latency and loss such as the TCP flags and other transport-level information are not available. Another challenge is that a single IPsec tunnel can carry different types of traffic (eg., TCP, UDP, ICMP) as well as multiple connections, and for latency and loss estimation it is necessary to identify a string of related packets from the same transport-level connection.

Ironically, measuring loss in IPsec traffic is a simpler task than in TCP data streams. This is due to the fact that every IPsec connection is uniquely identified by a SPI (Security Parameters Index) number, and that each IPsec packet has a unique sequence number which is steadily increasing within the context of a given SPI. Therefore loss can be measured by tracking gaps in these sequence numbers. The only limitation is the fact that any calculated loss, will be for the network path from the IPsec endpoint sending the packet to the monitoring point, which can be resolved by strategically positioning monitors to obtain loss measurements over network paths of interest. We, therefore focus the remainder of this paper on inferencing the latency of encrypted traffic.

Other than the IP level information available at the monitoring point, the location of the monitoring point on the connection path provides important information which can be utilized for performance monitoring. Examples include obtaining an RTT value from the monitoring point to each end point of the connection in addition to the end-to-end RTT measurement, or aggregating performance measurements for the connections sharing the same network path. In addition to the location of a monitoring point, the type of traffic can also impact the measurements taken. In this context, we identify three common scenarios : (i) End-to-End, where the encrypted traffic flows between two end hosts. (ii) Remote Access, in which a host tunnels all it's communication with a remote network through an IPsec gateway. (iii) VPN-to-VPN, where two VPNs with multiple hosts behind them, connect to each other using an IPsec tunnel. We should note that if the IPsec transport mode is employed in the second and third scenarios, then these scenarios will be similar to that of the End-to-End case in terms of measurement complexity. While all three scenarios occur in the Internet today, in this paper we focus primarily on the first case, and provide preliminary results for the second case.

3. RTT ESTIMATION ALGORITHM

Any production network tool needs to be necessarily lightweight and scalable to handle the huge traffic volumes being monitored. Given the IP-level encryption, a monitor

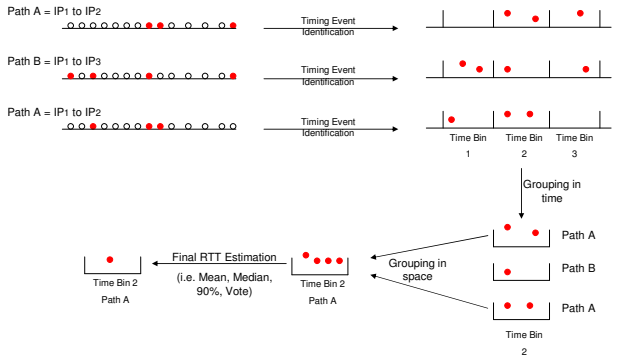

Figure 1: This figure represents the overall process flow for our methodology consisting of (1) Timing Event Identification, (2) Grouping measurements in time, (3) Grouping measurements in space, and (4) Final RTT Estimation which are explained in the text . The red circles indicate timing events of interest, and the white circles represent other events.

can only directly track a few features, such as the time it observed each packet, the size of each packet, as well as some IP address information. Hence, our RTT estimation approach is based on identifying specific packet exchanges using observable network-level features, and inferring RTT measurements from them.

To be suitable for accurate RTT measurements, candidate packet exchanges should involve minimal client- or server-side processing time, as compared to the end-to-end RTT. Also, from a monitoring point perspective, an exchange of at least three packets is required to obtain an RTT estimation between the monitor and each endpoint, as shown in Figure 2. One common network event which exhibits the above properties is the TCP handshake. In addition, TCP handshakes occur every time a connection is initiated. Also since most TCP connections are usually short lived,eg., 22.7 s for WWW, 10.1 s for https, and 89.8 s for ftp data [14], we expect that most time intervals containing network traffic would also contain at least one handshake.

