Characterizing Per-Application Network Traffic Using Entropy

Vladislav Petkov
School of Engineering
University of California Santa Cruz
Santa Cruz, CA, USA
Email: vladi@soe.ucsc.edu

Ram Rajagopal
Dept. of Civil and Environmental Engineering
Stanford University
Stanford, CA, USA
Email: ramr@stanford.edu

Katia Obraczka
School of Engineering
University of California Santa Cruz
Santa Cruz, CA, USA
Email: katia@soe.ucsc.edu

***Abstract*—The Internet has been evolving into a more heterogeneous internetwork with diverse new applications imposing more stringent bandwidth and QoS requirements. Already such new applications such as YouTube, Hulu, and Netflix are consuming a large fraction of the total bandwidth. We argue that, in order to engineer future internets such that they can adequately cater to their increasingly diverse and complex set of applications while using resources efficiently, it is critical to be able to characterize the load that emerging and future applications place on the underlying network. In this paper, we investigate entropy as a metric for characterizing per-flow network traffic complexity. While previous work has analyzed aggregated network traffic, we focus on studying isolated traffic flows. Per-application flow characterization caters to the need of network control functions such as traffic scheduling and admission control at the edges of the network. Such control functions necessitate differentiating network traffic on a per-application basis. The "entropy fingerprints" that we get from our entropy estimator summarize many characteristics of each application's network traffic. Not only can we compare applications on the basis of peak entropy, but we can also categorize them based on a number of other properties of the fingerprints.**

I. Introduction

There is no question that emerging and future Internet applications have become much more diverse and complex than the original Internet's "killer apps" , namely e-mail, file transfer, remote login, and even early Web-based services. Application diversification will not only continue but will likely become even more accentuated as the Internet becomes the preferred medium for access to information, communication, and entertainment replacing or complementing the phone, TV, radio, movies, newspapers, books, etc. This trend is already visible today with services like Skype, YouTube, Hulu, and Netflix, to name a few. Teleconferencing and distance learning applications are also becoming more popular as well as media streaming, games, interactive TV, peer-to-peer and social networking. While these applications currently may only represent a small percentage of the Internet's users, they already consume more than half of the total bandwidth [1]. And, as they become more popular, they will consume an even more disproportionate amount of the Internet's overall resources.

We argue that, in order to engineer future internets such that they can adequately cater to their increasingly diverse and complex set of applications while using resources efficiently, it is critical to be able to characterize the load that emerging and future applications place on the underlying network. To this end, in this paper, we explore ways to understand the correlation between the nature of an application and the complexity of traffic it generates. We investigated different metrics to characterize the complexity and behavior of application traffic in a systematic way. As a starting point, we explored self-similarity, which has become a well-known metric in the networking community and measures whether traffic preserves its burstiness at different time scales. We then looked into entropy, which, from Information Theory is defined as *a measure of information, choice and uncertainty* [2]. We found that, while self-similarity is not a strong indicator of traffic behavior, the entropy of packet inter-arrival times can generate application "entropy fingerprints" that can be used not only to clearly distinguish one application from another, but also provide a summary of the application's complexity over multiple time scales which can be used to quantitatively compare the complexity of one application's traffic to that of another.

Around the same time we were conducting our study, Riihijarvi et al. [3] also proposed the use of entropy as a complexity metric for network traffic. There are two distinctions between the works. First, Riihijarvi et al. [3] focus on aggregated network traffic where packets from multiple source-destination pairs and multiple applications are present. Our work targets per-(application) flow traffic, in which packets of a single application between a single source-destination pair are considered in isolation. Per-application flow characterization targets network control functions such as traffic scheduling and admission control at the edges of the network, which necessitates differentiating network traffic on a per-application basis. Rihiijarvi et al., on the other hand, explore traffic model validation and anomaly detection applications to which aggregated network traffic is better suited.

While both efforts agree on the fact that self-similarity is not a strong indicator of traffic complexity (for both isolated and aggregated traffic), the second distinction between our approach and theirs is the way the entropy analysis is conducted. We take an approach similar to that used in the neuroscience community to study neuron spike trains [4]. We map the packet

arrival times of each trace to a binary series and estimate the entropy of this series. Riihijarvi et al., on the other hand, use the SampEn estimator [5] directly with the unprocessed time-series of packet inter-arrival times. We found that our approach of using the binary series in conjunction with our Plug-in Packet Timing Entropy (PPTEn) estimator captures more of the underlying application characteristics than the SampEn multiscale approach.

