End-to-End Detection of Compression of Traffic Flows by Intermediaries

Vahab Pournaghshband
Computer Science Department
University of California, Los Angeles
vahab@cs.ucla.edu

Alexander Afanasyev
Computer Science Department
University of California, Los Angeles
afanasev@cs.ucla.edu

Peter Reiher
Computer Science Department
University of California, Los Angeles
reiher@cs.ucla.edu

***Abstract*—Routers or nodes on the Internet sometimes apply link-layer or IP-level compression on traffic flows with no knowledge of the end-hosts. If the end-host applications are aware of the compression already provided by an intermediary, they can save time and resources by not applying compression themselves. The benefits from these savings are even greater in mobile applications.**

We present a probing technique to detect the compression of traffic flows by intermediaries. Our technique is non-intrusive and robust to cross traffic. It is entirely end-to-end, requiring neither changes to nor information from intermediate nodes. We present two different but similar approaches based on how cooperative the end-hosts are. Our proposed technique only uses packet inter-arrival times for detection. It does not require synchronized clocks at the sender and receiver. Simulations and Internet experiments were used to evaluate our approach. Our findings demonstrate an accurate detection of compression applied to traffic flows by intermediaries.

I. Introduction

On the Internet, every packet sent goes through numerous routers or intermediaries until it gets to the intended receiver. While routing the traffic, these intermediaries, are potentially capable of making serious changes to what happens to a traffic stream on the network. One class of intermediaries makes no changes to the content of the traffic, giving the appearance that nothing has been done to the stream other than routing it to the destination. This transparency property may make end-to-end detection of such intermediaries harder in most cases. The class of such intermediaries is very broad (e.g., performance enhancing proxies, VPN gateways, Internet censors, and network dissuasion [29]), and some intermediaries have been deployed worldwide for decades. Investigating the detectability of such intermediaries leads to two questions: (i) can the sender or receiver (or both if they cooperate) determine that something of this kind has been done if they pay attention, and/or (ii) is it possible for such an intermediary to work by stealth and remain undetected? Another example of these intermediaries is that of network compression which happens at intermediate nodes, rather transparently to the end-hosts. As an example of determining the detectability of third party influences of this kind, in this paper we investigate the feasibility of detecting network compression on the path.

One way to increase network throughput is to compress data that is being transmitted. Network compression may happen at different network layers and in different forms:

Application layer. Compression at the application layer is widely used, specially for applications that use highly compressible data such as VoIP and video streaming. At the application layer, both compression and decompression happen at the end-hosts.

TCP/IP header. Often header compression is possible when there is significant redundancy between header fields; within the headers of a single packet, but in particular between consecutive packets belonging to the same flow. This is mainly achieved by first sending header field information that is expected to remain static for most of the lifetime of the packet flow. Since these methods are used to avoid sending unchanged header fields for a network flow, no data compression algorithm is actually applied here [6].

Early TCP/IP header compressions such as CTP [17] and IPHC [11] were designed for slow serial links of 32 kbps or less to produce a significant performance impact [8]. More recent header compressions have since been developed, such as ROHC [19].

IP payload. IPComp [33] is the de facto method in this case. LZS [12], Deflate [27], and ITUT v.44 [5] are the well-known compression algorithms that work with IPComp. IPComp is generally used with IPsec. IP payload compression is something of a niche optimization. It is necessary because IP-level security converts IP payloads to random bitstreams, defeating commonly deployed link-layer compression mechanisms that are faced with payloads which have no redundant information that can be more compactly represented. However, many IP payloads are already compressed (images, audio, video, or zipped files being FTPed), or are already encrypted above the IP layer (e.g., SSL). These payloads will typically not compress further, limiting the benefit of this optimization. In general, application-level compression can often outperform IPComp because of the opportunity to use compression dictionaries based on knowledge of the specific data being compressed. This makes the mechanism less useful, and hence reduces the need for IPComp [9].

Link layer. Commonly used link compression schemes include the Stacker [13] and the Predictor [31][8]. The

Stacker compression algorithm is based on the Lempel-Ziv [12] compression algorithm. The Predictor compression scheme works by predicting the next sequence of characters in a data stream using an index to look up a sequence in a compression dictionary. There is no information on how widely link-layer compression is used in practice.

Except for application-layer compression, compression happens at intermediate nodes, often without the knowledge of end-users. For example, in January 2013, a researcher discovered that Nokia had been applying compression to its users' data without their knowledge [3]. In this case, the intermediary was surreptitiously decrypting and re-encrypting user packets in order to effectively apply compression. Users surely would have preferred to know that this was happening, both because of the security risk and because it would render their own application-level compression unnecessary.

