Revisiting Route Caching: The World Should Be Flat

Changhoon Kim[1], Matthew Caesar[2], Alexandre Gerber[3], and Jennifer Rexford[1]

[1] Princeton University,
[2] UIUC,
[3] AT&T Labs–Research

Abstract. Internet routers' forwarding tables (FIBs), which must be stored in expensive fast memory for high-speed packet forwarding, are growing quickly in size due to increased multihoming, finer-grained traffic engineering, and deployment of IPv6 and VPNs. To address this problem, several Internet architectures have been proposed to reduce FIB size by returning to the earlier approach of *route caching*: storing only the working set of popular routes in the FIB. This paper revisits route caching. We build upon previous work by studying flat, uni-class (/24) prefix caching, with modern traffic traces from more than 60 routers in a tier-1 ISP. We first characterize routers' working sets and then evaluate route-caching performance under different cache replacement strategies and cache sizes. Surprisingly, despite the large number of deaggregated /24 subnets, caching uni-class prefixes can effectively curb the increase of FIB sizes. Moreover, uni-class prefixes substantially simplify a cache design by eliminating longest-prefix matching, enabling FIB design with slower memory technologies. Finally, by comparing our results with previous work, we show that the distribution of traffic across prefixes is becoming increasingly skewed, making route caching more appealing.

1 Introduction

Packet forwarding on core Internet routers is an extremely challenging process. Upon receiving an IP packet, routers have just a few nanoseconds to buffer the packet, select the *longest-matching prefix* covering the packet's destination, and forward the packet to the corresponding outbound interface. To allow this process to happen quickly, routers often make use of special-purpose high-speed memory such as TCAM and SRAM. Unfortunately, the need for multi-homing and fine-grained traffic engineering, the desire to mitigate prefix hijacking by advertising more-specific routes, and the continuing rapid growth of the Internet have rapidly increased the number of routes that an Internet router has to maintain. The accelerating deployments of protocols with large address spaces such as IPv6 and VPNs, combined with the rapidly increasing link speeds, stand to worsen this problem even further.

Unfortunately, the special-purpose memory used for high-speed packet forwarding is orders of magnitude more expensive, more power-hungry, bigger in physical size, and generates substantially more heat than conventional DRAM. This is because manufacturing fast memory requires more transistors per bit [1]. Due to these factors, it

S.B. Moon et al. (Eds.): PAM 2009, LNCS 5448, pp. 3–12, 2009.

will be increasingly challenging to manufacture a line card with large, fast memory at a reasonable price and power-consumption budget. To save expenses, service providers may therefore be forced to provision with little headroom for growth, making their networks unable to handle sudden spikes in table size. Providers will also be forced to upgrade their equipment more often, a substantial problem given maintenance of operational networks can be much more expensive than hardware costs. To keep up with these demands, future routers must reduce the number of routes stored in the FIB.

In this paper we revisit *route caching*, where the FIB stores only the frequently-used routes and other routes are retrieved from a larger but slower memory (e.g., DRAM) on a miss. This approach is motivated by the fact that Internet traffic exhibits high degrees of temporal locality (as packets are grouped into flows, which are often transmitted in bursts) and spatial locality (as many hosts access a small number of popular destinations). In fact, route caching *was* once widely used in Internet routers. In the late 1980s and early 1990s, most routers were built with a route caching capability, such as *fast switching* [2, 3]. Unfortunately, these designs were not able to keep up with fast-increasing packet forwarding rates, due to the large cost of cache misses, such as lower throughput and high packet loss ratio. While this limitation of route caching is yet to be addressed, revisiting it with modern Internet traffic seems worthwhile because recent research results indicate that route caching might be both *possible* and *necessary*:

Route caching may be possible: New Internet routing architectures (such as ViAggre [4], or SEATTLE [5]) can improve feasibility of route caching by reducing the cost of a cache miss. For example, when a cache miss happens, a router can immediately forward a packet via a backup default route, without forcing the packet to be queued (while waiting for the cache to be updated from the slow memory) or to traverse the "slow path" (through the router CPU). The backup default route indirectly leads the packet to an alternate router that always maintains a correct route to the destination in its FIB (more explanation in Section 2.1). This "fall-back" mechanism has substantial performance benefits, because packets can always be sent immediately after the cache lookup completes. Since a route cache can be much smaller than a full FIB and takes substantially less time for a lookup, ensuring line-rate packet forwarding becomes easier.