The remainder of the paper is organized as follows: in Section II we present the datasets that we use in the paper. Section III explains how entropy can be used as a complexity metric for network traffic and presents our PPTEn estimator and a brief overview of the SampEn estimator [5][3]. The results of our entropy analysis of the datasets are presented in Section IV but we must leave out our self-similarity analysis due to space constraints. Section V outlines some possible applications of this work and Section VI concludes the paper.

II. Datasets

In this section we describe application data we use in our study. We start by presenting a taxonomy of network applications that we feel is representative of a large portion of today's network traffic in Table I. In our taxonomy, applications fall into one of three categories according to their network traffic characteristics: streaming media, real-time or best-effort. We further subdivide the real-time category to differentiate voice over IP (VoIP), video conferencing, and remote access applications. For each application we indicate whether it uses buffering, has traffic that tends to be bursty, has traffic that is affected by available bandwidth, has traffic patterns that depend on a application codec of some sort or has its traffic pattern greatly influenced by the presence or absence of user interaction.

From the applications in the taxonomy, we chose from the ones whose traffic is not affected by user interaction to make up the traffic dataset that we use in the remainder of the paper. We collected *tcpdump* [13] network traces and isolated the network traffic of the chosen applications. The application traces that form our dataset are listed in Table II.

TABLE II
NETWORK TRACES THAT FORM OUR DATASET

Real-time	VoIP	Skype iChat GoogleTalk
	Video conferencing	Skype iChat GoogleTalk
Media streaming		Hulu.com - "24" Hulu.com - "Chuck" Hulu.com - "American Dad" Netflix.com - "One Last Thing" Abc.com - "Castle" Webcam stream

We will analyze these application traces using entropy estimation algorithms in the remainder of the paper, but before we do, we describe the nature of each of the flows from the perspective of the cumulative distribution function (CDF) of their inter-arrival times and 5-second snapshots of their inter-arrival times. We make hypotheses in these two sections about the complexity of each application that we will come back to when we analyze the entropy estimates.

A. Real-time flows

The real-time flow group of traces consists of both voice over IP and video conferencing network traffic from Skype, GoogleTalk, and iChat. The traces were collected on the sender side so as to prevent deterioration of patterns due to network queueing. Durations of the flows are around 10 minutes and data rates range from $38Kbps$ to $630Kbps$. In the context of the real-time flows, we use the terms audio and VoIP interchangeably in this paper.

From previous research on Skype traffic identification [14] [15] we know that we can expect to find patterns (and thus a high degree of predictability) in Skype audio flows. From the cumulative distribution function (CDF) of the packet inter-arrival times in Figure 1(a), we can see that there are 4 distinct inter-arrival times in Skype VoIP flows. This supports the claim that there are patterns in Skype audio traffic. The CDF for the Skype video conferencing flow in the same figure has similar distinct inter-arrival times to the VoIP one, with the addition of a new one at close to 0 seconds. The "staircase" in the Skype video CDF has smoother corners and non-horizontal steps, which means that there are inter-arrival times distributed throughou the [0,40]ms range. We expect the complexity of the Skype video conferencing flow will be higher than its VoIP counterpart, although the 5-second Skype flow snapshots in Figure 2(b) shows that some pattern is still evident.

The CDF of iChat audio (Fig. 1(a)) indicates that there is one fundamental packet inter-arrival time in the flow with the occasional extra packet. The 5-second flow snapshot in Figure 2(a) solidifies that observation. iChat audio has a more distinct pattern than Skype audio and can be expected to have lower complexity. iChat video has a similar packet inter-arrival pattern to iChat audio but has an additional distinct inter-arrival time of around 1ms as can be seen in both the CDF (Fig. 1(a)) and the flow snapshot (Fig. 2(b)).

GoogleTalk audio has the widest range of packet inter-arrival times according to its CDF in Figure 1(a), but from the flow snapshot (Fig. 2(a)), it looks like the inter-arrival times around 60ms and 100ms dominate and as a result the flows behavior is less complex than that of Skype audio. GoogleTalk video seems to have nothing in common with its audio counterpart. Both the CDF and the flow snapshot show completely different behavior with no overlap in inter-arrival time concentrations. Due to the curved nature of its CDF "staircase", this flow is expected to be the most complex of the six.

B. Media streaming flows

We collected traces of three show episodes from Hulu.com, one episode from Abc.com and one movie from Netflix.com. Additionally, we collected a trace from a webcam streaming

TABLE I
A TAXONOMY OF COMMON NETWORK APPLICATIONS. APPLICATIONS WHOSE TRAFFIC WILL BE STUDIED IN THIS PAPER ARE MARKED WITH "*".

