An Information Theoretic Approach to Network Trace Compression

Yong Liu[a], Don Towsley[a], Jing Weng[a], and Dennis Goeckel[b]
[a]Computer Science,
University of Massachusetts,
Amherst, MA 01003
{yongliu, towsley, jweng}@cs.umass.edu

[b]Electrical & Computer Engineering,
University of Massachussetts,
Amherst, MA 01003
goeckel@ecs.umass.edu

November 5th, 2004

Abstract

In this paper, we propose an information theoretic framework within which to study the redundancy present in packet header traces. Packet level and flow level models are developed that capture both temporal and spatial correlation present in packet headers, which can be exploited for packet trace compression. Information theoretic bounds are established for lossless packet header compression. Dependencies between the potential compression ratio and network parameters, such as the average route length and average flow size are derived. Valuable insights are obtained to guide the design of efficient packet trace compression algorithms.

1 Introduction

The collection of network traces is essential for the engineering and management of today's networks, and for performing research leading to better traffic models, and network architectures and protocols. The collection of such traces poses tremendous challenges due to the high speeds of current network links. For example, the collection of 60 byte packet headers on an OC-48 link can easily generate 600Gbytes of data in an hour, and even trace collection at a gateway of a university or a company can produce over 30Gbytes of data in an hour. Clearly, the size of such traces precludes their widespread collection, either over long periods of time at a single monitoring site , or concurrently at a large set of distributed monitors without the use of some form of compression.

There exists considerable redundancy in a packet trace collected at a single monitor. Packets from the same flow share the same flow level information, such as source IP, destination IP, port, protocol, etc. In a packet header trace, this shared flow level information is recorded multiple times for each packet belonging to the flow. In addition, flows from the same subnet use only a small range of IP addresses. This applies also to port number as most of the applications tend to use only one of a small number of port numbers, such as 80 for http, 21 for ftp, etc. This suggests that a compression algorithm that removes this redundancy may be able to produce substantially smaller traces.

Observations made from measurements also demonstrate considerable redundancy within a network due to spatial correlation. A single flow can pass through several measurement monitors, and the packets belonging to these flows will contain almost exactly the same packet header information, such as source IP address, destination IP address, packet size, etc. The time stamps for these packets will differ only by small

amounts. This suggests the potential for distributed compression to take advantage of this spatial correlation, resulting in further reduction in trace sizes.

The focus of this paper is to identify and quantify the potential benefits of compression described above. We focus on two scenarios. The first concerns trace collection at a *single* monitor whereas the second concerns the simultaneous collection of traces at a multiple monitors distributed throughout a network. We propose two traffic models with which to examine these scenarios, a *packet-level* and a *flow-level* model. Each of these models the flow of traffic through a network (or a single router) as stochastic processes. We then compute the entropy rates for these stochastic processes, i.e., the information associated with these processes per unit time. Given a trace collected over a finite interval of time, information theory tells us that the raw trace can be compressed to a file of size corresponding to the product of the entropy rate and trace duration. Hence, using these models, we identify the maximum benefit that can be achieved through lossless compression in both the single monitor and distributed monitor settings. Ultimately, we combine the two models into a *hybrid* model before applying them to finite duration traces.

The above models focus on the information contained in the 5-tuple normally associated with an IP flow, namely that containing the source and destination addresses, source and destination ports, and protocol field. We also evaluate the remaining fields of the IP header to determine their information content. We then apply a combined packet-level and flow-level model, augmented to account for the remaining IP header fields to a set of one hour traces collected at the gateway of a research university to identify the potential benefits of lossless compression. We find it possible to compress these traces to files that are roughly $1/8$ the size of the original traces. The evaluation of the benefits of the joint compression of traces collected at multiple sites is more difficult as we do not have access to traces collected at a set of monitors. Instead, we consider several synthetic traces based on topologies obtained through the rocketfuel project, [3]. In this setting, we observe the potential of distributed loss compression to further reduce the marginally compressed traces at the individual monitors by a factor inversely proportional to the average flow path length.

A number of studies have focussed on the development of trace compression algorithms. Both [9], and [13] present flow-based compression algorithms which produce compressed traces that are approximately 25% of the size of the raw trace. Both of these studies focussed on the compression of a trace collected at a single monitor. There has been little work on the problem of the efficient collection of traces at multiple monitors. One exception is the work on *trajectory sampling* [7] in which information regarding a small subset of flows is collected at multiple monitors within the network. This approach records complete packet header information at only one monitor while recording only the information that changes at the remaining monitors. None of these works have attempted to quantify the potential benefits of compression. We will discuss each of these in greater detail elsewhere in the paper.

The remainder of the paper is structured as follows. In Section 2 we present the packet level model. The section concludes with a discussion of some of the deficiencies of this model motivating the flow-level model, which is presented in Section 3. Section 3 concludes with a motivation and description of a hybrid packet/flow model. As these models focus on the header fields ordinarily associated with identifying a flow, Section 4 evaluates evaluates the remaining fields of the IP header and augments the previous packet- and flow-level models to account for them. Section 5 applies these models to a set of one hour traces and synthetic networks and Section 6 discusses the implications of our work to the design of new compression algorithms. Section 7 summarizes the paper.

2 Packet Level Model

In this section we introduce a packet-level trace model with which we determine the gains that can be obtained through the compression of traces collected at a single monitoring point, and more important, through the *distributed compression* of traces collected at monitoring points scattered throughout the network, i.e., trace compression that accounts for the spatial correlation present in a network. Before introducing this model, however, we review key concepts in information theory required by our framework.

2.1 Some Concepts from Information Theory

We begin by introducing the concepts of entropy and entropy rate and their relation to data compression [6].

Definition 1 Shannon entropy. *Let X be a discrete random variable that takes values from χ. Let $p(x) = P(X = x)$, $x \in \chi$. The entropy of X is defined by*

$$H(X) = -\sum_{x \in \chi} p(x) \log_2 p(x)$$

Now consider a stochastic process $X = \{X_n\}_{n=1}^{\infty}$ where X_n is discrete valued.

Definition 2 ***Entropy Rate.*** *The entropy rate of a discrete valued stochastic process X is defined by*

$$H(X) = \lim_{n \to \infty} \frac{H(X_1, X_2, \ldots, X_n)}{n}$$

when the limit exists.

The entropy rate represents the information rate conveyed by the stochastic process X. It provides an achievable lower bound on the number of bits per sample required for lossless compression of the process. With lossless compression, every single bit of data that was originally in the packet header trace remains after the file is uncompressed. All of the information is completely restored.

Definition 3 Joint Entropy Rate. *The joint entropy rate of a collection of many stochastic processes $\{X_n^{(i)}\}_{n=1}^{\infty}, i = 1, 2, ..., N$ is defined by*

$$H(X^{(1)}, X^{(2)}, \ldots, X^{(N)}) = \lim_{n \to \infty} \frac{H((X_1^{(1)}, \ldots, X_n^{(1)}), \ldots, (X_1^{(n)}, \ldots, X_n^{(N)}))}{n} \quad (1)$$

when the limit exists.

The joint entropy rate represents the information rate conveyed by the joint stochastic process. It is also an achievable lower bound on the number of bits required per sample for the joint lossless compression of all the processes.

Let us place this in the context of a network monitoring application. Let X_i be the header of the i-th packet and M the size of the header. $\{X_i\}_{i=1}^{\infty}$ is a stochastic process representing packet headers. We are interested in quantifying the benefit gained from compressing a packet header trace gathered from one network monitor or traces collected at a set of network monitors.