However, performing compression and decompression requires many resources at the intermediate nodes, and the resulting overhead can overload the intermediary's queue, causing delay and packet losses. Further, not all commercial routers come with compression capabilities [16]. Thus, some intermediaries apply compression, some do not, and generally they do not tell end-users whether they do. While managing resources effectively at end-hosts is not as crucial as it is at routers, it is still beneficial—particularly for mobile devices where resources are limited. Wasting these resources on redundant compression is undesirable. End-hosts can benefit from recognizing when compression has already being applied on a network connection.

Ideally, end-hosts and intermediaries should coordinate their compression decision, but practical problems make that ideal unlikely. Therefore, since the end-hosts have the greatest interest in proper compression choices for their data, they could detect if intermediate compression is present and adjust their behavior accordingly. An end-to-end approach to detect compression by intermediaries can help to save end-hosts' resources by not compressing data when intermediaries are already doing so.

This paper describes a method to allow end-to-end detection of intermediary-provided link-layer or IP-level compression. Such an end-to-end approach does not require any changes to or cooperation from intermediary nodes, making its deployment and use much more practical. We propose two end-to-end approaches based on a receiver's cooperativeness in the detection process. A *cooperative receiver* is willing to make necessary changes on its machine or system to fully cooperate with the sender in the detection process. A *responsive receiver* only responds to the sender's requests as long as they do not require any changes on the receiver's machine. For example, our approach assumes that the receiver responds to the sender's ICMP requests. This paper deals with a single-sender/single-receiver path of a communication network. Our approach is resilient to cross traffic and other Internet variabilities, and is non-intrusive, making it both practical and deployable. Also, our proposed solution uses only regular unicast probes, and thus it is applicable in today's Internet.

Simulation and Internet experiments show that our approach works. Our approach detects both software and hardware compression. We use only the relative delays between arrival times of our probing packets for detection, so clock synchronization is unnecessary. The approach requires no special network support. However, it is not designed to detect compressions that are not based on entropy of data, such as dictionary-based or TCP/IP packet header compression.

While IPComp is not widely used, there is no evidence related to how commonly link-layer compression is deployed. Knowing this is essential before performing further research in this direction, which suggests an investigation of the prevalence of link-layer compression as a next step. We plan to use our findings here in a longitudinal study of the prevalence of this type of compression in the Internet.

End-to-end detection of compression by intermediaries is also valuable for bandwidth availability and capacity estimation. Bandwidth availability and path capacity are among the most important characteristics of Internet paths. IP-level and link-layer compression directly influence the estimation of these path properties, since compression has a considerable effect on the assessment of both capacity and bandwidth. Not taking such effects into consideration will lead to bandwidth or capacity under- or overestimation.

While investigating the detectability of link-layer compression is valuable by itself, we believe this work is the beginning of a much broader area of research — that is, exploring the detectability of the class of intermediaries that influences traffic but leaves the packet payloads within the traffic stream unchanged. This area, in turn, feeds into the highly important research question of what, in general, can an end user know about what happens to the packets he submits to the Internet.

The remainder of the paper is organized as follows: Section II presents related work, followed by detection methodology in Section III. Implementation, simulations, Internet evaluation, and discussion are presented in Sections IV, V, VI, and VII respectively. Section VIII concludes the paper.

II. Related Work

While the problem of detecting compression has not been addressed in the literature in the past, the detection of the presence of redundancy elimination (RE) on bandwidth-constrained links has. RE-enabled routers identify, and then, remove redundant packets (i.e., multiple copies of the same packet) [4]. Han et al. [15] briefly outline an approach to detecting RE-enabled routers on the path and leave elaboration and implementation to future work. The detection of compression, however, is different from detecting RE-enabled routers on the path, since the two third parties are looking to reduce (or ideally eliminate) different kinds of redundancies.

Because of the nature of network compression effects on the available bandwidth and the fact that our approach is inspired by the algorithms used to estimate bandwidth, in this section we present end-to-end techniques and tools for measurements of the available bandwidth and capacity of a network path. The

problem of bandwidth estimation has been extensively studied in the past. Many approaches are designed for cooperative end-hosts, and some are designed to work with responsive hosts.

End-to-end active probing schemes for bandwidth estimation are classified into three categories [30]: Packet Pair/Train Dispersion (PPTD), Self-Loading Periodic Streams (SLoPS), and Trains of Packet Pairs (TOPP). In this section we briefly describe each of these techniques.

A. Packet Pair/Train Dispersion (PPTD)

The packet-pair technique was first introduced by Keshav [20]. In this technique, the source sends multiple pairs of packets to the receiver. Each packet pair consists of two packets of the same size sent back-to-back. Then, the dispersion of a packet pair is used to measure the capacity of the path. The dispersion of a packet pair, δ_i, at a particular link of the path, is defined as the time distance between the last bit of each packet. With the assumption of no cross traffic, δ_i is:

$$\delta_i = \max\left(\delta_{i^*}, \frac{L}{C_i}\right) \tag{1}$$

where δ_{i^*} is the the dispersion prior to the link C_i, L is the packet size, and $\delta_0 = L/C_0$.

