FastCARS: Fast, Correlation-Aware Sampling for Network Data Mining

Jia-Yu Pan, Srinivasan Seshan, Christos Faloutsos
Computer Science Department, Carnegie Mellon University
Pittsburgh, Pennsylvania 15213, USA

***Abstract*—Technology trends are making it more and more difficult to observe and record the large amount of data generated by high speed links. Traffic sampling techniques provide a simple alternative that reduces the volume of data collected. Unfortunately, existing sampling techniques largely hide any temporal relationship in the recorded data.**

Our proposed method, "FastCARS" naturally captures statistics for packets that are 1, 2 or more steps away. It has the following properties: (a) provides accurate measurements of a full trace's statistics, (b) is simple and can be easily implemented, (c) captures correlations between successive packets, as well as packets that are further apart, (d) generalizes previously proposed sampling methods and includes them as special cases, and (e) is scalable and flexible to account for prior knowledge about the characteristics of traces.

We also propose several new tools for network data mining that use the information provided by *FastCARS*. The experimental results on multiple, real-world datasets (233Mb in total), show that the proposed *FastCARS* sampling method and these new data mining tools are effective. With these tools, we show that the independence assumption of packet arrival is not correct, and that packet trains may not be the only cause of dependence among arrivals.

***Index Terms*—Traffic analysis, sampling**

I. INTRODUCTION

The ability to monitor and characterize network traffic has proven to be critical to the design and operation of today's networks. However, as links have gotten faster and faster, it has become more difficult to observe key traffic characteristics or record packet data in real-time. Already, most network monitors rely on sampling techniques [1] [2] to provide measurements of high speed links. The ability of these sampling techniques to preserve data characteristics is necessary for network data mining applications which aim at revealing patterns and correlations that are crucial to the understanding and development of today's and future networks.

Today's sampling techniques are targeted towards enabling tasks such as usage-based billing, capacity planning and network research. These techniques can typically answer questions about the traffic such as: *What is the distribution of packet sizes on this link? Which destinations are popular? or How long are typical connections?* However, a significant weakness of existing schemes is that they do not answer questions about the temporal correlation of the traffic. For example, some interesting traffic characteristics include: *How is the arrival time of packet n related to that of packet $(n+1)$ or $(n+2)$? Is there correlation among arrivals?* In the past, analysis of such packet content characteristics [3] and arrival correlation [4], using full (unsampled) packet traces, have led to the discovery of important phenomena such as "packet trains", which is defined as a set of sequential packets that have the same source/destination IP addresses and port numbers, and self-similar traffic pattern. Clearly, such traffic characteristics are critical to the design of routers, routing algorithms and caching techniques. It is necessary that this kind of analysis should be possible on sampled data, rather than on full trace.

We would like a sampling method that is informative and efficient. It should provide sufficient information for accurate estimates of both average and temporally correlated statistics. It should also be simple and require low computation when implemented on routers. To achieve these objectives, we propose a new method, Fast, Correlation-Aware Sampling (*FastCARS*), to do sampling and data mining on router traffic. We show that our method can support the traditional uses of network sampling (provide interarrival time distribution), as well as statistics about packets separated at different steps, which can be used for further data mining on the sampled traffic. In particular, we use *FastCARS* to explore the independence of interarrival time. We show that packet interarrival times are not independent and packet trains may not be the only cause of dependence among arrivals.

This paper is organized as follows. We summarize previous work in Section II. Section III describes our sampling method. Section IV presents results from using our sampling technique. Finally, we present our conclusions in Section V.

II. RELATED WORK

Network sampling has played an important role in network measurements for the past decade. In order to describe the desirable properties of a sampling technique, we begin by defining the term *step*.

***Definition** 1:* We call the separation between samples, **steps**. A **n-step histogram** is a histogram of measurements obtained from pairs of sampled packets that are n steps away (separated by $(n-1)$ packets). Histograms of different steps provide aggregated statistics which reveal short and long term correlations (i.e. **temporal correlation**).

