Synthetic Trace Generation for the Internet: An Integrated Model

W. Shi, M. H. MacGregor and P. Gburzynski
Department of Computing Science,
University of Alberta,
Edmonton, AB, T6G 2E8, Canada
{wgshi, macg, pawel}@cs.ualberta.ca
Phone: (780)492-1930, fax: (780)492-1071

Keywords: Temporal Locality, Workload Modeling, Synthetic Trace Generation, Parallel Forwarding, Load Balancing

Abstract
We document a synthetic trace generation model that captures two salient features of IP destination address traces that are critical to the performance of Internet forwarding systems. First, the degree of temporal locality in the trace can be controlled. Second, the trace can include a selected number of large flows that represent, for example, the result of transferring large files over high-speed links. To combine these features, we integrated the frequencies of large flows into the least recently used stack model (LRUSM). Synthetic traces generated using this technique accurately reproduce the behavior of real traces and can be used for parallel forwarding system performance evaluation.

INTRODUCTION

The phenomenal growth of the Internet presents scalability problems to its infrastructure. In a sense, the Internet has become a victim of its own popularity. In particular, the devices connecting multiple networks, known as routers or gateways, have to forward a huge amount of data from network to network in time to prevent the Internet from degrading in service or even collapsing.

Due to the widening gap between the performance of CPU and that of memory, using cache to improve overall performance has become a standard approach for the design of many computer systems. Routers with caches can take advantage of locality in their workloads, i.e., Internet traffic. Moreover, to forward at high speeds with the flexibility to accommodate new standards and services, one current trend in router design is to put multiple network processing units (NPU), also called *routing engines* or *forwarding engines*, on the same line card to forward incoming Internet packets in parallel. For such systems, to distribute workloads evenly among the processors is critical to reaching the performance potential provided by parallelism.

Workload characterization plays an important role in system performance evaluation. Accurate modeling of workload provides insights into system design. It is important to be able to generate accurately characterized workloads synthetically for use in trace-driven simulators. First, this allows parametric studies that are not possible with real traces. Second, it enables studies of system behavior in traffic regimes that are projected future scenarios, in which case real traces cannot be gathered at all.

We examine two characteristics of Internet traffic that are important to parallel forwarding system performance: temporal locality and flow size distribution.

We define a *flow* as all packets with the same destination address. This is a coarser aggregation than some other flow definitions, e.g., the five-tuple of the destination IP address and transport-layer protocol number, the source IP address and port number, and the transport layer protocol number. As a result, the notion of *flow size* distribution is the same as *IP address popularity* distribution.

We show that temporal locality in IP destination sequences can be captured by a mixed Weibull and Pareto distribution and that flow size distribution is Zipf-like. We show further that the flow size distribution and, in particular, the presence of several large flows in Internet traffic are the major sources of load imbalance for a parallel forwarding system. The above observations motivate the development of a synthetic workload model that captures both temporal locality and flow size distribution in IP destination address sequences.

After a brief discussion of related work, we describe our experiments, and we present the locality model used for aggregate traffic. The flow size distribution and the impact of large flows on parallel forwarding system load imbalance are discussed next. Then, we describe the integration of the aggregate traffic temporal locality and flow size distribution models and show simulation results. Finally, we summarize the work.

RELATED WORK

Temporal locality refers to the phenomenon that when an item is accessed, it is likely to be accessed in the near future. Temporal locality in the network environment originates from the *packet train* behavior of network traffic [1]. It is shown in [2] that significant forwarding performance improvement can be achieved by using simple route caches in a gateway. Ref. [3] extends the work in [2, 4] to multimedia environments.

Abundant research has been done to model temporal locality in Web document reference strings and synthetic workload generation. Refs. [5–8] use the least-recently-used stack model (LRUSM) to characterize temporal locality. Ref. [9] studies inter-reference times to show that locality exists in Web proxy access logs. It is pointed out in [10] that document popularity is related to temporal locality and a two-parameter model is proposed to capture both short- and long-term locality.