We emphasize that we do not make any assumption as to the relation of the handshake RTT to the RTT of the entire connection. Rather, we treat the measurements obtained from each inferred handshake to represent a sample estimate of the RTT along the corresponding network path at the time of the handshake. As we will discuss later in this section, by grouping multiple such estimates for a given time interval we derive a single overall RTT estimate for a given time interval and path.

As illustrated in Figure 1, the technique for path RTT estimation consists of the following four steps:

1. TCP Handshake Identification

A TCP handshake consists of three SYN, SYN-ACK, ACK packets, transmitted in the stated order. As many TCP connections are user-initiated, we expect some idle time before the connection initiation. Of course there are exceptions to this assumption. For example when a web page is accessed after the initial handshake which starts the connection, a number of other connections could be initiated to obtain embedded objects. Fortunately, as discussed later, for our passive measurement technique to work, we do not need to identify all handshakes. It is sufficient to identify enough handshakes to obtain a reliable measure of the RTT.

Another characteristic of TCP handshake packets is that they carry no data payload and their sizes are limited to a known small range. IPsec encapsulation in general involves adding headers and trailers of predictable size and potential padding for confidentiality. However, in practice, experiences with a large ISP and with multiple enterprise networks indicate that such padding is rarely used. This is due to a number of reasons. First, padding only provides a small improvement in traffic flow confidentiality. Second, there are additional costs in terms of network utilization as well as performance. These costs should not be underestimated as they can increase the bandwidth cost a corporation would have to pay to provide remote VPN access to its employees by two to three folds based on the packet sizes reported in[14], with little additional benefit. Consequently, encrypted TCP handshake packets still have sizes that fall in a narrow predictable range.

Combining the above observations, we use the following steps to identify a TCP handshake:

• *SYN Detection:* Identify a packet to be a potential SYN, if (i) its size is within a given range, and (ii) its appearance is preceded by at least $t_{IdleSYN}$ seconds of idle time during which no packets were observed originating from the client and destined for the server.

• *SYN-ACK Detection:* Given that a potential SYN packet was detected in the opposite direction, a packet going from the server to the client which is preceded by $t_{IdleSYN-ACK}$ seconds of idle time and whose size is within a given range, is identified as a potential SYN-ACK packet.

• *ACK Detection*: Given that a potential SYN and SYN-ACK packet pair were identified over a *link*, then the first packet between the same client-server pair, going in the same direction as the SYN, whose size is in the given range, will be identified as a potential ACK.

The above steps for identifying a TCP handshake, are shown in Figure 2 using the terms *client* to identify the initiator of the TCP connection and *server* for the host being connected to. In addition to the above, we require that there are no other exchanges (in terms of packets with a packet size in the range of interest) between the same client and server pair, during an ongoing potential handshake.

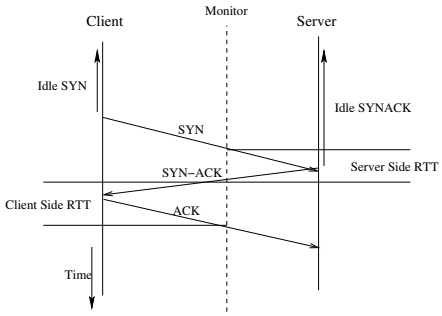

Figure 2: The TCP handshake process.

2. Grouping Measurements in time

In Section 1 we pointed out that performance measurements are of value to network operators when grouped over a period of time. We consider time intervals of width T seconds, and use the above model to identify all handshakes within each such time bin. As a consequence, the probabil-

ity of obtaining more than one measurement in each time bin increases, in turn benefiting the accuracy of our RTT estimates as we discuss later.

3. Grouping Measurements in space

In addition to grouping in time, we also group measurements on a common network path as follows. We first decouple the end-to-end client to server RTT estimation problem to separately estimate the RTT components from the client to monitoring point (or *client*-side) and monitoring point to the server (or *server*-side) (see Figure 2). To estimate the client to measurement point RTT, we consider all communications from that client in a given time bin to all servers, and aggregate across all the individual *client*-side measurements which traverse the same path between the client and the measurement point. Here we have assumed that the network path remains constant in a given time bin. The same approach is used to obtain the *server*-side RTT for a given server and time interval.