Traffic class		Application	Transport protocol	Application protocol or codec	Buffered	Bursty	Bandwidth dependent	Codec dependent	User dependent
Streaming media		YouTube.com	TCP	H.264/MPEG-4 AVC [6]	•	•		•	
		Hulu.com *	TCP	H.264 [7]	•	•		•	
		Netflix.com *	TCP	VC1 Advanced Profile [8]	•	•	•	•	
		ABC.com *	TCP	TrueMotion VP7	•	•	•	•	
		Webcam stream *	UDP	RTP				•	
Real-time	VoIP	Skype *	UDP,TCP	ISAC, iLBC, G.729, iPCM-wb, EG.711A/U, PCM A/U, SVOPC [9]				•	
		iChat *	UDP	AAC-LD [10]				•	
		GoogleTalk *	UDP	PCMA, PCMU, G.723, iLBC, ISAC, IPCMWB, EG711U, EG711A [11]				•	
	Video conference	Skype *	UDP,TCP	TrueMotion VP7 [9]				•	
		iChat *	UDP	H.264/AVC [12]				•	
		GoogleTalk *	UDP	H.264 SVC, H.264, H.263-1998 [11]				•	
	Remote access	ssh	TCP	Secure Sockets Layer (SSL)					•
		VNC	TCP	Remote Framebuffer (RFB)				•	•
Best-effort		BitTorrent	TCP	BitTorrent protocol		•	•		
		File transfer	TCP, UDP	FTP/SFTP		•	•		
		Web browsing	TCP	HTTP		•	•		•

video using the Real-Time Protocol (RTP). The webcam streaming application is set apart from the others by the fact that it does not leverage client-side buffering and therefore doesn't have the burst-pause-burst network traffic pattern that is visible in the other traces.

From the CDFs in Figure 1(b) it is difficult to distinguish between the Hulu, ABC, and Netflix flows, but the webcam flow is visibly different. In the flow snapshots in Figure 2(c), the Netflix flow is easily differentiable, but the Hulu and ABC flows look very similar. This similarity is unexpected because the video codec used by ABC.com and Hulu.com is not the same (Table I). Although the flow snapshots show the presence of a 2s inter-arrival time, its occurrence compared to the other inter-arrival times is so low that the CDFs don't show it (the 0-100ms CDF window appears to represent close to 100% of inter-arrivals).

Based on the CDFs and flow snapshots, our expectation is that the complexity of the webcam flow will be the lowest of the media streaming flows. The Netflix flow will have the highest complexity because it has less visible pattern than the Hulu and ABC flows. The remaining 4 flows will be very similar in complexity.

How the complexity of the media streaming flows will compare to that of the real-time flows is a more difficult estimation to make. Our hypothesis is that the larger amount of data in the media streaming flows will make their complexity higher than that of the real-time flows.

III. PACKET TIMING ENTROPY ESTIMATION

In this section we describe why entropy can be used as a complexity metric, we present our Plug-in Packet Timing Entropy (PPTEn) estimator and the theory behind it and we briefly describe the SampEn estimator used by Rihiijarvi et al. [3].

A. Entropy Rate

Given a stochastic process $X = (X_n : n = 0, 1, ...)$ taking values in a discrete domain $\mathcal{D}$, its entropy rate $H(X)$ is defined as:

$$H(X) = \lim_{n\to\infty} \frac{1}{n} H(X_1, ..., X_n), \tag{1}$$

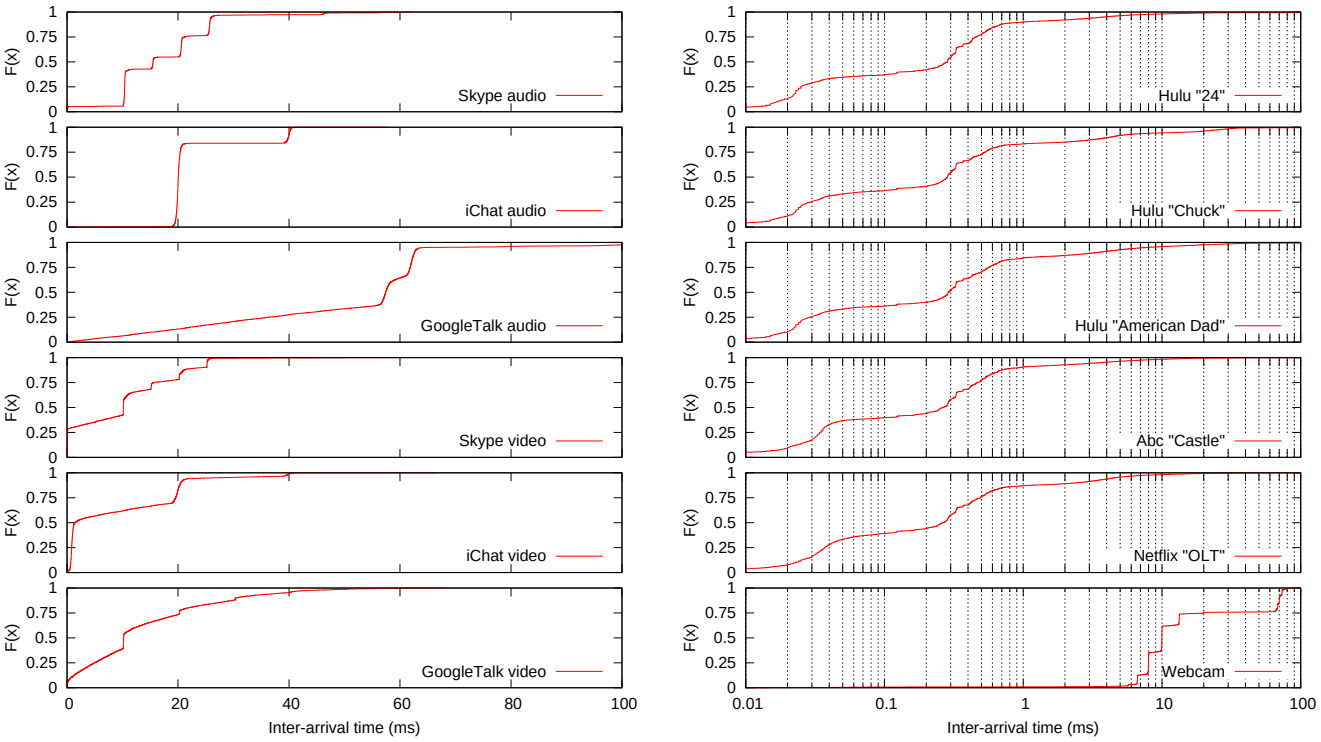
where $H(X_1, ..., X_n)$ is the entropy of the set of random variables $\mathbf{X_n} = \{\mathbf{X_1}, ..., \mathbf{X_n}\}$ with joint probability $P(X_1, ..., X_n)$, and is given by

$$H(X_1, ..., X_n) = -\sum_{\mathbf{X_n}\in\mathcal{D}^\mathbf{n}} P(X_1, ..., X_n) \log P(X_1, ..., X_n).$$

Standard information theoretic results [16] show that $0 \leq H(X) \leq 1$, and for a *stationary* stochastic process, the rate is given by the residual entropy:

$$H(X) = \lim_{n\to\infty} H(X_n|X_{n-1}, ..., X_1), \tag{2}$$

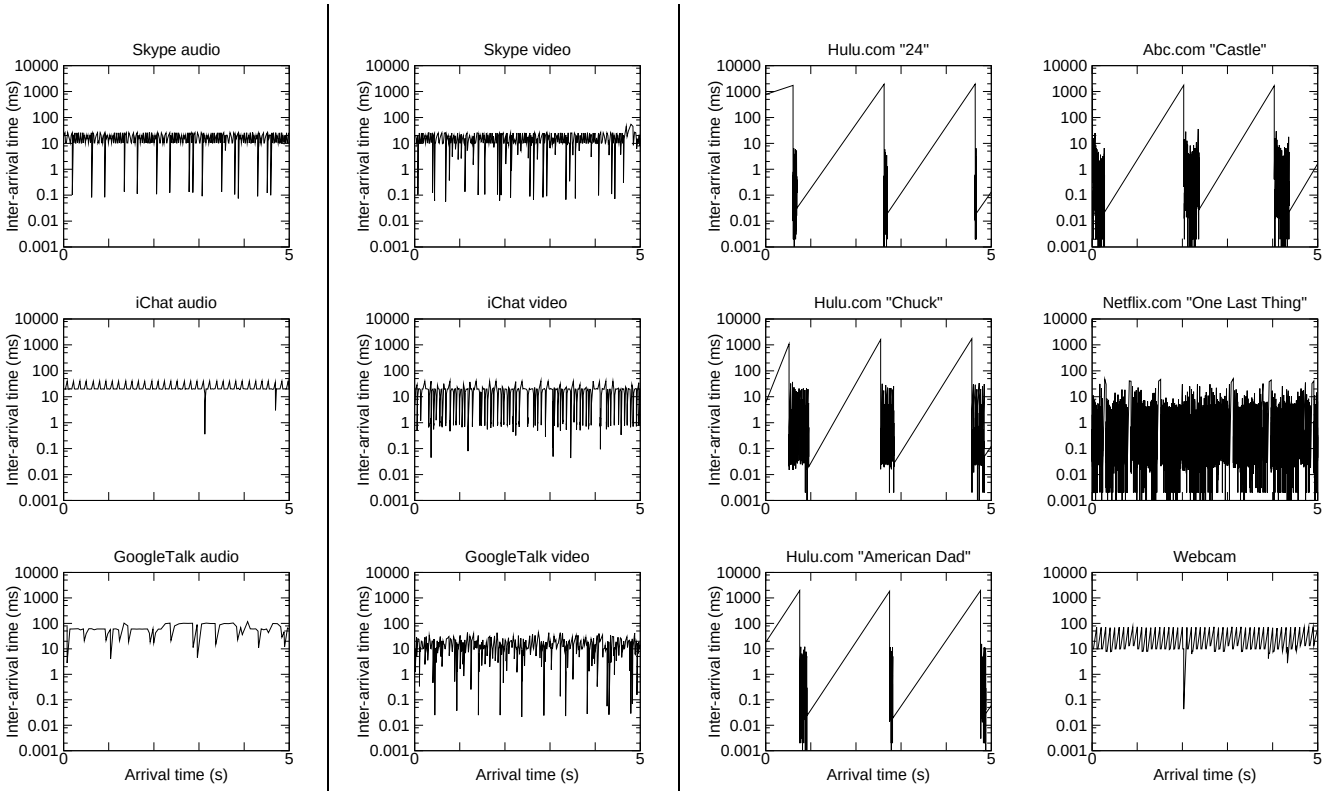
where conditional entropy is defined with respect to conditional probabilities. Notice that this result also suggests that a stationary Markov process with memory length l has rate given by $H(X) = H(X_n|X_{n-1}, ..., X_{n-l+1})$, and that in some sense non self-similar processes have this property as the correlation to X_r for $r << n$ becomes small quickly. The importance of this property is that entropy rates can be reliably estimated for such processes, using finite memory estimators [17].