In [11], it is found that the reuse distance (a synonym of stack distance in the LRUSM) complementary cumulative distribution function (CCDF) for IP destination address traces can be captured by the sum of a two-parameter Weibull and a Pareto distributions:

$$C(x) = pW(x) + (1-p)P(x), \quad 0 < p < 1, \tag{1}$$

where $W(x)$ is the CCDF of the Weibull distribution:

$$W(x) = e^{-(x/d)^c}, \quad c, d > 0,$$

and $P(x)$ is the CCDF of the Pareto distribution:

$$P(x) = (1 + bx)^{-a}, \quad a, b > 0.$$

A trace generation scheme that is based on LRUSM and Eq. 1 is proposed to produce synthetic traffic with specified locality parameters.

Internet traffic is known to be *bursty*. Many models have been developed to capture this important feature. Recently, [12] used connection level information to reveal that burstiness in network traffic is due to large files transmitted over high-bandwidth connections. Such flows are called *alpha* flows and the rest of the traffic consists of the mixture of *beta* flows which represent other types of network traffic.

Ref. [13] studies the flow patterns of measured Internet traffic, and points out that network streams can be classified by both size (*elephants* and *mice*) and lifetime (*tortoises* and *dragonflies*). Tortoises are flows lasting more than 15 minutes, and represent a small portion of the flows (one or two percent) but carry fifty to sixty percent of the total traffic.

Flow-level analysis reveals details of network traffic that can have significant impact on forwarding system performance. Specifically, the few alpha flows in the aggregate traffic are the major contributors to load imbalance in parallel forwarding systems. Flow level characteristics also justify some cache designs for forwarding systems, e.g., the multi-zone cache [14], and Web systems, e.g., the GreedyDual-Size [9] and the GreedyDual* [15].

TRACES

We used three real-world traces to start this work. *UofA* is a 71-second trace (1 million entries) gathered at University of Alberta campus network in 2001. *AuckIV* and *IPLS* are packet header traces from NLANR [16] where AuckIV is a 5-hour trace (4.5 million entries) collected at the University of Auckland Internet uplink by the WAND research group between Feb. and Apr. 2000. and IPLS is a 10-minute trace (44.8 million entries) collected at an OC-48 link between Indianapolis and Cleveland in Aug. 2002. The three traces cover traffic patterns ranging from campus networks to a major Internet backbones.

TEMPORAL LOCALITY CHARACTERIZATION

The LRUSM [17] was studied by researchers in the context of program page reference behavior. Imagine a stack as a one-dimensional array, say A, containing all possible addresses, each of them a single array element. When an address is referenced, the index of the address, say *idx*, is output as the *stack distance* or *reuse distance*, and the LRU algorithm is used to update the stack: all the addresses before $A[idx]$ are moved down by 1 position and the $A[idx]$ is put at position 0 of the array, that is, at the top of the stack. In the LRUSM, each entry in an address trace is a reference to the same address on the stack and produces a stack distance. Therefore, corresponding to an address trace, there is a sequence of stack distances, which is called the *distance string*. As an example, a distance string can look like "$38, 1, 0, 143, 1,$ $162, 1, 0, 40, 97, 1, 150,$ $63, 311, 80, 312, 1, 3, 0, 313, 127$". A "0" in the string indi-

