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

Jia-Yu Pan, Srinivasan Seshan, Christos Faloutsos [1]

December 2002
CMU-CS-02-167

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

[1] 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. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation, DARPA, or other funding parties.

Keywords: traffic analysis, network data mining, sampling

Abstract

Measuring traffic on routers is vital for finding patterns, traffic modeling, and anomaly detection. Unfortunately, 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. Real world data is seldom temporally independent and data observed at one time is likely to have important correlations with data observed at close-by instants in time. A good sampling method should be able to give measurements that take this correlation into account. 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 full trace's statistics, (b) is simple and scalable for implementation, (c) captures correlations between successive packets, as well as packets that are further apart, (d) evenly separate sampling efforts over time, and (e) generalizes previously proposed sampling methods and includes them as special cases.

We also propose several new tools for network data mining and demonstrate the good quality of the information provided by *FastCARS*. These tools include: (a) The *n-step histograms* which give correlated statistics at different levels of temporal correlation, (b) the *convolution test* which could be used to examine the dependence level between packet arrivals. (c) the *n-step packet-size/delay graph* which provides accurate bandwidth estimation and load monitoring, and (d) the *n-step flow graph* which effectively visualizes flow patterns hidden in a trace.

The experimental results on multiple, real-world datasets (479Mb 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. The provided tools may be useful in applications such as monitoring link load and traffic flows.

1 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 [4] [5] 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 [6] and arrival correlation [7], 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. It's performance should be predictable, in the sense that the collection of desired statistics is guaranteed after a certain amount of samples taken.

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. Besides, we propose 2 novel tools for network data mining, based on the information obtained from *FastCARS*. We demonstrate the use of these tools which leads to better insights of the network traffic.

This report is organized as follows. We summarize previous work in section 2. Section 3 describes our sampling method. Section 4 presents results from using our sampling technique. Several tools for network data mining are introduced in section 5. Finally, we conclude in Section 6.

2 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 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. [2] 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 (deterministic or random) are small. Today's routers incorporate sampling techniques similar to those described in [2]. Cisco's *NetFlow* monitoring system supports 1 out of p packets, i.e., Event(p) [4]. 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 [5], 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

Table 1: Comparing the sampling methods

Property	Event(p)	Back-to-back	Random	Conf(p,q)	*FastCARS*
1-step histogram	×	○	○	○	○
n-step histogram (n>1)	×	×	○	○	○
Scalability	○	○	○	×	○
Predictability	○	○	×	○	○

sampling only gives information about the interarrival time between samples, rather than that between back-to-back packet pairs. In [3], 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.

Table 1 summarizes the comparison of these sampling methods.

3 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.

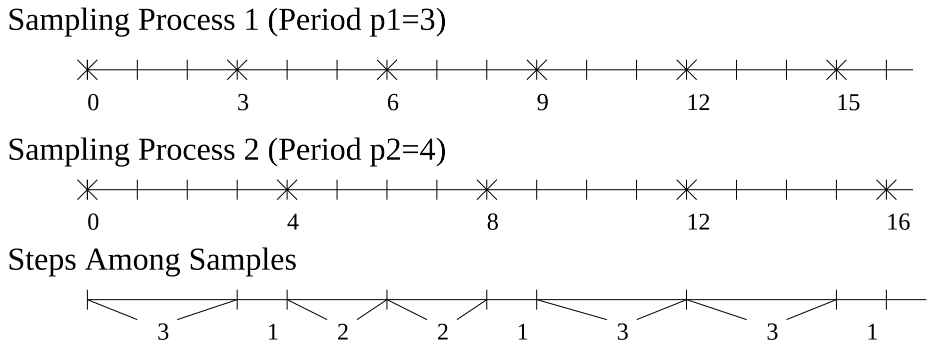


Figure 1: How *FastCARS* works *FastCARS*(3,4) has two processes with sampling intervals 3 and 4. The top two lines indicate the packet numbers sampled by the two processes. The bottom line shows the steps among samples. In this case, we collect interarrival times of 1, 2, and 3 steps, each of them twice,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

Fig. 2 compares *FastCARS* with other sampling methods, namely, **event-driven**, **back-to-back**, and **configured run-length** sampling. Configured run-length sampling (Conf(p,q)) 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 $n(\geq p\text{-}1)$ 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 (the maximum rate the samples can be collected, or the minimum rate the original data has to be reduced) using the following lemma of data reduction rate.

