Edge-to-Edge Achieved Transfer Throughput Inference Using Link Utilization Counts

Demetris Antoniades and Constantine Dovrolis
School of Computer Science
College of Computing
Georgia Institute of Technology
Atlanta, Georgia
Email: [danton,constantine]@gatech.edu

***Abstract*—We propose a methodology to infer edge-to-edge achieved transfer throughput using link utilization counts. Our method treats variations in the link utilization time-series as possible transfer starting or ending events. Iteratively following these variations to the neighboring routers, we then identify the path the transfer traversed through the monitored network. Our evaluation shows that this method can identify events larger than 3 Mbit/sec and longer than 2 minutes in duration with more than 95% recall. Additionally, we show that event detection is strongly correlated with the traffic in the busiest router in the path. We discuss how a number of applications such as throughput prediction and DDoS attack source detection can use the inferred information.**

Keywords-throughput inference; edge-to-edge; SNMP; network performance monitoring.

I. Introduction

The Simple Network Management Protocol (SNMP) [1], [2] is widely used to monitor aggregated link usage from network components (routers, switches, etc.). Such data, provide a valuable resource for network administrators, aiding decisions about network routing, provisioning and configuration. SNMP data is simple to collect and maintain, providing a low disk space option for a log of historical network usage.

On the other hand, Netflow data provides detailed information about end-to-end performance. Using Netflow, one can have information about the communication between two hosts, the amount of packets and data transferred between them and the achieved throughput. The enhanced information given by Netflow comes with additional archival cost and many privacy concerns. To reduce the cost of collecting Netflow data, aggressive sampling (i.e., 1:1000 packets) is often employed, even for relatively low-speed networks [3]. Sampling affects the accuracy of Netflow data and may limit its applications [4]. Netflow records also include the IP addresses and port numbers used by the participating endpoints. Such content raises significant user privacy concerns [5], [6].

In this work, we propose leveraging SNMP link utilization data to accurately identify edge-to-edge (e2e) information about the achieved throughput of large network transfers. We have developed a methodology for inferring network events from SNMP traffic utilization time series data. Our method is the result of two main observations. First, looking at the time series of a link's utilization, we observe events where the utilization of the link increases (or decreases) to a different level, deviating from the link's normal behavior up to that point. These events could be considered as starting (or ending) points of high-throughput transfers. Second, these events propagate from the input links of a router to the output links of the same router, and from there to a neighboring router, allowing us to infer the actual route that the specific event followed.

Figure 1 illustrates these observations over a network example. Each router connects an organization's internal network to other organizations or intermediate routers. Using SNMP link usage data one can form the utilization time-series for each interface, which represents the traffic transferred between two connected routers. Looking at the time-series between $R7$ and $R9$ one can observe an increase in the link utilization. This increased utilization lasts for some time and then drops. Such behavior can be attributed to a transfer initiated from $R7$'s access network towards some destination. Following this increase from $R9$ to the next router and so on, we can observe that the corresponding transfer continues through $R12$ and $R14$. After $R14$ the transfer either continues to another network or is destined to a host in the access network served by $R14$. Note that the involved router interfaces do not have the same traffic variations in general. At the point that this transfer starts or ends, however, their traffic level changes in a similar fashion. Other transfers can be identified in different parts of the network at the same time. For example we can also observe a transfer between $R1 - R11$ and two transfers between $R2 - R6$.

The work presented in this paper is, to the best of our knowledge, the first that suggests the possibility of inferring edge-to-edge information from aggregated link utilization measurements. In a related work, Gerber et. al. used flow records to estimate the achievable download speed [7]. Similarly to our work, their algorithm eliminates the need for, network intrussive, active measurements. In contrary to this work, the use of flow records makes their solution expensive to deploy. Our contributions can be summarized as follows:

1) We propose a methodology to identify events in SNMP utilization time series.
2) We propose a methodology to map events in an input interface of a router to the output interface of the same router the event is switched to.

