


MACIEJ CZEKAJ  
ERNEST JAMRO 

## FLOW CACHING EFFECTIVENESS IN PACKET FORWARDING APPLICATIONS

**Abstract** *Routing algorithms are known to be potential bottlenecks for packet processing. Network flow caching can function as a general acceleration technique for packet processing workloads. The goal of this article is to evaluate the effectiveness of packet flow caching techniques in high-speed networks. The area of focus is the data distribution characteristics that lead to the effectiveness of caching network flows (connections). Based on a statistical analysis and simulations, the article sets the necessary conditions for the effective use of caches in packet forwarding applications. Public domain network traces were examined and measured for data locality. Software simulations show a strong correlation between the flow packet distance metrics and the cache hit rate.*

**Keywords** computer networks, caching, SDN

**Citation** Computer Science 20(2) 2019: 145–163

## 1. Introduction

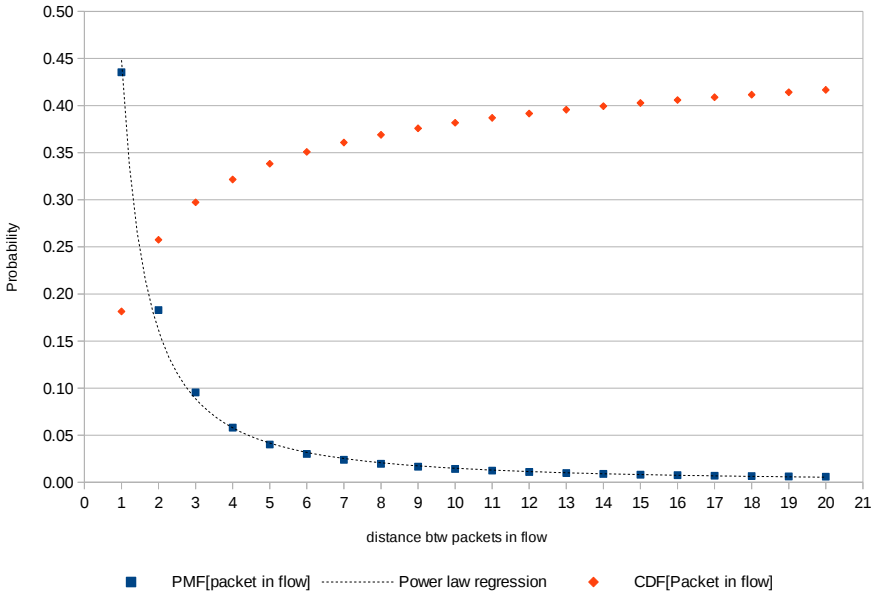
Routing algorithms are known to be potential bottlenecks for packet processing if the routing table is large [3]. The principle of network packet locality allows us to accelerate the task of routing, which is based on pattern-matching the destination subnet by a simple dictionary lookup. Moreover, if the packet processing pipeline is longer (e.g., after routing, when there is a policing stage, QoS stage, etc.), then caching can accommodate for all of the stages, effectively bypassing the whole packet processing procedure with a simple hash table lookup. Of course, only a subset of traffic can be accelerated by the cache, and the cache hit ratio should be high in order to justify the effort.

Caching is a common technique for accelerating data access that is known to work in various fields; e.g., CPUs, storage devices, Web serving, Content Delivery Networks, etc. The main notion that explains the usefulness of caching is called the *locality principle*. It is based on the observation that data placement on the Internet is highly non-uniform and most likely follows a power law such as Zipf's [1]. In other words, the majority of the data is placed in relatively few locations, forming a good opportunity for employing various types of caches in the traffic path.

In addition, the power law distribution function is described as *self-similar* or *scale-free*, which suggests that the statistical processes behind the data pattern have long-term memory [7]. This characteristic can be observed on various levels; e.g., from web page popularity [1] through TCP connections [34] to IP traffic [9]. The *scale-free* data phenomenon forms a natural limit to the effectiveness of caching [28], as some fraction of the traffic is not going to be local.

This paper is specifically focused on packet forwarding applications such as switches, routers, IP gateways, or software-defined network overlays. Translating the general problem of caching to packet forwarding, the locality principle applies to the packet arrival time or network address distribution. Given the current packet arriving at the network interface, there is a high probability that many subsequent packets are going to be from the same network flow. The traffic is bursty in nature [24], and a single burst contains packets from relatively few flows.

Figure 1 presents the statistics from an HTTP traffic sample [23]. The  $x$  axis presents a distance between packets from the same flow measured in the packets; e.g.,  $x = 2$  means the probability that the second packet is from the same flow. On the  $y$  axis, there are two data series: a histogram (or sampled probability mass function, PMF) of the flow packet distance, and the cumulative distribution function (CDF) computed from the given histogram. The histogram is well-fitted to power law function  $f(x) = \alpha x^{-\lambda}$ . The network flow is defined as 5-tuple:  $(IP_{src}, IP_{dst}, Port_{src}, Port_{dst}, protocol)$ . Given this interpretation, the CDF chart shows a c.a. 40% likelihood that the next packet is going to be from the same flow, a 60% chance that at least one of the two packets meets the criteria, and as high as 80% that at least one of the five packets is from the same flow.



**Figure 1.** Network traffic histogram (PMF) fitted to  $\alpha x^{-\lambda}$  and its CDF

The definition of flow is application-dependent, but unique flow is typically defined by a combination of addresses and ports taken from Layers 2 through 4 of the OSI model (e.g., Ethernet, IP, TCP). As it turns out from further analysis, the specific definition of the flow as either an IP flow (IP source and destination address), 5-tuple (TCP or UDP connection), or any other does not affect the overall picture. Given the most narrow definition of flow selected from the L2, L3, and L4 address fields, the same traffic property holds. Choosing any definition that is less “constrained” (contains fewer fields) can only improve the performance.

