Leveraging Zipf's Law for Traffic Offloading

Nadi Sarrar
TU Berlin / T-Labs
nadi@net.t-labs.tu-berlin.de

Steve Uhlig
TU Berlin / T-Labs
steve@net.t-labs.tu-berlin.de

Anja Feldmann
TU Berlin / T-Labs
anja@net.t-labs.tu-berlin.de

Rob Sherwood*
Big Switch Networks
rob.sherwood@bigswitch.com

Xin Huang
Deutsche Telekom Inc. R&D Lab
xin.huang@telekom.com

ABSTRACT

Internet traffic has Zipf-like properties at multiple aggregation levels. These properties suggest the possibility of offloading most of the traffic from a complex controller (e.g., a software router) to a simple forwarder (e.g., a commodity switch), by letting the forwarder handle a very limited set of flows; the heavy hitters. As the volume of traffic from a set of flows is highly dynamic, maintaining a reliable set of heavy hitters over time is challenging. This is especially true when we face a volume limit in the non-offloaded traffic in combination with a constraint in the size of the heavy hitter set or its rate of change. We propose a set selection strategy that takes advantage of the properties of heavy hitters at different time scales. Based on real Internet traffic traces, we show that our strategy is able to offload most of the traffic while limiting the rate of change of the heavy hitter set, suggesting the feasibility of alternative router designs.

Categories and Subject Descriptors

C.2 [**Internetworking**]: Routers

General Terms

Design, Measurement, Performance

Keywords

Heavy Hitters, Software Router, Software Defined Network

1. INTRODUCTION

A reoccurring property of Internet traffic at various levels of aggregation is its consistency with Zipf's law [8, 26, 23, 3], i.e., the amount of traffic per flow is consistent with a Zipf distribution. A flow can be as fine as the traffic between a given IP source-destination pair or as coarse as all the traffic sent to a single network. For this reason, a small fraction of flows capture most of the traffic. However, heavy hitters are known to be volatile [24], making their selection for traffic engineering difficult [19].

We propose to utilize the Zipf properties of Internet traffic to offload most of the traffic from a complex controller to a simple forwarder. We show that by taking advantage of the properties of Internet traffic, a simple forwarder can with limited communication overhead be used to boost the performance of a complex forwarding system, by offloading most of the traffic. The principle behind such a controller-forwarder system lies in bundling a controller, e.g., a complex and flexible software-centric system, together with a forwarder, e.g., a high-performance hardware-centric packet forwarding device. This approach follows the split architecture idea: most of the control plane load should be handled by the software-oriented part of the system, while most of the packet forwarding should be performed by a fast and simple device.

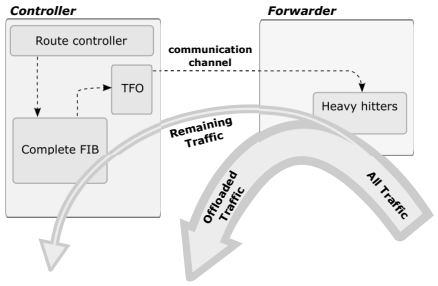
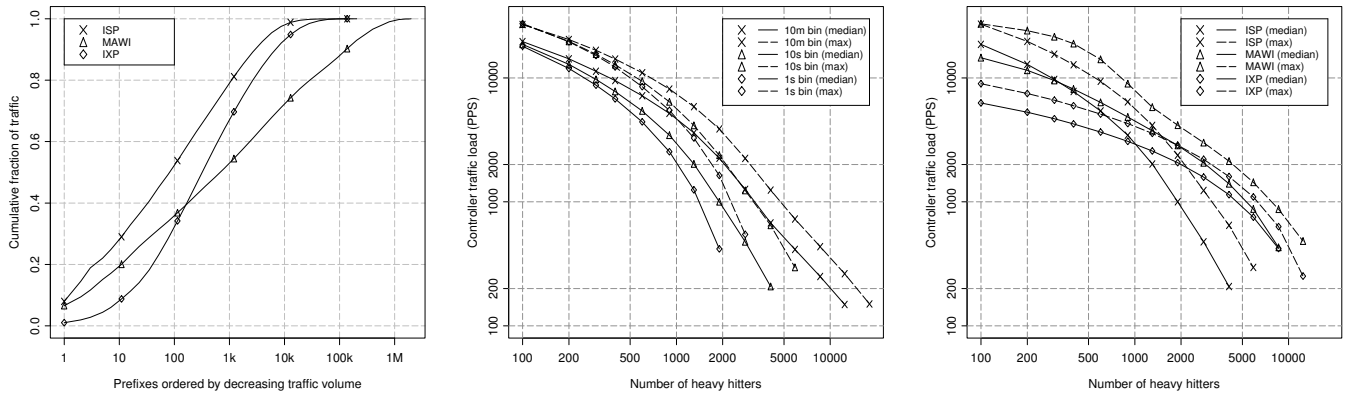


Figure 1: Traffic offloading system principle.