Statistics of samples n steps apart are prefixed by the term **n-step**. For example, the interarrival time between a pair of back-to-back packets is an instance of **1-step interarrival time**. We will show in the following sections that n-step histograms give us information about the traffic characteristics, and reveal temporal correlation between packets. For example, the 1-step histogram is used to estimate packet interarrival time distribution, and the 2-step histogram is used to explore the independence of interarrival times. One important property for a sampling

This material is based upon work supported by the National Science Foundation under Grants No. IRI-9817496, IIS-9988876, IIS-0083148, IIS-0113089, IIS-0209107 and by the Defense Advanced Research Projects Agency under Contract No. N66001-00-1-8936.

technique is that it be **correlation-aware**, i.e., it should provide statistics for n-step histograms for arbitrary n.

We classify past techniques into four categories: event-driven sampling, random sampling, configured run-length sampling and back-to-back sampling. Each of the existing sampling methods has its merits. However, none of them successfully satisfies all the favorable requirements.The following definitions describe these techniques.

***Definition** 2:* The deterministic **event-driven sampling method** with sampling period p (**Event(**p**)**) samples events numbered 0, p, $2p$ and so on. In the case of network traffic, events are packet arrivals.

***Definition** 3:* The **random sampling method** is a variant of the event-driven sampling method where its sampling interval is a random variable following a specific distribution.

***Definition** 4:* The **configured run-length sampling method** (**Conf(**p,q**)**) with sampling period p and run length q samples a sequence of q events in every sampling cycle. If the sampling starts on packet 0, then in the $(k+1)$-th sampling cycle ($k \geq 0$), Conf(p,q) will sample packets numbered kp, $(kp+1), \ldots, (kp+q-1)$.

***Definition** 5:* The **back-to-back sampling method** (**back-to-back(**p**)**) with sampling period p samples packets numbered 0, 1, p, $(p+1)$, $2p$, $(2p+1)$ and so on. Note that back-to-back(p) is equivalent to Conf(p,2). In general, a back-to-back sampling with sampling period p and step s (**back-to-back(**p, s**)**) samples packets numbered 0, s, p, $(p+s)$, $2p$, $(2p+s)$ and so on.

These different techniques have been evaluated in past work. Claffy et al. [5] compared several sampling methods by their errors on estimating packet interarrival times and packet sizes. This study concluded that the event-driven sampling method performs better than other methods and that the performance differences between sample selection patterns are small. Today's routers incorporate sampling techniques similar to those described in [5]. Cisco's *NetFlow* monitoring system supports 1 out of p packets, i.e., Event(p) [1]. Juniper's routers provide some additional flexibility. They allow administrators to apply packet filters before the sampling is done and to request that a configured run length of packets be collected with each sampling event [2], i.e., Conf(.,.). The ability to collect a set of packets with each sample enables the evaluation of temporal correlations between transmissions. However, this ability comes at the cost of recording significantly more data.

Event-driven sampling methods have great difficulty in measuring traffic characteristics such as packet interarrival time. The problem is that the sampling only gives information about the interarrival time between samples, rather than that between back-to-back packet pairs. In [6], interarrival times of the packets between two adjacent packet samples were assumed to be the same, and were estimated by dividing the sampled interarrival time by the number of gaps in between (**naive averaging estimation**). The estimated distribution is biased toward the overall mean of the interarrival time and does not give enough emphasis at the extreme values as we will show later in Fig. 3.

Back-to-back sampling can provide a good estimate of interarrival times. However, it only gives us information about packets 1-step away and does not give information about packets separated by more steps which is important if sequential packet arrivals are correlated.

Random sampling and configured run-length sampling could provide n-step histograms. However, their computation overhead can be high, and for random sampling, the size (number of samples) of the collected histograms is not predictable.

III. Proposed Method

Unlike previous work, our main goal is to provide a sampling method that provides accurate statistical estimation, and is also simple, predictable and capable of capturing temporal correlation. To achieve these objectives, we propose a fast, correlation-aware sampling method (*FastCARS*).

We propose to use a combination of multiple deterministic event-driven sampling processes with sampling intervals that are *relatively prime numbers*. For every sampled packet, its header information, such as time stamp on arrival, packet size, source/destination addresses, source/destination ports, and protocol, is stored for subsequent processing.

Definition** 6:* ***FastCARS$(p_1, p_2, \ldots, p_n)$ sampling method consists of n event-driven sampling processes, where $p_1, \ldots, p_n$ are relatively prime numbers. The i-th process has sampling period p_i, which takes one sample every p_i events.