4. Path RTT Estimation

By grouping measurements in time and space, we substantially increase the probability of obtaining more than one measurement in a given time bin, for the same network path. To obtain a final representative RTT value for that path and specific time bin, we explore different aggregation techniques: (i) mean, (ii) median, to capture the center of mass, (iii) 90% percentile value, and (iv) a *voting*-scheme based value. The voting approach is designed to remove outliers which could have been caused by either an isolated/transient network delay which is of little interest to the network operator, or an incorrect identification of a TCP handshake. The scheme works by quantizing the obtained RTT values in each time bin with predetermined quantization steps, after which the quantization step with the maximum population is selected as the representative estimate for a given time bin. We use non-uniform quantization steps which are set relative to the values being quantized. In our evaluations we use four (value range, quantization step size) combinations: (i) ((> 10 msec), 10 msec), (ii) ((0.1 msec, 10 msec], 0.1 msec), (iii) ((0.001 msec,0.1 msec], 0.001 msec). Furthermore if the results of the vote are smaller than 0.001 msec or 1 μsec, the vote value is set to 0.1 μsec.

4. EXPERIMENTAL EVALUATION

In the following section, we describe the dataset and experimental parameters we used in our evaluation, and present the obtained results.

4.1 Dataset

Obtaining a dataset with which we can directly evaluate our technique is a difficult task, as we require access to both the encrypted as well as clear traffic in order to validate our approach. We collected IP packet traces at the Internet access link of an enterprise network located behind a firewall, from April 17th to 25th, 2006. We then simulated IPsec encryption by removing the encrypted fields from the data set.

We classify IP addresses as either local or remote where a local IP address represents an IP address within the access network being monitored (i.e. client). All other IP addresses are categorized as remote (i.e. server). All local IP addresses which had some traffic destined towards them but themselves did not originate any traffic were eliminated. This phenomenon is mainly caused by network scans. Furthermore, for our study, we focus on remote IPs which were either visited by multiple local IPs or were visited over an extended period of time. In particular we only consider a remote IP if the sum of the number of local IPs communicating with it and the number of time bins in which at least one local IP communicates with it exceeds 50. After the filtering step, our *Enterprise* dataset consisted of 1 billion packets, 97.72%, 1.97%, and 0.16% of which are TCP, UDP, and ICMP packets respectively, and contains 2.5 million TCP connections initiated from local to remote IPs. There were a total of 1012 local, and 2011 remote IP addresses.

4.2 Experimental Parameters

Our algorithm requires a few parameters. First is the *Size of handshake packets*. These packets do not carry any payload, and only consist of the TCP and IP headers. These headers each range from 20 to 60 bytes according to the TCP/IP standards. Thus we assume a minimum, and maximum size of 40 and 120 bytes respectively for handshake packets [1]. The second parameter is the *Time bin length*. This value is greatly dependent on the human actionable time and the particular network management application. We selected a bin size of 300 seconds which is commonly used by network operators for many network management tools such as SNMP pollers.

The third, and last parameter is the *Idle time threshold*. We found that $t_{IdleSYN}$ and $t_{IdleSYN-ACK}$ are highly correlated, with a correlation coefficient of 97.24%. Therefore for the remainder of this work we have set $t_{IdleSYN} = t_{IdleSYN-ACK}$ and identify them collectively as the idle time. We believe this will generalize to other datasets, since the SYN and SYN-ACK represent the initial packets in every connection, and their presence is highly correlated. We have experimented with a range of idle time thresholds and have found two values of 1 and 50 seconds to represent two reasonable choices which strike a balance between the number of SYN and SYN-ACK packets identified, and the identification accuracy. We will discuss this tradeoff later.

4.3 End-to-End Scenario

We evaluate our handshake-based RTT estimation methodology along a number of axes, namely (i) How available are the handshakes, (ii) how well do we detect them (iii) how accurate are the RTT estimates.

