On estimating the number of flows

Bruce Spang
Stanford University
bspang@stanford.edu

Nick McKeown
Stanford University
nickm@stanford.edu

ABSTRACT

There are a number of suggested sizes for router buffers that depend on the number of flows using the router. In an ideal setting, this is an easy quantity to measure. Unfortunately, in a practical setting there are numerous challenges in both both defining and estimating the number of flows. We will discuss some of the complications which arise when trying to estimate the number of flows in practice, and ways of working around these complications. We support our arguments with data from real network traces, and provide recommendations to network operators who need to estimate the number of flows in their networks.

1 INTRODUCTION

Prior work suggests that in order to optimally size a router buffer, we need to know the number of TCP flows sharing a link. For example, in [2] Appenzeller et al. argue that a buffer for a link of capacity C carrying n flows each with RTT R should be at least as large as $C \cdot R/\sqrt{n}$ to keep the link fully utilized.

On the other hand, Dhamdhere et al. in [9] argue that as a consequence of the Mathis model of TCP [16], the packet loss in TCP is proportional to the square of the number of flows. Therefore, a buffer should grow with the number of flows in order to limit loss.

In order to test these assertions or follow their recommendations, a network operator must first determine how many TCP flows share the link. It is not at all obvious how to do so.

In this short paper we describe how the number of flows can be estimated in practice. It is more challenging than existing work suggests, and we argue it is an interesting research problem on its own. We start by considering what "the number of flows" means in an idealized setting, and then discuss how practice differs from (and complicates) this idealized setting. We examine different ways to estimate the number of flows, supporting our arguments with data from real network traces.

We do not come up with one-size-fits-all recommendations for estimating the number of flows, but do suggest some ways of estimating the number of flows which may be feasible in practice.

2 THE IDEALIZED NUMBER OF FLOWS

In the idealized setting of existing buffer sizing work, the number of flows is easy to estimate. In this setting, a flow is a set of packets belonging to a particular TCP flow. Each flow starts with a SYN packet, ends with a FIN or RST packet, and continually sends packets in between, according to the congestion control algorithm. The number of flows at any instant is the number of TCP flows which have sent a SYN packet and not yet sent a FIN or RST packet.

If we are able to observe all packets in this idealized setting, measuring the number of flows is easy. Starting with no flows, we just keep one counter of the number of flows, increment it for each SYN packet, and decrement it for each FIN or RST packet. At any instant, this is the exact number of flows. This can be measured periodically to observe how the number of flows changes over time, and used to size router buffers.

3 ESTIMATING THE NUMBER OF FLOWS IN PRACTICE

There are a number of complications which prevent this idealized method from working well in practice. Our measurements may miss the start of a flow, the end of a flow, and some of the packets in between. In this section, we elaborate on each of these points and describe techniques for estimating the number of flows.

Throughout this section, we will support our claims using data from a real network. We use the 2019 CAIDA anonymized internet trace dataset [5], specifically the 201901117-125910 trace, a complete packet trace of about 50 seconds of traffic on a link from São Paulo to New York.

3.1 Missing the start of flows

In practice, we may not be able to collect SYN packet to determine the start of each flow. This may happen for many reasons. For example, a flow may begin before our measurement does, routing changes may result in a halfway-complete flow suddenly appearing at a router, or a protocol of interest might not have an identifiable type of starting packet.

This is a significant issue in the CAIDA dataset. Figure 1 shows how the fraction of flows for which we have observed a SYN packet changes over the duration of the trace. For each second in the CAIDA trace, we count both the number of flows, and the number of flows for which we have previously seen a SYN packet. The number of flows with a SYN packet gradually increases over time, but remains well under half of all flows. We expect this would become less of a concern after a long period of measurement, but limits the usefulness for short measurements.

To resolve the issue for short measurement periods, we define the start of the flow as the first time a packet with a particular five tuple appears at a router. This makes measurement more complicated.