Definition** 7:* *FastCARS*$(p_1, p_2, \ldots, p_n)$ starts all n sampling processes at the same time. We can further generalize *FastCARS* by specifying the start times of the n processes. We denoted the generalized *FastCARS* by ***GFastCARS$(p_1, p_2, \ldots, p_n, s_1, s_2, \ldots, s_n)$, where packet s_i is the first packet sampled by the i-th process.

The next lemma shows that *GFastCARS* reinforces previous sampling methods and includes them as special cases.

Lemma** 1:* ***GFastCARS *GFastCARS*$(p_1, \ldots, p_n, s_1, \ldots, s_n)$ includes other deterministic sampling methods as special cases:

- Event(p) = *GFastCARS*$(p, 0)$
- back-to-back(p, s) = *GFastCARS*$(p, p, 0, s)$
- Conf(p, q) = *GFastCARS*$(\underbrace{p, \ldots, p}_{q}, 0, 1, \ldots, (q-1))$ ■

Fig. 1 shows how *FastCARS* works. As shown, the *FastCARS*(3,4) method samples at periods of 3 and 4 packet arrivals. The figure also shows that the samples collected are either 1-step, 2-steps, or 3-steps away from each other, allowing the corresponding n-step histograms.

In general, when the sampling periods $(p_1, \ldots, p_n)$ are chosen to be relative primes, $p_{min} = \min_{i=1 \ldots n} p_i$ and $L = lcm(p_1, \ldots, p_n)$, *FastCARS* guarantees us samples of steps ranging from 1 to p_{min} every L packet arrivals. This creates a more predictable sampling result, which random sampling can not give us. *FastCARS* is also tunable in the sense that the sampling intervals can be chosen such that samples of particular steps which are of special interests will occur more often.

FastCARS is a simple generalization of the event-driven sampling which can be efficiently implemented. Event-driven sampling of sampling interval p can be implemented using a counter to keep track of how many packets to be skipped before taking the next sample. *FastCARS* could be implemented similarly with one counter per sampling process.

Fig. 2 compares *FastCARS* with other sampling methods, namely, **event-driven**, **back-to-back**, and **configured run-length** sampling. Configured run-length sampling (Conf(p,q))

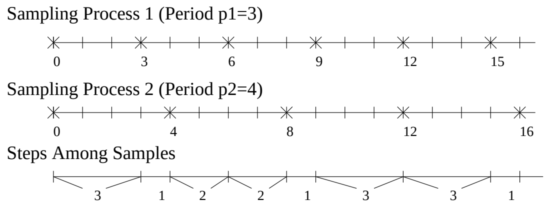

Sampling Process 1 (Period p1=3)
0
3
6
9
12
15
Sampling Process 2 (Period p2=4)
0
4
8
12
16
Steps Among Samples
3
1
2
2
1
3
3
1

Fig. 1. How *FastCARS* works *FastCARS*(3,4) has two processes with sampling intervals 3 and 4. The top two lines indicate the packet numbers sampled by the two processes. The bottom line shows the steps among samples. In this case, we collect interarrival times of 1, 2, and 3 steps, each of them twice, in every 12 packet arrivals.

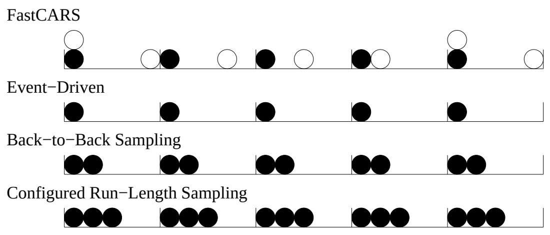


Fig. 2. Comparison of Different Sampling Methods Each bin contains n packets (not shown). Sampling methods compared are *FastCARS*(n, $n-1$), event-driven, back-to-back, and the configured run-length (with run-length 3) sampling. Each ball shown (either filled or empty) indicates a sample taken. For *FastCARS* , filled balls are samples of the sampling process of sampling interval n, and empty balls are those of interval $(n-1)$.

and random sampling also give us n-step histograms. However, *FastCARS* has important advantages over these techniques. *FastCARS* is computationally simpler than random sampling. In addition, random sampling does not provide guarantees about the sampling rate for different n-step histograms. *FastCARS* includes Conf(p,q) as a special case. The major problem of Conf(p,q) is that it requires bursts of recording activity (no action for (p-q) events, frenetic action for the next q events). *FastCARS* spreads the recording activity evenly (Fig. 2). To collect an n-step histogram, Conf(p,q) must be configured to collect q=(n+1) packets at a time, which implies if histograms of large n($\geq p$-1) are needed, Conf(p,q) must collect *every* packet.