Figure 1 illustrates the principle controller-forwarder system architecture with the use case of an IP router. The controller speaks routing and signalling protocols in the same fashion as route controllers on existing routers. It can take advantage of the extensive CPU and memory resources of commodity PCs. Furthermore, the controller makes use of a forwarder, a device optimized for high-performance packet forwarding, to offload most of the traffic. Through a communication channel, the controller modifies the set of heavy hitters which are offloaded and handled by the forwarder. In this paper we investigate the feasibility of such an offloading system based on Internet traffic traces.

We first show that in principle, the achievable offloading gain is in the order of more than 90 percent when offloading the traffic contributed by 0.2 % of the flows. However, the volatility in heavy hitters incurs significant communication overhead, which is contrary to our design goals of keeping the forwarder simple. To address this challenge, we propose our heavy hitter selection strategy called Traffic-aware Flow Offloading (TFO), which relies on traffic statistics over multiple time scales. Previous work [19, 24] has identified the volatility of heavy hitters in Internet traffic as a challenge for traffic engineering, which also applies to our controller-forwarder context. In addition, we require that the benefit of introducing a forwarder is not counterbalanced by the complexity of the communication between the controller and the forwarder. Therefore, TFO trades off the overhead of modifying the set of heavy hitters against the expected increase in offloading gain.

*Rob Sherwood did the majority of the work while employed at Deutsche Telekom Inc. R&D Lab.



(a) Fraction of traffic captured per prefix. (b) Impact of bin size on controller load (ISP). (c) Controller load with 10s bins (all traces).

Figure 2: Feasibility experiments: IP prefix popularity (a) and baseline controller load (b, c).

Our results show that TFO is able to offload most of the traffic to the forwarder while keeping the number of modifications to the set of heavy hitters low, contrary to traditional route caching strategies [13]. This enables us to discuss both the benefits as well as challenges that Internet traffic properties pose for router design. The contributions of this paper are the following:

Traffic offloading: Based on traffic traces from a large European ISP, a large European IXP, and a transcontinental link we show the potential of traffic offloading and identify the challenges of heavy hitter selection strategies.

Traffic-aware Flow Offloading (TFO): We propose TFO, a heavy hitter selection strategy that leverages Zipf and the associated stability properties of heavy hitters across different time scales.

TFO and traditional caching: We evaluate TFO and compare it to traditional caching strategies. We observe a cross-over point. TFO outperforms traditional caching when the number of heavy hitters and the communication overhead have to be limited. However, when these constrains are relaxed traditional caching strategies might outperform TFO.

In addition, we release a library implementation of TFO as free software to the community [1], which we have used to implement an IP router by coupling a PC running a software router with an OpenFlow-enabled switch [17]. For a system description of our prototype we refer to [20]. The focus of this paper however lies in understanding the potential of traffic offloading, which we believe is crucial not only for our example prototype but for a wide range of high-capacity, programmable packet forwarding systems.

The rest of the paper is structured as follows. In Section 2 we revisit properties of heavy hitters including traffic share and churn. In Section 3 we first explore the limitations of traditional caching strategies and then propose and evaluate TFO. We present our system prototype, related work, and discuss long-term perspectives of our work in Section 4. We summarize in Section 5.

2. FEASIBILITY STUDY

In this section we perform feasibility experiments based on Internet traffic traces to understand the fundamental challenges and quantify the potential benefits of traffic offloading. We start by revisiting the properties of Internet traffic as seen from three different vantage points in the network. Then, we run evaluations under two optimistic assumptions: (i) future traffic is known and (ii) heavy hitter modifications are instantaneous. The results suggest a need to reduce the churn in the selected heavy hitters in order to limit the communication overhead. They also confirm previous results [24] which show that the variability of heavy hitters differs across ranks. Both observations shall be taken into account when designing a heavy hitter selection strategy such as TFO.

[1]TFO software implementation: `http://www.fibium.org/`

	ISP	IXP	Backbone
Network	residential	hundreds of ASs	transcontinental link
Speed	1 Gbps	1.5 Tbps	150 Mbps
Duration	2 days	4 days	3.5 days
Sampling	none	$1{:}2^{14}$	none

Table 1: Dataset details.