Route caching may be necessary: New protocols with larger (e.g., IPv6) or even flat address spaces (e.g., ROFL [6], LISP [7], AIP [8]) have been proposed to facilitate the Internet's growth and configuration. However, deploying these protocols would significantly increase FIB sizes beyond the capacities of much currently-deployed equipment, and is predicted to require several million FIB entries within several years if current growth trends continue. When FIBs fill up, conventional routers crash or begin behaving incorrectly [9], forcing operators to deploy new line cards or even routers with a larger memory. Alternatively, in such a case, the use of caching would only increase the volume of the traffic handled via the "fall-back" mechanism, instead of incurring hard crashes, improving availability and extending times between router upgrades.

We start by describing our traffic traces collected from over 60 routers in a tier-1 ISP, and justify caching flat *uni-class* (i.e., /24) prefixes (Section 2). We then characterize the working sets of popular prefixes (Section 3) and evaluate route-caching performance under our uni-class model (Section 4). Despite the significantly larger number of uni-class prefixes as compared to CIDR, the cache size needed for a reasonably small miss

rate is comparable to the size of FIBs in conventional Internet routers. Moreover, a uniform prefix length allows use of *hashing* for faster lookups, greatly reducing the number of memory accesses per lookup (e.g., looking up an item in a chained hash table with N bins and N items requires 1.58 memory accesses on average, whereas a lookup in a trie-based approach, such as [10], takes much more than that). Therefore, caching uni-class prefixes may be implementable with a slower and cheaper memory (e.g., RLDRAM [1], or even regular DRAM). Finally, by comparing our results with previous work, we show that Internet traffic today is more amenable to caching (Section 5).

2 Measurement Methodology and Route-Cache Design

Our data sets were collected from a tier-1 ISP's backbone in the U.S. First, we collected *unsampled packet-level* traces over a one-week period at an access router servicing regional DSL subscribers in the ISP. Second, we collected *flow-level* traffic records from more than 60 edge routers in different geographical and topological regions. Since the volume of the traffic transited by those routers was extremely large, we utilized a *sampling* technique (Sampled NetFlow [11]) to reduce the overhead of collecting traffic. These two sets of traces are complementary to each other; the flow-level traces allow us to compare behavior across different routers, while the packet-level traces allow us to study finer-grained behavior at a single access router. Additionally, using unsampled packet-level traces allows us to validate the accuracy of our methodology for using sampled flow-level traces to study route caching.

DSL traces: We collected IP packet headers that originated from roughly $20,000$ regional DSL subscribers in the USA using our network traffic monitoring platform. Our monitor is attached to the access router aggregating traffic from subscribers, and was configured to monitor *inbound* traffic sent from DSL subscribers to the rest of the Internet. Note that we deliberately captured inbound traffic because destination addresses accessed by the inbound traffic are much more diverse than those by outbound traffic and are thus very challenging for our route-caching study. We ran our monitor for 8 consecutive days from Feb 29 through Mar 7, 2008, and captured roughly 40 billion packets, corresponding to an average of $\sim 65,000$ packets per second.

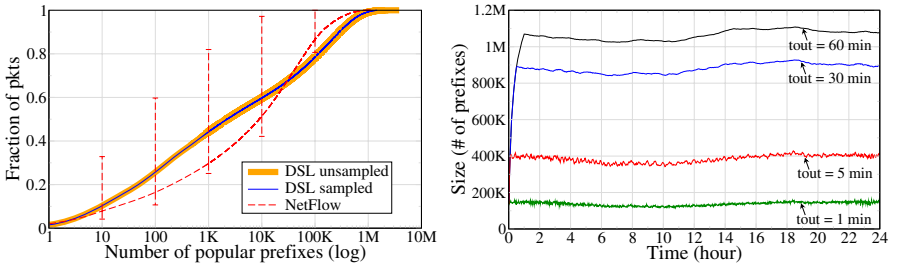
NetFlow traces: To study differences in workload across routers, we collected NetFlow records from inbound traffic to the ISP via two representative POPs (Points of Presence) containing over 60 routers, respectively located on the east and west coasts of the USA. To avoid overloading the router CPU, we were forced to configure NetFlow to perform deterministic sampling with a sampling ratio of $1/500$ [11]. The active and inactive time-out values were set to 60 and 15 seconds respectively. We ran NetFlow for 15 hours on January 23 - 24, 2008, collecting the information of ~ 330 billion packets (roughly 100K pkts/sec per edge router on average). Some of our analysis (e.g., estimating cache miss rate) requires packet-level arrival information. To construct packet records from flow records, we post-processed the trace to distribute all counted packets in a flow evenly over the measured duration of the flow. To check for sampling inaccuracies, we generated sampled DSL traces by applying the NetFlow sampling algorithm to our unsampled DSL traces. There was no statistically significant difference between the results acquired with sampled and unsampled traces.