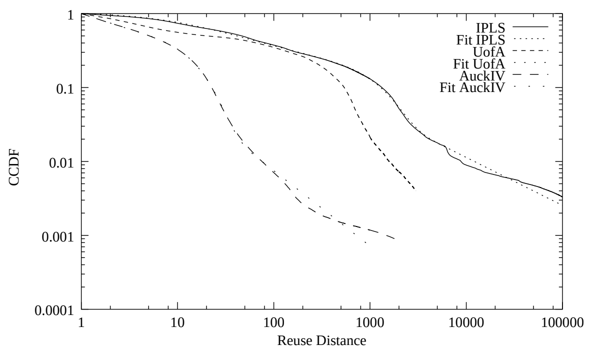

Figure 1: Fitting Reuse Distances CCDF's

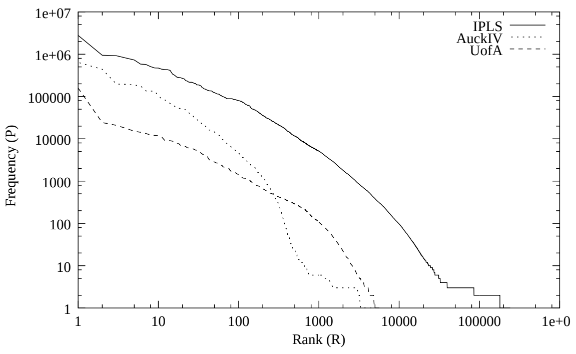

Figure 2: Zipf-like Internet Flow Size Distributions

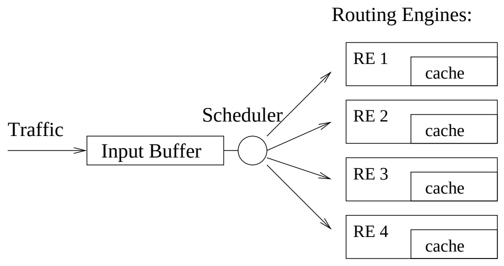

Figure 3: A Parallel Forwarding System

cates that the address at the top of the stack, which was just referenced, is referenced again; a "1" means that the address referenced just prior to the last reference (now at position 1 of the stack) is referenced again, and so on.

The distribution of the reuse distance, therefore, can answer the question "what is the probability that an address just referenced will be referenced again in the future?". The LRUSM can be used to generate synthetic address traces with temporal locality quantified by reuse distance distribution parameters. Specifically, a sequence of reuse distances, $r_1, r_2, \ldots, r_i, \ldots$, is generated to reference a stack of unique IP addresses. For reuse distance r_i, the address at the r_ith position in the stack is produced and the stack is updated. The address sequence thus generated retains the desired temporal locality.

We use Eq. 1 to characterize the temporal locality in IP destination address sequences. Fitting this function to the measured reuse distance CCDF's of the three traces yields the results shown in Fig. 1.

FLOW LEVEL TRAFFIC CHARACTERISTICS AND LOAD BALANCING IMPLICATIONS

The deficiencies of the LRUSM as a tool to capture temporal locality have long been recognized. For example, in Web workload characterization, it is criticized for not being able to tell the difference between short-term temporal correlation and long-term popularity. When applied to IP traffic, using the LRUSM to capture temporal locality ignores per-flow information. This makes it useful only in characterizing the locality of *aggregate* traffic. When it is used to produce synthetic traces, the individual flows within the trace tend to be similar in size and arrival patterns. This contradicts the traffic patterns observed in [12, 13].

Zipf's Law Applies to Internet Traffic

Zipf's law [18] is the observation that frequency of occurrence of some event (P) as a function of rank (R) is a power-law function

$$P(R) \quad \sim \quad 1/R^a \qquad (2)$$

with the exponent a close to 1.

Fig. 2 shows that, although scales differ, each of the three flow size distributions can be described by a Zipf-like distribution with an α value around 1.2. Therefore, we believe that Zipf's law is adequate for the purpose of modeling the largest flows. We assume that the flows with small ranks (R values) in Fig. 2 are the alpha flows.

Many distributions related to the Internet have been observed to follow Zipf-like distributions [19] and IP address is one. Given that Web requests dominate Internet traffic, the Zipf-like distribution of IP destination address can be the result of Zipf-like Web request size distributions [20, 21].

Implications for Load-balancing

A forwarding system with four routing engines (RE) is shown in Fig. 3. Incoming traffic is dispatched by the

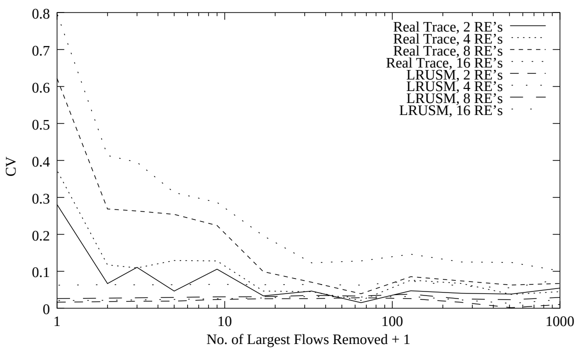

Figure 4: Effect of Largest Flows on Load Balancing (UofA Trace)

scheduler which is often hash-based [22,23]: one or more header fields (in this paper, the IP destination address) of an incoming packet are selected as a key to a hash function and the return value is used to select the RE to forward the packet to. Since the header fields are usually constant for all the packets in one flow, such a scheduler preserves packet order within flows. Packet order preservation is desirable because re-ordered TCP segments give TCP false congestion signals and thus are detrimental to end-to-end system performance [24,25].