2.1 Datasets

We base our study on traffic traces from three different network locations (Table 1):

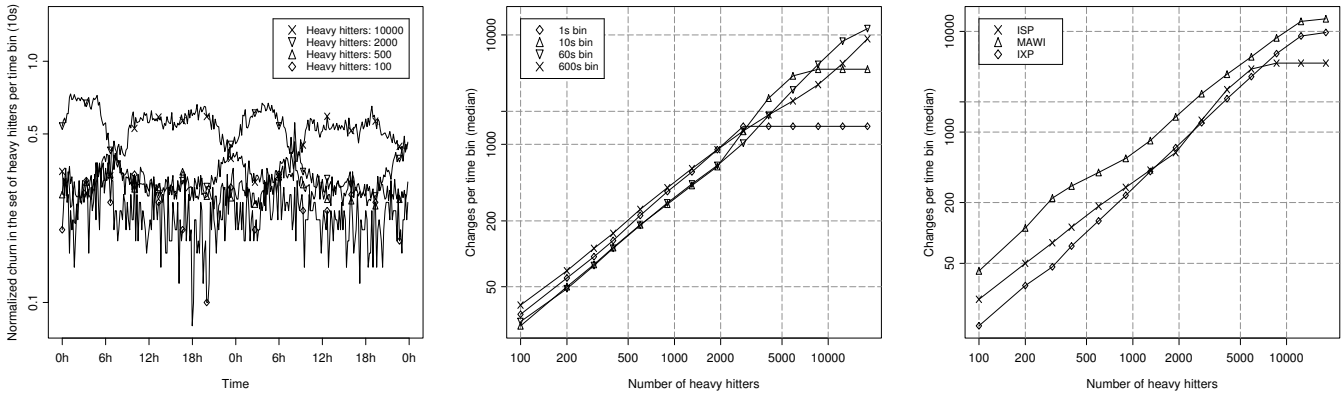
Residential ISP: The ISP trace relies on passive, anonymized packet-level observations of residential DSL connections collected at an aggregation point within a large European ISP. Overall, the ISP has roughly 10 million (4 %) of the 251 million worldwide broadband subscribers [18]. The monitor operated at a broadband access router connecting customers to the ISP's backbone. The trace started in August 2009 and covers 48 hours.

Transcontinental link: The traffic trace from a transcontinental link between Japan and USA also consist of passive, anonymized packet-level observations. The trace was collected in April 2010, covers 3.5 days, and is publicly available from the MAWI working group of the WIDE project [12].

European IXP: In contrast, the trace from a major European IXP consists of anonymized sFlow records which were captured in April 2011. The sFlow probes were configured to export sampled packets at a sampling ratio of $1{:}2^{14}$. Due to the sampling, some underestimation of small flows is possible, and therefore there may be some bias towards the heavy hitters [4, 7].

While anonymizing the ISP and IXP traces, we map each destination IP address to its longest-matching prefix based on publicly available BGP data. The same technique could not be applied to the trace from the transcontinental link, but fortunately the anonymization uses consistent scrambling of /24 prefixes [12].

These three datasets allow us to look at the problem of offloading from different perspectives: network location, network size,



(a) Churn over time (ISP). (b) Churn for different time bins (ISP). (c) Churn for the three traces.

Figure 3: Churn rate when using the bin-optimal strategy.

and time. This gives us confidence that our results can be generalized beyond these samples. Moreover, we note that our proposal exploits traffic properties that have already been observed in the past and are not expected to change, as the popularity of Internet flows is expected to retain its consistency with Zipf's law.

2.2 Consistency with Zipf's law

Since we aim at leveraging the Zipf properties of Internet traffic for traffic offloading, we first confirm that the three traces are consistent with Zipf's law.

Figure 2(a) shows the relative share of the total traffic as a function of the number of top destination prefixes. These results confirm that all three traces exhibit the Zipf property—a limited number of prefixes account for a majority of the traffic. However, one difference between the traces lies in the number of heavy hitters whose traffic has to be aggregated to capture a desired fraction of the traffic. For example, the most popular 10 destination prefixes of the residential ISP trace and the transcontinental trace already capture more than 20 % of the total traffic, but less than 10 % in the case of the IXP. With 1,000 prefixes one can capture more than 50 % of the traffic. This holds for all three traces. For lower ranks the amount of traffic that each prefix is contributing decreases rapidly. The next 1,000 prefixes combined capture less than 5 % of the traffic. Another difference is that the transcontinental link sees traffic for a larger number of prefixes. However, the least popular half of the prefixes contribute less than 1 % of the traffic. This may be an artifact of the anonymization as described above and the resulting assumption of a flat /24 address space.