Measuring the dispersion at the receiver, δ_R, is what is used to estimate the path capacity, C (H is the number of hops between the end-hosts):

$$\delta_R = \max_{0\le i\le H}\left(\frac{L}{C_i}\right) = \frac{L}{\min_{0\le i\le H}(C_i)} = \frac{L}{C} \tag{2}$$

$$C = \frac{L}{\delta_R}. \tag{3}$$

Jain et al. [18] extended the packet-pair probing technique to packet trains, where more than two packets are sent back-to-back. The dispersion of a packet train at a link is defined as the time between the last bits of the first and last packets in the train.

PPTD probing techniques typically require cooperative end-hosts. It is, however, possible to perform PPTD measurements with only a responsive receiver. In that case, the receiver is expected to, for instance, respond to ICMP messages. However, the reverse path capacities and cross traffic may affect the results.

B. Self-Loading Periodic Streams (SLoPS)

SLoPS is a another methodology for measuring end-to-end available bandwidth. In this technique, the sender periodically sends a number of equal-sized packets to the receiver at a certain rate. This measurement methodology involves monitoring the arrival time variations of the probing packets. If the sending rate is greater than the path's available bandwidth, it will overload the queue of the bottleneck, which results in an increasing trend of one-way delay.

On the other hand, if the stream rate is lower than the available bandwidth, the probing packets will go through the path without overloading the queue: thus we do not expect to see an increasing trend. In this approach, the sender, through iterations, attempts to adjust the sending rate to get close to the available bandwidth.

C. Trains of Packet Pairs (TOPP)

Unlike the Self-Loading Periodic Streams technique that measures the end-to-end one-way delays of a packet train arriving at the receiver, TOPP [25] increases the sending rate (R_S) until a point where the sender is sending faster than the path capacity (C). The receiver cannot receive faster than the available bandwidth at the bottleneck ($R_R < R_S$), so further increasing the sending rate above the available capacity means that the packets will get queued at the intermediate routers. As long as the sender is still sending within the path capacity, the receiving rate is not more than the available capacity. Thus, the ratio of sending rate to receiving rate is close to unity (i.e., $R_R = R_S$). Hence, TOPP estimates the available bandwidth to be the maximum sending rate such that $R_S \approx R_R$. The following equation is used to estimate the capacity C from the slope of R_S/R_R versus R_S:

$$\frac{R_S}{R_R} = \frac{R_S}{R_S + R_C}C \tag{4}$$

where R_C is the average cross traffic rate. TOPP is quite similar to SLoPS. In fact, most of the differences between the two methods are related to the statistical processing of the measurements.

Table I summarizes some of the publicly available bandwidth estimation tools and the methodology used in their underlying estimation algorithm.

TABLE I
END-TO-END BANDWIDTH ESTIMATION TOOLS

Tool	Measurement Metric	Methodology
bing	Path capacity	PPTD
bprobe	Path capacity	PPTD
nettimer	Path capacity	PPTD
pathrate	Path capacity	PPTD
sprobe	Path capacity	TOPP
cprobe	Available bandwdith	PPTD
pipechar	Available bandwdith	PPTD
pathload	Available bandwdith	SLoPS
IGI	Available bandwdith	SLoPS
pathchirp	Available bandwdith	SLoPS

III. Detection Methodology

A. Assumptions

In our approach to detecting compression on a path, we assumed that the network consists of a series of store-and-forward nodes; each of them is equipped with a FIFO queue and has a constant service rate. We also assume that the packet delay results from propagation delay, service time and variable queuing delay. Lastly, our approach is based on the assumption that, in the absence of compression, packets of the

same size are not treated differently based on the entropy of their payload.

B. Approach Overview

To detect if compression is provided on the network we exploit the unique effects of compression on network flows. Assuming the original packets were of the same size, compressed low entropy data packets are expected to be considerably smaller than compressed packets containing high entropy data, which in turn leads to a shorter transmission delay. The added processing delay (d_C) caused by compression/decompression methods for low entropy packets is not greater than the high entropy packets ($d_{C_L} \leq d_{C_H}$ where L: low entropy, H: high entropy) [32]. Based on these facts, the sketch of our approach is as follows:

Send a train of fixed-size packets back-to-back with payloads consisting of only low entropy data. Then send a similar train of packets, except these payloads contain high entropy data instead. We then measure the arrival times of the first and the last packet in the train, independently for low entropy (t_{L_1} and t_{L_N}, where N is the number of packets in a single train) and high entropy (t_{H_1} and t_{H_N}) packet trains. Since the number of packets in the two trains is known and all of the packets have the same uncompressed size, the following inequality will hold if some kind of a network compression is performed on the path:

$$\Delta t_L = t_{L_N} - t_{L_1} < \Delta t_H = t_{H_N} - t_{H_1} \tag{5}$$

The inequality (5) suggests that the total set of highly compressible low entropy packets gets to the destination faster than the set of less compressible high entropy packets. Conversely, if the packets are not being compressed by any intermediary, then the two sides of the inequality (5) should be almost equal. This suggests that a threshold should be specified to distinguish effects of compression from normal Internet variabilities:

$$\Delta t_H - \Delta t_L > \tau \tag{6}$$

The underlying rationale behind this approach is that because of the presence of compression and decompression, the receiving party should sense a relatively higher bandwidth when the train of low entropy data is sent, since the same amount of data is received, but in shorter time.