Thanks to the flexibility of *FastCARS*, we can tailor its parameters to the application. For example, if samples of n consecutive packets are needed, we can run *FastCARS* with n sampling processes. The sampling rate of each process could be tuned to meet the data storage limitation using the following lemma of data reduction rate.

***Lemma** 2:* **Data Reduction Rate** *FastCARS*($p_1, \dots, p_n$), where gcd(p_i, p_j)=1 for $1 \leq i \neq j \leq n$, takes samples at an average rate of $\frac{1}{\sum_{i=1,\dots,n} \frac{1}{p_i}}$. ■

IV. Experimental Results

We present experimental results showing that information collected by *FastCARS* can be used for typical measurement applications as well as novel data mining applications. In particular, we show that *FastCARS* gives accurate estimation of interarrival time distribution and provides n-step histograms for inspecting correlation at multiple aggregation levels. Results on other data mining tasks such as finding relations between packet size and interarrival time can be found in [7].

TABLE I
Summary of Traces

Trace	Location	Collected Time (GMT)	Link Speed
AIX	AIX/MAE-West Interconnection	Sunday June 10 2001 15:55:50	OC12c PoS
COS	Colorado State University	Monday August 20 2001 00:47:57	OC-3
IND	QuestPOP at IUPUI (Indianapolis)	Tuesday August 21 2001 22:47:04	OC12c ATM

Our experiments are done on the packet header traces obtained from the National Laboratory for Applied Network Research (NLANR[1]). Traces are 90-secs long. A previous study [8] suggests that the network is relatively stable within time spans shorter than 15 minutes. We, therefore, assume the measured packet arrivals form a stationary process. Experiments are done mainly on three traces, which we name **AIX**, **COS** and **IND**. Table I summarizes the details of these traces. The trace collectors are located at aggregation points within HPC networks, the vBNS and Internet2 Abilene. Therefore, the network traffic considered in this paper is traffic in which many independently originated flows are multiplexed.

A. Accuracy of FastCARS: Interarrival Time Distribution

In this section, we show that *FastCARS* can give an accurate estimation of the interarrival time distribution.

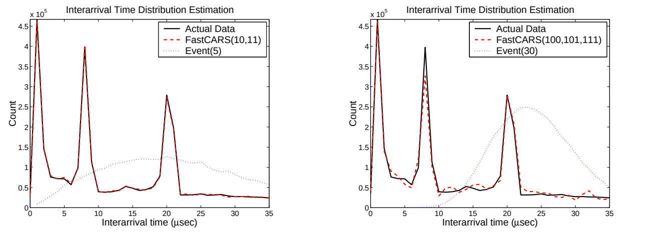
Fig. 3 compares our estimation with the actual interarrival time distribution collected from the full trace (AIX), and also with the results from the event-driven sampling method. Results on traces COS and IND are similar and not shown here.

We use the 1-step histogram collected by *FastCARS* to estimate the interarrival time distribution. As mentioned in Section III, relatively prime sampling intervals guarantee collections of 1-step histograms. We investigate the effects of different numbers of processes and different sampling intervals on *FastCARS*, and choose sampling intervals (10,11) and (100,101,111) as examples to demonstrate how *FastCARS* works. The estimation of interarrival time using samples from event-driven sampling is done by the naive averaging estimation (Section II). Our comparison of the quality of the estimation from different sampling methods is fair, allowing for a similar number of samples for all methods. For example, *FastCARS*(10,11) takes 713,435 samples on trace AIX, and it is compared with Event(5), which takes 747,407 samples. Even with slightly more samples, Event(5) still performs badly. Estimates from *FastCARS* samples are very close to the actual distribution, while, those from event-driven sampling are biased towards the distribution mean.