Definition 4 ***Marginal Compression Ratio.*** *Given stationary stochastic process $\{X_i\}_{i=1}^{\infty}$, the marginal compression ratio is defined as the ratio of the entropy rate and record size,*

$$\rho_m(X) = \frac{H(X)}{M}$$

Suppose that we are collecting traces at several points within the network. We are interested in quantifying the benefits of performing distributed compression on these traces. We define,

Definition 5 ***Joint Compression Ratio.*** *Given a collection of N jointly stationary stochastic processes $\{X_i^{(n)}\}_{i=1}^{\infty}, i = 1, 2, \ldots, N$, the joint compression ratio is defined as the ratio of the joint entropy rate and the sum of the entropy rates of the individual processes.*

$$\rho_j(X^{(1)}, X^{(2)}, \ldots, X^{(N)}) = \frac{H(X^{(1)}, X^{(2)}, \ldots, X^{(N)})}{\sum_{i=1}^{N} H(X^{(i)})}.$$

In the context of network trace compression, the joint compression ratio quantifies the potential benefits of performing distributed compression of the traces collected at several point in the network beyond simply compressing each trace independent of each other. According to Slepian and Wolf [5], if two discrete alphabet random variables X and Y are jointly distributed according to some arbitrary probability distribution $p(x, y)$, then X can be compressed without having access to Y without losing any compression performance with respect to the case where X is compressed with access to Y. More formally, without having access to Y , X can be compressed using $H(X|Y)$ bits where

$$H(X|Y) \quad = \quad \sum_{y} P_Y(y) \sum_{x} P_X(x|y) \log_2 P_X(x|y)$$

The quantity, $H(X|Y)$ is often interpreted as the *uncertainty* remaining in the random variable X given the observation of Y. This is the same compression performance that would be achieved if X were compressed while having access to Y . Hence, distributed compression can achieve the same benefits as joint compression. The joint compression ratio reflects the benefits of distributed compression of the traces.

Definition 6 ***Differential Entropy.*** *Let X be a continuous random variable with a density $f(X)$. The differential entropy of X is defined by*

$$h(X) = -\int_S f(x) \log f(x) dx$$

where S is the support set of the random variable.

In reality, every variable is measured with finite resolution. With a resolution of $\Delta = 2^{-n}$, i.e., an n-bit quantization, a continuous random variable X is represented by a discrete random variable X^{Δ}. The following theorem relates the discrete entropy of X^{Δ} to the differential entropy of X.

Theorem 1 *If the density of $f(X)$ of continuous random variable X is Riemann integrable, the entropy of an n-bit quantization of X is approximately $h(X) + n$.*

If X follows an exponential distribution with rate λ, its differential entropy

$$h(X) = -\int_0^\infty \lambda e^{-\lambda x} \log_2(\lambda e^{-\lambda x}) dx = \log_2 \frac{e}{\lambda}$$

With an n-bit quantization, the discrete entropy of X is $H(X^\Delta) = \log_2 \frac{e}{\lambda} + n$. In the following, whenever there is no confusion, we use the notation $H(X)$ for a continuous random variable X to represent its discrete entropy $H(X^\Delta)$.

2.2 Packet-level Model

We model a network as a directed graph $G = (V, E)$, where $v \in V$ represents a router in the network and edge $(v_1, v_2) \in E$ corresponds to a link between routers v_1 and v_2. Let $\mathcal{F}$ denote a set of packet flows that traverse the network. In this section, we make the following assumption on network flows $\{f \in \mathcal{F}\}$:

- Packets from flow f arrive according to a Poisson process with rate λ_f. The packet inter-arrival time is an exponential random variable δ_f.
- The route of a flow f is fixed. It is represented by a tuple $f = (v_1^{(f)}, v_2^{(f)}, \ldots, v_{l_f}^{(f)})$, where $v_j^{(f)}$ is the j-th router traversed by f and l_f is the path length. For each node v, let $C^{(v)} \subseteq F$ denote the set of flows that pass through it.
- There is no packet loss in the network and packets incur constant delay on each link: let $D_{i,j}$ denote the delay that the j-th packet incurs while traversing the i-th link, $i \in E$, we assume that $D_{i,j} = D_i$, $\forall j$.

These assumptions will be relaxed in the following sections and their implications will be studied.

We model packet arrivals by a *continuous time* process. However, packet monitoring tools use a high resolution clock to provide timestamps. For example, packets captured by the Endace DAG card [1] have a time stamp of 64 bits. The most significant 32 bits represent the number of seconds since midnight, January 1st, 1970 and the least significant 32 bits form a binary fraction, representing the factional part of the timestamp in a specified second. Henceforth, we assume all continuous time variables are quantized with 32 bits and the total length of a quantized raw timestamp is 64 bits.

The behavior of the network is described by the stochastic process $\{\phi_j = (\delta_j, \theta_j)\}$ where δ_j is the time between the arrivals of the $j-1$th and the jth packets to the network, and θ_j is the flow identifier (ID) of the j-th packet. Here $\{\delta_j\}$ is a sequence of iid exponential random variables with parameter $\lambda = \sum_{f \in \mathcal{F}} \lambda_f$ quantized using n bits and $\{\theta_j\}$ is an iid sequence of rv's with distribution $P(\theta_j = f) = \lambda_f / \sum_{g \in \mathcal{F}} \lambda_g$.

For now, we ignore all information associated with each packet header except for the flow identifier, which covers five fields within the TCP/UDP/IP packet header: the source IP address, destination IP address, source port, destination port, and protocol. We will observe later (Section 4) that the rest of the header contains little additional information beyond the flow ID and timestamp associated with the packet. Hence the additional header information has little effect on the compression ratio.

Note that the above stochastic process, along with the route information associated with each of the flows provides sufficient information to simulate the network. Suppose that we wish to record a sample path in a compressed format. From Section 2.1 we know that we need a number of bits per packet equal to $H(\phi)$

where

$$
\begin{aligned}
H(\phi) &= h(\delta) + 32 + H(\theta) && (2)\\
&= \log\frac{e}{\lambda} + 32 - \sum_{f\in\mathcal{F}} \frac{\lambda_f}{\lambda}\log\frac{\lambda_f}{\lambda} && (3)\\
&= \sum_{f\in\mathcal{F}} \frac{\lambda_f}{\lambda}(\log\frac{e}{\lambda_f} + 32) && (4)\\
&= \sum_{f\in\mathcal{F}} \frac{\lambda_f}{\lambda} H(\delta_f), && (5)
\end{aligned}
$$

where δ_f is the packet inter-arrival time for a flow f. From the first term in (3), per packet information decreases with the aggregate packet rate. This is because a higher packet rate means shorter packet-interarrival time, and thus the shorter the bit sequence to represent them. The last term in (3) corresponds to the balance between the rates of all flows. The more balanced the flow rates, the longer the bit sequence needed to represent the flow ID. In the most balanced case, $\lambda_f = c, \forall f \in \mathcal{F}$, then we need $H(\theta) = \log_2\frac{\lambda}{c} = \log_2|\mathcal{F}|$ bits for the flow ID. If there are only several, say $m << |\mathcal{F}|$, high rate flows dominate in packet rate, we only need approximately $\log_2|m| < \log_2|\mathcal{F}|$ bits for the flow ID. Equation (5) suggests that per-packet information in an aggregate packet stream equals the average of the per-packet timing information over all component flows. The number of bits per unit time needed for compression is then $\lambda H(\phi)$. Note that this is for the case of constant link delays. If link delays are random and independent of each other, then it becomes necessary to add an information term corresponding to the entropy of the quantized version of the delays.