As we will discuss later, the controller typically cannot afford delegating the responsibility for too many flows to the forwarder, because of the associated communication overhead. The most relevant range seems to lie between a few hundred to a few thousand heavy hitters. The above results confirm previous work [13, 22, 24, 11], which has shown that it is possible to offload most of the traffic with a limited number of heavy hitters. However, these works do not study how to derive a heavy hitter selection strategy which trades off the expected benefit of modifying the heavy hitters against the extra work of applying those changes. Instead, previous work has focused on dynamic heavy hitter detection mechanisms [27, 5].

2.3 Remaining load at the controller

In our context, traffic offloading aims at limiting the rate at which the controller needs to forward packets by offloading most of the traffic load to a specialized forwarding device. By design, the controller is optimized for running complex routing protocols but often it does not satisfy the requirements of packet forwarding. Some known limitations of controllers include the available forwarding bandwidth and delay. Our goal is to limit the amount of traffic the controller has to handle in order to avoid long packet queues and packet loss. As a baseline, we now quantify the amount of traffic the controller has to handle based on our traffic traces.

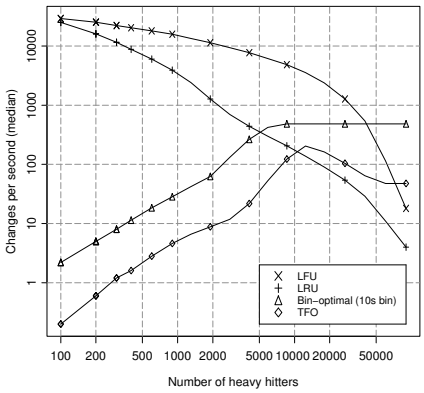
For this purpose, we rely on a practically infeasible heavy hitter selection strategy called bin-optimal. The bin-optimal strategy assumes that future traffic is known, and that the set of heavy hitters to be used on the forwarder can be updated instantly. The bin-optimal strategy uses fixed time bins. We choose 1s, 10s, and 10 minutes as time bins to capture the varying stability properties of heavy hitters across different time scales. The bin-optimal strategy selects for each time bin the top n heavy hitters, where n is the number of entries that the forwarder keeps.

Figures 2(b) and 2(c) show the traffic load in packets per second (PPS) which remains at the controller vs. the number of heavy hitters which are delegated to the forwarder, on a log-log scale. Overall, we see that even for small values of n, in the order of a few thousands, the traffic load at the controller is relatively low. When taking the ISP trace as an example, with $n = 1,000$ the median traffic load at the controller is 3,400 PPS (with a maximum of 6,400 PPS) which corresponds to a relative reduction in load of 92 %. With shorter time bins the offloading gain can be further increased. Figure 2(b) shows both the median as well as the maximum traffic rate at the controller for the three considered time bins.

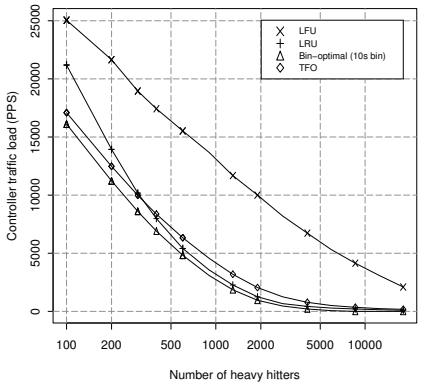
Figure 2(c) compares the controller traffic load for all three traces under different numbers of n and fixed 10s time bins. We observe that with all considered traces, the offloading gain increases rapidly with increasing n. All three traces incur only up to around 1,000 PPS at the controller when offloading the top 5,000 prefixes.

2.4 Churn in heavy hitters

Heavy hitters enable significant offloading but are also challenging because of their dynamics [19, 23, 24, 26, 3]. We examine the impact of variability in the heavy hitters when delegating prefixes to the forwarder. The bin-optimal strategy relies on the foreknown heavy hitters for each time bin. In this case, the churn in the heavy hitters stems entirely from the natural traffic variability. We now quantify the corresponding number of changes to the set of heavy hitters between consecutive time bins, as incurred by the bin-optimal strategy.



(a) Churn rate in comparison (ISP).



(b) Controller traffic load (ISP).

Figure 4: Controller load and churn for different strategies (ISP).

Figure 3(a) shows the fraction of heavy hitters that are modified for time bins of 10s, as a function of time for the ISP trace. For small values of n such as a few hundreds we observe churn in the order of 15 %. When increasing n up to a few thousands, around half of the heavy hitters are modified. As expected, we notice that changes among the top ranked prefixes occur much less often than among the lower ranked ones. We also point out that the churn is subject to a time of day effect. Interestingly, the direction of the time of day effect inverts as the number of heavy hitters is increased. This is due to the reduced variability and traffic volume during the night.