In an N-RE system, let B_i be the total busy time of the ith RE, $i \in \{1, \ldots, N\}$. The coefficient of variation (CV) of busy time is used as a measure of the degree of load imbalance in the system:

$$CV_B = \frac{\sigma_B}{\mu_B} \tag{3}$$

where σ_B is the standard deviation and μ_B is the mean.

CV_B is chosen for its independence from the units of measurement. It reflects the fact that the traffic is biased toward a few flows, and reveals the system's ability to balance the load. With the simulation input fixed, CV_B measures the latter. In the ideal case where each RE has exactly the same load, σ_B and CV_B would both become zero.

To appreciate the importance of flow-level specifics in Internet forwarding systems, we show the change in load imbalance after removing a number of the largest flows from the UofA trace and its LRUSM-generated synthetic trace (Fig. 4). We use the Internet checksum algorithm over IP destination addresses to generate random RE indices [26,27]. We assume that the RE's have identical capability and that each packet requires the same amount of processing.

For the real trace, removing one or two of the largest flows drastically reduces CV_B values. Removing more large flows helps to improve load imbalance situations, especially for systems with more RE's. The reduction in CV_B diminishes when even more large flows are removed and CV_B values never become zero. The above observations imply that removing large flows is useful but does not eliminate load imbalance. For the synthetic trace, however, load imbalance is insignificant compared with that observed when using the real trace and there are no obvious changes in load imbalance when the largest flows are removed. The reason for both observations is that without taking flow-level information into account, LRUSM generates homogeneous flows. Traces generated using only the LRUSM will thus produce inaccurate results when applied to simulations of multi-RE systems. Flow-level statistics must be added to generate realistic synthetic workloads.

THE INTEGRATED MODEL

The goal of our synthetic trace generation algorithm is to capture both the temporal locality and the nature of the flows making up Internet traffic. To realize this goal, we implement alpha flows in the framework of the LRUSM. In the following discussion, we consider the *"number of inter-arrival packets"* (NIP), the number of packets between two successive packets of the flow in question, as the measure of packet inter-arrival time. Characterizing the distribution of NIP makes it easier to integrate flows into the LRUSM. The basic idea in our algorithm is to reorder LRUSM-generated reuse distances to match the NIP's generated from a packet arrival model for alpha flows.

A stack is first initialized to contain all the distinct IP addresses to be used. Let $r_1, r_2, \ldots, r_T, \ldots$ be a sequence of reuse distances generated by the LRUSM. Suppose that N alpha flows are to be integrated into this sequence. At time T, the arrival model of the N flows is used to produce the $\{$*next address, NIP*$\}$ pair, say $\{A, NIP_A\}$. Thus, the address A is scheduled at the future time t, $t = T + NIP_A$, in the generated traffic.

While $0 \leq i < NIP_A$, the $T + i$th address is generated using the LRUSM. To generate the tth address, i.e., A, we search the stack for address A (alternatively, the stack position of the alpha flows can be tracked during the generation process). Assuming that A is found at index r_A in the stack, we search forward in the reuse distance sequence $r_t, r_{t+1}, \ldots$, for r_{t+m} equal to r_A. r_{t+m} is then removed from the reuse distance sequence, the address A is output, and the stack is updated. Alternatively, A is output and later when the generating process proceeds to the point where r_A appears in the reuse distance sequence, it is deleted. We generate another pair

```
0. Initialization
     Generate a reuse distance sequence in array RD[ ]
     Populate the array Stack[ ] with unique IP addresses
     Identify the flow's destination address A
     p <-- 0
1. Generate next arrival time t
2. While (p < t) Do
3.     Find i, so that i>=p, RD[i] != StackIndex(A)
4.     Swap RD[i] and RD[p]
5.     UpdateStack(RD[p])
6.     p <-- p + 1
7. Find i, so that i>=p, RD[i] == StackIndex(A)
8. Swap RD[p] and RD[i]
9. UpdateStack(RD[p])
10. p <-- p + 1
11. Go to step 1
```

Figure 5: Synthetic Trace Generation: LRUSM + 1 Alpha Flow

$\{A,\ NIP_A\}$ and the above process repeats.