Depending on the volume of traffic, it may be possible to record all unique flow identifiers and in doing so, determine when a new one arrives. Some routers may be able to do this automatically, using NetFlow or sFlow. If the router can collect packet headers or flow identifiers, perhaps via a SPAN port on the router, it can send those to a collector. This collector could then increment a counter each time it sees a new unique flow identifier and decrement a counter each time a flow ends. Even in large internet service providers, this could be a practical solution for short periods of time. Continuously collecting full IPv4 and TCP headers for a 100 Gb/s link creates a stream of only 2.6 Gb/s[1], or 866Mb/s if we pare it down to only the five tuple.

[1] With 39 bytes of header and maximum length 1500 byte packets.

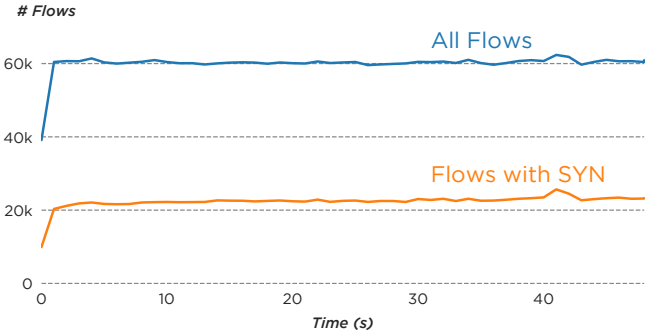

Figure 1: Number of flows per second for which the CAIDA trace contains a SYN packet, compared to all flows per second.

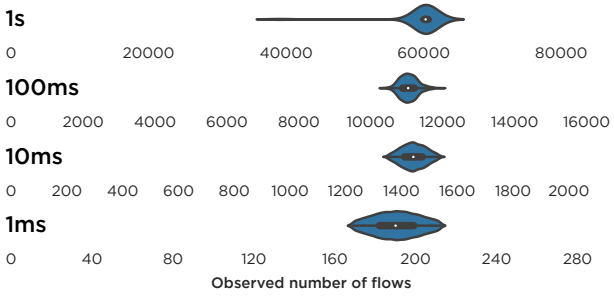

Figure 2: Number of unique five tuples in CAIDA packet trace during intervals of various durations.

If the switch is programmable, it is possible to estimate the number of unique flow identifiers using the switch itself and subtract the number of flows which have ended. For instance, the authors in [15] implement NetFlow using a programmable switch. Estimating the number of unique identifiers could also be done using the HyperLogLog algorithm [13], as done in [3, 10, 14, 20].

3.2 Missing the end of flows

In practice, we may not be able to tell when a flow has finished sending data. Again, this may happen for many reasons. A client may choose not to (or be physically unable to) send a FIN or RST packet at flow completion. A non-TCP protocol may not have an identifiable packet which marks the end of a flow.

In existing buffer sizing work, it is also not clear when a flow contributes to the buffer size. Does a flow contribute when it has started (i.e. a SYN) but not yet finished (i.e. a FIN, RST, or timeout), even if it sends no packets? Or conversely, perhaps a flow only contributes when it has a packet physically present in a buffer? In the case of the Appenzeller result [2], which predicts a buffer size that decreases with n, there must be a point at which most flows do not have packets in the buffer. What is the correct value of n in this case?

Or, consider video streaming flows (which make up a large fraction of current Internet traffic). Video servers typically send chunks of compressed video in an on/off pattern lasting many RTTs. If, for example, a video server sends 3 second chunks with 3 seconds of silence in-between, and the RTT is 100ms, then the silence period is 30 RTTs. Should we consider the flow to be actively participating in the buffering effect during the silence period, or not?