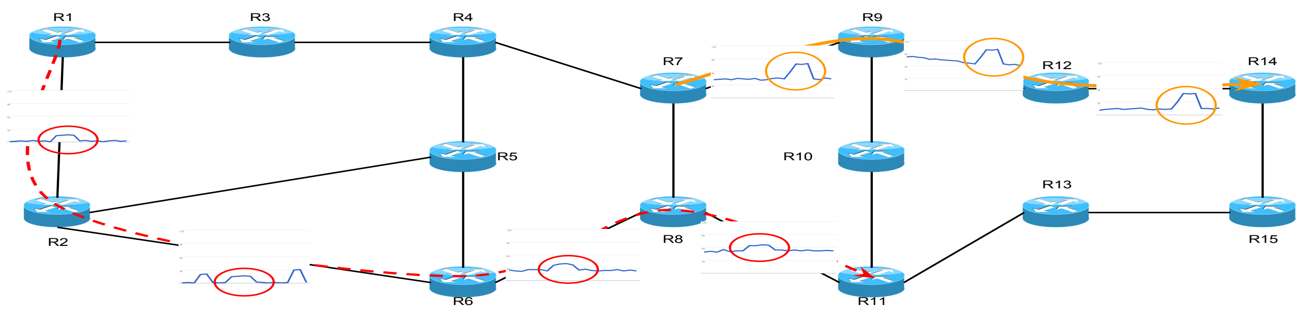


Figure 1. Several traffic utilization increase and decrease events can be identified in each observation period.

3) We evaluate our methodology and show that it can accurately identify events in a real network.

The rest of the paper is organized as follows. Section II provides a detailed description of our methodology also listing the specific challenges we came across in each step. In Section III we provide a detailed evaluation of the proposed methods. Section IV list a number of sample applications that can benefit from our method. Section V discusses open challenges for our method. Finally, Section VI concludes our work.

II. Methodology

Out method consists of the following three steps:

Event inference: In the first step, we identify transfer-start and transfer-end events in the link utilization time series. This is an online processing step. We first transform the utilization time series into a 2-step differentiated time series, and identify as "events" those differences that are larger than a specific, user defined, threshold. The threshold can be defined based on the utilization and variations of the interface(s) of interest. We also propose a simple outlier detection method able to identify events by examining if the link's utilization at the current time period (last 30 seconds) has deviated significantly from the utilization of the link in a recent time window.

Mapping incoming events to outgoing interfaces: After we identify an event (either transfer-start or transfer-end) at an input interface, we proceed to identify the output interface at the same router that the event is forwarded to. Our algorithm considers all transfer events that appear at any router input interface in that time period, and tries to find the most likely outgoing interface that each of those transfer events also appears in.

Identify edge-to-edge path: This step aims to identify the next router that each identified transfer is forwarded to. This step is accomplished easily when we have the network topology of the given network, including the IP address of every router interface in that network. If this information is not available it can be inferred by a matching process between the current output interface and all other input interfaces.

Note that all the three steps of our algorithm can be executed in real-time as new traffic utilization data become available for each link.

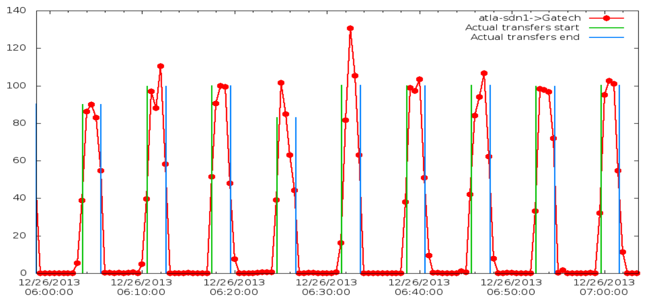


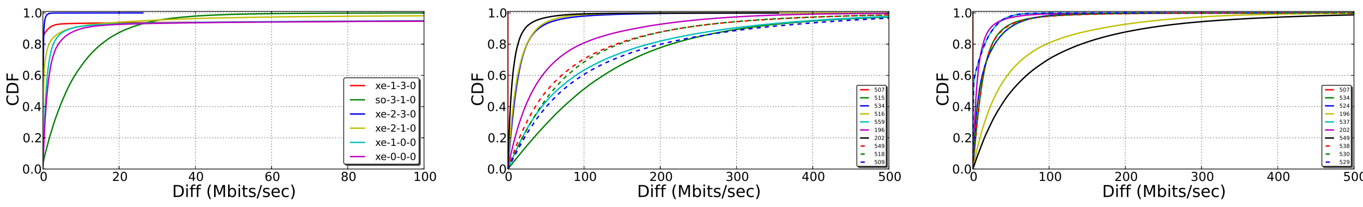
Figure 2. SNMP utilization time series of a link during a number of 100 Mbits/sec transfers traversing that link.

A. *Event inference*

SNMP periodically, every Δ seconds, reports the number of bytes that traversed a specific router link over the previous interval $(t-\Delta)$. Using these byte counts, we can extract the average throughput utilization $U_i(t)$ for a link i over the interval $(t-\Delta,t)$, $U_i(t)=\frac{Bytes(t-\Delta,t)\times 8}{\Delta}$ (1).

Using the link utilization time series we are interested in identifying changes in the utilization of a link I that are created when a high-throughput flow starts or ends. We refer to these changes as flow events e at interface I and denote them with $I(e)$.