As shown on Figure 3(b) for the ISP trace, relying on larger bin sizes does not significantly reduce churn. Moreover, on Figure 3(c), we see that even though our network observation points are quite diverse, similar trends apply with regards to churn. We explain the larger churn in case of the MAWI trace again with the assumption of a flat /24 address space.

3. TRAFFIC OFFLOADING WITH TFO

In Section 2, based on Internet traffic observations, we have identified the main challenges behind heavy hitter selection strategies for traffic offloading: maintaining a high offloading ratio while limiting the changes to the set of heavy hitters. In this section we first examine how well traditional caching strategies perform in the context of the offloading problem (Section 3.1). Traditional caching strategies are not appropriate in our context due to their high churn

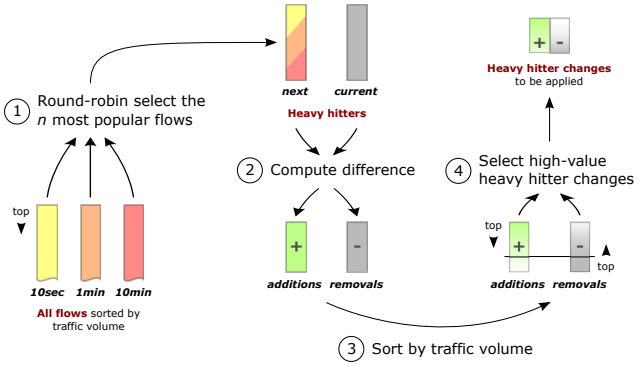


Figure 5: Traffic-aware Flow Offloading.

and the fast reaction times they require. This leads us to develop and evaluate our heavy hitter selection strategy called Traffic-aware Flow Offloading (TFO). Based on our traffic traces, we show that TFO achieves low churn, one order of magnitude less than the bin-optimal strategy from Section 2.3, while achieving a similarly high offloading gain.

3.1 Limitations of traditional caching

Traditional caching techniques react on individual misses: when a cache miss occurs, the missing object is fetched and then placed into the cache. During the time interval between the event of a cache miss and the completion of a cache update, incoming packets need to be buffered or will lead to packet re-ordering or loss.

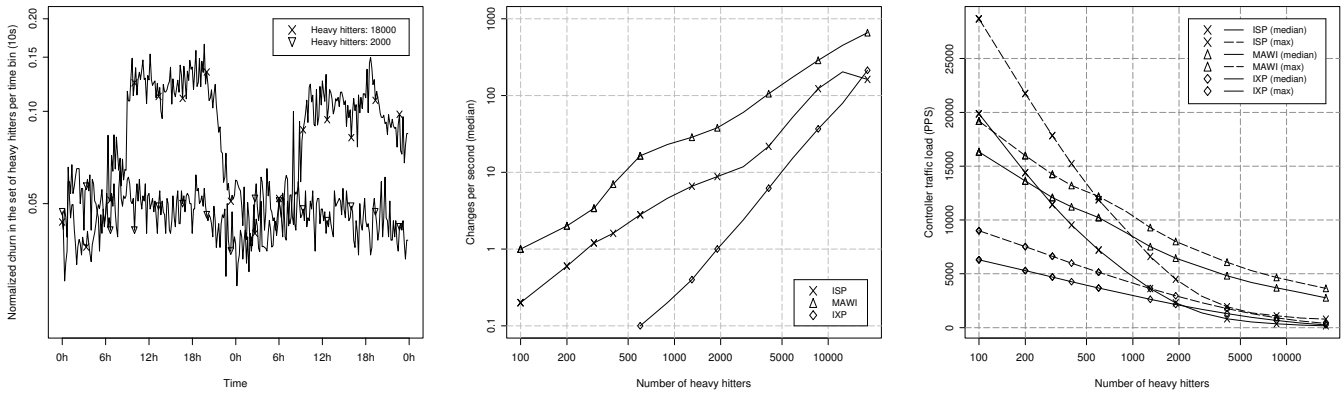
The main difference among traditional caching techniques lies in the strategy to determine which cache entry to evict and replace upon a cache miss. Two common cache replacement strategies are least recently used (LRU) and least frequently used (LFU). We implement both strategies based on the optimistic assumption that cache replacement is instantaneous, to be favorable toward their offloading gain.