Lemma 2 ***(Data Reduction Rate)*** *FastCARS*($p_1, \ldots, p_n$), *where gcd*(p_i, p_j)

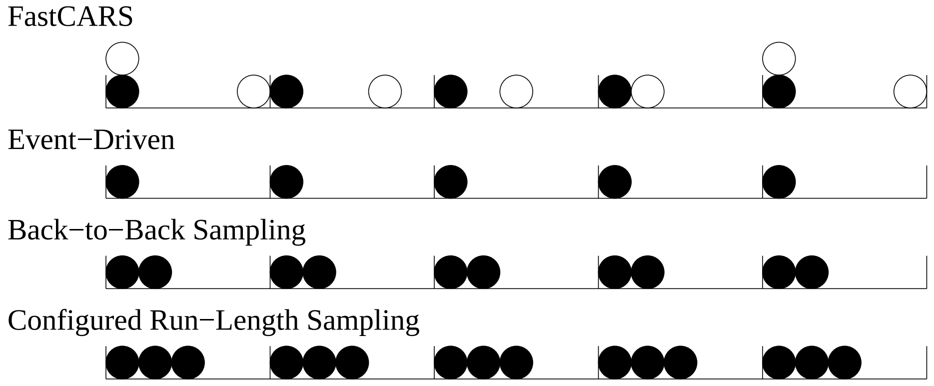


Figure 2: Comparison of Different Sampling Methods Each bin contains n packets (not shown). Sampling methods compared are $FastCARS(n, n-1)$, event-driven, back-to-back, and the configured run-length (with run-length 3) sampling. Each ball shown (either filled or empty) indicates a sample taken. For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

=1 for $1 \leq i \neq j \leq n$, *takes samples at an average rate of* $\frac{1}{\sum_{i=1,\ldots,n} \frac{1}{p_i}}$. ■

4 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 this section, we show that *FastCARS* gives accurate estimation of interarrival time distribution and provides n-step histograms for inspecting temporal correlation at multiple aggregation levels. Besides, the information provided by *FastCARS* make possible data mining tasks on network traffic, which will be discussed in the next section.

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

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

Table 2: 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

2 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.

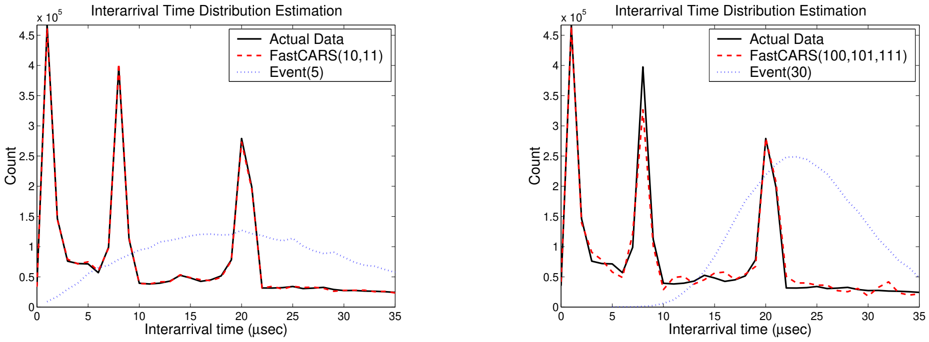
In the following, we present experiments to answer the following questions: *How accurate is the information FastCARS provides? Do the samples preserve the main characteristics of full trace? How can we use the n-step histograms to explore the correlation among packet arrivals?*

4.1 Accuracy of *FastCARS*: Interarrival Time Distribution

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

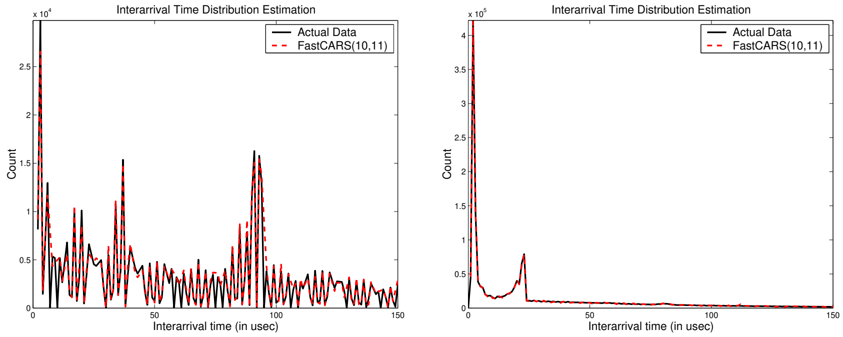
Fig. 3 compares our estimation with the actual interarrival time distribution collected from the full trace (AIX), and also with the results from the event-driven sampling method.

We use the 1-step histogram collected by *FastCARS* to estimate the interarrival time distribution. As mentioned in Section 3, 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 2). Our comparison of the quality of the estimation from different sampling methods is fair, allowing for a similar number of samples



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

Figure 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.



(a) Trace COS   (b) Trace IND

Figure 4: Estimating Interarrival Time Distribution (Traces COS and IND) *FastCARS*(10,11) also gives accurate estimations on these traces.

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.

Figure 4 shows that *FastCARS* also gives accurate estimation of interarrival time distribution on traces COS and IND. The actual interarrival time distribution is compared with the estimation obtained by using the 1-step histogram of *FastCARS*(10,11).