Figure 2 plots the SNMP traffic utilization time series as seen in a single network interface. A number of 100 Mbits/sec transfers were active during this time period, traversing the link. Vertical lines show the actual start (green) and end (blue) times of each transfer. We can observe the transfers to gradually appear in the SNMP utilization time series. This can be explained by two reasons: (i) the actual flow events are not aligned with the utilization reporting times and (ii) the utilization time series is an average over all the events at that Δ. Depending on when the flow event appears relatively to the interval start it will affect the average differently. Considering these observations, just using the difference in the utilization between consecutive intervals gives misleading information regarding the flow's throughput since it will only account for the difference in the interval t. If the flow started in the end of interval $t-1$, then the difference will be close to the actual flow throughput. However, if the flow started towards the beginning or the middle of $t-1$ then the difference will be far from the actual flow throughput. V takes into account these non-alignment and averaging effects.



(a) ESnet router (b) Commercial router: Busy links (c) Commercial router: Highly variate links

Figure 3. CDF of event magnitude for different links and routers.

To identify the events we first transform the utilization time series of I to the 2-step differential time series V, where $V_i(t) = U_i(t+1) - U_i(t-1)$ (2). V provides the time series of the link utilization difference between two intervals with distance $2 \times \Delta$. The previous equation will result to the time-series of the utilization difference between two intervals with distance $2 \times \Delta$. This difference is more likely to be closer to the actual flow throughput since it allows for the transition of the traffic utilization from the base ($U_i(t-1)$ in this case) to base + flow throughput ($U_i(t+1)$), in the case of a flow start. In the following we describe two methods to identify these transitions.

1) Threshold based event identification: After this point we consider each value in the $V_i(t)$ time series as a possible transfer start (positive values) or end (negative values). We leave it up to the user to decide which of the events she considers significant. Figure 3 plots the CDF of the magnitude of all events for all the interfaces of an ESnet router located in Atlanta (left) and for the 10 most busiest (middle) and most variable (right) interfaces of an edge router connecting GaTech to the commercial Internet. We calculate variability for a link by estimating the Coefficient of Variation ($CoV = \frac{\sigma}{\mu}$) for the traffic time series of the interface. We then select the links with the highest CoV values for Figure 3(c). Note that the events' magnitude can vary significantly at different links. Some links allow for the identification of rather minor events, i.e., less than 3 Mbit/sec, while in other links most of the events are larger than 100 Mbits/sec. Depending on the link (or path) of interest the user can appropriately set the threshold value for event identification.

2) Outlier based event identification: The first step of this process is to identify significant changes in the differentiated time series V. To identify them we need an outlier detection method that is (i) robust to the utilization variability, (ii) robust to any periodicity in the time-series (iii) does not assume any predefined distribution of the data and (iv) is able to detect outliers online as new data become available. A simple such method is running a robust moving window average over the data, and estimating the Median Absolute Deviation (MAD) from the median during the observation window. This method is also known as the Hampel identifier [8], [9]. The method is controlled by two parameters (a) the size N of the observation window and (b) the multiplicative factor c that defines how strict the outlier detection method is. If c is large then a value should be significantly far from the median to be identified as an outlier. If c is small then values that create small deviations can also be considered outliers. We empirically examine appropriate values for these parameters in the next section.

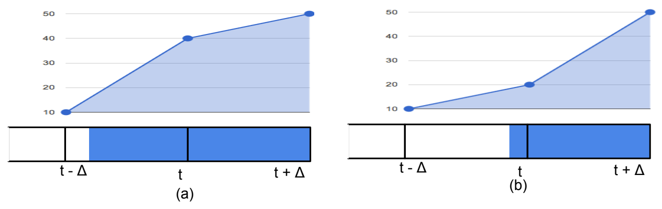


Figure 4. Event time approximation.

Eliminating surrounding outliers: As noticed earlier, a transfer event gradually appears in the utilization time series. As a result a single transfer event might correspond to more than one outliers. Each of the outliers corresponds to one step in the gradual appearance of the flow in the utilization time series. To accurately estimate the throughput of the event and also avoid having multiple values for each event we decide to keep only a single value for each sequence of consecutive outliers of the same trend. To do so, we sum, for each such sequence, the non-overlapping outliers, i.e., $V_i(t)$ for not consecutive t's. For each outlier sequence we then keep the maximum of these sums.