An important question is when searching for the reuse distance, r_A matching the stack index of A, how far into the future is the algorithm expected to look. The answer depends on the rates of the alpha flows. Suppose that the arrival rate of flow A is λ_A, then $E[NIP_A] = 1/\lambda_A$, where $E[NIP_A]$ is the expected value of NIP_A. The definition of reuse distance implies that $E[r_A] \leq E[NIP_A]$. If K_A is the number of steps we must look ahead, then the upper bound of $E[K_A]$ can be expressed as

$$UpperBound(E[K_A]) = \frac{1}{P(E[NIP_A])}, \qquad (4)$$

where $P(x)$ is the reuse distance probability density function which can be derived from Eq. 1.

In practice, we would like K_A to be as small as possible to avoid lengthy searching. Fortunately, because alpha flows are high-rate flows, any two successive packets from an alpha flow are not far-apart, and therefore the reuse distances for alpha flow addresses should be relatively small. According to Eq. 4, this, in turn, indicates that K_A should be small.

The algorithm that combines the LRUSM and one alpha flow is shown in Fig. 5. The function $StackIndex(A)$ returns the index of address A in the stack. $UpdateStack(n)$ output the address at stack index n and updates the stack using LRU. The next arrival time is generated by the flow model (step 1). It is trivial to incorporate more alpha flows: we only need to mix all the alpha flows and generate NIP's for each flow. This process is independent from the LRUSM, so the reuse distance search and stack update procedures need not be changed.

In summary, to generate synthetic traces, our algorithm needs an alpha flow arrival model, an LRUSM, and two groups of parameters:

- **Locality Parameters**: the five parameters, a, b, c, d, p, of the reuse distance distribution (Eq. 1).
- **Popularity Parameters**: α, the shape parameter, and N, the scale of the Zipf-like function (Eq. 2), and the number of alpha flows to incorporate.

SIMULATION RESULTS

We experiment with two traces: the UofA trace and the first 1 million entries of the IPLS trace. Two synthetic traces are generated for each: one based on the LRUSM model and Eq. 1 [11], the other based on the integrated algorithm described in Fig. 5. In the latter, for simplicity, we use the Poisson model to generate alpha flow arrivals. This is adequate for our discussion of load balancing. Twenty alpha flows are generated in the second synthetic trace. Although using the Zipf-like formula to model the popularity of flows is one of our goals, to better show the effect that large flows have on load imbalance, we calculated the rates of these flows from the real traces. The destination addresses of these flows are also used in the corresponding synthetic flows. This is necessary for simulating the load of the RE's under hash-scheduling.

We thus have two sets of traces, with three traces in each set. For each set, we compare the CCDF of the reuse distances, the flow popularity distributions, and the load balance curves as large flows are removed from the trace. The results for the UofA and IPLS sets are shown in Figs. 6 and 7. In each sub-figure, the result for the trace that is based on the integrated model is identified as "synthetic - LRUSM+20flows".

Both figures show that the real trace, the LRUSM-based synthetic trace, and the integrated-model-based trace have identical temporal locality: the CCDF's for the traces in each set are almost identical. This result shows how well Eq. 1 captures the temporal locality of aggregate traffic.

However, the flow popularity curves of the three traces differ from each other. In both figures, the curve for the LRUSM-based synthetic trace is quite different than those for the real trace and the integrated model. For the integrated-model-based synthetic traces, the most popular 20 flows match those of the real trace. As indicated in Fig. 4, a relatively small number of alpha flows