4.2 Testing the Independence Hypothesis of Packet Arrivals

The hypothesis that packet arrivals are independent facilitates tasks such as traffic analysis and modelling. However, is 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. 5 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) [1] 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 [9]. The fit of the QQ-plot to the 45-degree line demonstrates the goodness of fit between two histograms.

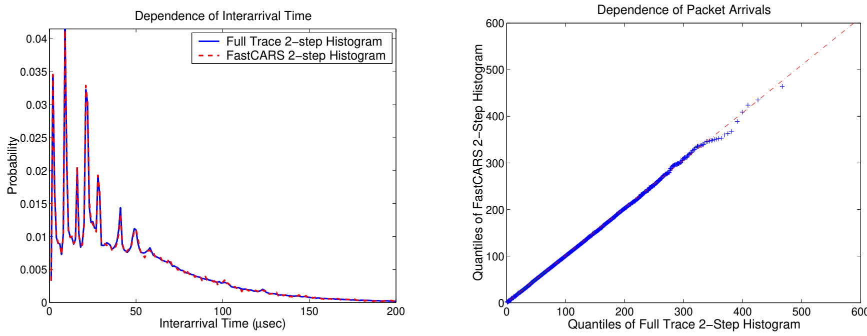
The QQ-plot in Fig. 5(a.2) goes along the 45^o line indicates that *FastCARS* gives accurate 2-step interarrival time distributions. The big deviation from the 45^o line shown in Fig. 5(b.2) indicates that the actual 2-step histogram is different from the convolution of the 1-step histogram. Therefore, 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. 6 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. 6(a)). In Fig. 6(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. 5(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. 5(b.2), which is 0.99368. This shows that, as expected, the dependence of packet arrival time diminishes as the separation between packets increases.

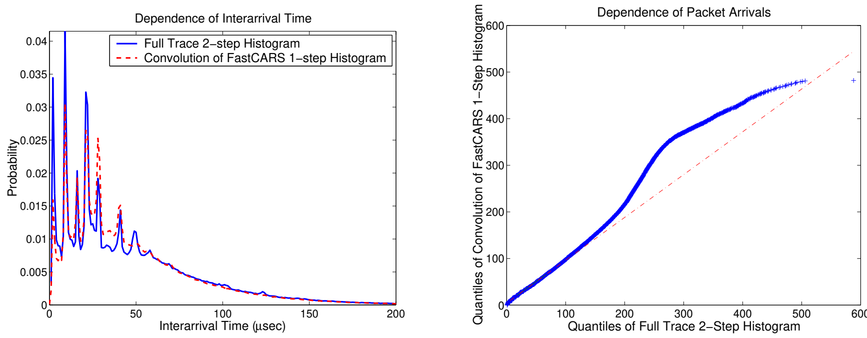
4.3 Dependence Assumption and "Packet Train" Phenomenon

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 [6]. Packets within a packet train are not expected



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

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



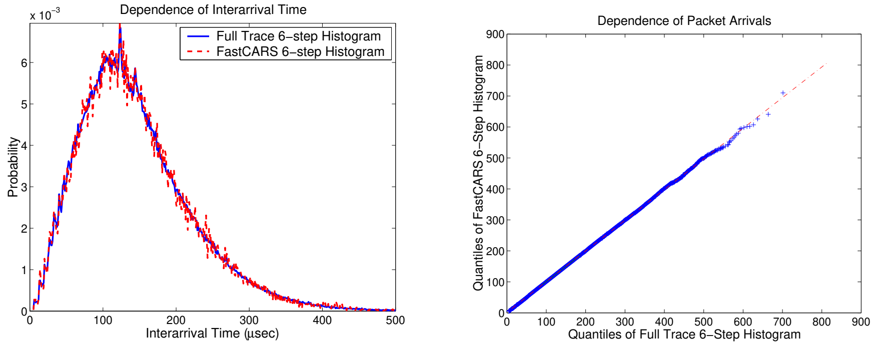
(b.1) Interarrival Time Histogram (b.2) QQ-plot

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

Figure 5: 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.

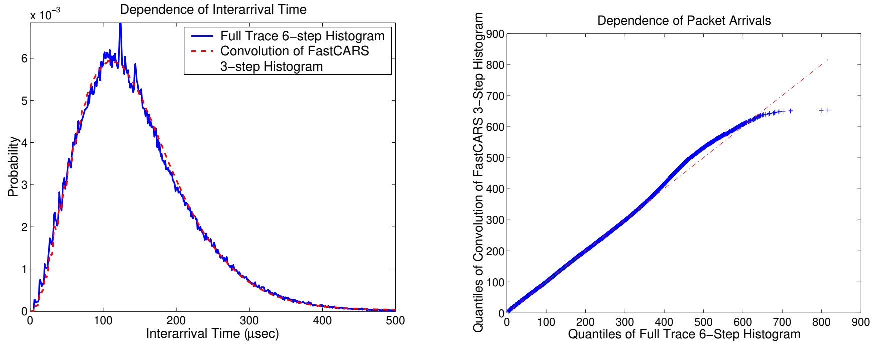
to act independently, and may affect traffic characteristics. It may cause the independence hypothesis of packet arrivals to be incorrect.