B. Testing the Independence Hypothesis of Packet Arrivals

The hypothesis that packet arrivals are independent facilitates tasks such as traffic analysis and modelling. However, is

[1] http://moat.nlanr.net/Traces/Traces/



(a) FastCARS(10,11) - Event(5) (b) FastCARS(100,101,111) - Event(30)

Fig. 3. Estimating Interarrival Time Distribution (Trace AIX) *FastCARS* gives better estimation that event-driven sampling does. (a) *FastCARS*(10,11) VS Event(5) sampling, (b)*FastCARS*(100,101,111) VS Event(30) sampling.

this assumption realistic? A connection usually sends a flow of packets and the transmission times and contents of these packets are not independent. *Do these packets make the independence hypothesis false? Are there other dependences among packets?* In this section, we show how the histograms gathered from *FastCARS* can answer these questions.

We use the 1-step and 2-step histograms collected by *FastCARS* to check the independence hypothesis of packet arrivals. The idea is as follows. *The distribution (histogram) of the 2-step interarrival time will be similar to the convolution of the 1-step interarrival time distribution (histogram), if the (1-step) packet interarrival time is independent.* In the remainder of this paper, we refer to this test of independence as **convolution test**.

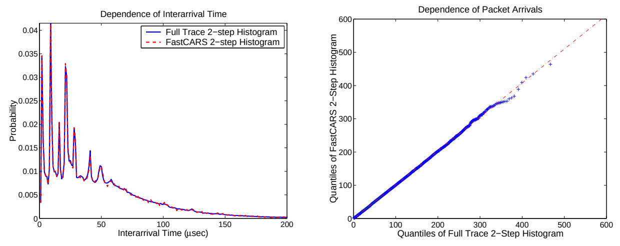
More formally, let $f_i(.)$ be the probability mass function of the i-step interarrival time distribution (time is discretized into micro-seconds). For 3 consecutive packet arrivals (x_1, x_2, x_3), let T_1 (T_2) be the random variable of the interarrival time between x_1 and x_2 (x_2 and x_3). T_1 and T_2 follows $f_1(.)$. Let D be the random variable of the 2-step interarrival time between x_1 and x_3, and D follows $f_2(.)$.

Definition 8: **Convolution Test** If the packets' (1-step) interarrival times are independent and identical distributed, then the probability distribution of 2-step interarrival time (f_2) is the convolution of the 1-step interarrival time distribution (f_1). This is due to

$$\begin{aligned} f_2 &= Pr(D=d) = Pr(T_1+T_2=d) = \sum_{t=0}^{d} Pr(T_1=t, T_2=d-t) \\ &= \sum_{t=0}^{d} Pr(T_1=t)Pr(T_2=d-t) = f_1 \otimes f_1, \end{aligned}$$

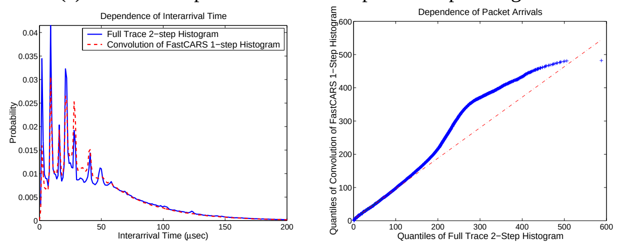
where $Pr(E)$ denotes the probability of an event E, and $\otimes$ is the convolution.

Fig. 4 compares the 2-step histogram and the convolution of the 1-step histogram, both obtained from *FastCARS*(10,11), with the actual 2-step histogram collected from full trace AIX. Results on traces COS and IND are similar and not shown here. The histograms are normalized before doing convolution and comparison. We use the quantile-quantile plot (QQ-plot) [9] of the two histograms as a visualization of the similarity between two histograms, which is actually related to the Kolmogorov-Smirnov test of similarity of two distributions [10]. The fit of the QQ-plot to the 45-degree line demonstrates the goodness of fit between two histograms.



(a.1) Interarrival Time Histogram (a.2) QQ-plot

(a) Actual 2-Step & *FastCARS* Sampled 2-Step Histograms



(b.1) Interarrival Time Histogram (b.2) QQ-plot

(b) Actual 2-Step & Convoluted Sampled 1-Step Histograms

Fig. 4. Dependence of Interarrival Time (Trace AIX) Histograms are collected by *FastCARS*(10,11). Compares actual 2-step interarrival time histogram to (a) the sampled 2-step interarrival time histogram, and (b) the convoluted 1-step interarrival time histogram. Correlation coefficients of the QQ-plots: (a.2) 0.99994, (b.2) 0.99368.