IV. Implementation

We used UDP packets to generate our probe train. When the receiver is cooperative, with the help of two TCP connections before and after the probe UDP train, the sender is able to send experiment parameters to the receiver, and the receiver uses the second TCP connection to send the recorded arrival times of the received packets to the sender for further analysis.

If the receiver does not cooperate, but is responsive, we attached an ICMP ping request packet to the head and tail of the UDP probing train. In this way, the sender performs the detection by analyzing the difference between the arrival time of the two ICMP ping reply packets, without relying on the receiver to provide any measurement information.

Our techniques are not designed to handle receivers who are neither cooperative nor responsive.

A. Content of packet's payload

The payloads of the low entropy packets are filled with 0's. The payloads of the high entropy packets are filled with random bytes read from `/dev/random`, independently for each new experiment.

B. Packet size

Dovrolis et al. [10] argued that a maximum transmission unit (MTU) is not optimal for accurate bandwidth estimation. However, we used large packet size probes (1100 bytes) since the larger the packets, the more apparent are the effects of compression, which in turn leads to a more accurate detection. We emphasize that while our technique is based on measuring the bandwidth, for detecting compression we do not need to estimate the bandwidth *accurately*.

C. Inter-packet departure spacing

In our experiments we use an empirically set value of 100 μ-sec. In general, this number should not be too small to ensure that our experiment does not result in queuing overflows in the intermediate routers. Conversely, this value should not be too large, since sending the packets at a slow rate removes the aggregate effects of compression on the traffic flow.

D. Number of measurements

The presence of cross traffic, on average, differs at certain times of the day and follows a particular pattern [34]. By performing measurements throughout the day, we hope to capture the effects of time-dependent cross traffic variation. We ran each scenario at every hour throughout an entire day, resulting in a total of 24 measurements.

E. Threshold

The selection of this value highly depends on the time precision and resolution of the machine performing the measurement time analysis. If the time precision on a typical machine with a typical operating system is 10 ms [26], then we believe any value at least an order of magnitude higher (e.g. 100 ms) is a suitable selection for this parameter.

F. Number of packets

A careful selection of this number is vital, since a small train may not be sufficient to introduce a noticeable gap of aggregate compression, and on the other hand, a large number of packets would make our approach highly intrusive. In the next section, we show, through simulations, that this number is positively correlated with the available bandwidth. In our experiments we used 6000 packets for each train. Section VI.B shows that this number is sufficient without being intrusive (Section VII.A).

V. Simulations

We simulated our approach using the `ns-3` simulator [2] under different network scenarios using the simple topology depicted in Fig. 1.

Fig. 1. Topology used for simulation

We used simulation to investigate how network characteristics can influence our detection rate. Each of the following subsections refer to a particular scenario. Within each subsection, we first describe the scenario setup, then finally present the obtained results.

A. Scenario I

In the first scenario we tested our approach using the topology in Fig. 1 when the capacity of all three links are equal (Fig. 2). As these results show, we can detect compression on a 100 Kbps or 5 Mbps link with a 100 msec threshold and a small to moderate number of packets.

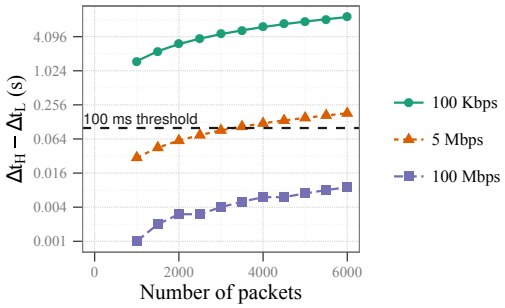

Fig. 2. Comparison of number of probe packets required for detection for different bandwidths (in log-scale).

While the results verify our proposed approach, they show that in our detection mechanism, as the path capacity gets higher, more probe packets are required for the detection of compression effects. With 100 Mbps links, we cannot detect at a threshold of 100 msec even with 6000 packets. The trend of the curve suggests we would need several thousand more packets. This observation suggests that both the path capacity and the available bandwidth of the path in question have direct effects on the number of packets required for each measurement. It also suggests that a relative orders-of-magnitude larger number of probe packets is required for detection in high speed networks (e.g., 1Gbps). However, this is not a major disadvantage for our approach, since compression on high-speed networks is rarely deployed because the hardware required for compression for high-speed networks is expensive. In addition, the typical hardware compression components have proved to be unable to compress/decompress fast enough for a high-speed network, thus creating a bottleneck on that link.