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*.



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

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

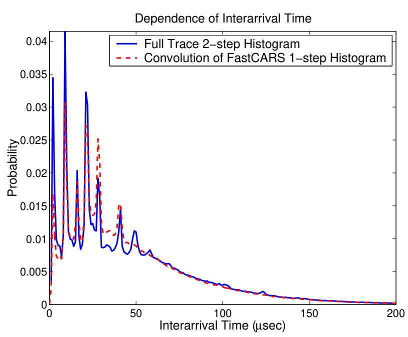


(b.1) Interarrival Time Histogram (b.2) QQ-plot

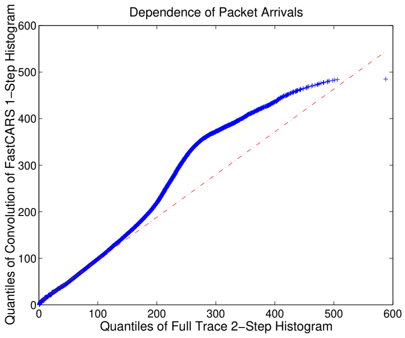
(b) Actual 6-Step & Convoluted Sampled 3-Step Histograms

Figure 6: 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.

Fig. 7 shows the result of the convolution test on trace AIX, after packet trains are removed. Since the discrepancy in the QQ-plot in Fig. 7 remains significant (compared to Fig. 5(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.



(a) Interarrival Time Histogram



(b) QQ-plot

Actual 2-Step & Convoluted Sampled 1-Step Histograms

Figure 7: 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.

5 Putting *FastCARS* to Work

Beside providing accurate information for the n-step interarrival time distribution, *FastCARS* makes possible data mining tasks of finding correlations among the packet attributes (header fields). In particular, we propose two novel tools for network data mining based on the information given by *FastCARS*, namely, the "n-step packet-size/delay graph" and the "n-step flow graph". We found that the n-step packet-size/delay graph shows how packet size is distributed and how traffic burstiness effects the queueing of packets. The n-step flow graph highlights interesting traffic flow patterns such as "big flows" and "packet trains".

5.1 Observing load patterns of a link: n-step packet-size/delay graph

We are interested in the correlation between the size of a packet and the time delay between this packet and the one that arrives next. Observations of this interaction give insights into questions such as "*Are longer delays caused by router overload? How bursty are traffic patterns? What is the impact of packet size on interarrival time? What is the distribution of packet* *sizes? What percentage of packets of a particular size are queued in the router before being forwarded?*" These insights are also useful in traffic monitoring and capacity planning.

We introduce the n-step packet-size/delay graph to see the interaction of packet size and router's forwarding delay. We begin by defining the word, "*delay*".

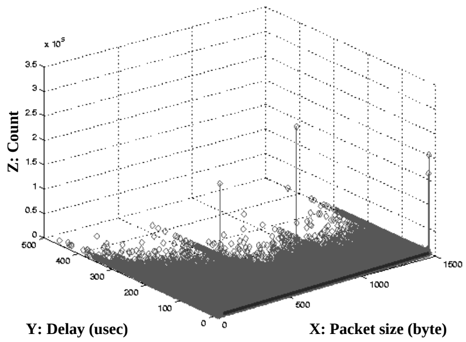
Definition 9 ***(Delay)*** *The difference between the timestamps on packets n steps away is called the* n***-step delay****. Hence, the* 1*-step delay is the difference on arrival time of the two back-to-back packets.*

Definition 10 ***(n-step packet-size/delay graph)*** *Given a set of packet pairs, each has packets n steps apart. A* ***n-step packet-size/delay graph*** *is a 3-dimensional graph, where x-axis is the size of first packet in a packet pair, y-axis is the n-step delay of packets in a pair, and z-axis is the count of a particular (x,y) tuple occurs in the given set of packet pairs. We denoted the n-step packet-size/delay graph as a function $SD_n(x, y, z)$.*

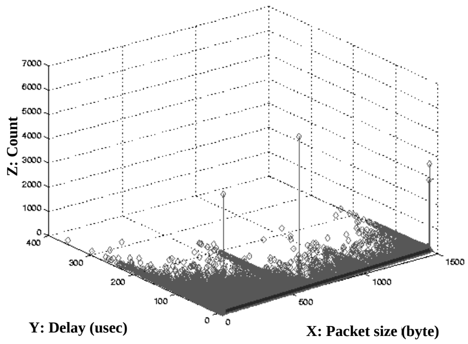
Fig. 8(a) shows the *1*-step packet-size/delay graph of all the packet pairs in the full AIX trace (without sampling). The graph shows interesting features such as spikes at particular packet sizes and the widely spreaded delay values. Compare to Fig. 8(b), where only the 1-step samples collected by *FastCARS*(10,11) are used, we found that the main features of the graph remain even when we are only using samples. This shows that *FastCARS* samples retain the desirable information of the full trace. We can therefore work on the *FastCARS* samples instead of the full trace whose huge amount of data often causes problems in processing and visualization.