Caching as a means for speeding up packet processing is popular in both academia and the developer community [31]. On the other side, flow caching is subject to strong opposition due to its non-deterministic behavior and security issues [17]. Certainly, it is not a panacea to all networking challenges. Therefore, it is greatly desired to evaluate the risks and benefits associated with employing a cache as a front-end to a forwarding application. This paper focuses on the potential benefits, but the risks mentioned in [17] should be properly addressed for any real-world deployment.

While trying to improve the performance of a packet forwarding application, a designer meets following dilemmas:

- How to tune the cache size if the deployment site is not known (or could be any site for that matter)?
- How to assess cache effectiveness against potential data patterns that can be met “in the field”?

The main contribution of this paper is to develop a method for predicting cache effectiveness against a given traffic profile and confirm it by simulation results. The effectiveness becomes a function of cache size, but it also depends on traffic complexity benchmarks. It builds upon the results obtained from previous publications that only addressed this issue in fragments. The metrics introduced in the paper are developed by contrasting several traffic data sources used in academic research and running an independent set of simulations to confirm their predictive power.

## 1.1. Paper organization

This paper is organized as follows. Section 2 provides a tour of the scientific literature on the subject of caching and its effectiveness. Both hardware and software solutions are taken into account, as caching is not limited to software alone. Section 3 shows the original measurements and statistics based on publicly available packet traces typically used for research purposes. Simulations of the cache performance are discussed in Section 4. Finally, Section 5 gives a summary of the paper.

## 2. Related work

Caching as a way to improve the performance of a routing gateway has been known for a few decades (as extensively discussed in [15]). This work identifies the key ingredients of a successful approach to the problem: the measurement of inter-packet arrival time for identically addressed packets and a time of last reference chart similar to Figure 1. The paper gives promising simulation results: a fully-associative cache with as few as 20 cache entries had an over 90% hit ratio. In the paper, the cache was only indexed by an IPv4 destination address. Note that the total number of unique addresses for the gateway in the 24h period was 1250, so cache capacity accounted for merely 1.6% of all of the IPv4 addresses used in the network.

A deeper analysis of cache performance with a focus on cache replacement policies can be found in [20]. The cache is also indexed by a destination IP address, so the results are expected to be comparable (except for differences in the data set). The WAN traffic trace used in [20] appears to have a much worse data locality, which results in cache sizes that approach 10,000 entries and more. Also, there are stark differences between the performance of different cache replacement policies. There are two main takeaway points from this paper. First and foremost, the data locality may vary across different Internet links. The second point is that the Least Recently Used (LRU) replacement policy can be suboptimal; it is worth seeking alternatives when the experiments show low cache performance. This is somewhat in line with the observations from the CPU data cache research, where LRU can cause so-called “performance cliffs” (drastic hit rate reductions for working sets larger than the threshold value) for data sets that slightly exceed the cache capacity [4].

An attempt to employ a cache in an Application Specific Integrated Circuit (ASIC) designed for ethernet switching is discussed in [12]. The simulation results

for a campus LAN network and datacenter LAN show high data locality and a high cache hit ratio (60–70%) – even for a simple 4-entry cache. This may be explained by the fact that LAN traffic is less diverse than what can be found in WAN links. The very nature of WAN links assumes thousands of connections originating from equally as many hosts. This hypothesis will be explored further in the paper.

Another hardware approach discussed in [26] places a cache component onto a packet processing FPGA board. The cache functions as an offload to the main packet processing engine, which leads to either a reduction in the utilization of packet processor or, conversely, an increase in the overall throughput of the system. The experiments in [26] were carried out using WIDE project [38], backbone router traces. The caching approach resulted in a 60–90% cache hit ratio for a 4K 4-way set associative cache with an LRU replacement policy. Unfortunately, cache size was not a variable in that experiment, so it is difficult to assess the hit ratio as a function of cache size.

As an emerging trend, SDN places a strain on both the CPU and networking hardware. In [19], we see a hybrid approach for accelerating the SDN OpenFlow processing by using a hardware cache combined with software switches. The hardware cache module keeps track of the OpenFlow rules [22], which correspond more to an IP subnet than to a single IP flow (as can be found in previously discussed publications). Nonetheless, an SDN controller can store tens of thousands of hierarchical OpenFlow rules, which is beyond the capabilities of hardware switches. On the other side, hardware switches use Ternary Content Addressable Memory (TCAM) to process all of the rules in parallel, which makes it a constant time operation as opposed to a multi-dimensional tree search [16].

The result of using a small TCAM cache in front of a large software-based OpenFlow database gives very optimistic results. A 125-entry TCAM can serve 70–80% of the traffic.

### 3. Measurements

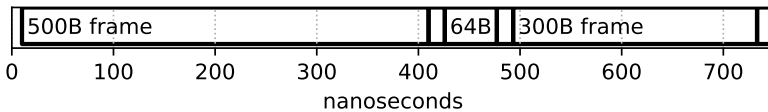
#### 3.1. Methodology

Among the freely available data sources, four popular representatives have been selected:

- Lawrence Berkeley National Laboratory Enterprise Tracing Project (LBNL) [21,27]. This is traffic collected from a single site’s internal network.
- Naples University traffic traces [13,14,23]. This is WWW traffic from the university’s Internet gateway.
- Waikato Internet Traffic Storage archives from WAND research group (WAND) [11,39]. The traffic used in the paper comes from New Zealand’s ISP core router.
- MAWI Working Group Traffic Archive (MAWI) [8,38]. This is the most recent traffic coming from Japan’s ISP’s backbone link.

The selection criteria for the traffic traces is their usage in other research so that the results can be compared. Also, they represent a wide spectrum of traffic profiles, which allows us to make general statements about network traffic.

For the sake of the measurements, the network flow is defined as a 5-tuple ( $IP_{src}, IP_{dst}, Port_{src}, Port_{dst}, protocol$ ) constructed from the inner IP header, which is consistent with the *NetFlow* standard used by network infrastructure devices [10]. The same definition of flow is used by the network gateways [35] and SDN controllers [5, 6], so the conclusions from this paper are relevant to engineering problems found in the data center network infrastructure. The definition of flow accounts for both IPv6 and IPv4, and the port numbers are from the TCP and UDP connections. In case there is no Layer 4, the respective fields are zeroed so all IP-only flows that do not have a transport layer are merged. Using an inner IP header ensures that IP overlays such as VXLAN or MPLS/GRE are decapsulated. This is important, as most SDN software is focused on processing inner flows. Obviously, IPSec tunnels cannot be treated this way, but IPSec processing is typically done at the gateway or network edge level; therefore, traces taken at the gateway should account for this traffic class.