The curves for LRU and LFU in Figure 4(a) show the amount of heavy hitter changes per second as a function of the number of heavy hitters, n. LRU outperforms LFU, consistent with previous work on route caching [13]. Figure 4(a) also shows the results for the bin-optimal strategy based on 10s bins. Both LRU and LFU imply a significantly higher churn rate as compared to bin-optimal for up to a few thousand of heavy hitters. For larger values of n, traditional caching can outperform bin-optimal in churn rate, this is when n gets close to the total number of active flows. With regards to the remaining traffic load at the controller as shown on Figure 4(b), LFU exhibits the worst results. For $n > 200$, all strategies except LFU result in similar remaining loads at the controller.

In summary, these results indicate that the traditional caching strategies LRU and LFU suffer from a major limitation, namely the high rate at which heavy hitters need to be changed, especially for values of n ranging up to a few thousands, i.e., the range of n in which offloading is likely to be relevant.

3.2 Traffic-aware Flow Offloading

In this section, we propose our heavy hitter selection strategy: Traffic-aware Flow Offloading (TFO). The design criteria of TFO stem from our observations in the earlier sections. The goal is to handle most traffic at the forwarder while limiting the number of changes to the heavy hitters. The idea behind TFO is to leverage the stability properties of heavy hitters, which have been shown to depend on their rank [24]. Therefore, we maintain traffic statistics at multiple time scales which enable TFO to take into account short-term as well as long-term trends for heavy hitter selection. To



(a) Churn over time (ISP). (b) Churn for the three traces. (c) Controller traffic load (all traces).

Figure 6: TFO evaluation.

compute a new set of heavy hitters, we further take into account the set of heavy hitters which are currently in place, in order to trade-off the cost of changing the set of heavy hitters with the expected increase in offloading gain.

Figure 5 illustrates the four different steps of TFO. We start by taking top heavy hitters from each time scale in a round-robin manner, until we reach a pre-defined value of n. The round-robin pre-selection gives equal chances to heavy hitters that are highly ranked at different time scales. Second, we compute the difference between the current heavy hitters at the forwarder and the set of heavy hitters from the first step. This gives us two sets of heavy hitters: those to be installed and the ones to be removed. The last two steps are key to reduce the churn while ensuring fast reaction times. In the third step, we sort these two sets into lists according to their traffic share in inverse orders, with respect to the smallest time scale. Finally, in the fourth step, we pick such modifications in which the additions are ranked much higher than the removals, e.g., they differ in traffic share at least by a factor of 2. This fourth step gives priority to the heavy hitters which are already in place, effectively reducing the churn. Consequently, the chosen factor affects the trade-off between offloading gain and communication overhead. In future work, we want to study how to optimally determine this factor. Note, that sudden high-ranked flows can (and should) still enter the set very quickly, i.e., in one round of set computation.

3.3 Evaluation of TFO

We now evaluate the performance of TFO based on our three traffic traces. Figure 4(b) shows, as a function of n, the remaining traffic load at the controller for the ISP trace. The offloading effectiveness of TFO is marginally worse compared to the bin-optimal strategy. However, the bin-optimal strategy involves a churn rate which is about an order of magnitude higher as compared to TFO, see Figure 4(a). When comparing the evolution of churn over time of the bin-optimal strategy on Figure 3(a), against TFO on Figure 6(a), we observe that TFO changes only less than 10 % of the heavy hitters between any two time bins whereas bin-optimal exceeds 70 %, for the ISP trace with $n = 2,000$. When letting n grow to values as large as 18,000, we observe that the rate of churn varies heavily and follows a strong time of day pattern in cases of TFO and bin-optimal. Smaller values of n, such as 2,000, are less sensitive to time of day effects and thus show only limited variations. We conclude that an offloading system that aims too high in its offloading gain will face strong churn rate variations. The key lies in understanding the trade-off when choosing n: On the one hand, n must be large enough such that the heavy hitters can capture enough traffic to retain a low enough traffic load at the controller. On the other hand, one should strive to keep n as small as possible to keep the associated rate of churn and its variation low.

Figure 4(a) also compares TFO with the traditional caching strategies LRU and LFU. For $n < 10,000$, TFO outperforms both LRU and LFU in terms of churn rate. However, at around $n = 10,000$ we observe a cross-over point at which LRU starts to outperform TFO. At such large values of n, the rate of churn also exhibits strong variations over time.

Figure 6(b) presents the median number of changes to the heavy hitters as a function of n for the three traces. We observe an impact of traffic aggregation on the rate of churn: the MAWI trace leads to the highest churn rate followed by the ISP trace, and finally the IXP one with the least amount of churn. Also, the top 500 heavy hitters in the IXP trace are so stable that their median churn is 0 over the four days. This result is important with respect to the scalability of the offloading problem, as our results suggest that the higher the traffic aggregation, the lower the churn.