B. Scenario II

In this scenario we examined the relationship between how effective the deployed compression is and the detection rate of our detection scheme.[1] To carry out this simulation, we set the link capacity of the first and last links to a fixed rate of 5Mbps. We then ran the simulation on numerous values for the link capacity of the compression link, $C = \{1, 4, 5, 6\}$ (Mbps).

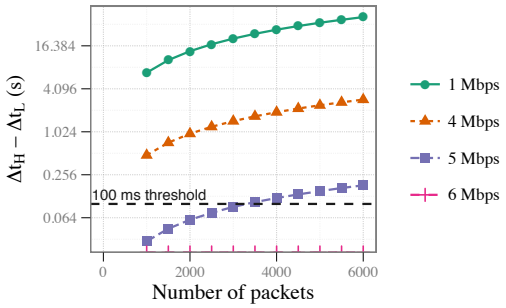

Fig. 3. Comparison of scenarios based on how narrow the compression bottleneck link is (in log-scale).

As the results show (Fig. 3), our detection works well only when the compression link is indeed the bottleneck of the path. There is a correlation between how narrow the compression link is compared to the overall path capacity, and the number of probe packets needed to detect compression. The narrower the link, the smaller the number of the probe packets needed for detection. In fact, sometimes an order-of-magnitude fewer packets are needed, while maintaining approximately the same detection accuracy.

This stems from the fact that the effects of compression are more significant as the bottleneck becomes very narrow. On the other hand, compression is ineffective when it is not on the bottleneck. The result is that our detection method becomes rather ineffective (6 Mbps case in Fig. 3). However, we know that employing compression when it is not placed on the bottleneck link is a poor practice. For this reason, we expect that similar scenarios are relatively rare to find in practice, and hence this makes handling such scenarios less important. To summarize, simulation has confirmed that our approach detects only *effective* compression.

VI. Internet Evaluation

In this section, we present our Internet evaluation to confirm that our method works on a real network. Here, we begin with the experiment setup, and follow with a demonstration of the results and an analysis.

A. Experiment Setup

To simulate the effects of link-layer, we used the Click Modular Software Router [21]. The experiment environment and topology setup is depicted in Fig. 4. The compression and decompression components, as well as the receiver, were all

[1]Compression efficiency is determined by how much the packet size is reduced by applying the compression algorithm [19].

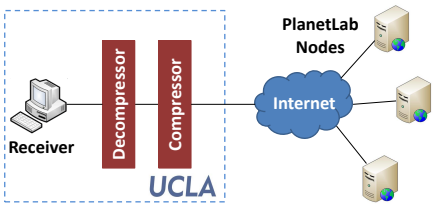

Fig. 4. Environment used in our experiments

located in UCLA. The senders are, however, remote PlanetLab [1] nodes. We implemented LZCompressor and LZDecompressor Click elements for our experiment.[2] Also, we reduced the sending transmission rate from the compression element to 1Mbps, making the compression link the narrow link of the path. Note that to confirm that our compression link is indeed the bottleneck, we used pathrate [10], a capacity estimation tools that has been proven experimentally to work well with PlanetLab nodes [22].

We could have placed both the sender and the receiver remotely and simulated a single bottleneck link on the path by routing the traffic stream through our local network. But by using a remote sender and a local receiver, we simulate a more common scenario, ensuring that the experiment matches the no-valley property, which derives from the provider-to-customer relationship and is the most commonly adopted routing policy by ASes [14], [35].

B. Results

A set of ten geographically distributed PlanetLab nodes connected via the open Internet were selected for our experiment (Table II). We define an experiment scenario uniquely as three elements: (1) a remote PlanetLab node, (2) whether we applied link-layer compression or not in the experiment, and (3) whether the cooperative receiver approach or the responsive receiver approach was used to detect compression in the experiment. For every scenario, we performed 24 individual experiments, within a span of 24 hours, running only one experiment every hour.

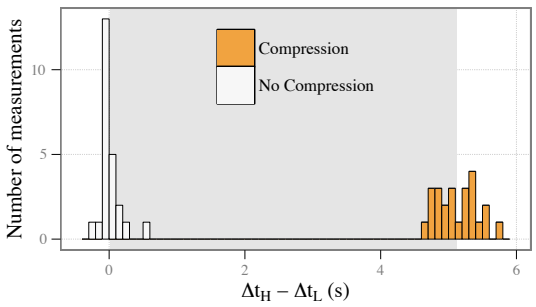

Fig. 5. Histogram of $\Delta t_H - \Delta t_L$ for two sets of 24 measurements from a PlanetLab node in Singapore (the gray region indicates the gap between the means of the two distributions).