**Figure 2.** Timing diagram of Ethernet frames in 10Gb/s link

Since this paper is primarily focused on packet forwarding and SDN applications, it is important to select the measurement method that is most appropriate for this field. In the case of a router, switch, firewall, or even a TCP/IP stack, the amount of work is proportional to the number of packets, not the network bandwidth or payload bytes. The latter is used in data processing workloads such as deep packet inspection [29] or encrypted traffic such as IPSec [33], where the payload is subject to intensive processing.

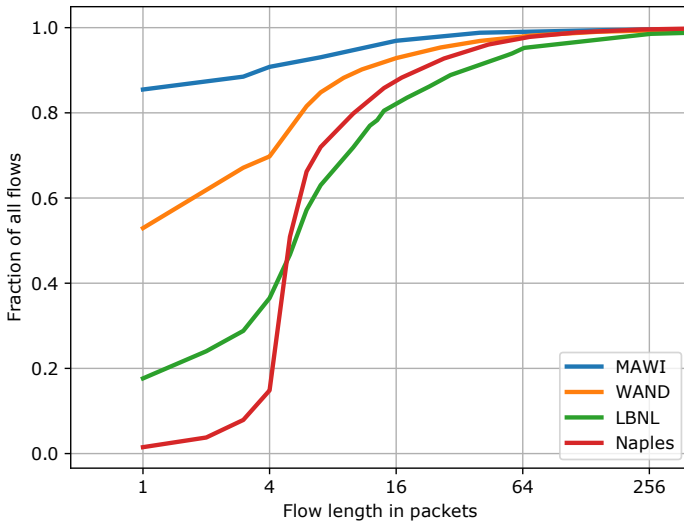
The reason for this distinction is based on the fact that packet forwarding only inspects network headers, which can be considered to be a fixed size. A typical Internet packet header can be as small as 42 bytes (UDP packet) or reach 100 bytes or more in the case of tunnels or excessive IPv6 options. The payload that follows is ignored (passed-through) by the network infrastructure and processed only at the endpoints (except for encrypted tunnels). Ethernet is a serial medium, where transmission time plays a major role. For instance, a 10Gb/s interface, such as XGMII, transmits 32 bits of data in one cycle [18] and this cycle lasts 3.2 ns. Figure 2 shows the timing diagram of the Ethernet frame being received by such a network interface. For a 10Gb/s link, even the smallest frame takes 48 ns to transmit. Each frame is preceded by a preamble (8B) and followed by an inter-frame gap (12B), so the transmission time adds up to 67.2 ns. The typical maximum transmission unit (MTU)

of IP networks is 1500B (counted in the IP layer), so a frame is 1518B after adding the Ethernet header and Frame Check Sequence (FCS). This amounts for a 1200-ns transmission time for the MTU. Despite the difference in payload length and transmission time, the smallest and the largest packets are considered to be an equal load for the packet forwarding engine (hardware or software alike) since the forwarding operation is only based on the limited-size header.

The statistical results presented below were produced by a custom traffic analysis program suite created for the purpose of the publication. The traffic analysis part used the PcapPlusPlus library [30], and the statistical processing (e.g., regressions) was programmed in NumPy [25].

### 3.2. Results

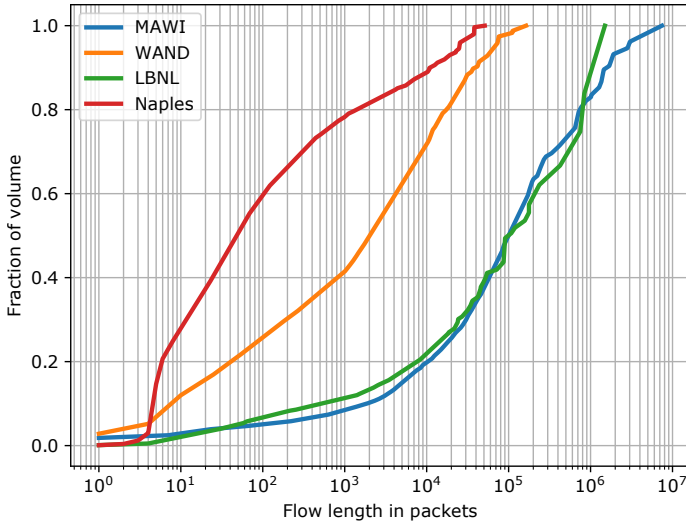
Figure 3 represents the flow length breakdown measured in packets for all traces. This is a cumulative distribution function (CDF), so point  $(x, y)$  on the chart means that fraction  $y$  of all flows have a length of  $x$  or less; e.g., in a WAND packet trace, c.a. 70% of all network flows have no more than four packets. The curved shapes differ among traces in a significant manner. MAWI traces are notorious for extremely short flows (more than 80%), but so do the others (except for the WWW traffic from Naples).



**Figure 3.** CDF of flow length in packets

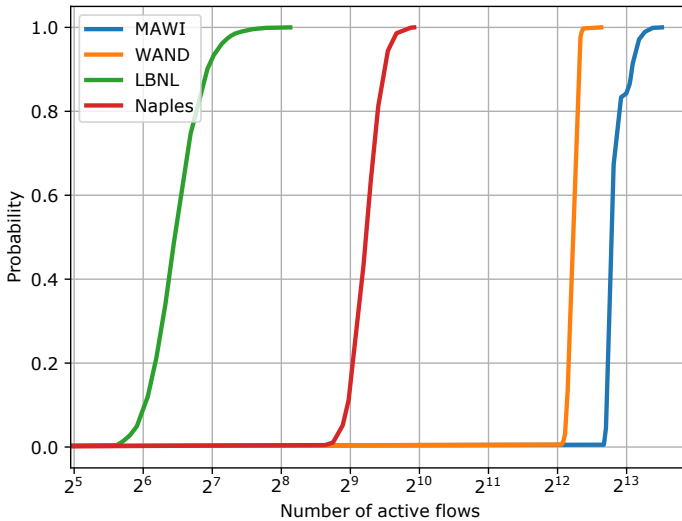
Despite the fact that most of the flows are short, the bulk of the traffic remains in longer flows (which is shown in Figure 4). For the same MAWI trace that is dominated by short flows, they represent only a tiny fraction of the overall volume.