In all of these cases, we will need to make a judgement about when a flow is no longer affecting the buffer size. A natural way to do this is using a timeout—if a flow does not send a packet within some amount of time T, we will consider it finished. It is important to get this timeout right to get a good estimate for buffer sizing. While it is not clear what the right definition is for existing buffer sizing work, we conjecture that a timeout on the order of a few RTTs (e.g. 100ms-1s) would be fairly accurate.

In the CAIDA trace, the length of this timeout has a major impact on the number of flows we estimate. Figure 2 shows the number of flows in the trace for various timeout values. We count the number of flows which send a packet within each timeout-sized interval, and ignore outlying intervals (e.g. the first and last second of the trace). There are about sixty thousand unique flows each second, and as we look at smaller scales there are many fewer flows. The number of flows does not shrink proportionally to the timeout value. Depending on which timeout we use, the work of Appenzeller et al. [2] would suggest a reduction of buffer size between 15 and 250. Clearly, the choice of timeout is very important to get the correct buffer size.

3.3 Sampling packets

So far we have described estimators for the case where every packet during some measurement interval is either collected or analyzed on-the-fly. In today's large operator networks, this is often difficult to accomplish. Typically, packets are sampled before they are analyzed, for example by including every Nth packet in the sample or including each packet independently with some probability [1, 18]. This makes it much harder to correctly estimate the number of flows. We do not yet have practical recommendations for estimating the number of flows using sampling.

Naïvely estimating the number flows by counting the unique flows in the sample is biased. If packets arrived with a uniformly random order, a flow with many packets during the interval will be in the sample with high probability, while a flow with just a few packets is less likely to be in the sample. This is a very real concern in practice: Figure 3 shows a log-log plot of the distribution of flow sizes for the CAIDA trace. The flow length distribution is very skewed: many flows have only a few packets, while just a few flows have many packets. This suggests that naïvely estimating the number of flows by counting the number in a sample is unlikely to give a good estimate, since there are many single-packet flows which may not be included at all.

In [11], Duffield et al. suggest estimating the number of SYN packets as a proxy for estimating the number of flows. Suppose each packet is independently included in a sample with probability p. If N_S is the number of SYN packets in the sample, then $\hat{N}_s = N_s/p$ is an unbiased estimate of the number of SYN packets. We ran this SYN estimator on part of the CAIDA trace, including each packet independently with a certain probability and estimating the number of SYN packets. We repeated each estimate ten times. Figure 4 shows the results. As expected, this is a very accurate estimator for the

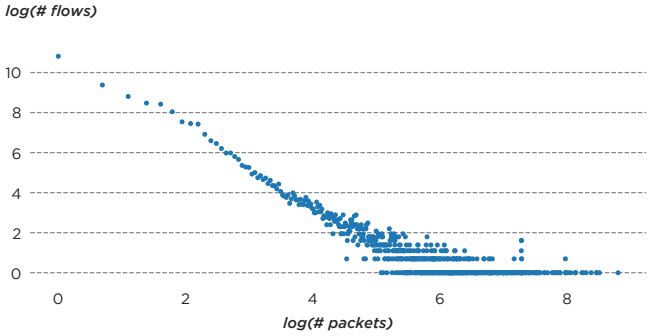

Figure 3: Distribution of flow sizes in one second of a CAIDA packet trace.

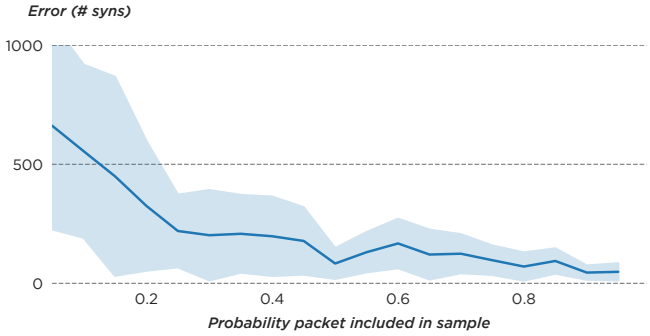

Figure 4: Error in estimate of number of SYN packets from a sample of packets. The shaded blue region is a 95% confidence interval over ten trials.

number of SYNs, especially as the fraction of packets included in a sample increases. As we have already argued in Section 3.1, however, the number of SYN packets in a trace is an inaccurate estimate of the number of flows.