[2]Our code and implemented `ns-3` and Click elements are available publicly on http://lasr.cs.ucla.edu/vahab/Taracom. The implementation details are presented in our technical report [28].

To illustrate how the aggregate data looks, we depicted the measurement histogram for one node when testing with the cooperative receiver (Fig. 5). Looking at the histogram, we observe that each set of data can be described by a normal distribution. The noticeable gap between the two distributions is an indication of the compression effects. Also, another observation is that the measurements are slightly more spread for compression than non-compression. As discussed in Section IV, we use different sets of random bytes for the payloads of high entropy probe packets used for each experiment. This results in inconsistent compression ratios, which we suspect are responsible for the wider spread of timings when compression is applied.

Table II summarizes our results for all the experiment scenarios we performed by illustrating the normal distribution parameters in each scenario. The results confirm the existence of a significant gap in the presence and absence of network compression for all scenarios tested.

An observation from Table II is that the distributions for the responsive approach scenarios are relatively more spread compared to those of the cooperative approach. This is because in the responsive receiver approach, the ICMP reply packets travel the reverse path back to the sender, adding additional variability to the delay observations. In addition, since the effects of double compression are not very different than effects from single compression, a different kind of observation from our results suggests that, except for our own intermediary compression, there is no effective compression provided between the end-hosts selected in our scenarios in the duration of our experiment.

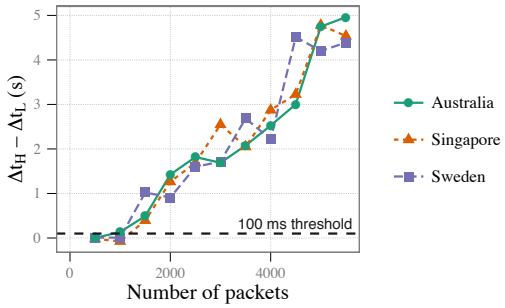

Fig. 6. Comparison of the number of probe packets used for detection when using only a single measurement from three geographically diverse distant PlanetLab nodes.

Fig. 6 demonstrates the difference between the duration of the packet trains between high and low entropy for three different nodes, when considering only their first measurements. As can be seen, the plots are not as linear as those presented in the simulations in Section V.A where the measurements were performed in a clean and more controlled environment, yet they follow a similar increasing trend.

VII. Discussion

A. Desirable Characteristics

In general, for any end-to-end active network measurements there is a set of desirable, and in some cases necessary,

			Cooperative Receiver				Responsive Receiver			
			Compression $\Delta t_H - \Delta t_L$		No Compression $\Delta t_H - \Delta t_L$		Compression $\Delta t_H - \Delta t_L$		No Compression $\Delta t_H - \Delta t_L$	
Location	**IP Address**	**Est RTT**	μ	σ	μ	σ	μ	σ	μ	σ
Singapore	203.30.39.238	201	5119	300	54	320	5010	430	78	450
California	128.111.52.59	5	5052	640	-42	290	4870	890	18	360
Brazil	200.17.202.195	188	5000	450	-46	130	5120	470	-35	620
Czech Republic	147.229.10.250	190	4389	440	-34	230	4102	830	25	1020
New Zealand	130.195.4.69	136	4632	530	79	410	5103	1050	-10	1205
Massachusetts	75.130.96.13	80	5388	410	13	230	5202	410	28	730
Canada	216.48.80.12	94	4624	440	161	700	4700	450	130	410
Sweden	192.16.125.11	185	1601	1220	115	390	2440	1300	150	1380
South Korea	143.248.55.128	173	4823	1140	155	550	4322	930	94	1100
Australia	130.194.252.8	188	4827	500	50	550	4730	780	-12	1240

TABLE II
SUMMARY OF THE RESULTS FROM THE INTERNET EXPERIMENTS (MSEC).

characteristics that should be satisfied.

1) Working with uncooperative intermediaries: Obtaining any information directly from routers is usually impossible, so end-to-end measurements should not rely on cooperation from them.

2) Robust against cross traffic: Cross traffic is usually present and can significantly affect the network measurements. Hence, measurement mechanisms should be accurate even in the presence of cross traffic. In addition, no assumption should be made on the characteristics of the cross traffic when considering its impact on the measurement.

We ran our experiment for a span of 24 hours and from different parts of the world. While we cannot truly confirm that during our measurements the network path experienced high volumes of cross traffic and congestion, the experiments were performed in realistic scenarios, accurately discovering the effects of compression in all of the scenarios. Normally (unless there is an ongoing long-term DDoS attack), cross traffic and high levels of congestion on a particular path persist for only a limited amount of time [24].

3) Non-intrusive: Any detection approach using network measurements should not significantly affect the traffic in the path or the throughput of the other connections. If some active probing is necessary, it should be minimized. Also, a non-intrusive network measurement technique should not affect the actual property being measured. Our experiment consists of two sets of 6000 packets of 1100 bytes each; the sets are sent one minute apart. Each set adds up to a total size of 6.6 MB, which is equivalent to the size of a typical high-quality MP3 song [7].