Notably, 90% of all packets from MAWI belong to longer connections (more than 100 packets). This supports the locality principle, but only partially, since the flows must also be localized in time.



**Figure 4.** CDF of flow length in packets weighted by traffic volume

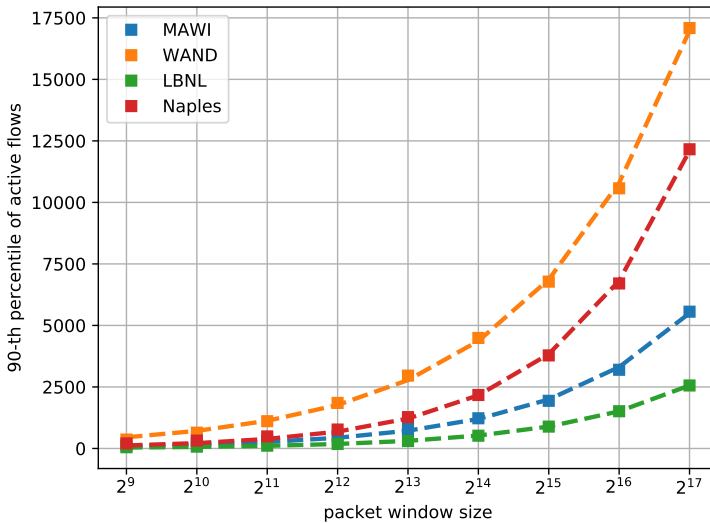
The metrics that support the flow locality in time domain concentrate on the duration of the flow as well as the number of simultaneous flows. The number of active connections in one millisecond is shown in Figure 5.



**Figure 5.** CDF of number of active flows in millisecond time frame



There is a clear split between the ISP traces (MAWI, WAND) and site-local traces (LBNL, Naples), as the ISPs aggregate more connections. Despite this, the number of simultaneous connections does not surpass 10,000 (even in the worst case – MAWI). There are two issues with a flow activity estimator based on the time window. First, a 1-ms time window is chosen arbitrarily just for comparison with [24]. It is worth examining different window sizes in order to observe the trend in the data. Second, a fixed time window may not be “fair” to all packet traces, as they represent different link speeds; e.g., for a fast 10Gb/s link, a 1-ms time window is enough to transmit 833 MTU-sized packets, but a 1Gb/s link can transmit only 83 of them. Moving from the time domain to the packet sequence domain makes it independent of link-speeds and more in line with the network device processing model described in Section 3.1.



**Figure 6.** Number of active flows as function of packet window size. Data is fitted to power law function  $\alpha x^\lambda$

Figure 6 shows the number of active flows as a function of the packet window size. This measurement was done in a sequence domain, and the active flow number is a 90<sup>th</sup> percentile number from the active flow distribution of the given packet window. Let us consider a window size of  $2^{16}$  and the number of active flows which is bigger than 90% of the samples (i.e.  $y = 0.9$  on a CDF chart). For that criteria the result for a WAND trace is 10,573 flows and 6706 for Naples, respectively. The 90<sup>th</sup> percentile rule was selected to avoid the need to display two-dimensional data, especially when the data distribution in the second dimension is similar to that in Figure 5. The data points are fitted to power law curve  $f(x) = \alpha x^\lambda$  with a high degree of similarity.

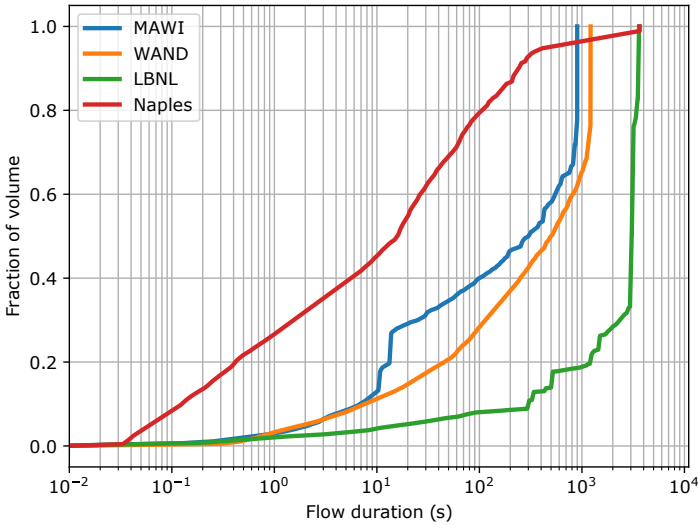
This means that increasing the window size by multiplying its size by a constant gives a proportional increase in the number of active flows; e.g.:

$$f(cx) = c'f(x) \quad (1)$$

for some constants  $c$  and  $c'$ . Using the function from Figure 6 gives a more specific result:

$$f(cx) = \alpha(cx)^\lambda = c^\lambda f(x) \quad (2)$$

It is worth noting that the ranking of traces (i.e., those with bigger active flows number) measured in a time-independent manner is different than the one computed using the time window. This observation becomes useful when confronted with the simulation results from Section 4.