In practice, we do not have access to $\{\phi_j\}$. Instead, we can instrument the routers to gather packet traces. Note that each link can be modelled as an M/D/∞ system. Since packet arrivals to the network are described by a Poisson process, and network route and delay are fixed, packet arrivals to every router are also described by Poisson processes. Let $\{\phi^{(v)} = (\delta_j^{(v)}, \theta_j^{(v)})\}$ denote the inter arrival times and flow identifiers for the stream of packets entering router v. Here $\{\delta_j^{(v)}\}$ is described by an n-bit quantized exponential distribution with rate $\lambda^{(v)} = \sum_{f\in C^{(v)}}\lambda_f$, and $P(\theta^{(v)} = f) = \lambda_f/\lambda^{(v)}$ for $f \in C^{(v)}$ and zero otherwise. Similar to the network scenario, in order to describe a packet trace collected at node v we need a number of bits per packet equal to $H(\phi^{(v)})$ where

$$
\begin{aligned}
H(\phi^{(v)}) &= h(\delta^{(v)}) + 32 + H(\theta^{(v)})\\
&= \log\frac{e}{\lambda^{(v)}} + 32 - \sum_{f\in C^{(v)}} \frac{\lambda_f}{\lambda^{(v)}}\log\frac{\lambda_f}{\lambda^{(v)}}
\end{aligned}
$$

The rate at which information arrives at node v per unit time is then $\lambda^{(v)}H(\phi^{(v)})$. Note that this is true for the case of constant link delays. In the case that delays on the different links are mutually independent iid sequences of random variables, it is necessary to introduce a new sequence of random variables at each monitor to represent the out of order arrival characteristics of the packets. We will not pursue this in this paper.

In the absence of compression, each packet requires 168 bits, 104 bits to encode the flow identifier and 64 bits for the timestamp. Hence node v generates $168\lambda^{(v)}$ bits of uncompressed trace per unit time. The aggregate rate at which uncompressed trace at the nodes is generated per unit time is $168\sum_{v\in V}\lambda^{(v)}$.

Now we can answer the question: what is the maximum benefit that can be achieved through compression? We have a *marginal compression ratio*[1]

$$\begin{aligned}\rho_m(\phi) &= \frac{\sum_{v\in V}\lambda^{(v)}H(\phi^{(v)})}{168\sum_{v\in V}\lambda^{(v)}} \\ &= \frac{\sum_{v\in V}\{\log\frac{e}{\lambda^{(v)}}+32-\sum_{f\in C^{(v)}}\frac{\lambda_f}{\lambda^{(v)}}\log\frac{\lambda_f}{\lambda^{(v)}}\}}{168\sum_{v\in V}\lambda^{(v)}}\end{aligned}$$

The compression ratio $\rho_m(\phi)$ provides a lower bound on what can be achieved through lossless compression of the original network lossless trace.

We are also interested in quantifying how well marginal compression comes to achieving the entropy rate of the network. We have

$$\rho_j(\phi) = \frac{\lambda H(\phi)}{\sum_{v\in V}\lambda^{(v)}H(\phi^{(v)})} \tag{6}$$

where the numerator is the lower bound on joint compression and the denominator is the lower bound of marginal compression of each trace separately. The compression ratio ρ_j shows the benefit of joint compression. According to (5),

$$\lambda H(\phi) = \sum_{f\in\mathcal{F}}\lambda_f H(\delta_f). \tag{7}$$

Similarly,

$$\lambda^{(v)}H(\phi^{(v)}) = \sum_{f\in C^{(v)}}\lambda_f H(\delta_f). \tag{8}$$

Therefore,

$$\rho_j(\phi) = \frac{\sum_{f\in\mathcal{F}}\lambda_f H(\delta_f)}{\sum_{v\in V}\sum_{f\in C^{(v)}}\lambda_f H(\delta_f)} = \frac{\sum_{f\in\mathcal{F}}\lambda_f H(\delta_f)}{\sum_{f\in\mathcal{F}} l_f\lambda_f H(\delta_f)} \tag{9}$$

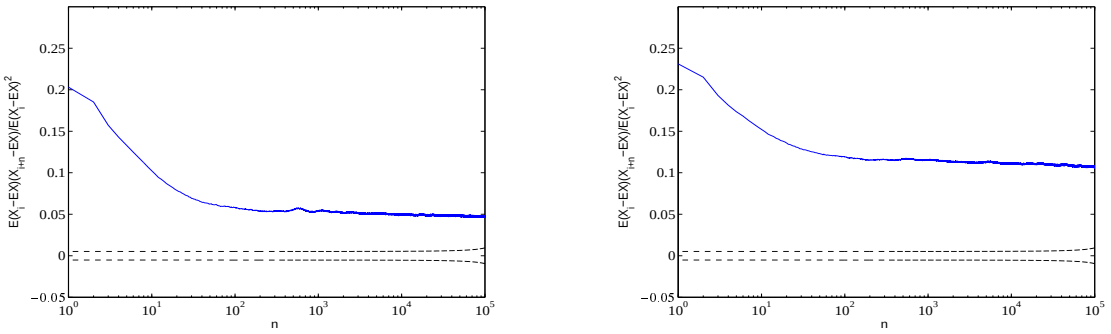
Equation (9) indicates as flow route lengths increase, the gain of joint compression increases as well.

2.3 Limitation of Packet-Level Model

In the remainder of this section, we examine some of the assumptions underlying the packet-level model. The packet level model assumes independence between packets. In a real network environment, this is generally not valid. We have collected a number of one hour traces from the outgoing gateway at a major research university. In Figure 1(a) and 1(b), we plot the autocorrelation functions of the source address and destination address for one of these traces used in Section 5. The dotted lines correspond to the 95% confidence interval. The plotted auto-correlation functions in Figure 1(a) and Figure 1(b) illustrate significant temporal correlation in the trace. This is true for other fields of the packet header as well. One explanation of this temporal correlation is that packets from the same flow share lots of common information and they tend to closely spaced in time.

In the packet level model, we assume packets arrive according to Poisson processes for all the flows. Therefore the aggregate packet arrival process is still Poisson and the inter-arrival times are independent. If the packet arrival process is not Poisson, the entropy calculated from the packet level model (2) can no longer serve as the lower bound for packet trace compression. Let's illustrate this by a simple example. Suppose there are two flows f_1 and f_2 traversing one node v. Within the measurement time interval $[0, T]$,

[1]This is over all of the traces collected at all of the monitors.



(a) Source IP Auto-correlation

(b) Destination IP auto-correlation

Figure 1: Auto-correlation in Packet Addresses

f_1 generates K_1 packets and f_2 generates K_2 packets. As illustrated in Figure 2, we denote by $\{\delta_{1,i}\}_{i=1}^{K_1}$ and $\{\delta_{2,i}\}_{i=1}^{K_2}$ the packet inter-arrival times for flow f_1 and f_2.

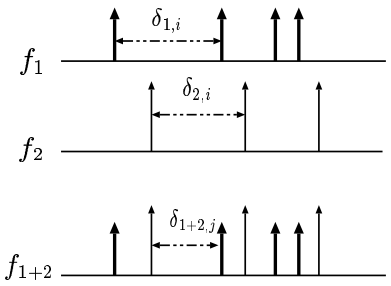


Figure 2: Interleaved Packet Stream from Two Flows

Assume $\{\delta_{1,i}\}$ and $\{\delta_{2,i}\}$ are i.i.d. sequences and are independent with each other. Therefore the entropy of packet trace of f_1 and f_2 at node v can be calculated by

$$H(\{\delta_{1,i}\}_{i=1}^{K_1}, \{\delta_{2,i}\}_{i=1}^{K_2}) = K_1 H(\delta_1) + K_2 H(\delta_2). \tag{10}$$