(a) VoIP and video conferencing

(b) Media streaming

Fig. 1. CDFs of packet inter-arrival times for real-time flows and media streaming flows



(a) VoIP

(b) Video conferencing

(c) Media streaming

Fig. 2. Packet arrival patterns for VoIP flows, video conferencing flows, and media streaming flows

The entropy rate of a sequence is a measure of *how predictable* a sequence is based on past observations. An intuitive interpretation is how much new information the outcome of X_n brings, when we have observed the past. A constant sequence has entropy rate 0, and a sequence of independent fair coin tosses has entropy rate 1, as every new coin toss brings one whole bit of information. Sequences with lots of repeated patterns have low entropy rates, as previous pattern help to predict more recent outcomes. In fact a periodic sequence, without randomness, has entropy rate 0 once a complete period has been observed.

B. Multiscale Plug-in Packet Timing Entropy Estimator

Markov plug-in estimator. A standard estimator for the entropy rate of an independent and identically distributed (i.i.d.) sequence X is given by Maximum Likelihood Estimator (MLE) of the empirical entropy loss:

$$\hat{H}(X) = \max_P -\frac{1}{n}\sum_{r=1}^{n} \log P(X_r), \tag{3}$$

where P is the discrete distribution over the m atoms of the discrete domain that defines each element of X. The probability distribution $P = \hat{P}$ that solves the above optimization problem is the empirical estimator [18]

$$\hat{P}(k) = \frac{1}{n}\sum_{r=1}^{n} \mathbf{1}(X_r = k), \tag{4}$$

that counts the number of instances the k-th member of the discrete domain happens in the sequence. The *plug-in* entropy estimate is obtained by plugging in $\hat{P}$ in the definition of $\hat{H}$ in place of P. The plug in estimator has some important properties when the sequence is independent and identically distributed. It is biased so that

$$\mathbb{E}[\hat{H}] - H = -\frac{m-1}{2n} + O(1/n^2), \tag{5}$$

and it has asymptotic variance given by $\mathrm{Var}[\log P(X_1)]/n^2$. A simple extension can be used for estimating the *entropy rate* in Eq. 1, when X is not independent, but is Markov, is by using a conditional plug-in estimator $\hat{P}(X_n = k|X_{n-1} = r)$ following the standard empirical estimator. The residual entropy formulae (Eq. (2)) then shows that inserting this as a plug-in in the conditional entropy definition

$$\begin{aligned} H(X) &= \sum_r \hat{P}(r) H(X_n|X_{n-1} = r) \\ H(X_n|X_{n-1} = r) &= \sum_k -\hat{P}(k|r)\log\hat{P}(k|r), \end{aligned} \tag{6}$$

where $\hat{P}(k,r) = \hat{P}(X_n = k|X_{n-1} = r)$. In general we can consider Markov processes X with memory length l, where X_n is conditionally independent of the past given $X_{n-1}, ..., X_{n-l}$. Extending the proof in [18] we obtain

Theorem 1. *The conditional entropy estimator is a biased, consistent and asymptotic normal estimator for a* stationary *Markov process X with memory length l, with bias given by*

$$\mathbb{E}[\hat{H}] - H = -\frac{m^l \times (m-1)}{2n} + O(1/n^2). \tag{7}$$

Improvements in performance are obtained by using a *coverage adjusted* entropy estimator [19], that rescales $\hat{P}$ appropriately, although for large n relative to m, both estimators are very similar and moreover, the resulting bias is unknown. To abbreviate the name of the procedure, we call it PPTEn.