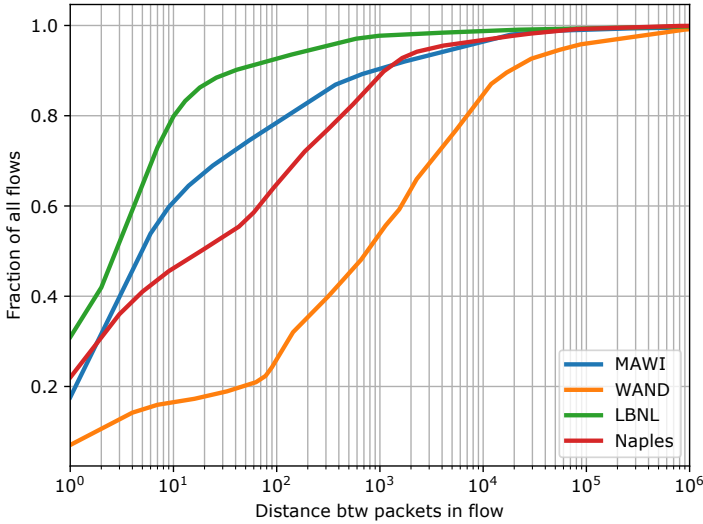


**Figure 7.** Flow duration (s) as fraction of total volume (CDF)

Figure 7 represents a CDF plot of the flow duration in seconds weighted by the volume of traffic in the packets. As an example, point  $(10^0, 0.3)$  (which is close to Naples curve) means 30% of the traffic belongs to connections that last for one second or less. This figure clearly shows that most of the traffic is localized in long-lasting connections. The short flows do not contribute to the overall bandwidth, despite their quantity (shown in Figure 3).

Both the time-domain and packet count statistics seem to be in line with the locality principle, but this is not definite proof. It is always possible that both time and volume are independent random variables. The last piece of the puzzle is to show that the traffic is *bursty* in nature. The burstiness benchmark is postulated by [15] as a *flow distance* distribution. It is examined in detail by [24] and [9],

although they only use a time-domain analysis. The flow distance statistics can be obtained by computing the cumulative distribution function of the distance between packets from the same flow. It is important to measure the distance in units, which represent a packet ordering that is independent of time. This time-less view is the best approximation of the way traffic is processed by the network infrastructure.



**Figure 8.** CDF of flow distance

Figure 8 shows the flow distance plot for all of the traces. According to [15], flow distance distribution is a measure of flow locality and should predict the performance of the flow cache. Let us consider a fully-associative cache with an LRU policy. The order in which the flow was referenced last time with respect to the other flows becomes a criterion for replacement. In other words, the “oldest” referenced flow is the first to be replaced. The replacement policy activates when the number of referenced flows is greater than the cache capacity.

According to Figure 8, c.a. 30% of the packets from the WAND trace have a flow distance of 100. If the cache capacity is 100, then the complementary 70% of traffic bandwidth with flow distances greater than 100 are less likely to hit the cache. It should be noted that the hypothesis of similarity between the CDF function and the cache behavior is empirical in nature and may not hold in general. Still, further experiments prove that the flow distance metrics correctly predict the performance of the cache for all selected data sets. If the CDF function is interpreted as a benchmark, it ranks the data sets with respect to cache effectiveness (with WAND having the lowest rank).

Most of the metrics (e.g., flow duration, flow packet count, and active connections) are not able to produce similar rankings, and each of the mentioned charts ranks

the packet traces differently; e.g., LBNL has the lowest number of active connections in a 1-ms time window, but WAND is ranked in the middle, not as an extreme. The only exception to this rule is the active flow metrics based on the packet sequence window shown in Figure 6. This ranks the traces in the same way as in Figure 8. Both metrics are time-less and based on packet sequence numbers, which suggests that this approach has more predictive power than the others. The possible explanation of this fact is partially given in Section 3.1 – this method is independent of link and bandwidth characteristics.

## 4. Simulations

In this section, the theoretical predictions of flow distance distribution are confronted with the simulation results. The method for determining the flow is the same as in Section 3.1 (i.e., 5-tuple). The software cache simulator was implemented as a part of the traffic analysis program used to produce the statistics in Section 3. The implementation is a special variant of a hash table with a fixed bucket size (1 to 4, depending on the variant) and additional LRU state and logic.

Four types of caches were implemented:

- fully associative, LRU replacement policy,
- 4-way associative, LRU replacement policy,
- 2-way associative, LRU replacement policy,
- direct mapped (one-way), random replacement policy.

The direct mapped cache is implemented as a hash table with a bucket size of 1. The flow replacement happens when a new key is inserted into a bucket already occupied by an older flow. In this way, the randomizing property of the hash function is used for a “random policy”. This is the simplest implementation for both the software and hardware, as it requires no additional memory for the LRU state.

Figure 9 is a detailed look at the relationship between the cache hit ratios and the CDF. This data is based on the WAND network trace. For cache sizes of between 4 and 65,536 entries, the CDF curve behaves like an approximation of the cache hit ratio. Another important conclusion is that a direct mapped cache with random replacement is a viable alternative for the implementation (although it is consistently worse than all of the other caches). There is a slight increase in performance from the random through the 2-way and 4-way up to the fully associative cache, which is expected based on the literature (e.g., [36]). On the other hand, a marginal improvement in the performance results in a significant increase of complexity when comparing a fully associative cache to simpler alternatives.

This is an important point, especially for hardware designers. It seems that the real choice lies between a 2- or 4-way LRU cache and a direct-mapped random cache. The choice between a stateless and stateful replacement policy such as LRU depends on the implementation trade-off. Maintaining the state (i.e., the history of last accesses) consumes more memory and other resources (energy, processing time,

circuit logic) in the software and even more in the hardware, where it results in multiple memory banks being activated in parallel [32]. Also, state representation becomes more complex when the associativity is greater than two [2].