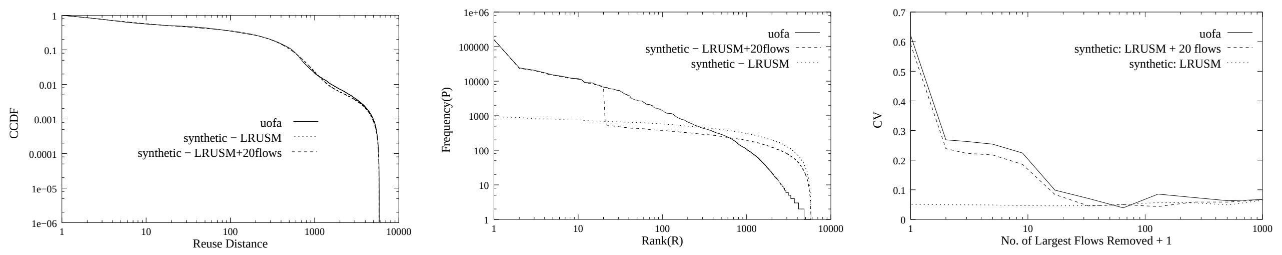

Figure 6: Simulation Results for UofA Traces (Left to Right: Reuse Distance CCDF's, Flow Popularity Distributions, Load Imbalance for an 8-RE System)

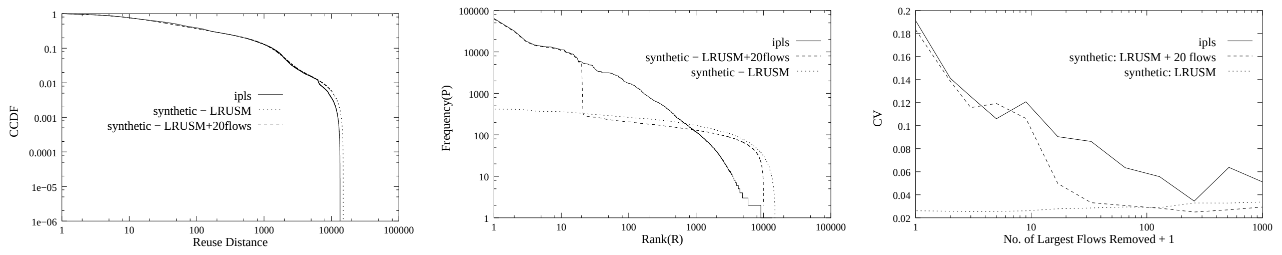

Figure 7: Simulation Results for IPLS (1st 1 Million Entries) Traces (Left to Right: Reuse Distance CCDF's, Flow Popularity Distributions, Load Imbalance for an 8-RE System)

contribute the most to the load imbalance in a parallel forwarding system.

This is confirmed by the load balance simulation results shown in the right sub-figures of Figs. 6 and 7. *CV* values when using the real trace and that from the integrated model are relatively close.

SUMMARY

In this paper, we propose a generative model that captures two features of the workload for Internet forwarding systems: the temporal locality in aggregate IP destination address sequences and the biased address popularity distributions. With a Zipf-like curve, we are able to capture the flow size distribution of Internet traffic. We developed an algorithm to integrate the LRUSM and the flow size distribution model to produce more realistic synthetic traffic. Simulations using real traces and simulations using traces generated by our integrated model give the same results. This indicates that we have successfully captured both the temporal locality and flow-size distribution of real traffic. The parametric nature of the synthetic workload model enables the generation of traffic over a wide range for parametric studies. It also enables us to generate traffic representing future scenarios not yet seen, but for which new systems concepts and designs must be developed and tested.

References

[1] R. Jain, S. Routhier, "Packet trains: Measurements and a new model for computer network traffic," *IEEE Journal of Selected Areas in Communications*, vol. SAC-4, no. 6, pp. 986–995, September 1986.

[2] D. Feldmeier, "Improving gateway performance with a routing table cache," in *IEEE INFOCOM 1988*, New Orleans, LA, USA, April 1988, pp. 298–307.

[3] X. Chen, "Effect of caching on routing-table lookup in multimedia environment," in *IEEE INFOCOM 1991*, Bal Harbour, FL, USA, April 1991, pp. 1228–1236.

[4] R. Jain, "Characteristics of destination address locality in computer networks: a comparison of caching schemes," *Computer Networks and ISDN Systems*, vol. 18, no. 4, pp. 243–254, May 1990.

[5] M. F. Arlitt, C. L. Williamson, "A synthetic workload model for Internet mosaic traffic," in *Summer Computer Simulation Conference (SCSC95)*, Ottawa, Canada, 1995, pp. 852–857.

[6] V. Almeida, A. Bestavros, M. Crovella, A. Oliveira, "Characterizing reference locality in the WWW," in *The IEEE Conference on Parallel and Distributed Information Systems (PDIS 1996)*, Miami Beach, FL, USA, December 1996, pp. 92–107.