The average per-packet information is

$$\frac{K_1}{K_1+K_2} H(\delta_1) + \frac{K_2}{K_1+K_2} H(\delta_2).$$

On the other hand, the aggregated packet trace can also be described by the sequence $\{\delta_{1+2,j}, \theta_{1+2,j}\}_{j=1}^{K_1+K_2}$, where $\delta_{1+2,j}$ is the inter-arrival time between the jth and $j-1$th packet in the aggregated trace, $\theta_{1+2,j}$ is the flow id of the jth packet. There is a one-one mapping between the sequence $\{\delta_{1+2,j}, \theta_{1+2,j}\}_{j=1}^{K_1+K_2}$ and the flow based sequence $\{\{\delta_{1,i}\}_{i=1}^{K_1}, \{\delta_{2,i}\}_{i=1}^{K_2}\}$. Therefore,

$$H(\{\delta_{1+2,j}, \theta_{1+2,j}\}_{j=1}^{K_1+K_2}) = H(\{\delta_{1,i}\}_{i=1}^{K_1}, \{\delta_{2,i}\}_{i=1}^{K_2}). \tag{11}$$

Packet level model (2) will calculate the entropy $H(\delta_{1+2})$ and $H(\theta_{1+2})$ according to the marginal distribution of δ_{1+2} and θ_{1+2}. For general packet arrival pattern within a flow, it is no longer true that $\{\delta_{1+2,j}, \theta_{1+2,j}\}_{j=1}^{K_1+K_2}$ are independent, therefore

$$(K_1+K_2)(H(\delta_{1+2}) + H(\theta_{1+2})) > H(\{\delta_{1+2,j}, \theta_{1+2,j}\}_{j=1}^{K_1+K_2}), \tag{12}$$

together with (10) and (11), we have

$$H(\delta_{1+2}) + H(\theta_{1+2}) > \frac{K_1}{K_1+K_2}H(\delta_1) + \frac{K_2}{K_1+K_2}H(\delta_2). \tag{13}$$

That is to say the per-packet information calculated by the packet-level model in (2) is larger than the real average per-packet information. In this case, the packet level model is no longer suitable to describe the information content in an aggregated packet trace. Instead, ***flow-based model***, e.g. $\{\delta_{1,i}\}_{i=1}^{K_1}$ and $\{\delta_{2,i}\}_{i=1}^{K_2}$, can be employed to characterize packet traces.

3 Flow Level Model

3.1 Network Flow Model

In this section we introduce a ***flow-based*** model that addresses the problems described at the end of the preceding section. As before, we represent the network as a directed graph $G = (V, E)$. Assume that flows arrive to the network according to a Poisson process with rate Λ. Let $\Theta_j \in \mathcal{F}$ be the id of the j-th flow that arrives to the network. As in the packet level model, the route of a flow $f \in \mathcal{F}$ is fixed, and is represented by a tuple $f = (v_1^{(f)}, v_2^{(f)}, \ldots, v_{l_f}^{(f)})$, where $v_j^{(f)}$ is the j-th router traversed by f and l_f is the path length. For each node v, let $C^{(v)} \subseteq F$ denote the set of flows that pass through it. When flow j comes in, it generates K_j packets. Packets within flow j arrive according to some point process with independent inter-arrival times $\{\delta_{j,i}\}_{i=2}^{K_j}$, where $\delta_{j,i}$ is the inter-arrival time between the $i-1$th and ith packet of flow j. It is assumed that the first packet arrives at the same time as the flow. As before, we assume the system uses 32 bits to quantize both the flow and packet inter-arrival time and the total length of a uncompressed timestamp is 64. The behavior of packet arrivals in the network is described by the stochastic process $\{(\Delta_j, \Theta_j, K_j, \{\delta_{j,i}\}_{i=2}^{K_j})\}$. We are interested in determining the minimum number of bits required to represent each flow. If we assume $\{\Delta_j\}$, $\{\Theta_j\}$ and $\{K_j\}$ are all mutually independent i.i.d. sequences, on average we need a number of bits per flow equal to $H(\Phi)$ where

$$H(\Phi) = H(\Theta) + h(\Delta) + 32 + H(K) + E[(K-1)(h(\delta)+32)]. \tag{14}$$

If we further assume K_j is independent of $\{\delta_{j,i}\}$, we have

$$H(\Phi) = H(\Theta) + h(\Delta) + 32 + H(K) + (E[K]-1)(E[h(\delta)]+32). \tag{15}$$

The per-flow information consists of two parts: one part is timing information about the flow arrival and flow ID, which is shared by all packets in the flow; the other part consists of all the packet inter-arrival information, which grows linearly with the number of packets within the flow if we assume packet inter-arrivals are independent. (note: If packet inter-arrival times are not independent, this part can be further compressed by exploiting the correlation.) The information rate per unit time is then $\Lambda H(\Phi)$.

In practice, traces are collected at individual nodes. Consider a node v in the network. Since flows arrive to the network according to a Poisson process and the delay between any two nodes in the network is constant, flows arrive to node v according to a Poisson process with rate $\Lambda^{(v)} = \Lambda \times P(\Theta \in C^{(v)})$. The behavior of packet arrivals at node v can be described by the stochastic process $\{(\Delta_j^{(v)}, \Theta_j^{(v)}, K_j^{(v)}, \{\delta_{j,i}^{(v)}\}_{i=2}^{K_j^{(v)}})\}$, where $\{\Delta_j^{(v)}\}$ is the sequence of inter-flow-arrival time at node v that follows exponential distribution with rate $\Lambda^{(v)}$, $\{\Theta_j^{(v)}\}$ is an i.i.d. sequence of flow ids seen by v, $\{K_j^{(v)}\}$ is an i.i.d, sequence of integer valued

random variables that denote the number of packets in the jth flow passing through v and $\{\delta_{j,i}^{(v)}\}_{i=2}^{K_j^{(v)}}$ is the inter-arrival time of packets within flow j. We need a number of bits per flow equal to $H(\Phi^{(v)})$ where

$$H(\Phi^{(v)}) = H(\Theta^{(v)}) + h(\Delta^{(v)}) + 32 + H(K^{(v)}) + E[(K^{(v)} - 1)(h(\delta_f^{(v)}) + 32)] \tag{16}$$

If we further assume $K^{(v)}$ is independent of $\{\delta_{f,i}^{(v)}\}$, we have

$$H(\Phi^{(v)}) = H(\Theta^{(v)}) + h(\Delta^{(v)}) + 32 + H(K^{(v)}) + (E[K^{(v)}] - 1)E[h(\delta_f^{(v)}) + 32] \tag{17}$$

The information rate per unit time is then $\Lambda^{(v)} H(\Phi^{(v)})$ at node v. In the absence of compression, each flow requires on average $(104 + 64)E[K^{(v)}] + 64$ bits with 104 bits to encode the flow identifier and 64 bits for timestamps of both packet inter-arrivals within a flow and flow inter-arrival.

Now we can answer the question: what is the maximum benefit that can be achieved through compression? From $\Phi^{(v)}$, we have a *marginal compression ratio*

$$\rho(\Phi^{(v)}) = \frac{H(\Phi^{(v)})}{168 * E[K^{(v)}] + 64} \tag{18}$$

$$= \frac{H(\Theta^{(v)}) + h(\Delta^{(v)}) + 32 + H(K^{(v)})}{168 * E[K^{(v)}] + 64} + \frac{(E[K^{(v)}] - 1)E[h(\delta_f^{(v)}) + 32]}{168 * E[K^{(v)}] + 64} \tag{19}$$

The compression ratio $\rho(\Phi^{(v)})$ provides a lower bound on what can be achieved through lossless compression of the raw network trace. From (19), the compression ratio at node v is a function of the *average* flow size $E[K^{(v)}]$ of all flows traversing that node. Since the information in flow ID $\Theta^{(v)}$ and flow arrival $\Delta^{(v)}$ is shared by all packets in the flow, the larger the average flow size $E[K^{(v)}]$, the smaller the per-packet share, therefore the smaller the compression ratio. When $E[K^{(v)}]$ is large, the compression ratio is bounded from below by $\frac{E[h(\delta_f^{(v)})]+32}{168}$, which is an indication of how compressible the packet inter-arrival time is in average. (Note: in this model, we assume packet inter-arrival time within a flow is independent with its flow size. When this assumption is not true, a tighter bound can be derived to explore the correlation.)