Figure 6(c) compares the traffic load at the controller for the three different traces when using TFO. The figure shows the median and maximum controller load as a function of n, in log-linear scale. The ISP trace gains most with increasing n. This is consistent with Figure 2(a), where the top heavy hitters for the ISP capture a larger fraction of the total traffic as compared to the other traces. For $n = 6,000$ we observe an offloading gain of more than 99 %, resulting in a median of only 500 PPS at the controller.

The MAWI trace has the widest CDF curve on Figure 2(a), which leads to the worst offloading gain except for very small n. Due to the anonymization technique of the MAWI trace, we consider a flat /24 address space which potentially slices heavy hitters into many /24 prefixes. Hence, the MAWI results are penalized in terms of offloading gain per heavy hitter. However, they still result in a very limited packet rate at the controller. For example, with $n = 6,000$, TFO still achieves an offloading gain of 86 %, leaving only 4,200 PPS (median) at the controller.

In contrast to MAWI, the IXP trace presents a very smooth S-shaped CDF curve on Figure 2(a). This stems from the level of aggregation at the observation point, which represents traffic exchanged between hundreds of ASs. Figure 6(c) shows that for most values of n, the IXP trace leads to the lowest controller load compared to the other traces. However, as mentioned earlier, due to the low packet sampling rate of this trace we expect a bias in favor of the observed traffic share of the heavy hitters. For $n = 6,000$,

TFO achieves an offloading gain of 90 %, leaving less than 1k PPS (median) at the controller.

Note that real traffic traces, especially those as large as the IXP one, contain certain types of attacks such a DDoS, scans, and other types of anomalies. Our results show that despite the presence of such events, neither the churn nor the offloading gain seem affected, even when considering the maximum churn and maximum traffic load at the controller.

4. DISCUSSION AND RELATED WORK

We revisited the Zipf property of Internet traffic and derived a traffic offloading strategy that achieves high offloading gain while limiting the overhead caused by churn in the heavy hitters. With recent work which aims to improve the forwarding performance of software routers [9] as well as their port density [6], we believe that traffic offloading is a viable alternative to traditional router design.

Currently, routers maintain large numbers of forwarding entries in specialized memory for fast per-packet lookups. Routers require ever increasing amounts of such specialized memory, which typically is either costly and / or energy hungry. Furthermore, the rate at which the forwarding entries need to be changed is significant, especially due to routing protocols dynamics [15, 25, 14]. For example, reported BGP update rates are in the order of tens or hundreds per second on average with peaks of several thousands per second, for about $350,000$ prefixes. Even with such a limited churn relative to the number of prefixes, the implication of BGP updates on the data plane can be an issue [14, 1]. Compared to today's BGP update rates, the dynamics in heavy hitters is much lower, both in relative and absolute numbers. Furthermore, BGP updates are expected to have limited impact on the heavy hitters as popular prefixes have been shown to be stable [10].

Our work shows that an offloading system has potential to improve router designs, by further decoupling the tasks of the route controller with those of the forwarding engines. The main advantage of the offloading approach is to limit the interactions between the control and data planes to what is strictly necessary from the viewpoint of the traffic. By taking advantage of the forwarding capabilities of programmable forwarding engines such as OpenFlow [17], NetFPGA [16], and ATCA [2], we believe that IP router designs which leverage open source routing software and traffic offloading may challenge existing router designs.

As we plan to apply traffic offloading and TFO to other domains, we intend to incorporate TFO into FlowVisor [21] to complement its slicing policy.

5. SUMMARY AND FUTURE WORK

Internet traffic follows Zipf's law: a small fraction of flows capture most of the traffic. We propose to exploit this property to offload most of the traffic from a complex controller to a simple forwarder. We show that the main challenge lies in achieving a high traffic offloading gain while limiting the churn, and propose a heavy hitter selection strategy called Traffic-aware Flow Offloading (TFO). We perform simulations of TFO on real traffic traces. The results highlight the effectiveness of the strategy used in TFO and suggest that traffic offloading should be considered to build feasible alternatives to current router designs. In future work, we want to study the impact of parameters on TFO such as the chosen time scales. We also plan to evaluate TFO under different traffic loads with flows of granularities other than BGP prefixes. Another future study will focus on possible attacks against the offloading system and adequate defense mechanisms.

6. REFERENCES

[1] AGARWAL, S., CHUAH, C., BHATTACHARYYA, S., AND DIOT, C. The Impact of BGP Dynamics on Intra-Domain Traffic. In *ACM SIGMETRICS* (2004).

[2] AdvancedTCA Specifications for Next Generation Telecommunications Equipment. `www.advancedtca.org`.