4.3.1 Availability of Handshakes

Intuitively, we want to track the RTT for an IP in time bins where that IP has some minimum level of activity. We adopt an inclusive notion of activity and define an *active* time bin for a given IP to be one which has at least one packet destined to and at least one packet sourced from that IP. We find that more than 50% of active time bins carry one or more handshakes for 77% and 87% of local and remote IPs respectively. We also find that for 75% of the IPs (local or remote), for 90% of the time, for every 2 active time bins we get at least one time bin with handshakes. Overall the results indicate that the handshake based approach can provide frequent RTT measurements well spread over active time bins for most IPs.

[1]If we were to monitor actual IPsec traffic, the range had to consider the IPsec encapsulation overheads.

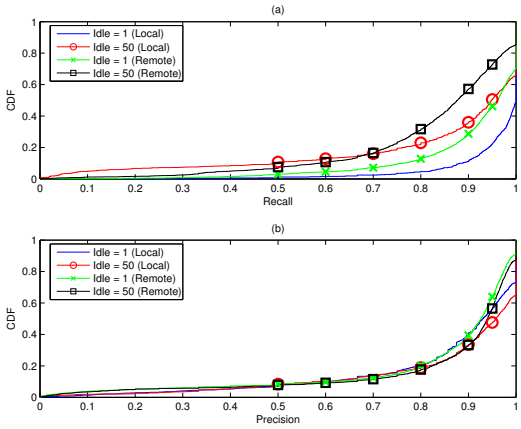

Figure 3: CDF of precision values in identifying time bins which carry handshakes.

4.3.2 *Identification of handshakes*

We next explore the effectiveness of our algorithm in identifying time bins which have at least one handshake. We utilize two widely accepted metrics of accuracy, Recall and Precision. In the context of our specific problem, they are computed as follows: *Recall* is the ratio of the number of time bins, for which handshakes are predicted, to the total number of time bins which actually had handshakes. *Precision* is the number of time bins for which handshakes were predicted correctly, expressed as a fraction of the total number of time bins predicted to contain handshakes.

Figures 3(a)-(b) depict the distribution of these Recall and Precision values across the different local and remote IPs. The results show that our algorithm is able to identify time bins which carry handshakes with high accuracy. For example, using an idle time value equal to 1 second, 80% of local IPs have recall above 94% and precision above 80%. As expected, the *Idle* time value plays an important role in our algorithm, and although a larger *Idle* time leads to higher precision values (i.e. more accurate identification of time bins with handshakes), at the same it also reduces the recall (i.e. the overall number of time bins identified to have handshakes).

Note that it is not necessary to be able to identify every individual handshakes for our technique to accurately estimate the final RTT. Recall that we obtain multiple RTT estimates in each time bin for a given network path, as a result of grouping measurements in space and time. A final and representative RTT value for that time bin and network path is obtained by aggregating over the available RTT estimates. The accuracy of the final estimated value depends both on the distribution of the estimated RTT values from individual handshakes in each time bin, and on the aggregation technique employed. Therefore, although identifying all available handshakes would result in an accurate final estimate, identifying a subset of available handshakes could still lead to accurate final RTT estimates, as we shall see next.

4.3.3 *Accuracy of RTT Estimation*

We next explore the accuracy of the *estimated* representative RTT value estimated in a time bin for each network path. We compare this value to the *actual* representative RTT for that time bin. The latter obtained by aggregating the RTTs of the actual handshakes in that time bin for the same path, for a given aggregation technique. Such a comparison can only be done with time bins for which both actual and identified handshakes are available. We compute the per-time-bin estimation error as the difference (*estimated* RTT - *actual* RTT).

Performance differences can typically be measured in relative (e.g. the round trip time increased by 50%) or absolute terms, (e.g. the round trip time increased by 10 msec). In practice, operators are often more interested in the absolute change then the relative one. For example, if the round trip time changes 500% from 10 msec to 50 msec there is basically no impact on the voice quality of a VoIP call, which we should note is one of the most time sensitive applications on the Internet today. On the other hand if the round trip time changes by a factor of 200% from 150 msec to 300 msec voice quality is severely impacted [8]. Therefore, in our evaluations, we focus on the absolute and not the relative estimation error.