[7] M. F. Arlitt, C. L. Williamson, "Internet Web servers: Workload characterization and performance implications," *IEEE/ACM Transactions on Networking*, vol. 5, no. 5, pp. 631–645, Oct. 1997.

[8] A. Mahanti, D. Eager, C. Williamson, "Temporal locality and its impact on Web proxy cache performance," *Performance Evaluation Journal: Special Issue on Internet Performance Modeling*, vol. 42, no. 2-3, pp. 187–203, 2000.

[9] P. Cao, S. Irani, "Cost-aware www proxy caching algorithms," in *USENIX Symposium on Internet Technology and Systems (USITS)*, Monterey, CA, USA, 1997, pp. 193–206.

[10] S. Jin, A. Bestavros, "Sources and characteristics of Web temporal locality," in *MASCOTS 2000*, San Fransisco, CA, USA, August 2000, pp. 28–35.

[11] W. Shi, M. H. MacGregor, P. Gburzynski, "Traffic locality charateristics in a parallel forwarding system," *International Journal of Communication Systems*, vol. 16, no. 9, pp. 823–839, November 2003.

[12] S. Sarvotham, R. Riedi, R. Baraniuk, "Connection-level analysis and modeling of network traffic," in *ACM SIGCOMM Internet Measurement Workshop*, San Francisco, CA, USA, November 2001, pp. 99–103.

[13] N. Brownlee, K. Claffy, "Understanding Internet traffic streams: Dragonflies and tortoises," *IEEE Communications*, vol. 40, no. 10, pp. 110–117, Oct. 2002.

[14] I. Chvets, M. MacGregor, "Multi-zone caches for accelerating IP routing table lookups," in *High Performance Switching and Routing (HPSR 2002)*, Kobe, Japan, May 2002, pp. 121–126.

[15] S. Jin, A. Bestavros, "GreedyDual* Web caching algorithm: exploiting the two sources of temporal locality in Web request streams," *Computer Communications*, vol. 24, no. 2, pp. 174–183, 2001.

[16] NLANR (National Laboratory for Applied Network Research) Measurement and Operations Analysis Team (MOAT), "Packet header traces," http://moat.nlanr.net.

[17] J. Spirn, *Program Behavior: Models and Measurements*, Elsevier-North Holland, N.Y., 1977.

[18] G. K. Zipf, *Human Behavior and the Principle of Least-Effort*, Addison-Wesley, Cambridge, MA, 1949.

[19] L. A. Adamic, B. A. Huberman, "Zipf's law and the Internet," *Glottometrics 3*, pp. 143–150, 2002.

[20] P. Barford, M. Crovella, "Generating representative Web workloads for network and server performance evaluation," in *ACM SIGMETRICS 1998*, Madison, WI, USA, July 1998, pp. 151–160.

[21] L. Breslau, P. Cao, L. Fan, G. Phillips, S. Shenker, "Web caching and Zipf-like distributions: evidence and implications," in *IEEE INFOCOM 1999*, New York, NY, USA, March 1999, pp. 126–134.

[22] G. Dittmann, A. Herkersdorf, "Network processor load balancing for high-speed links," in *2002 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS 2002)*, San Diego, CA, USA, July 2002, pp. 727–735.

[23] L. Kencl, J. Le Boudec, "Adaptive load sharing for network processors," in *IEEE INFOCOM 2002*, New York, NY, USA, June 2002, pp. 545–554.

[24] J. Bennett, C. Partridge, N. Shectman, "Packet reordering is not pathological network behavior," *IEEE/ACM Transactions on Networking*, vol. 7, no. 6, pp. 789–798, Dec. 1999.

[25] E. Blanton, M. Allman, "On making TCP more robust to packet reordering," *ACM Computer Communication Review*, vol. 32, no. 1, pp. 20–30, Jan. 2002.

[26] R. Jain, "A comparison of hashing schemes for address lookup in computer networks," *IEEE Transactions on Communications*, vol. 40, no. 3, pp. 1570–1573, October 1992.

[27] Z. Cao, Z. Wang, E. Zegura, "Performance of hashing-based schemes for Internet load balancing," in *IEEE INFOCOM 2000*, Tel-Aviv, Israel, March 2000, pp. 332–341.