Fig. 8(c), (d) are the graphs after we remove points (x,y,z) with small z values in Fig. 8(a), (b), respectively. Specifically, this is done by keeping only points with z that is greater than some thresholds ($z \geq 500$ to get (c) form (a), $z \geq 10$ to get (d) from (b)). The thresholded graphs (Fig. 8(c),(d)) show the significant features in the original graphs (Fig. 8(a),(b)), which are

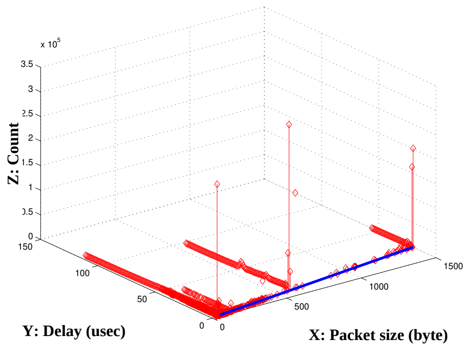
- most packet pairs have the minimal delay (spikes are at locates of minimal delay), and
- the minimum delay within a packet pair increases linearly as the size of first packet in the pair increases.



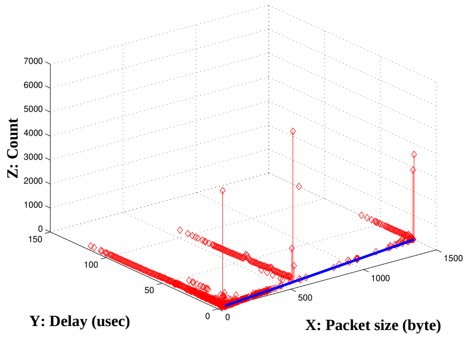
(a)Full Trace



(b)*FastCARS*(10,11) 1-Step Histogram



(c)Cleanup of (a)



(d)Cleanup of (b)

Figure 8: Correlation of Packet Size and Delay: *1*-step packet-size/delay graph (AIX trace) (a) data is from every 1-step packet pair in the full trace; (b) data is from the 1-step samples of *FastCARS*(10,11). (c),(d) is obtained from (a),(b) by dropping points that have count (frequency) less than 500, 10, respectively; The dark solid lines in the graphs are the *(approximated) link capacity lines*, defined by two points (x,y,z)=(40,1,0) and (1500,20,0).

The *1*-step packet-size/delay graph reveals how packet size is distributed. In Fig. 8 we found that the big spikes (counts) are at packet sizes 40, 576, and 1500 bytes. This set of popular packet sizes coincides with the well-known popular packet sizes in the Internet traffic: 40 bytes (TCP/IP header size), 576 bytes (maximum segment size of TCP/IP packet) and 1500 bytes (maximum Ethernet packet size).

The delay between back-to-back packets could be due to the time the second packet sitting in the queue waiting the first packet to complete its transmission. The bigger the size of the first packet is, the longer delay before the second packet can arrive is. On the other hand, the delay could also simply because that the second packet arrives late.

Definition 11 *Let $T_{delay}(p_1, p_2)$ be the delay between two back-to-back packets p_1 and p_2 of sizes x_1 and x_2 bytes, respectively. Then*

$$T_{delay}(p_1, p_2) = T_{transmit}(x_1) + T_{idle}(p_1, p_2), \tag{1}$$

where $T_{transmit}(x_1)$ is the time to transmit the entire packet p_1 (x_1 bytes) in the link, and $T_{idle}(p_1, p_2)$ is the time that the link is idle, i.e., the time interval from the time p_1 completes its transmission until p_2 arrives.

If the link is busy, the packets arrivals are not sparse and $T_{idle}(p_1, p_2)$ is small or zero. The delay between back-to-back packets are bounded from below by $T_{transmit}(p_1)$. In a *1*-step packet-size/delay graph, we define the line connecting the points of smallest delay ($T_{transmit}$) at each packet size be the "link capacity line". Often, we can model $T_{transmit}(x_1)$ as

$$T_{transmit}(x_1) = \frac{x_1}{B}, \tag{2}$$

where B is the (fixed) bandwidth of the link. As suggested by the model, we expect the link capacity line to be linear. However, because of the measurement noise, the measured link capacity line is not a straight line. We approximate the measured link capacity line by a straight line as follows:

Definition 12 ***((Approximated) Link capacity line)*** *If the network trace is collected from a busy link, we approximate the link capacity line as the line connecting the two points, each corresponds to the most frequent (packet size, delay) pairs at packet size* 40 *and* 1500*, respectively. Specifically, the two points defining the approximated link capacity line are (40, t_1, 0) and (1500, t_2, 0), where $t_1 = \underset{t}{argmin} SD_n(40, t)$ and $t_2 = \underset{t}{argmin} SD_n(1500, t)$.*

In Fig. 8, the dark solid lines in the graphs are the approximated link capacity line of the AIX trace. The two points which define the approximated link capacity line are (x,y,z)=(40,1,0) and (1500,20,0). In the following, we will use "link capacity line" to refer to the "approximated link capacity line".