Event time approximation: In both the above methods, the time t of the identified event gives the time of the event relative to the closest utilization interval. To more accurately approximate the event time T_e we use the following equation: $T_e = t - (\Delta \times \frac{|U(t) - U(t-\Delta)|}{|U(t+\Delta) - U(t-\Delta)|})$. Figure 4 illustrates how this approximation works. The plots show sample link utilization time series. The box underneath each plot shows a transfer starting at specific time. When the transfer started close to $t - \Delta$, it will count in the estimation of the throughput utilization U_t in the interval $(t-\Delta, t)$ and thus result to a larger value for $U_{t-\Delta}$ (Figure 4(a)). This results to a large fraction, moving T_e closer to the beginning of interval $t - \Delta$. In the opposite case that the transfer started closer to the end of the interval $(t - \Delta, t)$, the fraction will be small and the estimate T_e will be closer to t (Figure 4(b)).

B. Mapping input events to output interfaces

The previous section described how we identify transfer events in the utilization time series of a link. Our next step is to identify the output interface the event will be forwarded to. This procedure is not trivial and may not have a definite answer at all times. That is because an identified event $E_i(t)$ at input i may actually be the aggregation of a set of transfers $S(E_i(t))$, such as $E_i(t) = (e_x(t) \in S(E_i(t)))$, each with rate equal to $r(e_x(t))$, that appear at input i at time t. The rate change $V_i(t)$

we observe associated with $E_i(t)$ is $V_i(t) = \sum r(e_x(t))$. The mapping problem then becomes: *Given a rate change $V_i(t)$ for every input and output i of a router, at a given time t, determine the switch mappings $I(e) \rightarrow O(e)$ for every transfer $e(t)$ at that time step.*

One can easily show that this problem cannot be solved in the general case. Consider, the case of two aggregated events E_{I1} and E_{I2} at input interfaces $I1$ and $I2$. Suppose that all transfers in those aggregate events are switched to two outputs, creating the aggregate events E_{O1} and E_{O2} at outputs $O1$ and $O2$ as follows: $E_{I1} = \{e_1, e_2\}$, $E_{I2} = \{e_3, e_4\}$, $E_{O1} = \{e_1, e_3\}$ and $E_{O2} = \{e_2, e_4\}$. The only relationship we can find in this scenario, considering all possible combinations of inputs or outputs, is $V_{I1} + V_{I2} = V_{O1} + V_{O2}$. This is obviously not sufficient to solve the problem defined above. To solve the problem we consider the following four conditions:

Condition-1: Every transfer appears individually at both its input and output interface. Assuming that each event has a distinct rate, we can solve the problem by identifying for every input rate change an (approximately) equal rate change at one of the router outputs.

Condition-2: Every event appears individually at its input but not necessarily so at its output. Assuming that any possible combination of events has a distinct aggregate rate, we need to find the set of inputs I_j such that $\sum_{E_i \in I_j} V_{Ei} = O_j$, for every output interface O_j.

Condition-3: Every event appears individually at its output but not necessarily so at its inputs. Similarly to condition-2 we need to find the set of outputs O_j such that $\sum_{E_i \in O_j} V_{Ei} = I_j$, for every input interface I_j.

Condition-4: Every event appears individually at its input or output or both. Without loss of generality, consider that an event e appears individually at its input, say I_e. If this event appears individually also at its output, we can identify its switch mapping as in *Condition-1*. Otherwise, say that e is part of an aggregate event E_k at output k. Then, the rest of the events in E_k must appear individually at their inputs (based on Condition-4). So, we can identify their switch mappings as we did in *Condition-2*. Similarly, if an event appears individually at its output.

Our algorithm for identifying the switch mappings first considers all possible combinations of output interfaces for every input interface to find a matching aggregate rate as in *Condition-3*. Then, it considers all possible combinations of inputs for every output to find a matching aggregate event, as in *Condition-2*. Since small rate variations may occur internally in the router, when traffic switches from the input interface to the output, we use the similarity function $S = \frac{||V_I| - |V_O||}{max(|V_I|, |V_O|)}$ (3), to compare the input and output variations at each step. We consider the mapping that minimizes S as the most likely mapping between the input and output events of interest. An input combination might be also rejected if any of the individual interfaces in that combination (or smaller combinations) better matches the outgoing interface. Similarly for the reverse scenario. We evaluate appropriate similarity thresholds in the next section. Algorithm 1 presents the pseudocode of our switch mapping method.