We are also interested in quantifying how well marginal compression comes to achieving the entropy rate of the network. We have

$$\rho_j(\Phi) \quad = \quad \frac{\Lambda H(\Phi)}{\sum_{v \in V} \Lambda^{(v)} H(\Phi^{(v)})} \tag{20}$$

where the numerator is the lower bound on joint compression and the denominator is the lower bound of marginal compression of each trace separately. The joint compression ratio ρ_j shows the benefit of joint compression.

We apply flow-based model to an one hour trace collected at the outgoing link of a major research university connecting to a commercial service provider on July 22, 2004 starting at 09:30AM local time. There are $5,465,323$ flows and $57,976,722$ packets in the trace. As we don't have information about the flow length and packets arrival process of flows that starts or ends outside of the traces, the following results consider only $5,325,879$ flows that starts and ends within an hour. These flows corresponds to $34,370,698$ packets. Flow-based model shows that we need an average of 212.8 bits to describe a flow and can achieve. a marginal compression ratio $\rho_m = 0.1853$.

Although the flow-based model captures temporal correlation present in a trace, it cannot deal with flows that start or end outside of the trace. In the case of our one hour traces, these long flows account for more than 40%, of all packets in the traces. In the next section, we introduce a hybrid flow-packet model that accounts for both short flows and long flows that cross boundary of traces.

3.2 Hybrid Flow-Packet Model

In the packet-level model, we assume *persistent* flows, i.e., flows are always active and keep generating packets according to Poisson process. In the flow-based model, flows are *finite* in duration and generate a finite number of packets according to some flow size distribution. In reality, every flow is finite; at the same time, however, any packet trace is also finite. In a finite packet trace, any flow which is active throughout the duration of the trace appears *infinite*. Those long flows can account for a large portion of packets in a trace depending on the length of the trace. It is important to incorporate those flows in our model and characterize their information content. In this section, we develop a *hybrid* model, which captures persistent flows using the packet-level model and captures those finite transient flows using the flow-based model.

Again, we represent the network as a directed graph $G = (V, E)$. Packets are generated by two types of flows: persistent flows $\mathcal{F}_p$ and transient flows $\mathcal{F}_t$. Packets from a persistent flow $f_i \in \mathcal{F}_p$ arrive according to a Poisson process with rate λ_{f_i}. The aggregate packet arrival from all persistent flows is still a Poisson process with rate $\lambda = \sum_{f_i \in \mathcal{F}_p} \lambda_{f_i}$. Associated with each packet from persistent flows is a flow id θ with $P(\theta = f_i) = \frac{\lambda_{f_i}}{\lambda}$. Similar to (2), the number of bits required to represent a packet from persistent flows can be calculated as

$$H(\phi) = \log \frac{e}{\lambda} + 32 - \sum_{f_i \in \mathcal{F}_p} P(\theta = f_i) \log P(\theta = f_i) \tag{21}$$

The number of bits per unit time needed for compression is then $\lambda H(\phi)$.

Transient flows arrive to the network according to a Poisson process with rate Λ and let $\{\Theta_j \in \mathcal{F}_t, j = 1, 2, \cdots\}$ be an i.i.d sequence of random variables that denote the flow ids of transient flows. A transient flow j generates K_j packets, the packet inter-arrival time within flow j is $\{\delta_{j,i}\}_{i=2}^{K_j}$. Similar to (15), the number of bits required to represent a transient flow can be calculated as:

$$H(\Phi) = \sum_{f_j \in \mathcal{F}_t} P(\Theta = f_j) \log P(\Theta = f_j) + \log \frac{e}{\Lambda} + 32 + H(K) + (E[K] - 1)(E[h(\delta)] + 32). \tag{22}$$

The information rate per unit time is then $\Lambda H(\Phi)$.

Overall, the number of bits per unit time to describe a network with both persistent and transient flows can be calculated as:

$$H(\Phi) = \lambda H(\phi) + \Lambda H(\Phi), \tag{23}$$

where $H(\phi)$ and $H(\Phi)$ can be calculated as in equation (21) and (22) respectively.

In the absence of compression, each packet requires $104 + 64$ bits, 104 bits for the flow identifier and 64 bits for the timestamp and each flow in average requires $(104 + 64) * E[K] + 64$ bits with 104 bits to encode the flow identifier and 64 bits for timestamps of both packet arrivals within a flow and flow arrivals. Hence, node v generate $168\lambda^{(v)} + (168E[K] + 64)\Lambda^{(v)}$ bits per unit time. With marginal compression, we have a *marginal compression ratio*

$$\rho_m \quad = \quad \frac{\lambda^{(v)} H(\phi^{(v)}) + \Lambda^{(v)} H(\Phi^{(v)})}{168\lambda^{(v)} + (168E[K] + 64)\Lambda^{(v)}}$$

We are also interested in quantifying how well marginal compression comes to achieving the entropy rate of the network. Similar to flow-based model, we have

$$\rho_j \quad = \quad \frac{\Lambda H(\Phi) + \lambda H(\phi)}{\sum_{v \in V} (\Lambda^{(v)} H(\Phi^{(v)}) + \lambda^{(v)} H(\phi^{(v)}))}$$

where the numerator is the lower bound on joint compression and the denominator is the lower bound of marginal compression of each trace separately. The compression ratio ρ_j shows the benefit of joint compression.

4 The IP Packet Header

So far, we only studied the information content of packet header's time stamp and the flow ID Θ (or θ), which corresponds to the 5-tuple consisting of the source address, destination address, source and destination port numbers, and the protocol field. There are other fields in the IP header. Fig 3 shows all the fields in an IP header as defined in [10].

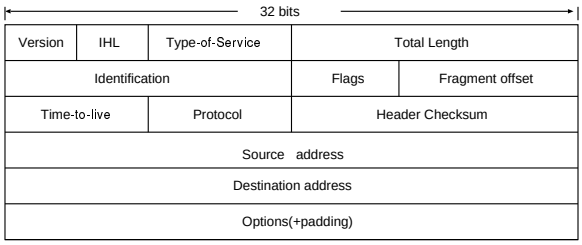


Figure 3: Format of IP Header

In this section we focus on the additional information conveyed by the remaining fields of the IP header. We do not consider the contents of the TCP/UDP header, leaving this for future work, with the exception of the port field, which is needed to characterize the flow ID. The discussion below is based on our analysis of an one hour trace collected at the outgoing link of a major research university connecting to a commercial service provider on July 22, 2004, starting at $09:30$AM local time.

We examine each of the fields in the IP header in turn.

- **Version**: The current version of IP is Version 4. Hence, this field is always set to four and conveys no information.
- **IHL**: This field, the IP Header Length, refers to the number of 32 bit words forming the header. This is typically five as is the case in our one hour trace and all other traces that we have examined. Hence it conveys no information.
- **TOS**: This is the Type of Service field, which is now known as Differentiated Services Code Point (DSCP). It is usually set to zero. the analysis of our trace indicates that the value of the TOS field is zero more than 99.2% ofthe time. In what follows, we will assume that it conveys no information as $H(TOS) \approx 0$ for this and other traces.
- **Size of Datagram($total\_len$)**: This fields refers to combined length of the header and the data in bytes. We can find practical distribution from real trace analysis and characterize the entropy with $H(l)$.