Packet timing entropy estimator. A sequence of packet arrival times $t_1, ..., t_n$ characterizes the packet arrival process. Typically we are interested in learning if there is some finite memory predictability for this process. The entropy rate captures such behavior. Unfortunately, packet arrival times usually belong to an unbounded integer domain, and therefore the estimator suggested in Eqn (6) may fail since it has a bias error proportional to m, the size of the domain (see Eqn. (7)). Furthermore, a standard discrete distribution implies that two elements x_1 and x_2 of the domain are not comparable, independent of a notion of distance between both. For example, small random jitters can increase entropy substantially. Finally, packet bursts can also lead to higher entropy, although we desire a process composed of periodic bursts have small entropy independent of the number of packets in the burst. Thus we separate packet timing from number of packets at a given time scale.

We consider a *rescaled* representation of the timing sequence to address these issues. Divide time into bins of size τ, the time scale unit. Create a timing pattern sequence s_k, such that $s_k = 1$ if there exists some t_r in the interval $[k\tau, (k+1)\tau)$ and $s_k = 0$ otherwise. Now using the plug-in estimator Eqn. (6) compute $\hat{H}(\tau, l)$, the entropy of the timing pattern sequence for time scale τ and memory length l. Notice that $m = 2$ since the sequence is binary. Theorem 1 shows that the estimator is guaranteed to be near consistent for memory lengths $l - 1 << \log_2 n$. For other memory lengths it may depend on the effective size of the conditioning sequences. Notice that the theoretical guarantees are usually conservative, and in practice performance may be better.

The multi-scale plug-in packet timing entropy estimator has low computation complexity in general, and in particular can be quickly computed for scales such that $\tau = 2^b\tilde{\tau}$, where $\tilde{\tau}$ is some reference smallest scale. This is an important and appealing property that allowed us to explore datasets in a more comprehensive way.

C. SampEn Entropy Estimator

The SampEn estimator works on the principle that the entropy rate of the sequence $\{t_1, ..., t_n\}$ for memory length l can be approximated by counting the number of vectors of size l selected as a contiguous subsequence of $T' = \{t_2 - t_1, ..., t_n - t_{n-1}\}$ within some distance r. The notion of time-scale is introduced by creating a sequence as a running averaging of T' of size τ. The estimator itself is not easy to compute, but it is shown to exhibit good behavior in various empirical sets.

PPTEn provides a complimentary view to SampEn by separating packet intensity and packet timing. Its simpler computation allows for determining quickly the time-scales of interest, and can be used as an input to a SampEn analysis. Furthermore, the bias tradeoff faced by PPTEn is favorable compared to SampEn. Finally, if coverage adjusted entropy is used, PPTEn is unbiased for finite samples but SampEn is not [19].

D. Entropy estimation related work

The entropy rate of a stochastic process is a measure of how predictable the process is, as it measures the amount of associated uncertainty or *information* [2]. In networking, entropy rate measurements have been used for attack detection and network behavior profiling [20], [21], [22], [23], [24]. We aim to use entropy as an indicator of the degree of predictability associated with a traffic process. The neuroscience community has investigated various estimators for the entropy rate associated with the arrival of neural spikes [4], i.e., the computation of the entropy of a sequence of 1s and 0s. If 1s are associated to a packet arrival, and 0s to no packet arrival for a discrete time interval, a packet flow maps to a spike train. Entropy then measures the presence of *patterns* in the arrival process, as patterns reduce entropy. We use a variation of the plug-in estimator that we developed to demonstrate that certain flow processes have *memory* and thus reduced entropy.

IV. Entropy-Based Traffic Complexity Analysis

A. PPTEn estimator results

The raw data that we extract from the flow traces consists of packet arrival timestamps and packet sizes. Though entropy analysis can apply to both packet arrival times and packet sizes, we focus on arrival times.