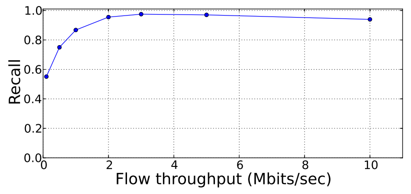
Algorithm 1 Mapping Input to Output interfaces

```
 1: procedure FIND_OUTPUT($E_i(t), E(t), V_o(t)$)
 2:     $IN \leftarrow$ all combinations of $E_i(t)$ with events $E(t)$ from the other input interfaces
 3:     $OUT \leftarrow$ all combinations of 2-step differential at time $t$ for all output interfaces $V_o(t)$
 4:     $RES \leftarrow \emptyset$
 5:     for all $o \in OUT$ do                          ▷ For all output event combinations
 6:         $S = \frac{absE_i(t) - |o|}{max(E_i(t), o)}$            ▷ Calculate similarity with input event
 7:         if $|S| \leq D$ then
 8:             $RES \leftarrow |S|$
 9:         end if
10:     end for
11:     for all $i \in OUT$ do                          ▷ For all input event combinations
12:         for all $e \in V_o(t)$ do                   ▷ for all single events in the output interfaces
13:             $S = \frac{|i| - |o|}{max(i, o)}$                  ▷ Calculate their similarity
14:             if $|S| \leq D$ then
15:                 $RES \leftarrow |S|$
16:             end if
17:         end for
18:     end for
19:     if $RES \neq \emptyset$ then
20:         $r \leftarrow \mathrm{argmin}(RES)$          ▷ Get pair with the maximum similarity
21:         return $r$
22:     else
23:         return NULL
24:     end if
25: end procedure
```

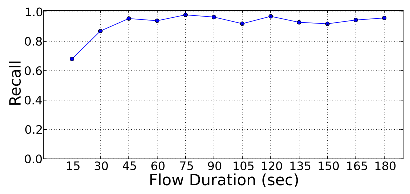
C. Identify the edge-to-edge path

This step aims to identify the edge-to-edge path the event will follow. In this paper we only consider the case where a complete connected view of the monitored network is available. That means that information for all routers the event traverses is available. If we already know which two routers a link connects, we can proceed from one router to the other. We can then infer the path by identifying the switch mappings for all routers in the path step by step, using the method described in the previous section. After we identify the outgoing interface in router R_1 we can then proceed to router R_2, that the corresponding link connects to. In this case the outgoing event $E_O(t)$ in interface O of R_1 becomes the incoming event in the interface I of R_2. Using the switch mapping method we can now identify the outgoing interface of the event in R_2.

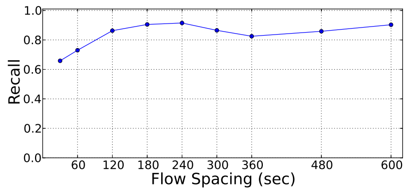
In the case where we do not know which two end-points a link connects, we need to identify this hop using some of the available information. Since a link between two routers is a physical link both end points will most probably observe the exact traffic (with minimal variation due to reporting synchronization). With this fact in mind we can compare either the traffic utilization of the current output host with all input interfaces in the network and identify the incoming interface with the closest traffic utilization time series. One option is to use the Euclidean distance to calculate the distance (d) between each two interfaces for a time window n, ($d = \sqrt{\sum_{t=1}^{n} (U_o(t) - U_i(t))^2}$ (4)), An alternative approach would be to use the 2-step differentiated time series for calculating the distance, instead of the actual traffic utilization time series. Note that we do not need to run the above step for every identified event. Physical links do not change often and thus only verifying the routers connecting the links every few days should be enough.



(a) Transfer throughput

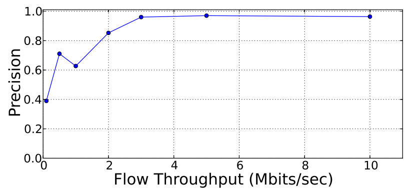


(b) Transfer duration

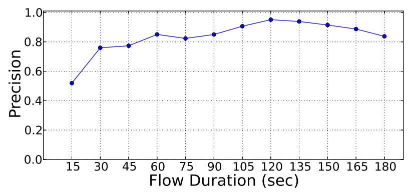


(c) Transfer spacing

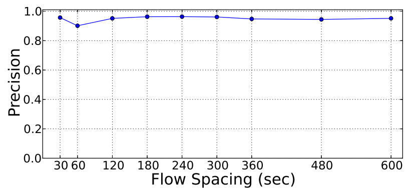
Figure 5. Recall when varying specific transfer properties ($c = 1$ and $W = 20min$).



(a) Transfer throughput



(b) Transfer duration



(c) Transfer spacing

Figure 6. Precision when varying specific transfer properties ($c = 1$ and $W = 20min$).

III. EVALUATION

A. Datasets

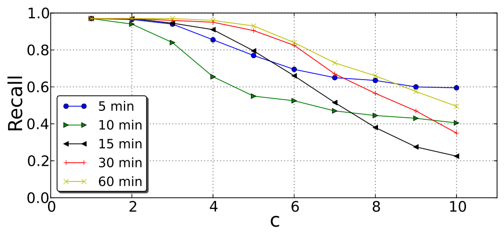
The methodology presented in the previous section aims at inferring the achieved throughput for transfers with a significant amount of data, i.e., high throughput and long duration transfers. In this section we try to identify the lower limits our method has in terms of (i) *transfer throughput*, (ii) *transfer duration* and (iii) *spacing* between two consecutive transfers. Additionally, we identify appropriate values for the method parameters that allow for high accuracy transfer inference.