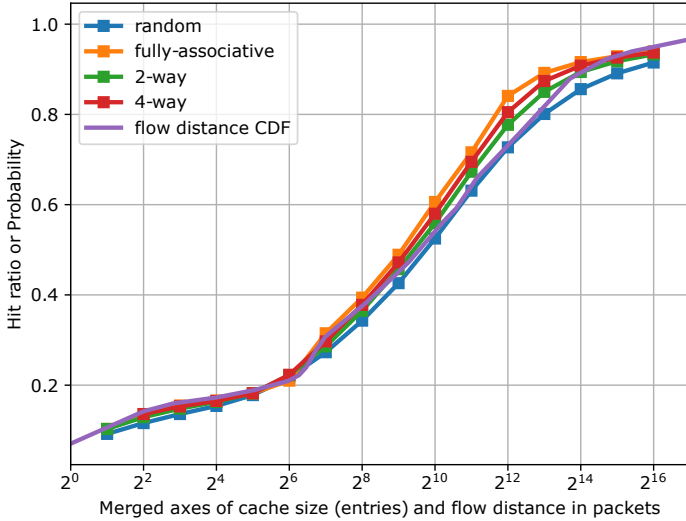


Figure 9. Cache hit ratio versus flow distance CDF, WAND trace

Figure 10 shows the results for a fully associative LRU cache and CDF functions for all traces.

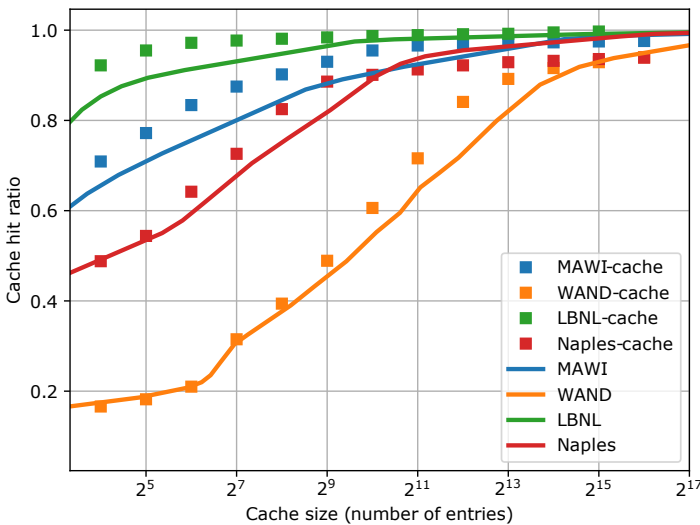


Figure 10. Cache hit ratio for fully associative cache combined with CDF function

In each analyzed case, the CDF can be treated as a lower bound of cache effectiveness. Except for WAND, the simulation shows good performance of the cache. Even with small cache sizes; e.g., for a 128-entry cache, the hit ratio is more than 70% for all traces other than WAND (which only achieves 30%). The 512-entry cache has a consistent effectiveness of at least 50% for all data sets.

When assessing cache effectiveness, it is worth expressing the cache size not in an absolute number but as a fraction of the number of active flows. For this task, the 90<sup>th</sup> percentile active flow benchmark from Figure 6 was selected. The window size for the measure should be chosen so that it spans a range of values that illustrate the change in behavior of the cache when its capacity is exceeded. In addition, when the cache size is bigger than the number of flows it is expected to handle nearly 100% of the traffic.

Table 1 shows the chosen active flows measurement for a window size of  $2^{16}$ .

**Table 1**  
Number of active flows in 64K-packet window

Trace	WAND	Naples	Mawi	LBNL
Active flows	10,573	6706	3198	1513

The key observation that leads to computing the hit ratio is the fact that a cache with a given size may be more effective for data sets with smaller active flows measure; e.g., for an LBNL dataset with 1513 active flows, a cache size bigger than 1500 has a nearly 100% hit rate simply because there are more cache entries than active flows. Replacement in a cache rarely occurs when all of the active flows fit the cache.

Looking at Figure 11 gives an overview of cache effectiveness across all traces. For data point  $2^{-6} \approx 1.5\%$ , the hit ratio varies from c.a. 35% (WAND) to 98% (LBNL). The worst performer (WAND) has a 50% hit ratio for a cache of 5% (512 entries) of the active flows.

The usefulness of the metrics developed in this article stretches beyond an analysis of the network data. Assuming that the traffic profile keeps its characteristics, it can be predicted how changing the bandwidth affects the cache or, conversely, how changing the cache size affects its effectiveness if the traffic bandwidth does not change. As an example exercise, let us answer the question of what happens when the traffic bandwidth is doubled. Doubling the bandwidth could be treated as doubling the window size in the active flow metrics. By the power law in Equation (2), the number of flows increases by a constant factor  $c' = 2^\lambda$ , with  $\lambda$  derived from the active flow curve. As a result, this "shifts" the cache effectiveness left on the efficiency curve with a factor of  $2^{-\lambda}$ . The WAND trace is fitted with  $\lambda \approx 0.65$ , so it gives a shift factor of 0.63. If the previous cache was 5% of all active flows, it is now 3.15%, and an expected hit ratio can be interpolated from Figure 11 to c.a. 35%.

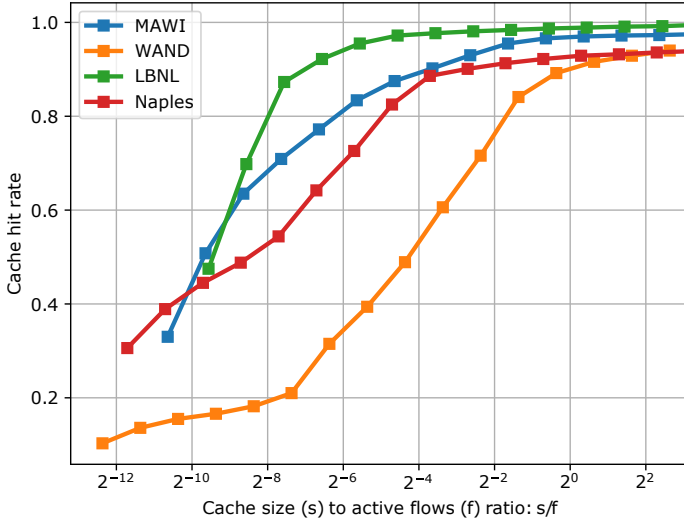


Figure 11. Fraction of active flows versus hit rate

## 5. Summary

The main point of this paper is to show the effectiveness of flow caching for packet forwarding. The methodology applied to the problem is based on a statistical analysis of flow locality and cache simulations. The statistics presented in Section 3 show that the locality principle manifests itself in various measurements: arrival time, flow address distribution, flow length, etc. This holds true for all of the examined data traces and is consistent with prior art on this subject from Section 2. The recommended method to predict caching effectiveness is to calculate two benchmarks. The first is the flow distance distribution – the simulation results in Section 5 show a high similarity between the flow distance and the cache hit ratio. The second benchmark is the number of active flows in a given packet sequence and limited to a certain window size. Combining these two results gives a cache effectiveness metrics that could be used for future predictions.