[3] CLAFFY, K. C., AND BROWNLEE, N. Understanding Internet traffic streams: Dragonflies and Tortoises. *IEEE Comm. Mag.* (2002).

[4] COHEN, E., DUFFIELD, N., KAPLAN, H., LUND, C., AND THORUP, M. Algorithms and estimators for accurate summarization of internet traffic. In *ACM IMC* (2007).

[5] CORMODE, G., KORN, F., MUTHUKRISHNAN, S., AND SRIVASTAVA, D. Finding hierarchical heavy hitters in data streams. In *VLDB* (2003).

[6] DOBRESCU, M., EGI, N., ARGYRAKI, K., CHUN, B., FALL, K., IANNACCONE, G., KNIES, A., MANESH, M., AND RATNASAMY, S. RouteBricks: exploiting parallelism to scale software routers. In *ACM SOSP* (2009).

[7] ESTAN, C., KEYS, K., MOORE, D., AND VARGHESE, G. Building a better netflow. In *ACM SIGCOMM* (2004).

[8] FANG, W., AND PETERSON, L. Inter-as traffic patterns and their implications. In *IEEE Global Internet* (1999).

[9] HAN, S., JANG, K., PARK, K., AND MOON, S. Packetshader: a gpu-accelerated software router. In *ACM SIGCOMM* (2010).

[10] J. REXFORD, J. WANG, Z. X., AND ZHANG, Y. BGP routing stability of popular destinations. In *ACM IMC* (2002).

[11] JAIN, R. Characteristics of destination address locality in computer networks: A comparison of caching schemes. *Computer Networks and ISDN* (1989/90).

[12] K. CHO, K. M., AND KATO, A. Traffic data repository at the wide project. In *USENIX ATC, Freenix track* (2000).

[13] KIM, C., CAESAR, M., GERBER, A., AND REXFORD, J. Revisiting route caching: The world should be flat. In *PAM* (2009).

[14] KUSHMAN, N., KANDULA, S., AND KATABI, D. Can you hear me now?!: it must be BGP. *ACM CCR* (2007).

[15] LABOVITZ, C., MALAN, G., AND JAHANIAN, F. Internet Routing Instability. In *ACM SIGCOMM* (September 1997).

[16] LOCKWOOD, J. W., MCKEOWN, N., WATSON, G., GIBB, G., HARTKE, P., NAOUS, J., RAGHURAMAN, R., AND LUO, J. NetFPGA–an open platform for gigabit-rate network switching and routing. In *Proc. of IEEE MSE* (2007).

[17] MCKEOWN, N., ANDERSON, T., BALAKRISHNAN, H., PARULKAR, G., PETERSON, L., REXFORD, J., SHENKER, S., AND TURNER, J. OpenFlow: enabling innovation in campus networks. *ACM CCR* (2008).

[18] OECD. Broadband Portal. www.oecd.org/sti/ict/broadband, 2008.

[19] PAPAGIANNAKI, K., TAFT, N., AND DIOT, C. Impact of flow dynamics on traffic engineering design principles. In *IEEE INFOCOM* (2004).

[20] SARRAR, N., FELDMANN, A., UHLIG, S., SHERWOOD, R., AND HUANG, X. Towards hardware accelerated software routers. In *ACM CoNEXT Student Workshop* (2010).

[21] SHERWOOD, R., GIBB, G., YAPA, K., CASSADO, M., APPENZELLER, G., MCKEOWN, N., AND PARULKAR, G. Can the production network be the test-bed? In *OSDI* (2010).

[22] SHYU, W., WU, C., AND HOU, T. Efficiency analyses on routing cache replacement algorithms. In *IEEE ICC* (2002).

[23] WALLERICH, J., DREGER, H., FELDMANN, A., KRISHNAMURTHY, B., AND WILLINGER, W. A methodology for studying persistency aspects of Internet flows. *ACM CCR* (2005).

[24] WALLERICH, J., AND FELDMANN, A. Capturing the variability of internet flows across time. In *IEEE Global Internet* (2006).

[25] WANG, L., ZHAO, X., PEI, D., BUSH, R., MASSEY, D., MANKIN, A., WU, S., AND ZHANG, L. Observation and Analysis of BGP Behavior under Stress. In *ACM IMW* (2002).

[26] ZHANG, Y., BRESLAU, L., PAXSON, V., AND SHENKER, S. On the Characteristics and Origins of Internet Flow Rates. In *ACM SIGCOMM* (2002).

[27] ZHANG, Y., SINGH, S., SEN, S., DUFFIELD, N., AND LUND, C. Online identification of hierarchical heavy hitters: algorithms, evaluation, and applications. In *ACM IMC* (2004).