To evaluate our methodology we create a number of artificial datasets, composed by TCP transfers of specific characteristics, between a machine in Georgia Tech (GT) and Lawrence Berkeley National Lab (LBL). The created transfers traverse only the ESnet network, for which we have access to all the intermediate router utilization data. We also know the routers and interfaces each link connects. We use the nuttcp network performance tool to create the transfers. Depending on the desired transfer characteristic, we keep all other characteristics constant and vary the value of the characteristic in question.

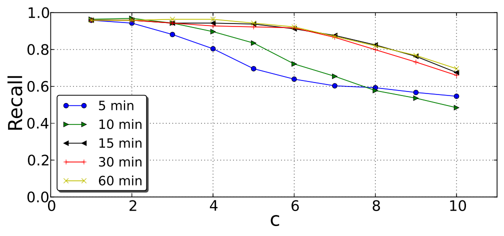
We evaluate the accuracy of our algorithm by calculating (i) recall and (ii) precision. Recall is defined as the number of true positives (TP), i.e., the number of actual transfer-events our method identified, over the total number of events we created. Precision is calculated as $\frac{TP}{TP+FP}$, where FP is the number of false positives, i.e., events detected by our method that were not part of the artificial dataset.

B. Outlier based event identification method

Minimum event throughput: Figure 5(a) plots recall as a function of the transfer's throughput. Our method manages to achieve more than 95% recall for transfers with throughput 2 Mbits/sec and larger. Figure 6(a) plots the precision of the method. Precision reaches values larger than 0.95 for transfer throughput larger than 3 Mbits/sec. We consider the latter value to be a reasonable throughput threshold.



(a) 120 seconds



(b) 180 seconds

Figure 7. Effect of TCP transfer duration in the selection of w and c values on method recall.

Minimum transfer duration: Figure 5(b) plots recall as a function of the transfer duration. We can observe that our method can achieve recall larger than 95% for duration larger than Δ. Looking at the method's precision (Figure 6(b)), we can see that our method can achieve precision larger than 0.9 when the duration of the transfer is close to 2 minutes.

Minimum transfer spacing: Figure 5(c) plots recall as a function of the interval between the transfers. Recall increases to values larger than 85% when the transfer spacing is 2 minutes. After that point it stabilizes to similar or larger values. The precision remains high as the transfer spacing increases (Figure 6(c)).

Method parameters: Figure 7 examines how the choice of W and c affect the method's recall as transfer duration varies. Small window sizes only work well with small c values,

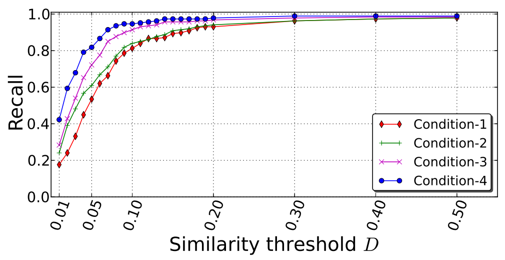


Figure 8. Recall as a function of the similarity threshold between input and output mappings.

resulting to low recall when $c > 2$. For both transfer duration values we can say that a window larger than 15 minutes and a c smaller or equal to 5 provide acceptable accuracy. The precision did not show any variability with the method's parameters when $c > 1$ and $W > 5$ minutes.

C. Traffic switch mapping

We now examine the accuracy of the method for inferring the outgoing interface an event, identified by the previous method, will be forwarded to. To evaluate this method we create a number of high throughput transfers (> 3 Mbits/s). We use information from Paris traceroute as the ground truth for the interface mapping. A true positive (TP) in this case is an event where one of the possible options given by Paris traceroute was included in the outgoing mapping result. Figure 8 plots recall for all different conditions explained in Section II-B as a function of the similarity threshold D. We can observe that *condition-1* is the one that provides the mapping in most of the cases. Using *condition-4* we can get an improvement of about 15% in all cases. Regarding the similarity threshold, a difference of 10% ($D = 0.1$) gives recall values larger than 80% in all cases. *Condition-4* gives more than 95% recall values for any value of D larger than 0.1.