2.1 Route Caching Model

To evaluate the performance of route caching, we define a simple and generic caching architecture. In this architecture, a packet forwarding unit (e.g., a line card, or a forwarding engine) incorporates hierarchical, two-level memory. The first level is a *route cache*, which is embodied by a small, but very fast memory containing only a subset of the entire routes. The second level is a *full routing table*, which is a slow, but large memory storing all routes. Once a packet arrives, the forwarding unit first looks up the packet's destination address in the route cache. If it finds a match, the packet is immediately forwarded based on the lookup result. If not, the forwarding unit forwards the packet using a backup route. The backup route indicates an alternate router, which can be either statically configured or dynamically chosen from a small set of alternate routers via a very simple computation (e.g., hashing) on the packet header. Note that this computation can be done in parallel with the cache lookup without increasing packet forwarding latency. The alternate router finally forwards the packet to the destination via a direct route residing in its FIB; to ensure this, administrators can run a well-provisioned router in each POP that always keeps the entire set of routes in its FIB or employ a routing protocol, such as ViAggre [4] or SEATTLE [5], where each router maintains a small amount of additional routing information in its route cache. Apart from this packet forwarding procedure, the forwarding unit separately updates its route cache by looking up the full routing table. If needed, an existing entry in the cache is evicted based on a cache replacement strategy.

Conventional routers store information about paths to CIDR (variable-length) prefixes in their routing and forwarding tables. Hence, at first glance, it appears that the cache should also store information in the same structure. However, caching CIDR prefixes presents a serious technical challenge arising from the need to perform longest-prefix matching: if multiple CIDR prefixes contain a destination address, the most-specific one – longest-matching prefix (LMP) – must be chosen to forward the packet. Unfortunately, in a route-caching system, only the contents of the cache are referred to when making a packet forwarding decision, and thus the *cache-hiding* problem arises: consider an empty cache, and suppose the full routing table contains two prefixes: $10.1.0.0/16$ associated with output interface `O1`, and $10.1.5.0/24$ associated with `O2`. Suppose a new packet destined to $10.1.2.3$ arrives. The router will find the LMP for this destination, which is $10.1.0.0/16$, and will install the route $[10.1.0.0/16 \rightarrow$ `O1`$]$ into the cache. Now, suppose the next packet the router receives is destined to $10.1.5.6$. Then, the router will discover $[10.1.0.0/16 \rightarrow$`O1`$]$ in the cache, and send the packet to `O1`. This, however, is incorrect because the LMP for $10.1.5.6$ is $10.1.5.0/24$, and hence the packet must have been sent to `O2`. Unfortunately, proposed solutions to this problem involve either a complicated data structure with on-the-fly computation to eliminate inter-dependency among prefixes [12], or grouping all prefixes containing a destination address as an atomic unit of caching operation (insertion, deletion, and update). The latter approach leads to cache thrashing because the size of an atomic prefix group in today's FIB can be larger than $25,000$. Worse yet, a cache storing CIDR prefixes still has to perform longest-prefix matching on every lookup.

To avoid these difficulties, we explore an alternative model which we refer to as "flat", *uni-class* caching. This model is identical to the technique that Feldmeier used in



(a) (b)

Fig. 1. (a) CDF of uni-class prefix popularity (b) Time series of working set size (DSL trace)