The simulations from Section 4 show that a moderately-sized flow cache can have a big impact on system performance. The exact number of cache entries that guarantee a high hit ratio may range from 16 to 1024 depending on the traffic profile. Modeling various cache designs shows that increasing the associativity of the cache reaps little benefit, so simpler 1-way or 2-way caches are preferred.

For the software, the performance cost of adding a flow cache is minimal; it may result in a large speedup of the processing pipeline. Packet processing on CPU is often dominated by random lookups in the data structures (e.g., a routing table). This kind of workload depends on memory subsystem latency, which increases with working set size. Contemporary data center CPUs have 256-512kB of private L2 and

more than 1MB of shared L3 cache per core. A 1024-entry flow cache can fit entirely in a CPU's L2 or L3 data cache, which leads to the better utilization of computing resources by avoiding costly CPU stalls caused by DRAM access.

A conclusion for hardware designers is that implementing flow caches need not be expensive. Even a simple direct-mapped cache can work efficiently, which means that the energy and logic cost paid by adding a cache is smaller as compared to CPU caches. In the latter case, the high associativity and replacement policy apparently matter more [2] than in the case of network processing.

Critics of network flow caching are often concerned that it causes a state explosion when an application operates on pattern matching rules; e.g., as in the Longest Prefix Match for IP routing. In this case, a single 24-bit subnet can potentially contain millions of flows. This is true in general, but the measurements in Section 3 show that the number of simultaneous flows is in fact limited and is subject to the same power law behavior as found in the page popularity ranking. If the principle of locality is taken into account, then the concern about state explosion must be re-examined on the new ground and simply rejected in some cases.

One of the possible research avenues concerning flow locality and caching is to characterize the traffic types that lead to local or non-local behavior. In other words, a predictor of cache-friendliness that does not require an actual statistical analysis is highly desired. If one exists at all, such a predictor could possibly stem from the structural properties of the network.

## References

- [1] Adamic L., Huberman B.: Zipf's law and the Internet, *Glottometrics*, vol. 3, pp. 143–150, 2002.
- [2] Al-Zoubi H., Milenkovic A., Milenkovic M.: Performance Evaluation of Cache Replacement Policies for the SPEC CPU2000 Benchmark Suite. In: *Proceedings of the 42nd annual Southeast regional conference*, ACM, pp. 267–272, 2004.
- [3] Asai H., Ohara Y.: Poptrie: A Compressed Trie with Population Count for Fast and Scalable Software IP Routing Table Lookup, 2015. <http://dx.doi.org/10.1145/2785956.2787474>.
- [4] Beckmann N., Sanchez D.: Talus: A simple way to remove cliffs in cache performance. In: *2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA)*, pp. 64–75, 2015. <http://dx.doi.org/10.1109/HPCA.2015.7056022>.
- [5] Bhuvaneshwaran V., Basil A., Tassinari M., Manral V., Banks S.: Benchmarking Methodology for Software-Defined Networking (SDN) Controller Performance. RFC 8456 (Informational), 2018. <http://dx.doi.org/10.17487/RFC8456>.
- [6] Bhuvaneshwaran V., Basil A., Tassinari M., Manral V., Banks S.: Terminology for Benchmarking Software-Defined Networking (SDN) Controller Performance. RFC 8455 (Informational), 2018. <http://dx.doi.org/10.17487/RFC8455>.



- [7] Borgnat P., Dewaele G., Fukuda K., Abry P., Cho K.: Seven Years and One Day: Sketching the Evolution of Internet Traffic. In: *IEEE INFOCOM 2009*, pp. 711–719, 2009. <https://doi.org/10.1109/INFCOM.2009.5061979>.
- [8] Cho K., Mitsuya K., Kato A.: Traffic Data Repository at the WIDE Project. In: *Proceedings of USENIX 2000 Annual Technical Conference: FREENIX Track*, pp. 263–270, 2000.
- [9] Çiftlikli C., Gezer A., Tuncay Özşahin A.: Packet traffic features of IPv6 and IPv4 protocol traffic, *Turkish Journal of Electrical Engineering and Computer Sciences*, vol. 20, pp. 727–749, 2012. <https://doi.org/10.3906/elk-1008-696>.
- [10] Claise B. (ed.): Cisco Systems NetFlow Services Export Version 9. RFC 3954 (Informational), 2004. <https://doi.org/10.17487/RFC3954>.
- [11] Cleary J., Graham I., McGregor T., Pearson M., Ziedins L., Curtis J., Donnelly S., Martens J., Martin S.: High precision traffic measurement, *IEEE Communications Magazine*, vol. 40(3), pp. 167–173, 2002.
- [12] Congdon P.T., Mohapatra P., Farrens M., Akella V.: Simultaneously Reducing Latency and Power Consumption in OpenFlow Switches, *IEEE/ACM Transactions on Networking (TON)*, vol. 22(3), pp. 1007–1020, 2014.
- [13] Dainotti A., Pescapé A., Rossi P.S., Palmieri F., Ventre G.: Internet traffic modeling by means of Hidden Markov Models, *Computer Networks*, vol. 52(14), pp. 2645–2662, 2008.
- [14] Dainotti A., Pescapé A., Ventre G.: A cascade architecture for DoS attacks detection based on the wavelet transform, *Journal of Computer Security*, vol. 17(6), pp. 945–968, 2009.
- [15] Feldmeier D.C.: Improving gateway performance with a routing-table cache. In: *IEEE INFOCOM’88, Seventh Annual Joint Conference of the IEEE Computer and Communications Societies. Networks: Evolution or Revolution? New Orleans, LA, USA*, pp. 298–307, 1988.
- [16] Gupta P., McKeown N.: Algorithms for Packet Classification, *IEEE Network*, vol. 15(2), pp. 24–32, 2001. <https://doi.org/10.1109/65.912717>.
- [17] [git.kernel.org, ipv4: Delete routing cache. https://git.kernel.org/pub/scm/linux/kernel/git/davem/net-next.git/commit/?id=89aef8921bfbac22f00e04f8450f6e447db13e42](https://git.kernel.org/pub/scm/linux/kernel/git/davem/net-next.git/commit/?id=89aef8921bfbac22f00e04f8450f6e447db13e42).
- [18] IEEE Standard for Information Technology – Telecommunications and Information Exchange Between Systems – Local and Metropolitan Area Networks – Specific Requirements Part 3: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications Amendment: Media Access Control (MAC) Parameters, Physical Layers, and Management Parameters for 10 Gb/S Operation. In: *IEEE Std 802.3ae-2002 (Amendment to IEEE Std 802.3-2002)*, pp. 271–272, 2002. <https://doi.org/10.1109/IEEESTD.2002.94131>.