D. Edge-to-edge throughput inference

In the previous subsection, we focused on the evaluation of the event identification and switch mapping methods individually. In this subsection we evaluate the accuracy of our method in inferring edge-to-edge events and reporting their achieved throughput. To do so, we create a second dataset where we create number of transfers between GaTech and LBL. Using nuttcp we limit the average transfer throughput to values in the range of 5 - 110 Mbits/sec. Using our method, described in the previous section, we then try to identify these transfers in the SNMP utilization data.

Figure 9 plots the recall value as a function of the achieved throughput of each event. The left plot shows recall for the whole path between Gatech and LBL (only routers belonging to the ESnet infrastructure). We can see that our method achieves recall values larger than 0.5 for transfer events larger or equal to 20 Mbits/sec. For transfers larger than 200 Mbits/sec, recall ranges to values larger than 0.8. To understand the low recall values for small throughput values, Figure 9 plots the recall values for each router in the path. We can observe that in most routers we can achieve recall values larger than 0.8 independent of the transfer achieved throughput. For one router in the path the recall values drop to values close 0.5 for small throughput values and increase as the transfer throughput increases to values larger than 20 Mbits/sec. Our intuition behind this behavior is that the specific router is a busiest network hub, than the other routers. This means that traffic from different interfaces ("noise" traffic) in that router mixes with the traffic we created. This "noise" traffic affects the IN/OUT interface mapping of small events. To verify this intuition Figure 10 (a), plots the average traffic in the two interfaces of interest (input and output) during the time of the actual events we try to identify. Traffic in a specific instance is calculated as the sum of the traffic on all interfaces of interest in that instance. We can observe that recall drops as the traffic in these interfaces increase. Figure 10 (b) plots the recall as a function of the average traffic in all interfaces of the router. Looking at the average traffic in all the router interfaces we can see how busy the router is and how additional traffic from other interfaces affects out method. We can observe that as the traffic traversing the router increases, the recall drops. This suggests that in an overall busy router the mapping process becomes more difficult since additional traffic from other interfaces might merge with the traffic of interest.

IV. Applicability of the Method

Improvements of throughput prediction: TCP throughput prediction applications are usually based on historical transfers between the end points of interest [10]. The extensive sampling and limited availability of NetFlow data usually limits the applicability of these type of prediction approaches. The achieved transfer throughput values inferred through our method can be used as additional samples in the presence of NetFlow data. Furthermore, in cases where NetFlow data are not available, our method can provide a sample of transfer throughput measurements.

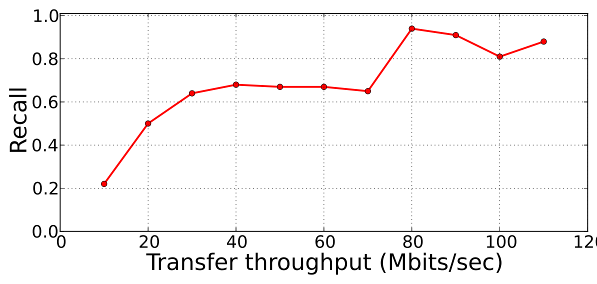
DDoS attack initiator inference: Spoofed traffic is a common method used by attackers to create Distributed Denial of Service (DDoS) attacks [11]. Using our method one can infer the actual source of spoofed traffic, by following the identified events to the source network and not relying to the IP address.

V. Challenges

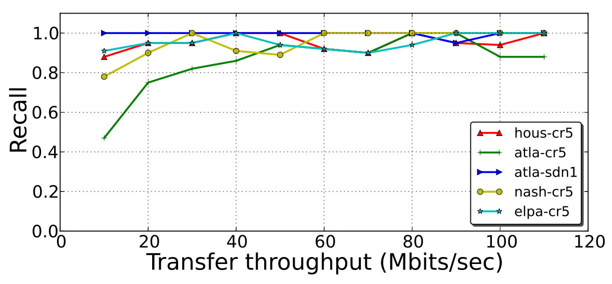
Incomplete Data: Access to SNMP data from every router in the network of interest is not always possible. For example, data may traverse routers that are not owned by the monitoring party, and thus cannot be collected. Additionally, equipment may fail and data might be lost for several reasons. We are exploring ways to take into account these cases, in our method, matching events that may appear to non-adjacent routers.

Transfer identification: An interesting next step would be to identify the actual transfers that traversed the path. This step would need to identify both transfer start and transfer end events and match them. This step is not trivial since transfer throughput might change through the duration of the transfer, or a number of transfers might be active during the same time at the path. We are considering several methods for transfer identification in our ongoing work.