Figure 4 plots the distribution of the RTT estimation errors, accumulated across all IPs. The graphs indicate that the error is low for the vast majority of the estimates, across the different aggregation techniques. Under the voting aggregation, more than 95.20% and 90% of the estimated RTT measurements are accurate to within 1 msec of the actual values for the local and remote IPs respectively. Furthermore, the error is within 10 msec for 96.59% and 95.87% in the local and remote IP cases respectively. The results also indicate that the choice of aggregation technique can significantly impact the RTT estimation accuracy. In particular voting consistently outperforms the other studied aggregation techniques.

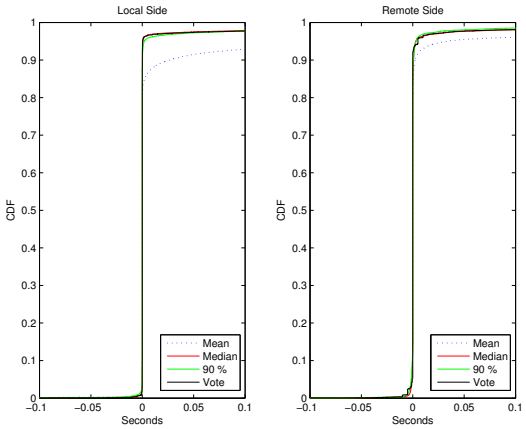

Figure 4: CDF of difference in actual and estimated RTTs for each time bin.

While the overall accuracy is high, the next question is: are time bins with low errors limited to a few IPs, or they are spread among different IPs? We consider the 90^{th} percentile (across all time bins) of the absolute error values (|estimated RTT - actual RTT|) per IP, based on the voting aggregation. The obtained results are quite encouraging. For the vast majority of the local (or remote) IPs, the error is very low. For eg., about 90% of the local IPs have 90% of their estimations accurate to within 10 msec.

4.3.4 *Alternate Data Set*

In order to investigate the effectiveness of our algorithm for other network environments, we employed a second dataset

which consists of packet traces obtained from a cable head-end located in a Tier-1 ISP. The data was collected from August 5th through 22nd, 2004. After the data processing step (described in Sec 4.1), our dataset consisted of 2.5 billion packets, 87.55%, 10.29%, and 0.35% of which are TCP, UDP, and ICMP packets, respectively. Furthermore it contains 20 million TCP connections initiated from the local to remote IPs, and 512 local, and 9876 remote IP addresses. For this set, using voting aggregation, 81.27% and 94.75% of the estimated RTT measurements are accurate to within 1 msec of the actual values for the local and remote IPs, respectively. Furthermore, the error is within 10 msec for 90.50% and 96.46% in the local and remote IP cases, respectively.

4.4 Remote Access Scenario

We next present initial results from exploring the Remote Access scenario described in Section 2. For this case, recall that only the gateway's IP is visible to the monitoring point. Therefore, the monitor has no easy way to distinguish, based on IP headers, between packets from different remote IPs located behind the same gateway. For the End-to-End case, the handshake identification algorithm has to deal with the possibility of incorrectly grouping together packets from different connections between a single local and a single remote IP. In contrast, in the Remote Access the algorithm has the task of identifying a handshake from a mixture of packets being exchanged between the local IP and potentially multiple remote IPs behind the same remote gateway. This makes the handshake identification and RTT estimation task harder.

Furthermore, as the VPN gateways represent multiple real servers behind them which most likely will have different latencies, we are only able to estimate the RTT on the local side (local IP to measurement point) in this scenario. Fortunately a network operator still has substantial control over which paths of his network are being monitored by placing the probe either close to the local IP or close to the VPN gateway.