This problem has also been considered in statistics, where there has been a long line of work on estimating the number of species in an unknown population. In the 1940s, Corbet spent two years in the field discovering many new species of butterflies, and was curious how many more species of butterflies could be discovered if he went back for another two years. Here, we could think of each new species of butterflies as a new flow, and the entire population of butterflies as the stream of packets in the router. Learning the number of species in the entire population would correspond to estimating the number of flows in the router.

In [12], Fisher, Corbet, and Williams gave statistical estimators for this problem. This was followed by a long line of work, reviewed in [4]. More recently, there has been a line of theoretical work proving upper and lower bounds for this problem [19, 21, 22], culminating in the work of Orlitsky et al. [17] which describes an efficient estimator with asymptotically optimal convergence rates.

To get a sense of how well these statistical estimators work for packet traces, we ran the estimators on a subset of the 2019 CAIDA passive packet trace [5]. Many of the estimators are available as part of the Python package `pydistinct` [6]. We also implemented the asymptotically optimal estimator of Orlitsky et al. [17] and the Duffield et al. estimator based on SYN packets [11].

We tested the statistical estimators in a standard networking sampling setting. We took the CAIDA trace, which consists of all packets observed by their routers during the capture interval. We took the first second of this trace, and counted the number of flows. We then took a number of samples of this sub-trace by including each packet in the sample independently with a certain probability. We ran estimators for the number of species on these samples to estimate the number of flows in the trace. We independently repeated each sample one hundred times. Figure 5 show the result of these estimates for a few of the more noteworthy estimators. The Chao estimator is from [7], the Chao-Lee from [8], and the Naïve estimator simply counts the number of unique flows in the sample.

We believe the reason the species estimators were unable to produce accurate results in the CAIDA trace is due to the packet

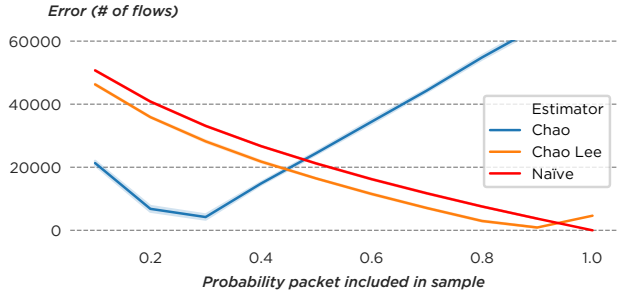

Figure 5: Error in estimates of number of flows from a sample of packets in the CAIDA dataset.

sampling model. The usual assumption for this work is that the sample is generated by picking each item in the population uniformly *with replacement*. To test this, we generated a synthetic dataset which matched this assumption. We took a subset of the CAIDA trace which contained 2,341 flows and generated samples by sampling from the packet trace with replacement. Figure 6 shows the results of this experiment as the length of these samples increase. As expected, all estimators significantly outperform the naïve estimate, and quite quickly converge towards accurate estimates of the number of flows. We include the Orlitsky estimator here because it is provably asymptotically optimal while the others are more heuristic, however our implementation has overflow problems that prevent it from working with larger samples.

4 DISCUSSION

Prior work has suggested that the size of a buffer depends on the number of flows passing through it; particularly for TCP New Reno where the flows interact through drops at the bottleneck. If we wish to measure the relationship between the buffer size and the number of flows in a production network, then we need to carefully decide which flows are currently contributing to the buffer dynamics.