We can estimate the link capacity (bandwidth) by (the inverse of) the slope of the link capacity line (Equation 2). For the AIX trace, the estimated link bandwidth is $\frac{(1500-40)}{(20-1)} \times 8 = 614.737$Mbps. Comparing to the actual bandwidth (622.080Mbps, bandwidth of a OC12c link), we found that the link capacity line gives a reasonable estimate of the link speed.

The *1*-step packet-size/delay graph also shows the packets queue up at the router upstream to the measured link. This is shown by the fact that big spikes (counts) are located at the link capacity line, which by definition indicates the $T_{idle} = 0$, i.e., the link is not idle. The zero link idle time suggests that the packets queue up in the upstream router and keep the link busy. We conclude this as an evidence that the traffic is bursty that packets arrive closely in time, and are queued up at the upstream router. However, although most of the packets are transmitted at full speed (inside the traffic bursts), there are packets which have longer delays ($T_{idle} > 0$, points not on the link capacity line). These information shown by the *1*-step packet-size/delay graph is useful for applications such as capacity planning.

In summary, samples given by *FastCARS* retain information of the full trace, therefore, data mining tools can be built on top the information provided by *FastCARS* without losing important features of the traffic. One advantage of building tools on top of the samples is that it is "lighter" and is easier to manipulate than the full trace. Particularly, the network data mining tool *1*-step packet-size/delay graph , built on top of the 1-step samples of *FastCARS*, supports the following observations:

Observation 1 *The most popular packet sizes in the traffic are* 40 *bytes (TCP/IP header size),* 576 *bytes (maximum segment size of TCP/IP packet) and* 1500 *bytes (maximum Ethernet packet size).*

Observation 2 *The (approximated) link capacity line determined by the 1-step packet-size/delay graph gives good estimate of the unknown link bandwidth.*

Observation 3 *The network traffic is bursty. In the 1-step packet-size/delay graph , there are big spikes at the link capacity line which indicates many packets are transmitted at link speed. Bursty traffic is an explanation for this observation that, within traffic bursts, most packets queue up in the upstream router and are transmitted at the link speed.*

5.2 Observing Traffic Flow Patterns: n-step flow graph

For traffic engineering applications, understanding the traffic patterns between communicating end points are important. This understanding could be obtained via observing traffic properties such as "which end point pair has intensive traffic?", "which pair consumes a lot of bandwidth?", "are there hot spots (end points) in the network?", and "do packet trains exist in the traffic?". A tool which reveals these kinds of information is useful for capacity planning and anomaly detection.

In this section, we propose a new tool, ***n*-step flow graph**, for network data mining, which highlights big flows and the existence of packet trains. Built on top of the reliable information provided by *FastCARS* sampling, an n-step flow graph reveals interesting traffic flow patterns which are hidden in the overwhelmingly large amount of trace data and can not be seen if no sampling is done.

We start by defining a couple of terms that will facilitate our following discussions.

Definition 13 ***(Flow)*** *A* ***flow*** *in the traffic is formed by the packets having the same source and destination IP addresses and port numbers. In other words, a flow f is identified by a tuple (sa, sp, da, dp), where sa (sp) is the source address (port number) and da (dp) is the destination address (port number) of the constituent packets of f.*

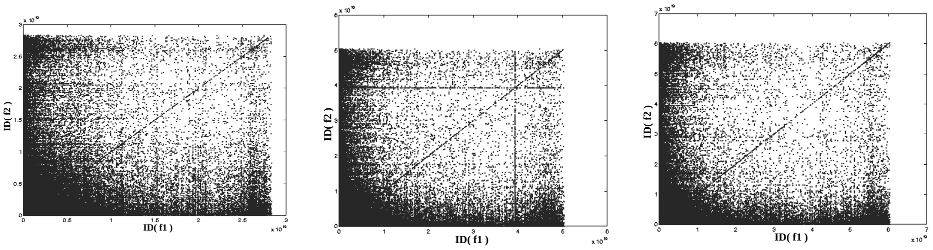
Definition 14 ***(Flow identifier)*** *The flow identifier of a flow f is defined as*

$$ID(f) = n_{sa}(sa) * 10^{15} + n_{sp}(sp) * 10^{10} + n_{da}(da) * 10^{5} + n_{dp}(dp), \tag{3}$$

where $n_{sa}(.)$ is a serialization function which map a source address (sa) to an integer ($n_{sp}(.)$, $n_{da}(.)$ and $n_{dp}(.)$ are similar, but for source port number (sp) and destination address and port number (da, dp), respectively).