- **Identification ($IPID$)**: This is a 16-bit number, which together with the source address uniquely identifies this packet. Modern versions of Linux randomly set the $IPID$ field for the first packet of a flow and then increments it for each successive packet. On the other hand, Windows, uses a global counter that is incremented each time a packet is sent out, regardless of the flow that it belongs to. Hence, the contents of this field contains information regarding other activities at the sender; in the case of Linux, the value for the first packet of a flow and in the case of Windows, the first IPID value along with the increments between packets of a flow. Let I_f be a r.v that denotes the first IPID value of a flow f and d_f denotes the difference of IPID value between two consecutive packets. Hence $\{I_f\}$ is an i.i.d. sequence of rv's with uniform distribution over 64K value space and $\{d_f\}$ is and i.i.d. sequence of rv's where we can have practical distribution from real trace analysis. We have

$$H(IPID) = H(I) + \sum_f K_f * H(d_f), \tag{24}$$

  where K_f is the number of packets within flow f.

- **Flag**: This field indicates whether the datagram is fragmented or not. The analysis of our trace indicates that this is typically set to 0. Hence, it contains little information and we ignore it for now. $H(Flag) \approx 0$ for this and other traces.

- **Fragment**: When the datagram is fragmented, his field indicates the position within the datagram that the fragment belongs. The analysis of our trace indicates that this value is typically set to 0. it contains little information and we ignore it for now. $H(Frag) \approx 0$ for this and other traces.

- **TTL**: This is the Time To Live field, which indicates the remaining number of hops /links that the packet may be routed over before it is removed from the network. Different Operating systems set the initial TTL differently. They choose values from $\{64, 128, 256\}$. Once a packet is in the network, each router decrements the TTL field by one. For a flow, it is determined by the first packet and remains unchanged afterwards. Let T_f be a r.v that denotes the first TTL value of a flow f. Hence $\{T_f\}$ is an i.i.d. sequence of rv's where we can find out practical distribution from real trace analysis. We denote the entropy in the TTL field as $H(T)$.

- **Protocol**: This field indicates the type of transport packet being carried. Our trace analysis indicates this to be primarily TCP, UDP, and some control protocols which accounts for more than 99.8% of the total traffic. This holds for all the traces we have examined. Once determined for a flow, it remains unchanged.

- **Checksum**: The header checksum corresponds to the 1's complement of the remaining fields of the IP header. Packets with an invalid checksum are discarded by all nodes in an IP network. Hence it is totally dependent on other fields. Hence, let $checksum$ denotes the field of checksum and contains no information beyond what is carried by the other fields.

- **Options**: This field indicates whether IP options are in effect. The analysis of our trace (Section 5) indicates that it is never used. Hence in our evaluation we will assume that it carries no information.

As described above, the only ones that convey any information are the TTL field, because they reflect the operating system that generated the packets, the IPID field, because it can reflect the activity of the end host sending the packets, and the packet length field. TTL field and the initial value of IPID field are shared by all packets in a flow. They only need to be recorded once per flow. For packets from a persistent flow, we ignore the overhead introduced by these two fields. On the other hand, we do have to record the increment in IPID

start time	packets	transient flows	persistent flows	original size (Gb)	predicted size (Gb)	ρ_m
2004-07-22 09:30	57,976,722	5,325,879	139,444	1.855	0.2746	0.1480
2004-09-22 10:00	387,766,186	34,868,897	1,185,542	12.409	1.722	0.1388
2004-09-23 01:00	408,677,406	32,225,458	1,482,371	13.077	1.500	0.1147
2004-09-23 13:00	438,144,794	27,658,438	580,827	14.020	1.528	0.1090
2004-09-25 10:00	318,278,620	52,309,680	1,412,361	10.184	1.726	0.1694
2004-09-26 01:00	358,380,592	49,877,711	1,296,820	11.468	1.532	0.1336
2004-09-26 13:00	374,112,428	33,641,117	1,143,797	11.971	1.658	0.1385
2004-09-29 10:00	386,969,290	30,513,707	1,403,630	12.383	1.417	0.1144
2004-10-05 10:00	426,552,282	26,926,647	565,459	13.649	1.486	0.1089
2004-10-06 01:00	422,589,830	69,453,421	1,875,242	13.522	2.301	0.1702
2004-10-06 13:00	480,253,220	66,839,365	1,737,823	15.368	2.057	0.1338
2004-10-07 10:00	407,247,232	36,620,681	1,245,103	13.031	1.813	0.1391
2004-10-08 01:00	416,196,650	32,818,373	1,509,645	13.318	1.528	0.1148
2004-10-08 13:00	412,129,998	26,016,222	546,340	13.188	1.433	0.1087
2004-10-09 10:00	231,111,618	37,983,622	1,025,557	7.395	1.248	0.1688
2004-10-10 01:00	231,327,068	32,195,004	837,070	7.402	0.987	0.1333

Table 1: Marginal Compression Ratio over Real Traces

field and the packet length for each individual packet, no matter it is from a persistent flow or a transient flow. Hence, the rate at which real information is generated in a network with persistent and transient flows is given by

$$H(S) = \lambda(H(\phi) + H(l^p) + H(d^p)) + \Lambda(H(\Phi) + H(T) + H(I) + E[K]H(l^t) + (E[K]-1)H(d^t)), \quad (25)$$

where λ is the aggregate packet arrival rate for all persistent flows, $H(\phi)$ characterizes the packet timing and flow ID information and can be calculated according to (2), l^p and d^p denotes the packet length and IPID increment of packets from persistent flows respectively, Λ is the arrival rate of transient flows, $H(\Phi)$ characterizes timing and flow ID information for all the packets within a flow and can be calculated according to (15), T and I denotes the TTL value and initial IPID value of a transient flow, and l^t and d^t denotes the packet length and IPID increment of packets from transient flows,

5 Empirical Results on Real Packet Traces

In this section we apply the hybrid model to several one hour traces taken at a major university gateway, We gathered one hour traces taken at different times of the day (1am, 10am, 1pm) over a period from Sept 22, 2004 to Oct 23, 2004. In table 1, we list the statistics of some of these traces and the marginal compression ratios predicted by our hybrid model.

We observe from Table 1 the potential to compress the raw trace to a size that is around 11%$\sim$17% of its original size.

Now we want answer the question: what is the benefit of joint compression? We apply the hybrid model to several pop-level topologies obtained from the Rocketfuel Project [3]. Detailed descriptions of the topologies that we use can be found in [4]. Unfortunately, the Rocketfuel project is concerned only with

	nodes	links	persistent flows	transient flows	persistent flow packets rate	transient flow packets rate	averge path length	ρ_j
C&W	33	107	21	805	183,079.8	467,870.7	3.7627	0.2241
Tele_G	10	34	2	79	150,022.8	569,704.8	1.6296	0.6182
Tele_E	43	107	28	1077	295,329.1	547,974.2	6.0172	0.1534

Table 2: Joint Compression Ratio

obtaining topoloical information but not workload information. We generate a workload in the following way. A persistent flow is generated between any pair of nodes with probability p. Packets within a flow are generated according to a Poisson process with an arrival rate generated using the methods in [8]. For each node $v \in V$, we pick two random numbers O_v, $Q_v \in [0, 1]$. Furthermore, for each node pair (v_i, v_j), we pick a random number $Z_{(v_i,v_j)} \in [0, 1]$. For v_i and v_j with Euclidean distance l, the traffic rate between v_i and v_j is

$$\alpha O_{v_i} Q_{v_j} Z_{(v_i,v_j)} e^{-l/2L}$$

where α is a scale parameter and L is the largest Euclidean distance among all pairs of nodes. The values of O_v and Q_v model the degree to which a node generates or attracts traffic. The distance l models traffic locality. This model, on average, generates larger traffic rates between close pairs of nodes than distant pairs of nodes. Similarly, we generate transient flows between any pair of nodes according to a Poisson process with an arrival rate generated using the methods in [8]. The distribution of flow sizes and packet inter arrival times within a flow are taken from the one hour trace collected at the outgoing link of a major research university connecting to a commercial service provider on July 22, 2004 starting at 09:30AM local time.