- [19] Katta N., Alipourfard O., Rexford J., Walker D.: Rule-Caching Algorithms for Software-Defined Networks. In: *Technical Report, Princeton University*, 2014.
- [20] Kim N., Jean S., Kim J., Yoon H.: Cache replacement schemes for data-driven label switching networks. In: *2001 IEEE Workshop on High Performance Switching and Routing (IEEE Cat. No.01TH8552)*, pp. 223–227, 2001. <http://dx.doi.org/10.1109/HPSR.2001.923636>.
- [21] LBNL/ICSI Enterprise Tracing Project. <http://www.icir.org/enterprise-tracing/download.html>.
- [22] McKeown N., Anderson T., Balakrishnan H., Parulkar G., Peterson L., Rexford J., Shenker S., Turner J.: OpenFlow: Enabling Innovation in Campus Networks, *SIGCOMM Computer Communication Review*, vol. 38(2), pp. 69–74, 2008. <https://doi.org/10.1145/1355734.1355746>.
- [23] Naples University traffic traces, Trace 1 14th June 2004 11:00-12:00. <http://traffic.comics.unina.it/Traces/ttraces.php>.
- [24] *Network Traffic Characteristics of Data Centers in the Wild*. Association for Computing Machinery, Inc., 2010. <https://www.microsoft.com/en-us/research/publication/network-traffic-characteristics-of-data-centers-in-the-wild/>.
- [25] NumPy, scientific computing with Python. <https://www.numpy.org/>.
- [26] Okuno M., Nishimura S., Ishida S.i., Nishi H.: Cache-Based Network Processor Architecture: Evaluation with Real Network Traffic, *IEICE transactions on electronics*, vol. 89(11), pp. 1620–1628, 2006.
- [27] Pang R., Allman M., Paxson V., Lee J.: The devil and packet trace anonymization, *ACM SIGCOMM Computer Communication Review*, vol. 36(1), pp. 29–38, 2006.
- [28] Park K., Kim G., Crovella M.: On the Effect of Traffic Self-similarity on Network Performance. In: *Proceedings Volume 3231, Performance and Control of Network Systems*, 1997.
- [29] Paxson V.: Bro: A System for Detecting Network Intruders in Real-Time, *Computer Networks*, vol. 31(23–24), pp. 2435–2463, 1999. [https://doi.org/10.1016/S1389-1286\(99\)00112-7](https://doi.org/10.1016/S1389-1286(99)00112-7).
- [30] PcapPlusPlus, a multiplatform C++ network sniffing and packet parsing and crafting framework. <https://seladb.github.io/PcapPlusPlus-Doc/>.
- [31] Pfaff B., Pettit J., Koponen T., Jackson E., Zhou A., Rajahalme J., Gross J., Wang A., Stringer J., Shelar P., Amidon K., Casado M.: The Design and Implementation of Open vSwitch. In: *12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15)*, pp. 117–130. USENIX Association, Oakland, CA, 2015. <https://www.usenix.org/conference/nsdi15/technical-sessions/presentation/pfaff>.

- [32] Powell M.D., Agarwal A., Vijaykumar T., Falsafi B., Roy K.: Reducing Set-Associative Cache Energy via Way-Prediction and Selective Direct-Mapping. In: *Proceedings of the 34th annual ACM/IEEE international symposium on Microarchitecture*, pp. 54–65, IEEE Computer Society, 2001.
- [33] Security Architecture for the Internet Protocol. <https://tools.ietf.org/html/rfc430>.
- [34] Sikdar B., Vastola K.S.: The Effect of TCP on the Self-Similarity of Network Traffic. In: *Proceedings of the 35th Conference on Information Sciences and Systems*, pp. 21–23, 2001.
- [35] Srisuresh P., Holdrege M.: IP Network Address Translator (NAT) Terminology and Considerations. RFC 2663 (Informational), 1999. <http://dx.doi.org/10.17487/RFC2663>.
- [36] Su C.L., Despain A.M.: Cache design trade-offs for power and performance optimization: a case study. In: *Proceedings of the 1995 international symposium on Low power design*, pp. 63–68. ACM, 1995.
- [37] The CAIDA UCSD Anonymized Internet Traces 2014–2018. [http://www.caida.org/data/passive/passive\\_2014\\_dataset.xml](http://www.caida.org/data/passive/passive_2014_dataset.xml).
- [38] WIDE project. <http://mawi.wide.ad.jp/mawi/>.
- [39] WITS: Waikato Internet Traffic Storage. <https://wand.net.nz/wits/>.

## Affiliations

**Maciej Czekaj**

AGH University of Science and Technology, mczekaj@agh.edu.pl

**Ernest Jamro** 

AGH University of Science and Technology, Faculty of Computer Science, Electronics and Telecommunications, al. Mickiewicza 30, 30-059 Kraków, jamro@agh.edu.pl,  
ORCID ID: <https://orcid.org/0000-0003-4632-2470>

**Received:** 21.02.2019

**Revised:** 09.04.2019

**Accepted:** 09.04.2019