his 1987 study on route caching [13]. In this model, a route cache automatically divides up a CIDR prefix into small, fixed-length (i.e., $/24$) sub-prefixes that are mutually non-overlapping, and then store only the single $/24$ sub-prefix matching to the destination[1]. For example, suppose a full routing table contains a prefix $10.1.0.0/16$, along with several more-specific subprefixes under it. In the uni-class model, $10.1.0.0/16$ is internally considered as a collection of 256 independent $/24$ routes, ranging from $10.1.0.0/24$ to $10.1.255.0/24$. Hence, if the destination of an incoming packet is $10.1.2.3$, only $10.1.2.0/24$ is stored in the route cache. Note that the full routing table still contains a single CIDR prefix, $10.1.0.0/16$, as the sub-prefix is generated upon cache update. The reason why we chose $/24$ as the length for our study is multifold. First, it is the most-specific prefix length in common use for inter-domain routing; most providers filter routes more specific than $/24$ to protect themselves against misconfiguration and prefix hijacking. Also, it is the largest source of FIB growth in today's Internet: from Route-Views traces collected from Nov 2001 through Jul 2008, we found that the number of $/24$ prefixes has increased from $60K$ to $140K$, while other prefix lengths increased at a much slower rate (e.g. $/16$s increased from $7K$ to $10K$, $/8$s increased from 18 to 19).

3 Analysis of Traffic Workload

Prefix popularity: Before directly evaluating performance of caching on our traces, we first analyze relevant properties of the workload. Specifically, since our uni-class model can significantly inflate the number of unique prefixes, we are interested in figuring out how small (or large) the set of frequently-accessed /24 prefixes is. Figure 1a plots the fraction of packets sent to a given top-x number of prefixes, for both DSL and NetFlow traces. We set the maximum value of the x-axis to be the maximum possible FIB size (9.3M), which is the number of unique $/24$ prefixes we count when deaggregating the 305K CIDR prefixes advertised in the Internet as of February 2008. We find that roughly one tenth (i.e., 0.93M) of the entire prefixes accounts for more than 97% of traffic (consistent with previous work [13, 14, 15]), and nearly 60% (i.e., 5.3M/9M) of the prefixes are never accessed. We found that this result holds across a variety of routers in

[1] Uni-class caching can still support more specific routes than $/24$, if desired, by maintaining a small secondary table for those routes.

the network, as shown by the *NetFlow* curve and error bars. Finally, we also confirmed that packet sampling does not noticeably affect the popularity distribution, shown by the *DSL sampled* curve closely matching the *DSL unsampled* curve.

Temporal analysis of working sets: Understanding the temporal dynamics of traffic destinations is important because it determines the stability and predictability of caching performance. Hence, we study how prefix popularity varies at different times and across different timescales as well. To address this, we leverage the notion of a *working set*, which is defined to be the set of prefixes accessed over a given period of time. The size of the working set and its variation are often used to estimate how large a cache will be needed to achieve a low miss rate. Figure 1b shows variation in the size of the working set over time, under four different definitions of the working set: the set of items accessed over last 60, 30, 5, and 1 minute. As one might expect, larger $tout$ values induce larger working set sizes. Interestingly, however, we found the size of working sets across a variety of timeout values to be highly stable. Over our entire set of traces, the standard deviation in working set size was $\sim 3.2\%$ of the average, and the maximum working set size was no more than $\sim 5.6\%$ larger than the average. This fact bodes well for deployability of caching solutions using small fixed-size caches, as cache sizes can be provisioned close to the observed mean without requiring large headroom. Our further analysis also confirmed that the contents of the working sets vary little.

Cross-router analysis of working sets: Understanding the similarity of working sets across routers is important because it allows a cache to be *pre-provisioned* with the contents from another cache, significantly reducing the cold misses after network events (e.g., router or line-card reboot, routing change due to a network failure). Thus, we need to understand how working sets vary across different routers, router roles, and regions within the ISP. We chose several representative access routers: 7 from the ISP's west coast POP, and 9 from its east coast POP. To quantify the similarity of working sets at two different routers r_1 and r_2, we define the *similarity factor* (sfactor) as the number of common prefixes maintained by both r_1 and r_2, divided by the sum of the number prefixes maintained individually by r_1 and r_2. We computed the *sfactor* and its standard deviation, across all pairs of routers within the east-coast POP (sfactor=59.1% with stdev=11.8%), the west-coast POP (sfactor=64.9% with stdev=17.6%), and between pairs of routers in different POPs (sfactor=50.7% with stdev=14.3%). Overall, despite the limited aggregation benefit of using $/24$ prefixes, working sets of routers within a POP tend to be quite similar, with an sfactor of $59 - 65\%$ on average. Working sets of routers in different POPs tend to differ more, with only a 50.7% overlap on average. Some of these differences were due to localized DNS redirection (e.g., large content distribution sites redirecting users to geographically-closer servers).