Multipath transfers: A common practice for load balancing network links is to use multiple paths to connect two networks. Some of the flows transferred simultaneously from a source

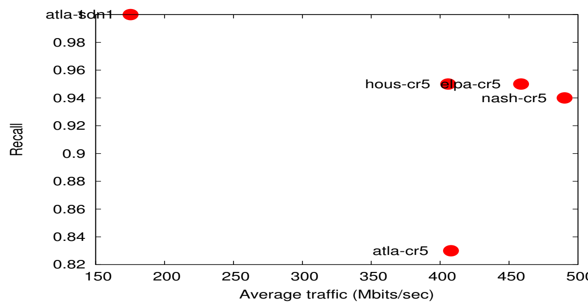


(a) Path recall vs. Actual transfer throughput

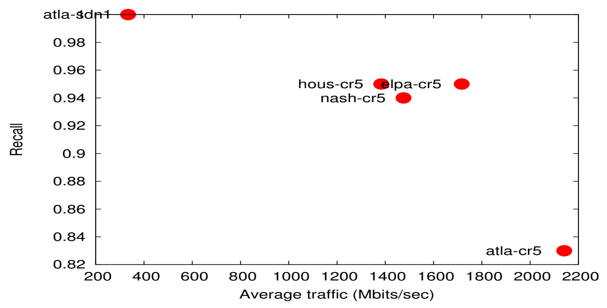


(b) Per router recall vs. Actual transfer throughput

Figure 9. Total path and per router Recall values as a function of the achieved transfer throughput.



(a) IN/OUT interfaces of interest



(b) All router interfaces

Figure 10. Per router in/out mapping recall vs. average traffic in router.

to a destination network are split among different paths that may not share any common routers apart from the source and destination ones. In this case, it is more difficult to identify a matching output link for an event observed in the source router, since split transfers will have lower magnitude. Currently, our methodology does not take these cases into account. We plan to further analyze the intermediate router variations in order to identify the several paths that the transfer could be traversed through. Also, correlation of non-adjacent router interfaces may give some indication for multipath transfers.

Research Vs. Commercial Networks: One may argue that ESnet, as a NREN, has very different traffic patterns than commercial networks and that our method would not apply in such data. Our results showed that we can also identify small magnitude events. In our future work we plan to investigate the applicability of our method in commercial environments with thousands of small flows starting and ending each measurement epoch.

VI. Conclusion

In this paper, we provide evidence that using SNMP link counts we can infer the achieved throughput of Edge-to-Edge transfers taking place in the network. Our method first identifies significant variations in the link counts, and tags them as possible transfer starting or ending points. Iteratively following these variations to the neighboring routers, we are then able to identify the path the specific transfer traversed through the monitored network. Our ongoing work is designed to further evaluate our methodology. Additionally, we plan to test the applicability of the inferred e2e transfers to a number of applications, such as throughput prediction, traffic matrix estimation.

VII. Acknowledgments

This research was supported by the U.S. Department of Energy under grant number DE-FG02-12ER26065.

References

[1] J. Case, M. Fedor, M. Schoffstall, and J. Davin, "Simple network management protocol (snmp)," 1990.

[2] D. Harrington, R. Presuhn, and B. Wijnen, "An architecture for describing simple network management protocol (snmp) management frameworks," rfc 3411, Dec, Tech. Rep., 2002.

[3] C. Estan and G. Varghese, "New directions in traffic measurement and accounting," in ACM SIGCOMM CCR. ACM, 2002.

[4] B. Choi and S. Bhattacharyya, "On the accuracy and overhead of cisco sampled netflow," in ACM LSNI, 2005.

[5] S. Coull, M. Collins, C. Wright, F. Monrose, and M. Reiter, "On web browsing privacy in anonymized netflows," in Proceedings of 16th USENIX Security Symposium. USENIX Association, 2007.

[6] M. Foukarakis, D. Antoniades, S. Antonatos, and E. Markatos, "Flexible and high-performance anonymization of netflow records using anontool," in SecureComm. IEEE, 2007.

[7] A. Gerber, J. Pang, O. Spatscheck, and S. Venkataraman, "Speed testing without speed tests: estimating achievable download speed from passive measurements," in Proceedings of the 10th ACM SIGCOMM conference on Internet measurement. ACM, 2010, pp. 424–430.

[8] L. Davies and U. Gather, "The identification of multiple outliers," Journal of the American Statistical Association, vol. 88, no. 423, 1993, pp. 782–792. [Online]. Available: http://www.jstor.org/stable/2290763

[9] P. Menold, R. Pearson, and F. Allgower, "Online outlier detection and removal," 1999.

[10] Q. He, C. Dovrolis, and M. Ammar, "On the predictability of large transfer tcp throughput," in ACM SIGCOMM CCR. ACM, 2005.

[11] F. Lau, S. H. Rubin, M. H. Smith, and L. Trajkovic, "Distributed denial of service attacks," in Systems, Man, and Cybernetics. IEEE, 2000.