Using the Enterprise dataset and the voting aggregation, we observe that more than 84.38% and 87.26% of the estimated RTT measurements are accurate to within 1 msec and 10 msec respectively, of the actual values for the local IPs. Furthermore, 61% of the local IPs have 90% of their estimations accurate to within 10 msec. These initial results are encouraging, and we are currently exploring ways to increase the RTT estimation accuracy in the *Remote Access* scenario.

5. CONCLUSIONS AND FUTURE WORK

In this paper, we investigated the challenging problem of monitoring performance of encrypted traffic, and developed a methodology for estimating RTTs along specific network paths. Simulations using traces obtained from two networks with different characteristics illustrate the effectiveness of our approach, and suggest the potential of harvesting network-level features. For example, for the enterprise data set, for the End-to-End IPsec deployment scenario, more than 95.20% and 90% of the estimated RTT measurements are accurate to within 1 msec of the actual values for the local and remote IPs respectively. The above technique has been deployed in a real network.

The work in this paper takes a first step towards solving the general problem of real-time performance monitoring of IP encrypted traffic. More research is needed in the community for developing a complete suit of encrypted traffic performance monitoring techniques such as what exists today for unencrypted traffic. We are working on improving the robustness and accuracy of the technique for the more complex network settings such as the *Remote Access* and *VPN-to-VPN* scenarios.

6. ACKNOWLEDGMENTS

We thank Jacobus Van Der Merwe and Seungjoon Lee for their helpful discussions on an earlier version of the paper, and the anonymous reviewers, whose suggestions benefited this paper.

7. REFERENCES

[1] J. Aikat, J. Kaur, F. Donelson Smith, and K. Jeffay. Variability in TCP round-trip times. *Internet Measurement Conference*, 2003.

[2] M. Allman, W. M. Eddy, and S. Ostermann. Estimating loss rates with TCP. *ACM Performance Evaluation Review, 31 (3)*, 2003.

[3] P. Benko and A. Veres. A passive method for estimating end-to-end TCP packet loss. *IEEE Globecom*, 2002.

[4] C. Cranor, T. Johnson, and O. Spatscheck. Gigascope: a stream database for network applications. *SIGMOD*, 2003.

[5] T. Dierks and C. Allen. The TLS protocol version 1.0. *RFC 2246*, January 1999.

[6] N. Fonseca and M. Crovella. Bayesian packet loss detection for TCP. *Infocom*, 2005.

[7] A. O. Freier, P. Karlton, and P. C. Kocher. The SSL protocol version 3.0. *http://home.netscape.com/eng/ssl3/draft302.txt*, November 1996.

[8] I.-T. R. G.107. The e-model, a computational model for use in transmission planning. *http://www.itu.int/*.

[9] S. Jaiswal, G. Iannaccone, C. Diot, J. Kurose, and D. Towsley. Formal analyis of passive measurements based inference techniques. *Infocom*, 2006.

[10] H. Jiang and C. Dovrolis. Passive estimation of TCP round-trip times. *Computer Communications Review*, 2002.

[11] S. Kent and R. Atkinson. Security architecture for the Internet protocol. *RFC 2401*, November 1998.

[12] J. Padhye and S. Floyd. On inferring TCP behavior. *ACM SIGCOMM*, 2001.

[13] C. Partridge, D. Cousins, A. W. Jackson, R. Krishnan, T. Saxena, and W. T. Strayer. Using signal processing to analyze wireless data traffic. *WiSE '02: Proceedings of the 3rd ACM workshop on Wireless security*, 2002.

[14] M. Roughan, S. Sen, O. Spatscheck, and N. Duffield. Class-of-service mapping for QoS: A statistical signature-based approach to IP traffic classification. *Internet Measurement Conference*, 2004.

[15] J. Wei and C.-Z. Xu. smonitor: A non-intrusive client-perceived end-to-end performance monitor for secured Internet services. *USENIX*, 2006.

[16] Y. Zhang, L. Breslau, V. Paxson, and S. Shenker. On the characteristics and origins of Internet flow rates. *ACM SIGCOMM*, 2002.