In table 2, we describe three topologies, Cable and Wireless (C&W), the main IP backbone of German Telekom (Tele G), and Colt Telekom Europe (Tele E). We also describe in the table the workload produced using the method described above. In addition, we list the joint compression ratio. Note that this describes the additional benefit of joint compression over marginal compression. We find in all cases that ρ_j is approximately equal to the inverse of the average flow path length. This is a reasonable result as joint compression mainly captures redundancy caused by duplicate records for the same packet at different nodes in the network.

6 Implications, Lessons, and Applications

In previous sections, we have established models to study the information content in packet headers. Application of these models to hour-long traces shows the potential of reducing the size of a raw trace from a single site to $1/6$ of the original size and, in the case of a collection of distributed traces by an additional factor of $1/6$ to $1/2$. In this section, we discuss several packet trace compression principles obtained from our models. Guidelines are presented which either have been adopted in previously proposed compression schemes or can be followed in new packet trace compression algorithms to achieve a better compression ratio. The fact that packet headers are highly compressible can also be exploited in other ways, such as online packet header compression for communication over low bandwidth channels and to discover hidden communication through covert channels that exploit unused fields in packet headers and timing information of packets. We will briefly discuss these two applications towards the end of this section. Last, the Section concludes with a discussion of the model assumptions, specifically focussing on those that can be relaxed.

6.1 Guidelines for Packet Trace Compression

6.1.1 Exploiting Information Structure

A naive way to do packet trace compression is to use generic data compression tools. Gzip [2] can achieve a marginal compression ratio around 2 when applied to our packet traces. This is far below the result predicted by our previous study. Packet header traces are highly structured data streams; their structure should be accounted for during compression.

Brute-force vs. By-field Since a packet header consists of several fields, and many fields are either constant or are described by highly skewed distributions (consequentially exhibiting low entropy), one can divide a packet trace into multiple traces, each corresponding to a single field. Data compression tools, such as gzip, bzip2, etc., can be used to compress these field traces. To further exploit correlation among different fields, e.g., the 5-tuple defining a TCP flow, one can place multiple fields into one trace, and then compress it using algorithms which can handle long symbols. This procedure can be conducted *online* if the compression algorithm only scans the data sequence once, such as gzip, dynamic huffman coding [17].

Packet-based vs. Flow-based Packets from the same flow share considerable information in addition to the tuple defining the flow. However, packets from different flows are interleaved in a packet trace. Hence there is less commonality between successive packets in a trace. In order to exploit the information redundancy within a flow, packet header compression algorithms can reorganize packets into flows and conduct flow based compression. Two previous works have investigated flow-based packet header recording and compression. The authors of [9] represent packet header traces as a series of *flow records* and *packet records.* The information shared by all packets from the same flow is only recorded once in a flow record. The per-packet records are maintained to store unique information of each packet. All per-packet records from the same flow stored together. This technique has been shown, [9], to produce a marginal compression ratio of between $1/3$ and $1/4$ can be achieved. However this approach requires greater storage than generic compression because of the need to identify flows and classify packets.

Another work [13] also adopted the idea of flow-based compression. Again, flow information is stored once for each flow. Unlike the previous work, packets are *not* reshuffled. Instead, a flow id is stored in each packet header as a pointer to the flow information. With a fixed time resolution of a micro-second, the author proposed to use a variable number of bits to store the packet inter-arrival time: 16 bits if the inter-arrival time is smaller than $2^{15}\mu s$; 32 bits if the inter-arrival time is smaller than $2^{31}\mu s$; and 96 bits otherwise. Packet timing information can be further compressed if optimal codes, e.g., Huffman code, are applied on packet inter-arrival time to achieve the entropy bound as calculated in Section 3.

Another major saving comes from the observation that many fields of a packet header are predictable given the previous packet header within the same flow, (in other words, a new packet header in a flow brings little new information). It is argued that, for a TCP flow, only the flow id (4 bytes) and inter-arrival time (2 bytes) is necessary to represent a in-sequence TCP packet (8 bytes time stamp + 40 bytes TCP/IP header). This easily leads to a compression factor of 8. Now the flow id takes a major portion of a compressed packet header. It can be further compressed since the flow size distribution is *skewed* in the Internet. Intuitively, assigning shorter flow ids to elephant flows will reduce the average flow id length for all the packets. We can even get rid of flow id in each packet header if we place all of the packet headers from the same flow together. This complies with our flow based model in Section 3, where flow id doesn't appear in a packet header information content. It comes, however, at the cost of packet reshuffling, which is very memory intensive because one has to store all the packets from all active flows. The memory requirement of packet reshuffling can be reduced by only storing the compressed headers in memory and the flow temporal locality

which will be explained in the following section.

6.1.2 Exploiting Temporal Locality

Users' connections are inherently transient. Network flows come and go and packets within a flow are closely spaced in time. This results in temporal locality in packet headers which can be explored both by packet based compression and flow based compression.

For packet based by-field compression, the compression ratio is bounded by the entropy of each field. Given a packet trace with duration T, we can use Huffman coding to construct a optimal code book C for the whole trace $Trace(0, T)$. Alternatively, we can construct a Huffman code book C_1 for the first half of the trace $Trace(0, \frac{T}{2})$ and another code book C_2 for the second half of the trace $Trace(\frac{T}{2}, T)$. We than use C_1 to compress $Trace(0, \frac{T}{2})$ and C_2 to compress $Trace(\frac{T}{2}, T)$. Since C_1 is optimized for $Trace(0, \frac{T}{2})$, $Trace(0, \frac{T}{2})$ is better compressed by C_1 than by C. Similarly, $Trace(\frac{T}{2}, T)$ is better compressed with C_2 than C. Therefore the whole trace $Trace(0, T)$ can be better compressed by using C_1 and C_2 than just C. On the other hand, we have to store two code books C_1 and C_2 instead of one C. In general, one can divide a packet trace efficiently according to time and compress them separately to explore the temporal locality in packet headers.

Temporal locality is more important for flow-based compression. Given a long trace, a large number of flows and their packets have to be stored in the memory. A long bit sequence has to be used to represent a flow id. Since most flows are transient, if we only count active flows within a time window, the smaller the window size, the smaller the number active flows. Equivalently, if we divide a long trace into multiple shorter traces according to time, we can reduce the memory requirement and use shorter bit sequences to represent flow ids. Similar to the argument in the previous paragraph, we can also do better in compressing flow id. On the other hand, time window will chop one flow, either longer than the time window or happens to cross one boundary of a time window, into multiple flows. The consequence is that the flow size (measured in packets) seen by the flow-based compression algorithm is now smaller than their original size. Ideally, the more packets in a flow, the higher the gain of flow-based compression. Too small a time window will degrade the performance of flow based compression. To illustrate, assume we have to use F bits to represent the flow information, B bits (in average) to record per-packet information, the total length of a flow f with N packets is $F + NB$ bits, the per-packet flow information overhead is $\frac{F}{N}$. Let's say due to a small time window, the flow is chopped equally into m smaller flows $\{f_1, \cdots, f_m\}$. Then we have to use F bits to represent flow information for each flow f_i which consists of $\frac{N}{m}$ packets. The per-packet flow information overhead is m times higher than the previous case. The total bits needed to represent $\{f_1, \cdots, f_m\}$ is $mF + NB$. The inflation in the compressed length for the flow is $\rho_I = \frac{mF+NB}{F+NB}$. If $\frac{N}{m} >> \frac{F}{B}$, i.e., each time window still have a large number of packets from the same flow, the inflation ratio ρ_I is close to 1. The choice of appropriate window size depends on the flow size distribution measured *both* in time and in packets.