4 Evaluation of Route Caching

LRU vs. LFU: In this section, we explore performance of caching algorithms directly on network traces, and start by comparing LRU (which keeps track of when each entry was used and evicts the one used the longest time ago) and LFU (which keeps track of how many times each entry is used and evicts the one used the smallest number of times while resident in the cache). Figure 2a shows that LRU outperforms LFU by a

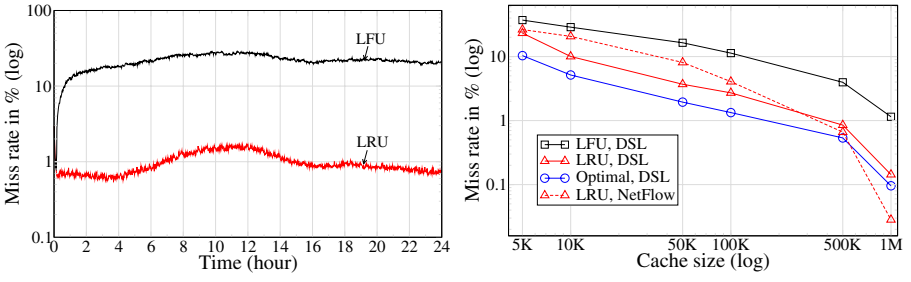


Fig. 2. Miss rate with the DSL traces, (a) Time series of miss rate, cache size = 500K, (b) Miss rates under LRU and the optimal strategy

large margin. Interestingly, the miss rate for LFU initially decreases during cold-start, which lasts for approximately 5 to 10 minutes as the cache fills. Several minutes after the cache reaches capacity, the miss rate sharply increases. This happens because the destinations of heavy (i.e., long and fat) flows become "stuck" in the cache, and the only way of evicting those is waiting for even heavier flows to overwrite those entries. LRU's miss rate also converges quickly to its long-term average and remains stable, whereas LFU's miss rate takes a few hours to converge and varies more. This happens because LFU tends to keep unnecessary entries for a long period of time. We observed these two findings across a variety of cache sizes and different input traffic mixes. Since LRU vastly outperforms LFU, we focus only on LRU for the rest of the paper.

LRU vs. Optimal: Next, we compare LRU's performance to an *optimal* scheme that has knowledge of future access times. The optimal caching algorithm (OPT) works by evicting entries that will not be needed for the longest time in the future. Note that implementing the optimal algorithm in practice is impossible (as it requires future knowledge), whereas implementing LRU (or an approximation of it) is possible – if not highly efficient or accurate. Figure 2b compares average miss rates over the unsampled DSL trace (solid curves). In a well-provisioned network, the cache would be large enough to attain a low miss rate. In this scenario, LRU performs nearly as well as OPT. For example, with a cache size of 500K, LRU's miss rate is only 0.3%-point higher than OPT's miss rate. We also study the fine-grained convergence behavior of OPT and LRU by measuring how fast their miss rates stabilize. We find that OPT's miss rate stabilizes within roughly 120 seconds, and that LRU converges almost as quickly, stabilizing within roughly 180 seconds. Given these results, it may be possible to design cache algorithms that outperform LRU on IP traffic, but it is unlikely the performance of these schemes will be substantially greater than that of LRU.

Cache size and miss rate: Figure 2b shows cache miss rates as a function of cache size for both the DSL traces (solid) and the NetFlow traces (dotted). The NetFlow curve shows average miss rate across all routers. Here we eliminated cold-start effects by measuring miss rates values only after the cache has reached capacity. We found that, with the DSL (NetFlow) traces, LRU attains a miss rate of 2.7% (4%) for a cache size of 100K, which is roughly 28% of the FIB size in conventional Internet routers. Based on the measured miss rates, we suggest some rough guidelines for determining a cache

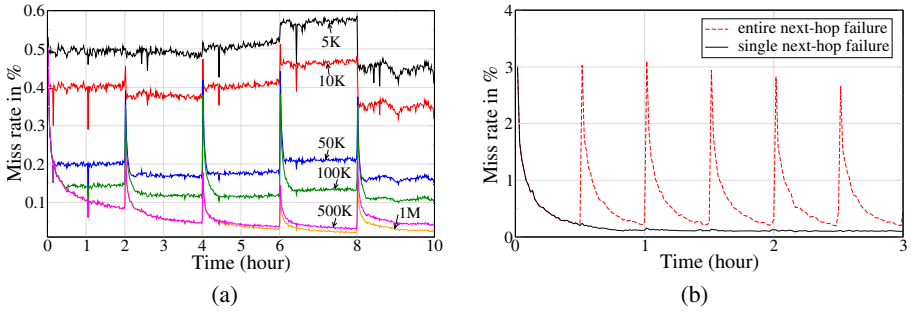


Fig. 3. Effect of routing change, (a) Inbound change (NetFlow traces), (b) Outbound change (DSL traces, cache size = 1M)