Definition 15 ***(n-step flow graph)*** *Let (p_1, p_2) be a pair of packets which are n steps away (p_1 comes before p_2). Let f_1 and f_2 be the flows to which p_1 and p_2 belong, respectively. The n-step flow graph is a scatter plot of $ID(f_2)$ (y-axis) versus $ID(f_1)$ (x-axis), for all pairs of sample packets (p_1, p_2) which are n-step apart.*

In our implementation, each of the functions $n_{sa}(.)$, $n_{sp}(.)$, $n_{da}(.)$, and $n_{dp}(.)$ is implemented as a lookup table. Take $n_{sa}(.)$ as an example and let T_1 be the lookup table corresponds to $n_{sa}(.)$. We start with an empty T_1 and add items into T_1 as we continuously examine newly arrived packets (with or without sampling). Specifically, if source address sa of the new packet p is not in T_1 yet, it is append to T_1, and $n_{sa}(sa)$ is assigned to point to this newly appended item in T_1. Note that in our implementation, the flow identifier roughly captures the popularity and the temporal order among the flows. Because of this first-in-first-assign property, the most popular value of, say source address sa, which will be sampled earlier than other less popular source addresses, will be assigned small values ($n_{sa}(.)$). Flows appears early in time are also likely to have small flow identifiers.



(a) 1-step (b) 2-step (c) 3-step

Figure 9: *n*-step flow graph (AIX trace, 5-sec long, no sampling) Here we shown the n-step flow graphs at (a) n=1, (b) n=2, and (c) n=3. Only a 5-second long portion of the full trace is used, due to the difficulty of handling longer trace without sampling. The horizontal and vertical patterns in (b) are due to big flows. The diagonal patterns at the three graphs suggests the existence of packet trains in the trace.

Fig. 9 shows the 1-step, 2-step and *3*-step flow graph using information from a 5-second period of the full AIX trace (without sampling, about 211K packets). We note that, without sampling, it is difficult to examine a long period of a full trace from a busy link. Because of the large amount of flows a trace contains, computation resources are strained and data visualization becomes difficult before we can go far in time.

In Fig. 9, we observe that there are (1) activities concentrated toward bottom-left of the graphs, which correspond to traffic of from/to the hot spots, (2) pairs of the horizontal and vertical lines (symmetric along the 45^o

line), which capture the existence of big flow, and (3) the diagonal pattern along the 45^o line, which suggests the existence of packet trains.

Observation 4 ***(Hot spots)*** *The activities in the n-step flow graph tend to be concentrated at the bottom-left part. The is due to the way we assign the flow identifiers, where flows sending out from popular source addresses have small flow identifiers. These source addresses are the* ***hot spots*** *in the network, i.e., they contribute more traffic than other end points.*

Observation 5 ***(Big flows)*** *A pair of horizontal and vertical lines, symmetric along the* 45^o *line, in the n-step flow graph indicates a big flow. The vertical line is formed by packet pairs with the first packet belonging to the big flow, while the second one belongs to any of the other flows. Reverse situation for the two packets in a pair causes the horizontal line. As shown, most of these pairs of lines are closed to the x-axis and y-axis and are associated with the hot spots. Big flows might also show up late in the trace, as shown in Fig. 9(b) (flow identifier around* 4×10^{19}*).*

Observation 6 ***(Packet trains)*** *The diagonal pattern along the* 45^o *line in the n-step flow graph suggests the existence of packet trains. The points on the* 45^o *line correspond to pairs of packets belonging to the same flow (same flow identifier). In Fig. 9, we observe that the* 45^o *pattern remains in the graph even when step n is increased from* 1 *to* 3*. We conclude with the following observations:*

- *The pattern spans the full range along the* 45^o *line, suggesting that the packet trains are quite common among pairs of end points.*
- *The pattern does not fade away as step n increases from* 1 *to* 3*, which suggests the packet trains are longer than* 3.

Because of the difficulty of processing the full trace, we need a sampling technique that can provide the same accuracy as we get to analyzing full trace. Information from sampling should still be able to expose hidden, interesting patterns. Fortunately, our *FastCARS* has this desirable property. Therefore, instead of working on full trace data, we can obtain the same results by working on data sampled by *FastCARS*.

Fig. 10 shows the n-step flow graph constructed using the information provided by *FastCARS*(10,11) on three different traces (AIX, COS and IND).