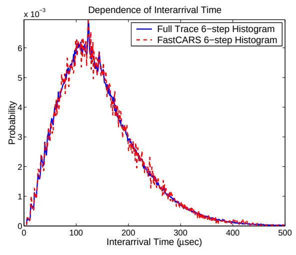
The QQ-plot in Fig. 4(a.2) goes along the 45^o line indicates that *FastCARS* gives accurate 2-step interarrival time distributions. The big deviation from the 45^o line shown in Fig. 4(b.2) indicates that the actual 2-step histogram is different from the convolution of the 1-step histogram. Therefore, by the convolution test, successive interarrival times are not independent.

As the number of steps increases, packet arrivals should be less dependent on each other. For example, packets 3 steps away are expected to be more independent of one another and, as a result, the sum of two 3-step interarrival times should be a good estimation of a 6-step interarrival time. This suggests that the convolution of 3-step histogram should be similar to the 6-step histogram.

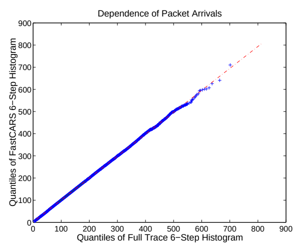
Fig. 5 shows the results on comparing the actual 6-step histogram from full trace (AIX) to (a) the sampled 6-step histogram and (b) the convolution of the sampled 3-step histogram. The 6-step interarrival time histogram from *FastCARS* is still a good estimation for the actual 6-step interarrival time distribution (Fig. 5(a)). In Fig. 5(b.2), the actual 6-step histogram fits well with the convolution of the 3-step histogram and is better than the fit in Fig. 4(b.2). That is, the correlation coefficient of the QQ-plot in this case, 0.99949, is much closer to 1 than the case of Fig. 4(b.2), which is 0.99368. This shows that, as expected, the dependence of packet arrival time diminishes as the separation between packets increases.

C. Dependence Assumption and "Packet Train" Phenomenon

A sequence of packets with same source, destination IP addresses and port numbers form a "packet train". It is known that the packet train phenomenon exists in network traffic [3]. Packets within a packet train are not expected to act indepen-

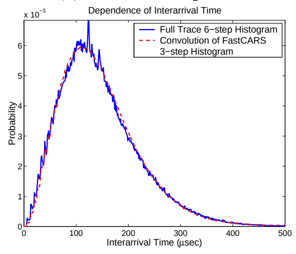


(a.1) Interarrival Time Histogram

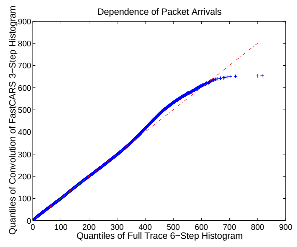


(a.2) QQ-plot

(a) Actual 6-Step & *FastCARS* Sampled 6-Step Histograms



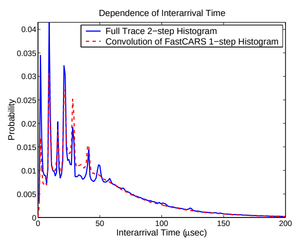
(b.1) Interarrival Time Histogram



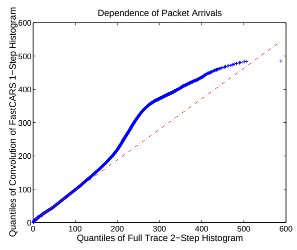
(b.2) QQ-plot

(b) Actual 6-Step & Convoluted Sampled 3-Step Histograms

Fig. 5. Dependence of Interarrival Time (Trace AIX) Histograms are collected by *FastCARS*(10,11). Compares the actual 6-step interarrival time histogram to (a) the sampled 6-step interarrival time histogram, and (b) the convoluted 3-step interarrival time histogram. Correlation coefficient of the QQ-plot in (b.2): 0.99949.