size: a cache size of 1M – which is less than $1/16$ of the maximum number of unique /24 prefixes in theory, and roughly $1/10$ of the total number of /24 prefixes used in the Internet today – may maintain a miss rate of roughly 0.1%. For a target miss rate of 1%, a cache size of roughly 500K may suffice. Overall, caching uni-class prefixes enables a route cache that is roughly an order of magnitude smaller than its full address space.

Impact of routing changes: Next, we evaluate the effect of network dynamics by manually injecting route changes into our traces, and evaluating how quickly caching algorithms converge after these changes are introduced. Two different route-change events can affect cache contents. First, routing changes upstream of the router may alter the distribution of *inbound* traffic arriving at router interfaces, leading to an abrupt change in the working set. We simulate this (Figure 3a) by randomly switching to a different router's NetFlow trace every hour while replaying against the same cache. While these events cause short-lived increases in miss rate, these spikes were roughly a factor of 3 above the average miss rate, and the miss rate stabilized in a few hundred seconds after these changes. Second, failures downstream of the router may alter the set of available *outbound* routes, causing multiple cache entries to become invalid simultaneously. We emulate (Figure 3b) the failure of multiple randomly-selected next hops by removing all cached entries associated with the given next hop upon its failure. Here, we find that for extreme scenarios where large numbers of next-hop routers fail, these spikes can be fairly large, increasing to 15 times the original value. However, for smaller numbers of failures, this value decreases substantially.

5 Related Work

Our paper is not the first to propose route caching. In fact, during the early 1990s, most Cisco routers were built with a route caching capability known as *fast switching* [2]. In these designs, packet forwarding decisions were made by invoking a user-level process that looks up a routing table (RIB) stored in slow memory. To boost lookup speeds, a route cache stored the results of recent lookups. Unfortunately, the large speed difference between the two lookup paths caused many packets to be kept in a buffer awaiting service in the slow path. In addition, upon routing changes or link failures, large groups of cached routes were simultaneously invalidated, dramatically decreasing

packet forwarding rate and increasing loss probability [16]. This limitation actually led to the abandonment of route-caching and motivated the evolution of today's caching-free routers. Our findings of large bursts of consecutive misses support these earlier observations about the limitation of route caching. However, the "fall-back" scheme (explained in Section 2.1) ensures full line-rate forwarding even on cache misses by immediately sending traffic to an intermediary. Several recent works suggest that constructing a reliable and efficient traffic indirection system is possible [4, 5, 7] and thus warrant revisiting route caching.

Also there has been research which recognized the difficulty of ensuring forwarding correctness when caching CIDR prefixes [12]. These approaches increase cache size and require a logarithmic searching algorithm even when prefixes are not overlapping. Recently, Iannone et al. studied the cost of route caching under the IETF LISP architecture [7] using traffic traces to and from a campus network [17]. Hence, their analysis results are applicable to estimating caching behavior at a stub network's egress router, whereas our results are suitable to understand caching performance in a large ISP's network, where route caching would be most beneficial. Moreover, although their study was based on caching CIDR prefixes, they did not address the problem of ensuring forwarding correctness with a subset of CIDR prefixes.

To understand how *modern* workloads change the performance of route caching, we compared our results with those of earlier studies. For example, in 1988, Feldmeier studied performance of caching /24 prefixes on traces captured at a gateway connected to the ARPANET [13]. Partridge repeated a similar analysis to Feldmeier's in 1995 [18] and confirmed Feldmeier's earlier findings. We compared our results against Feldmeier's to better understand how characteristics of Internet traffic have changed for the past 20 years. By comparing the cache size needed for a target hit rate with the number of unique /24 prefixes seen in the trace, we observed some interesting results. For example, when targeting a high hit rate (larger than 98%), route caching on modern traces performs better than in these earlier studies; achieving a hit rate of 98% today requires a cache size, normalized by the number of entire /24 prefixes in the traces, of 0.1 (i.e., 10%), whereas Feldmeier reported a normalized cache size of 0.23. Moreover, when targeting a lower hit rate than 98%, modern workloads are even more amenable to caching than 20 years ago. In particular, *when targeting a sub-95% hit rate, route caching today is an order of magnitude more efficient than it was 20 years ago*. For example, for a hit rate of 95%, we found modern workloads required a normalized cache size of only 0.008, while Feldmeier reported 0.096. Traditionally a sub-95% hit rate was not considered to be tolerable, but recent routing architectures that leverage the "fall-back" mechanism can easily tolerate such a rate.