4) Short measurement time: Short measurement time is desirable, but not always required. For instance, it is required for available bandwidth measurements because the average available bandwidth can change over time; therefore, it is important to measure it quickly. On the other hand, IP-level or link-layer compression on the path is likely to persist for a longer period, so a quick measurement is less vital.

5) Minimal performance overhead: Minimal overhead ensures that the measurement can be performed on typical machines with typical resources, and also that it does not interfere with other processes running on those machines.

B. Timestamp Precision and Resolution Effects on Detection Quality

End-hosts that perform Internet measurements can introduce delays and bottlenecks that are due to hardware or operating system effects and have nothing to do with the network behavior they are measuring. This problem is particularly acute when the timestamping of network events occurs at the application level.

Clearly the more accurately we can measure the time, the better the outcome of the detection process. This is also true as the end-hosts use more accurate time resolution. However, in devising our technique and in the implementation of our experiments, we intentionally used only standard hardware and software. For instance, we avoided using any special hardware components for precise timestamping of packet arrivals, capturing packets at the kernel level, or packet sniffing applications such as libpcap [23]. This was done to ensure that our detection technique works for typical end-hosts with typical machines and resources. Our results confirmed this.

VIII. CONCLUDING REMARKS

In this paper we examined the feasibility of detecting whether intermediaries have performed compression on the path. We presented an end-to-end packet-train-like approach that works in both cooperative and responsive environments. Our Internet experiments confirmed our detection approach. While this work constitutes a significance advance, we believe that this is just the beginning of further research in this direction.

Currently, there is no information about how commonly link-layer compression is deployed. This suggests that the next step will be an investigation of the prevalence of link-layer compression. We believe this realization is important before proceeding with further research in this direction. We plan to examine the prevalence of link-layer compression in the Internet, using our findings presented in this paper. Based on these results, we will be able to take further steps to complete this research.

For practical purposes, a desirable tool should be able to do the detection with just a few measurements—and ideally,

with only one. A lightweight probing technique could even be more attractive and useful, particularly for mobile applications where resources are limited. But, a tool that only takes one or a few measurements should respond to variability well, as it then requires dealing with clock skew and time precision, context-switching effects, congestion, etc. Besides, an automated method for assessing suitable values for detection process parameters based on the network environment is also beneficial for any automated network tool.

The existing bandwidth and capacity estimation techniques do not take into consideration the presence of compression. Another significant focus of future work would be to test and observe how accurately the current tools respond, and how they should be adjusted, in the presence of compression on the path.

An area that makes our approach particularly attractive is the mobile environment, since mobile devices typically have considerably lower available bandwidth. We will test our approach in a mobile environment where some of its characteristics are different from the wired Internet results reported here. For example, we will experiment with considerably higher loss rates. We will expand our approach so that it also works in this type of environment.

Finally, as suggested in the introduction, detecting compression is one important element of the more general problem of detecting all third-party influences on packets submitted to the Internet. Currently, users can learn very little about the fate of their traffic once it is sent. Ideally, for many good reasons, they should be able to know more. From a pure research point of view, it would be valuable to better understand what is possible to know about traffic handling on the Internet, and what can be done to acquire this knowledge. This work represents a step in improving that knowledge.

ACKNOWLEDGMENTS

The authors are grateful to Robert Chang, Matin Fouladinejad, and Mahdi Hosseini-Moghaddam for helping with the implementation of the Internet experiments.

REFERENCES

[1] "ns-3: An Open Simulation Environment," http://www.nsnam.org/.

[2] "PlanetLab: An Open network platform," http://www.planet-lab.org/.

[3] "Nokia hijacks mobile browser traffic, decrypts HTTPS data," http:www.zdnet.comnokia-hijacks-mobile-browser-traffic-decrypts-https-data-7000009655, 2013.

[4] A. Anand, A. Gupta, A. Akella, S. Seshan, and S. Shenker, "Packet caches on routers: the implications of universal redundant traffic elimination," in *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 4. ACM, 2008, pp. 219–230.

[5] J. Border and J. Heath, "RFC3051: IP payload compression using ITU-T V. 44 packet method," 2001.

[6] S. Casner and V. Jacobson, "RFC2508: Compressing IP/UDP/RTP headers for low-speed serial links," 1999.

[7] J. C. Chu, K. S. Labonte, and B. N. Levine, "Availability and locality measurements of peer-to-peer file systems," in *ITCom 2002: The Convergence of Information Technologies and Communications*. International Society for Optics and Photonics, 2002, pp. 310–321.

[8] Cisco Documents - Document ID: 14156, *Understanding Data Compression*. www.cisco.com/application/pdf/paws/9289/wan_compression_faq.pdf.