It is hard enough to decide which flows to count over what interval. Even after we have decided on these things, we still need to measure the number of flows. In practice, with current routers, we are unlikely to be able to count or capture all unique flows over a

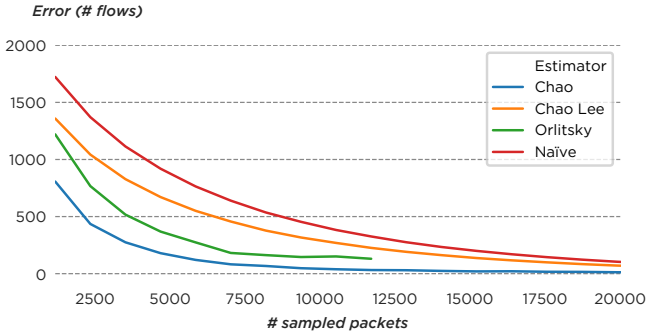

Figure 6: Error in estimates of number of flows from a synthetic packet trace generated from part of the CAIDA trace. There are 2,341 flows.

long period of time; instead, we must settle for a short measurement interval or an estimate of the number of flows.

We make the following recommendations for operators who are interested in measuring the number of flows for buffer sizing purposes:

If measurement infrastructure allows, it seems best to record all traffic during some measurement period and count the number of unique flows. Repeat a few times and over different durations to gain confidence. A measurement duration of a few RTTs is a reasonable place to start. As the duration of the measurement interval increases, the number of flows will also likely increase. If the goal is to reduce the buffer size using the Appenzeller et al. result [2], a shorter duration gives a more conservative estimate and may be more feasible to collect.

If collecting all traffic is infeasible, we suggest recording the number of SYN and FIN/RST packets over the period of several days. Looking at the difference between the two over time may give an estimate of the number of flows which is accurate to an order of magnitude or so. However, if the difference between the two continually grows over time, it is likely a bad estimate of the number of flows.

Estimating the number of flows, especially for the purposes of sizing router buffers, is not a straightforward task. We have described a few approaches to estimating the number of flows which may work well in practice. We hope that future buffer sizing work will give concrete and actionable ways for operators to determine the correct buffer size.

REFERENCES

[1] [n. d.]. NetFlow Configuration Guide - Using NetFlow Sampling to Select the Network Traffic to Track [Cisco Cloud Services Router 1000V Series]. https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/netflow/configuration/xe-16/nf-xe-16-book/nflow-filt-samp-traff-xe.html

[2] Guido Appenzeller, Nick McKeown, and Isaac Keslassy. 2004. Sizing Router Buffers. In *ACM SIGCOMM Computer Communication Review*. ACM, 281–292. https://doi.org/10.1145/1030194.1015499

[3] Barefoot. [n. d.]. In-Network DDoS Detection. https://barefootnetworks.com/use-cases/in-nw-DDoS-detection/

[4] J. Bunge and M. Fitzpatrick. 1993. Estimating the Number of Species: A Review. *J. Amer. Statist. Assoc.* 88, 421 (1993), 364–373. https://doi.org/10.2307/2290733

[5] CAIDA. 2019. The CAIDA Anonymized Internet Traces - 2019. http://www.caida.org/data/passive/passive_dataset.xml

[6] Edwin Chan. 2019. Chanedwin/Pydistinct. https://github.com/chanedwin/pydistinct

[7] Anne Chao. 1984. Nonparametric Estimation of the Number of Classes in a Population. *Scandinavian Journal of Statistics* 11, 4 (1984), 265–270. https://www.jstor.org/stable/4615964

[8] Anne Chao and shen-Ming Lee. 1992. Estimating the Number of Classes Via Sample Coverage. *J. Amer. Statist. Assoc.* 87 (March 1992), 210–217. https://doi.org/10.1080/01621459.1992.10475194

[9] A. Dhamdhere, Hao Jiang, and C. Dovrolis. 2005. Buffer Sizing for Congested Internet Links. In *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.*, Vol. 2. IEEE, Miami, FL, USA, 1072–1083. https://doi.org/10.1109/INFCOM.2005.1498335