6 Conclusion

An increasing number of network architectures make use of *route caching* to achieve scalability. Evaluating the feasibility of these techniques requires rigorous evaluation of the benefits and costs of caching. This paper revisits several earlier works from the late 1980s on route caching and evaluates the practicality of their techniques on modern workloads. To the best of our knowledge, this paper constitutes the first measurement

study of route-caching performance in a large ISP network. Key observations from our study are: (*i*) Working set sizes are stable over time, allowing route caches to be provisioned with relatively little headroom; (*ii*) Working sets of routers in a single POP are very similar to one another, introducing the possibility of pre-populating a cache; (*iii*) Uni-class caching eliminates complexity of longest-prefix matching and enables a cache using slower, cheaper memory; and (*iv*) Ensuring full line-rate forwarding upon cache misses is critical for the success of route caching. For future work, we plan to investigate theoretical models for the effects of sampling on estimating cache-miss rate.

References

1. Chang, E., Lu, B., Markhovsky, F.: RLDRAMs vs. CAMs/SRAMs: Part 1, http://www.commsdesign.com/design_corner/OEG20030603S0007
2. How to Choose the Best Router Switching Path for Your Network. Cisco Systems (August 2005), http://www.cisco.com/warp/public/105/20.pdf
3. Partridge, C., Carvey, P., et al.: A 50-Gb/s IP router. IEEE/ACM Trans. Networking (1998)
4. Ballani, H., Francis, P., Cao, T., Wang, J.: Making Routers Last Longer with ViAggre. In: Proc. NSDI (April 2009) (to appear)
5. Kim, C., Caesar, M., Rexford, J.: Floodless in SEATTLE: A Scalable Ethernet Architecture for Large Enterprises. In: Proc. SIGCOMM (August 2008)
6. Caesar, M., Condie, T., Kannan, J., Lakshminarayanan, K., Stoica, I.: ROFL: Routing on Flat Labels. In: Proc. ACM SIGCOMM (September 2006)
7. Farinacci, D., Fuller, V., Oran, D., Meyer, D., Brim, S.: Locator/ID Separation Protocol (LISP). Internet-Draft (work in progress) (December 2008)
8. Andersen, D., Balakrishnan, H., Feamster, N., et al.: Accountable Internet Protocol (AIP). In: Proc. ACM SIGCOMM (2008)
9. Chang, D., Govindan, R., Heidemann, J.: An empirical study of router response to large BGP routing table load. In: Proc. Internet Measurement Workshop (2002)
10. Eatherton, W., Varghese, G., Dittia, Z.: Tree bitmap: Hardware/Software IP Lookups with Incremental Updates. In: ACM Computer Communication Review (2004)
11. Sampled NetFlow, Cisco Systems, http://www.cisco.com/en/US/docs/ios/12_0s/feature/guide/12s_sanf.html
12. Liu, H.: Routing Prefix Caching in Network Processor Design. In: Proc. International Conference on Computer Communications and Networks (October 2001)
13. Feldmeier, D.: Improving Gateway Performance With a Routing-table Cache. In: Proc. IEEE INFOCOM (1988)
14. Rexford, J., Wang, J., Xiao, Z., Zhang, Y.: BGP Routing Stability of Popular Destinations. In: Proc. Internet Measurement Workshop (November 2002)
15. Jain, R.: Characteristics of Destination Address Locality in Computer Networks: A Comparison of Caching Schemes. Computer Networks and ISDN 18, 243–254 (1989/1990)
16. McRobb, D.: Path and Round Trip Time Measurements (slides 19-21), http://www.caida.org/publications/presentations/nanog9806/index.html
17. Iannone, L., Bonaventure, O.: On the Cost of Caching Locator/ID Mappings. In: Proc. ACM CoNEXT (December 2007)
18. Partridge, C.: Locality and Route Caches (1996), http://www.caida.org/workshops/isma/9602/positions/partridge.html