[9] S. Dawkins, "Internet Draft: Performance Implications of Link-Layer Characteristics: Slow Links," 1998.

[10] C. Dovrolis, P. Ramanathan, and D. Moore, "What do packet dispersion techniques measure?" in *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 2. IEEE, 2001, pp. 905–914.

[11] M. Engan, S. Casner, C. Bormann, and T. Koren, "RFC 2509: IP header compression over PPP," RFC 2509, February, Tech. Rep., 1999.

[12] R. Friend and R. Monsour, "RFC2395: IP payload compression using LZS," 1998.

[13] R. Friend and W. Simpson, "RFC1974: PPP Stac LZS Compression Protocol," 1996.

[14] L. Gao, "On inferring autonomous system relationships in the internet," *IEEE/ACM Transactions on Networking (ToN)*, vol. 9, no. 6, pp. 733–745, 2001.

[15] D. Han, A. Anand, A. Akella, and S. Seshan, "RPT: Re-architecting loss protection for content-aware networks," in *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation, NSDI*, vol. 12, 2011, pp. 6–6.

[16] HP Support FAQs, *Using Compression with HP Router Products*. http:www.hp.comrndsupportfaqspdfcomp.pdf.

[17] V. Jacobson, "RFC 1144: Compressing TCP," *IP headers for low speed serial links*, 1990.

[18] R. Jain and S. Routhier, "Packet trains–measurements and a new model for computer network traffic," *Selected Areas in Communications, IEEE Journal on*, vol. 4, no. 6, pp. 986–995, 1986.

[19] L. Jonsson, G. Pelletier, and K. Sandlund, "RFC 4995: The Robust Header Compression (ROHC) Framework," *Network Working Group*, pp. 1–40, 2007.

[20] S. Keshav, "A control-theoretic approach to flow control," *ACM SIGCOMM Computer Communication Review*, vol. 25, no. 1, pp. 188–201, 1995.

[21] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek, "The Click modular router," *ACM Transactions on Computer Systems (TOCS)*, vol. 18, no. 3, pp. 263–297, 2000.

[22] S.-J. Lee, P. Sharma, S. Banerjee, S. Basu, and R. Fonseca, "Measuring bandwidth between planetlab nodes," in *Proceedings of the 6th international conference on Passive and Active Network Measurement*, ser. PAM'05. Berlin, Heidelberg: Springer-Verlag, 2005, pp. 292–305.

[23] S. McCanne, C. Leres, and V. Jacobson, "Libpcap," 1989. [Online]. Available: http://www.tcpdump. org

[24] D. McPherson, R. Dobbins, M. Hollyman, C. Labovitzh, and J. Nazario, "Worldwide infrastructure security report, Volume V," *Arbor Networks*, 2010.

[25] B. Melander, M. Bjorkman, and P. Gunningberg, "A new end-to-end probing and analysis method for estimating bandwidth bottlenecks," in *IEEE Global Telecommunications Conference*, 2000, pp. 415–420.

[26] V. Paxson, "Strategies for sound internet measurement," in *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, ser. IMC '04. New York, NY, USA: ACM, 2004, pp. 263–271.

[27] R. Pereira, "RFC2394: IP payload compression using DEFLATE," 1998.

[28] V. Pournaghshband, M. Fouladinejad, and M. Moghaddam, "Taracom: End-to-End Detection of Third-Party Compression," UCLA, Tech. Rep., 2014. [Online]. Available: lasr.cs.ucla.edu/vahab/Taracom/Taracom.pdf

[29] V. Pournaghshband, L. Kleinrock, P. L. Reiher, and A. Afanasyev, "Controlling applications by managing network characteristics," in *IEEE International Conference on Communications (ICC)*, 2012. [Online]. Available: http://dx.doi.org/10.1109/ICC.2012.6364064

[30] R. Prasad, C. Dovrolis, M. Murray, and K. Claffy, "Bandwidth estimation: metrics, measurement techniques, and tools," *Network, IEEE*, vol. 17, no. 6, pp. 27–35, 2003.

[31] D. Rand, "RFC1978: PPP Predictor Compression Protocol," 1996.

[32] D. Salomon, *Data Compression.: The Complete Reference.*, 2004.

[33] A. Shacham, B. Monsour, R. Pereira, and M. Thomas, "RFC 3173: IP Payload Compression Protocol (IPComp)," 2001.

[34] A. Soule, A. Nucci, R. Cruz, E. Leonardi, and N. Taft, "How to identify and estimate the largest traffic matrix elements in a dynamic environment," in *Proceedings of the joint international conference on measurement and modeling of computer systems*, ser. SIGMETRICS '04. New York, NY, USA: ACM, 2004, pp. 73–84.

[35] F. Wang, Z. M. Mao, J. Wang, L. Gao, and R. Bush, "A measurement study on the impact of routing events on end-to-end internet path performance," in *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 4. ACM, 2006, pp. 375–386.