(a) Interarrival Time Histogram



(b) QQ-plot

Actual 2-Step & Convoluted Sampled 1-Step Histograms

Fig. 6. Packet Train and Dependence of Interarrival Time (Trace AIX, with packet trains removed) Histograms are collected by *FastCARS*(10,11). The number of packets removed is 155863, out of the total 3737038 packets. Correlation coefficient of the QQ-plot in (b.2): 0.99683.

dently, and may affect traffic characteristics. It may cause the independence hypothesis of packet arrivals to be incorrect.

Is it true that packet trains are the main reason that the independence assumption of packet arrivals is false? We test this hypothesis by removing consecutive packets in the same flow (packet trains) from the full trace, and then check whether (1-step) interarrival time histogram of the resulting trace set has the independence property. The independence check is done using our *convolution test*.

Fig. 6 shows the result of the convolution test on trace AIX, after packet trains are removed. Since the discrepancy in the QQ-plot in Fig. 6 remains significant (compared to Fig. 4(b.2)), the removal of packet trains does not make packet arrivals independent. This suggests that packet trains might not be the sole cause of the failure of the independence hypothesis of packet arrival.

V. Conclusions

In this paper, we present *FastCARS*, a fast, correlation-aware sampling method for network data mining, which is (1) **accurate** in providing traffic statistics, (2) **simple and scalable** for implementation, (3) **correlation-aware** in the sense that it easily captures information about n-step histograms and, therefore, reveals short and long term correlations among packet arrivals, (4) **non-bursty** since it evenly spreads the sampling efforts over time, and (5) **general** since it includes other deterministic sampling methods as special cases.

Using the information obtained from *FastCARS*, we also provide several new tools for network data mining, namely, the **n-step histograms** (Section II, definition 1), and the **convolution test** (Section IV-B, definition 8).

In addition, *FastCARS* and our tools enable the following observations on real-world traffic traces:

1) *FastCARS* preserves traffic characteristics and accurately estimates the interarrival time distribution (Section IV-A).
2) The assumption of independent arrivals is not correct (Section IV-B).
3) Packet trains are not the sole cause of the failure of the independence hypothesis of packet arrival (Section IV-C).

FastCARS can also be used in other areas that demand accurate, efficient, and correlation-aware sampling techniques. For example, *FastCARS* could be used to compare synthetic traces from traffic generators to real-world data. This would ensure that the traffic generators create traces that have the appropriate temporal correlations as well as the normally tested long-term aggregate distributions.

References

[1] Cisco Systems Inc., "Netflow services solutions guide," http://www.cisco.com/univercd/cc/td/doc/cisintwk/intsolns/netflsol/nfwhite.htm, August 2001.

[2] Juniper Networks Inc., "Junos 5.0 internet software configuration guide: Configure traffic sampling and forwarding," http://www.juniper.net/techpubs/software/junos50/swconfig50-interfaces/html/sampling-config.html, August 2001.

[3] R. Jain and 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.

[4] W. E. Leland, M. S. Taqqu, W. Willinger, and D. V. Wilson, "On the self-similar nature of ethernet traffic," *IEEE/ACM Transactions on Networking*, vol. 2, no. 1, pp. 1–15, February 1995.

[5] Kimberly C. Claffy, George C. Polyzos, and Hans-Werner Braun, "Application of sampling methodologies to network traffic characterization," *Proceedings of SIGCOMM 93*, pp. 194–203, 1993.

[6] Kedar Dhandhere, Hyang-Ah Kim, and Jia-Yu Pan, "The application and effect of sampling methods on collecting network traffic statistics," http://www.cs.cmu.edu/∼jypan/writing/network_sampling.ps.gz, unpublished, May 2001.

[7] Jia-Yu Pan, Srinivasan Seshan, and Christos Faloutsos, "Fastcars: Fast, correlation-aware sampling for network data mining," Tech. Rep., CMU-CS-02-167, Carnegie Mellon University, 2002.

[8] Vern Paxson, *Measurements and Analysis of End-to-End Internet Dynamics*, Ph.D. thesis, UC Berkeley, April 1997.

[9] John M. Chambers, W. S. Cleveland, B. Kleiner, and P. Tukey, *Graphical Methods for Data Analysis*, Wadsworth and Brooks/Cole, 1983.

[10] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery, *Numerical Recipes in C, Second Edition*, Cambridge University Press, 1992.