[10] Damu Ding, Marco Savi, Gianni Antichi, and Domenico Siracusa. 2019. Incremental Deployment of Programmable Switches for Network-Wide Heavy-Hitter Detection. In *2019 IEEE Conference on Network Softwarization (NetSoft)*. 160–168. https://doi.org/10.1109/NETSOFT.2019.8806649

[11] N. Duffield, C. Lund, and M. Thorup. 2005. Estimating Flow Distributions from Sampled Flow Statistics. *IEEE/ACM Transactions on Networking* 13, 5 (Oct. 2005), 933–946. https://doi.org/10.1109/TNET.2005.852874

[12] R. A. Fisher, A. Steven Corbet, and C. B. Williams. 1943. The Relation Between the Number of Species and the Number of Individuals in a Random Sample of an Animal Population. *Journal of Animal Ecology* 12, 1 (1943), 42–58. https://doi.org/10.2307/1411

[13] Philippe Flajolet and G. Nigel Martin. 1985. Probabilistic Counting Algorithms for Data Base Applications. *J. Comput. Syst. Sci.* 31, 2 (Sept. 1985), 182–209. https://doi.org/10.1016/0022-0000(85)90041-8

[14] Rob Harrison, Qizhe Cai, Arpit Gupta, and Jennifer Rexford. 2018. Network-Wide Heavy Hitter Detection with Commodity Switches. In *Proceedings of the Symposium on SDN Research (SOSR '18)*. ACM, New York, NY, USA, 8:1–8:7. https://doi.org/10.1145/3185467.3185476

[15] Yuliang Li, Rui Miao, Changhoon Kim, and Minlan Yu. 2016. FlowRadar - A Better NetFlow for Data Centers. *NSDI* (2016). https://dblp.org/rec/conf/nsdi/LiMKY16

[16] Matthew Mathis, Jeffrey Semke, Jamshid Mahdavi, and Teunis Ott. 1997. The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm. *Computer Communication Review* 27, 3 (1997), 67–82. https://doi.org/10.1145/263932.264023

[17] Alon Orlitsky, Ananda Theertha Suresh, and Yihong Wu. 2015. Estimating the Number of Unseen Species: A Bird in the Hand Is Worth $\log n $ in the Bush. *arXiv:1511.07428 [math, stat]* (Nov. 2015). arXiv:math, stat/1511.07428 http://arxiv.org/abs/1511.07428

[18] Peter Phaal and Marc Lavine. 2004. sFlow Version 5. https://sflow.org/sflow_version_5.txt

[19] Sofya Raskhodnikova, Dana Ron, Amir Shpilka, and Adam Smith. 2009. Strong Lower Bounds for Approximating Distribution Support Size and the Distinct Elements Problem. *SIAM J. Comput.* 39, 3 (Jan. 2009), 813–842. https://doi.org/10.1137/070701649

[20] Naveen Kr Sharma, Antoine Kaufmann, Thomas Anderson, Changhoon Kim, Arvind Krishnamurthy, Jacob Nelson, and Simon Peter. 2017. Evaluating the Power of Flexible Packet Processing for Network Resource Allocation. In *Proceedings of the 14th USENIX Symposium on Networked Systems Design and Implementation*. Boston, MA, USA, 17.

[21] Gregory Valiant and Paul Valiant. 2011. Estimating the Unseen - an n/Log(n)-Sample Estimator for Entropy and Support Size, Shown Optimal via New CLTs. *STOC* (2011), 685. https://doi.org/10.1145/1993636.1993727

[22] Yihong Wu and Pengkun Yang. 2016. Minimax Rates of Entropy Estimation on Large Alphabets via Best Polynomial Approximation. *IEEE Transactions on Information Theory* 62, 6 (June 2016), 3702–3720. https://doi.org/10.1109/TIT.2016.2548468