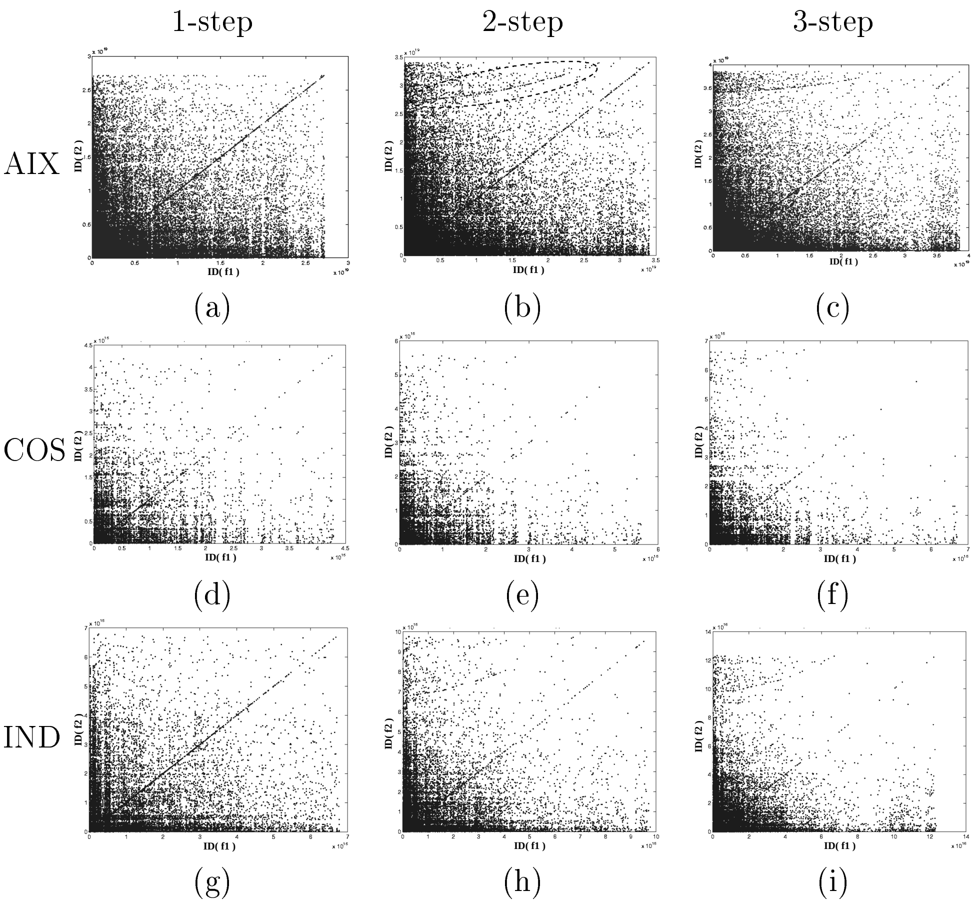


Figure 10: *n*-step flow graph (Trace AIX, COS and IND) Flow information is provided by *FastCARS*(10,11). Columns are graphs of different steps (1-, 2-, 3-step). Rows are results on different traces. Note the interesting pattern revealed by *FastCARS*, which is shown at the top-left corner of (b).

With *FastCARS*, we now can inspect a bigger portion (30 seconds long) of the trace. The resulting graphs are similar to those obtained using full trace information, where important features such as hot spots, big flows, and packet trains are preserved.

In addition, being able to examine longer trace gives more chances on discovering interesting patterns that are originally hidden in the full trace. An example of this is the linear pattern shown at the top-left corner of Fig.

10(b). This pattern suggests a correlation between certain consecutive flows, whose source addresses are the same or similar, and the destination addresses increases from a flow to another. This leads to one potential explanation of this pattern as a port scanning activity. However, the real causes of this pattern are not clear and need further inspection.

Observation 7 ***(FastCARS preserves flow information)*** *The information provided by FastCARS preserves the characteristics of traffic flows. With sampling, a longer trace can be examined to get a more global view of the patterns of traffic flows. Especially, previously hidden patterns may now be discovered.*

6 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 2, definition 1), the **convolution test** (Section 4.2, definition 8), the **n-step packet-size/delay graph** (Section 5.1, definition 10), and the **n-step flow graph** (Section 5.2, definition 15).

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

1. The n-step histogram preserves traffic characteristics and accurately estimates the interarrival time distribution (Section 4.1).

2. Convolution test shows the assumption of independent arrivals is not correct (Section 4.2).

3. Packet trains exist in real-world traffic (Section 5.2) and may contribute to the correlation among packet arrivals.

4. Packet trains are not the sole cause of the failure of the independence hypothesis of packet arrival (Section 4.3).

5. The n-step packet-size/delay graph outlines how packet sizes are distributed and how the burstiness of the traffic affects the link load, which leads to a good estimation of the link bandwidth.

6. The n-step flow graph enables a high-level view of network traffic and highlights hot spots, big flows and the packet trains.

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

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

[3] 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.

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

[5] 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.

[6] R. Jain and S. Routhier. Packet trains - measurements and a new model for computer network traffic. *IEEE Journal of Selected Areas in Communications*, SAC-4(6):986–995, September 1986.

[7] 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*, 2(1):1–15, February 1995.

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

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