Inspired by the neuron spike encoding used in [4], our approach is to encode packet arrivals in a simple binary sequence. In this approach, time is divided into bins of some size, τ and the binary value of each bin represents whether there was a packet arrival during that bin or not. The bin size, τ, is clearly an important parameter in this approach. With bin sizes too large, we risk losing information (multiple packet arrivals are treated the same as a single packet arrival). With bin sizes too small, no information is lost, but it turns out that the entropy estimates suffer because the abundance of empty bins drown out the effect of the few non-empty ones.

In their paper, Riihijarvi et al. [3] use several synthetic processes to verify that the output of their entropy estimator is in agreement with the expected complexity of the processes. We picked two of the same processes – the fractional gaussian noise process and the logistic map process – to verify that there is agreement between our entropy estimator and theirs. The plots in Figures 3 and 4 show that, in agreement with the SampEn estimator, the entropy ordering from our estimator of the 6 flows in order from lowest to highest entropy is:

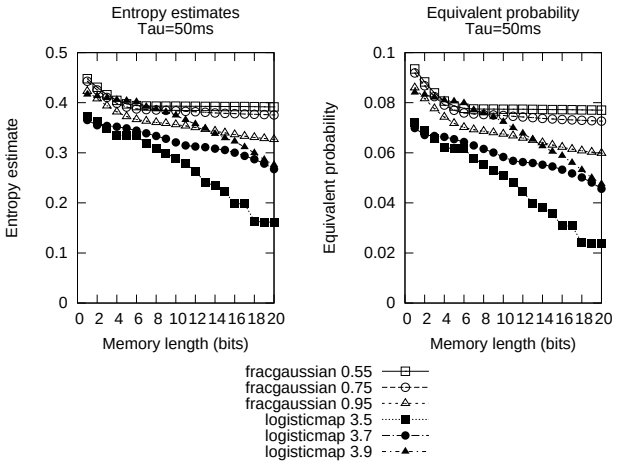


Fig. 3. Entropy estimates (a) and equivalent probability (b) of 2 types of synthetically generated flows. Using larger word lengths reduces the entropy estimate of the dataset because packet arrivals in a flow are not independent of each other.

Lowest entropy	Logistic map, $R = 3.5$
	Logistic map, $R = 3.7$
	Logistic map, $R = 3.9$
	Fractional gaussian, $H = 0.95$
	Fractional gaussian, $H = 0.75$
Highest entropy	Fractional gaussian, $H = 0.55$

1) Effect of word length: Equivalent probability is an alternative way of presenting the entropy information presented in Figure 3. Given an entropy estimate of a binary sequence, the equivalent probability is $p < (1 - p)$, that would yield that same entropy. Let's also assume that the entropy estimate is being used as a means of predicting what the next symbol in the sequence will be — a "1" or a "0" — and we pick whichever has a higher probability according to the estimator. In this scenario, the equivalent probability, p, becomes the probability that you will be wrong if you pick the symbol with higher probability. To restate that **in terms of word length: the more history you take into account when making your prediction, the smaller the chance your prediction will be wrong** (Figure 3).

2) Effect of Time Interval: Next we fix the word length at 15 bits and investigate the effect of time interval, τ, on the estimator output. Figure 4 shows the entropy estimate as a function of time interval.

Looking at the equivalent probability plot (Figure 4) as we did in the previous section, provides some intuition on the effect of τ on the entropy estimates. In general, **if τ is more than m times bigger or smaller than a flow's packet inter-arrival time, the effect will be a reduction in the probability of a wrong prediction**. Considering that we are predicting whether a packet will arrive or not during the next time interval, τ, a large τ will almost guarantee a packet arrival. Similarly, for a very small τ we can almost guarantee the absence of a packet arrival. This behavior is also readily observable in the application traces discussed below.

Ultimately, what τ will show a peak in the entropy estimate

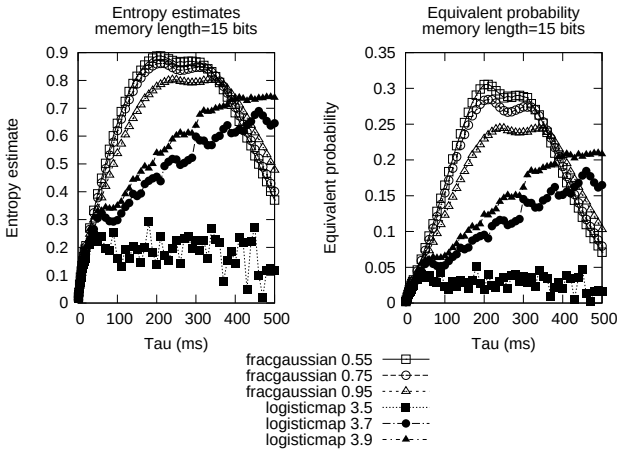


Fig. 4. Entropy estimates and equivalent probability of synthetic flows as a function of time interval, τ.