6.1.3 Exploiting Spatial Correlation

A distributed network measurement system collects traces from multiple network links to characterize network wide phenomena. Packet header traces can be compressed individually using the previously discussed principles. Network flows crossing multiple network monitors introduce information redundancy between packet traces collected at these monitors. The focus of this section is on how to exploit the *spatial correlation* present in packet traces to jointly compress them.

We observed that the joint compression ratio is roughly equal to the average path length. This observation was based on the assumptions that delays are constant and that packets for a given flow always take the same route through the network. However, these assumptions are generally not true in real network environment. Flow routes can change over time (albeit infrequently) and packets incur random delay on network links. To capture the route changes, the network behavior should include the information of where a particular packet appears. In other words, each monitor should at least record in some way the identities of all packets that it observes. To deal with random link delays, we *have to* store timing information of each packet on each monitor. The jointly compressed trace must contain the identity and timing information for each packet on each monitoring point. In an individually compressed flow-based trace at a monitor, flow id and inter-arrival time corresponds to most of the information associated with a packet. One may draw the conclusion that in a realistic network environment joint packet trace compression is not necessary. However, this is not true for the following reasons:

1. Flow routes change infrequently. We don't have to record packet ID at all monitors all the time. How to exploit the stability of network routes in joint compression of distributed packet traces deserves more study.
2. Are packet inter-arrival time on multiple links correlated? How can we exploit this spatial correlation to compress packet timing information.
3. So far we have focused on packet header trace compression. Some monitoring applications, especially network security related applications, require that packet payload information be recorded. Unlike headers, payloads don't change inside the network. Furthermore, they dominate the trace dominate header in size. Joint compression of full packet trace is definitely desirable.

Implementation of joint packet trace compression is much more complicated than individual packet trace compression. The major difficulty comes from how to correlate distributed traces. One option is to send all packet traces to a single service facility and compress them jointly in a centralized way. In Figure 4, all network monitors send their packet traces to a common packet trace storage E. In order to reduce bandwidth consumption in sending the original packet traces, packet traces can be compressed individually before the transmission. Another option is to jointly compress packet traces along their way to the common storage. For example in Figure 4, A first sends its trace to B; B compresses its own trace jointly with A's trace, then sends the compressed trace to E. How to optimally organize the routing and compression of packet traces is a trade off between the processing power and network connectivity on all monitors. Distributed

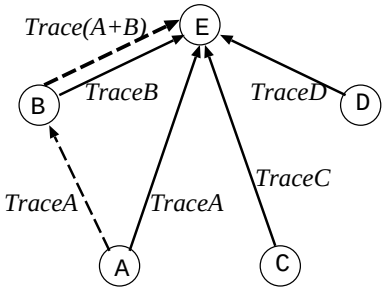


Figure 4: Joint Packet Trace Compression

data compression [6] aims at compressing correlated sources in a distributed way and achieving the gain of joint compression. How to compress packet traces without exchanging packet headers remains to be a challenging problem. It may also be possible to borrow ideas from *trajectory sampling* [7]to design joint compression algorithms.

6.2 Other Applications

6.2.1 Packet Header Compression to Improve Link Efficiency

Packet header compression is not only important for passive network monitoring. It has also been used to improve link efficiency in various network environment. For network applications, such as remote login, Voice over IP, network games, etc, the payload of IP packet is small. Transmitting IP headers incur a large overhead. Packet header compression is especially important when those applications run over low bandwidth and high bit error rate links, such as dial-up connection and wireless channels. As shown in Figure 5, packet headers can be compressed at the sender side of the link and uncompressed by the receiver. Header compression improves the efficiency of those expensive links. Several packet header compression

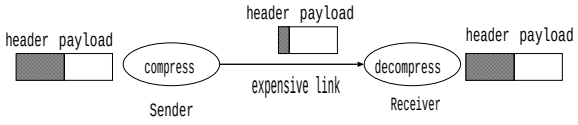


Figure 5: Header Compression to Improve Link Efficiency

standards have been developed within IETF ([11], [12], [14] and [16]) for different type of links and different applications. The models and principles developed in this paper may prove useful to guide the design of packet header compression schemes for new network environments and new applications.

6.2.2 Covert Channels Established using Packet Headers

A covert channel is defined to be a communication channel that can be exploited by a user to transfer information in a manner that violates a system's security policy. Covert channels in the Internet pose a big threat to network security [15]. Several schemes have been proposed to establish an embedded communications channel utilizing TCP/IP packet headers. Some schemes take unused/unchecked fields in packet headers to transmit data. Some schemes use timing between packets to convey information. Our information theoretic study in packet traces, both on packet timing and individual packet fields, can be used to detect covert channels built on packet headers: a significant increase in the entropy of packet headers suggests that they are being manipulated to hide communications.

7 Conclusions and Future Work

Using an information theoretic approach, we systematically study the information redundancy in packet headers. Both packet-level and flow-level models are developed to explore the temporal and spatial correlation of packet traces collected from distributed network monitors. We carefully study the information content in all fields of an IP header, including the timestamp, the flow Id defined by the 5-tuple, TTL field and IPID field, etc. Empirical data from an access link of a major university are feed into the proposed models to demonstrate the potential gain of conducting marginal and joint compression on packet traces. Various results obtained from our information theoretic study serve as lower bounds on the compression ratios which can be achieved by lossless compression algorithms. More importantly, our analytical models help us identify the major sources of information redundancy in packet header traces. Several important principles are obtained to guide the design of efficient packet trace compression algorithms.

Future work can be pursued the several directions. One immediate application of our study is to develop both marginal and joint packet compression algorithms according to the proposed guidelines. Their efficiency can be evaluated by comparing their compression ratios on real packet traces with those predicted by our models. In our models, we make several assumptions, such independency assumptions, on packet headers and their fields. The implication of violation of those assumptions on both analytical models and algorithm design deserves further study. How to extend our packet header models to study other emerging applications, such as covert channels, is another interesting direction to pursue.

References

[1] Dag card web site, `http://dag.cs.waikato.ac.nz/`.

[2] Gzip web site, `http://www.gzip.org/`.

[3] Rocketfuel project, `http://www.cs.washington.edu/research/networking/rocketfuel/`.

[4] Rocketfuel topology, `http://dmz02.kom.e-technik.tu-darmstadt.de/~heckmann/index.php3?content=topology`.

[5] J. Chou, D. Petrovic, and K. Ramchandran. A distributed and adaptive signal processing approach to reducing energy consumption in sensor networks. In *IEEE Infocom 2003*, April 2003.

[6] T. A. Cover and J. A. Thomas. *Information theory*. John Wiley & Sons, Inc., 1991.

[7] N. Duffield and M. Grossglauser. Trajectory sampling with unreliable reporting. In *IEEE Infocom*, March 2004.

[8] B. Fortz and M. Thorup. Internet traffic engineering by optimizing ospf weights. In *IEEE Infocom*, April 2000.

[9] G. Iannaccone, C. Diot, I. Graham, and N. McKeown. Monitoring very high speed links. In *Proceedings of ACM Internet Measurement Workshop*, November 2001.

[10] I. S. Institute. Internet protocol DARPA Internet program protocol specification. Sep 1981.

[11] V. Jacobson. Compressing TCP/IP headers for low-speed serial links. Feb 1990.

[12] S. P. M. Degermark, B. Nordgren. IP header compression. Feb 1999.

[13] M. Peuhkuri. A method to compress and anonymize packet traces. In *Proceedings of ACM Internet Measurement Workshop*, November 2001.

[14] V. J. S. Casner. Compressing IP/UDP/RTP headers for low-speed serial links. Feb 1999.

[15] J. C. Smith, `http://gray-world.net/cn/papers/covertshells.txt`.

[16] E. V. See. A template for ietf patent disclosures and licensing declarations. Sept 2004.

[17] J. S. Vitter. Design and analysis of dynamic huffman codes. *Journal of the Assocaition for Computing Machinery*, 34(4), October 1987.