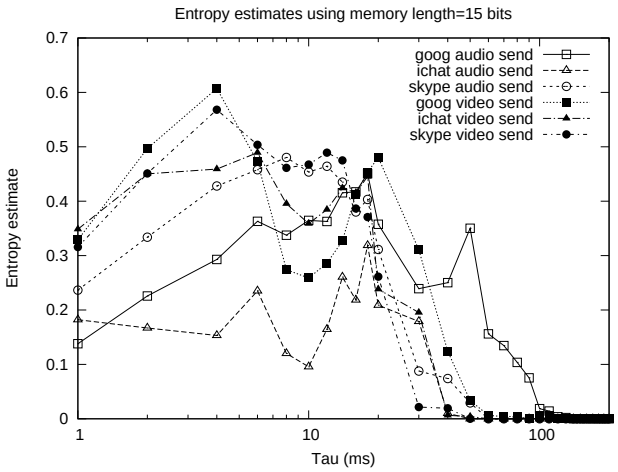


Fig. 5. Entropy estimates of VoIP and video conferencing flows using $m = 15$.

will depend on the flow itself. With low values of τ such as $1ms$, the entropy of the flow is likely to be low if the rate of packet arrivals is low (i.e. on the order of 1 packet every $20ms$) because in this case the bit string is mostly 0-bits with the occasional 1-bit, and the entropy estimator picks this up as a low entropy because the 1-bits are overpowered by the 0-bits. As τ increases we can reach the other extreme: a bit sequence with mostly 1-bits and the occasional 0-bit. This second case occurs when τ is approximately equal to the largest inter-arrival time in the flow.

For the remainder of the entropy estimates, we fixed the memory length of our estimator at $m = 15$ and varied τ between $1ms$ and $200ms$ for the real-time flows and between $0.001ms$ and $1000ms$ for the media streaming flows. The results are summarized in Figure 5 for real-time flows and Figure 6 for media streaming flows.

3) Real-time flow complexity: Depending on the packet arrival pattern of the flow being estimated, a different value of τ may be appropriate. It is necessary to observe the entropy estimates over a range of τ values to get a more complete picture. Figure 5 presents the entropy estimates of the VoIP and video conferencing traces that come from Skype, GoogleTalk and iChat.

From close inspection of the packet inter-arrival CDFs and the flow snapshots, we set the expectation for the trends that the entropy results should match in Section II. To summarize, we expect the entropy of VoIP flows to be lower than that of video conferencing flows, we expect that the iChat audio flow will have the lowest entropy and that GoogleTalk video will have the highest entropy.

First inspection of the entropies in Figure 5 may lead you to conclude that the first expectation — that VoIP flows have lower entropy than their video conferencing counterparts — has not been met. For τ in the range [8,18]ms, the entropy of GoogleTalk audio is larger than the entropy of GoogleTalk video. Similar discrepancies exist for Skype. However, if you note the entropy value of each flow at its highest point, you'll notice that the ordering that we expect is exactly what the estimator gives us:

Lowest entropy	iChat audio
	GoogleTalk audio
	Skype audio
	iChat video
	Skype video
Highest entropy	GoogleTalk video

This observation means that we can't pick some value of τ that will be appropriate for all the flows and compare their entropies using that single value of τ. The location of the peak entropy is related to the distribution of the inter-arrival times of the flow. For example, Figure 1(a) shows that GoogleTalk audio has a larger inter-arrival time than the other real-time flows and correspondingly, we see a peak in its entropy for a larger value of τ.

Due to the nature of the way we create the binary sequences that are fed to the estimator, all fluctuations in inter-arrival time smaller than τ are filtered out. Increasing τ increases the time scale at which the entropy estimates apply. Multiple peaks in the entropy of a flow mean that the flow exhibits complex behavior at multiple time scales. For example, the GoogleTalk video flow has a peak at $\tau = 4ms$ and then another at $\tau = 20ms$. The peaks correspond to the two larger sections of the flow's CDF in Figure 1(a).

4) Media streaming flow complexity: The PPTEn estimates for the media streaming flows are shown in Figure 6. We have used a wider range of τ values for these flows because their CDFs (Fig. 1(b)) and flow snapshots (Fig. 2(c)) indicate that they have inter-arrival time patterns at two very different time scales.

An examination of the peak entropies of the flows yields the following ordering:

Lowest entropy	Hulu.com "24"
	Webcam
	ABC.com "Castle"
	Hulu.com "Chuck"
	Hulu.com "American Dad"
Highest entropy	Netflix.com "OLT"

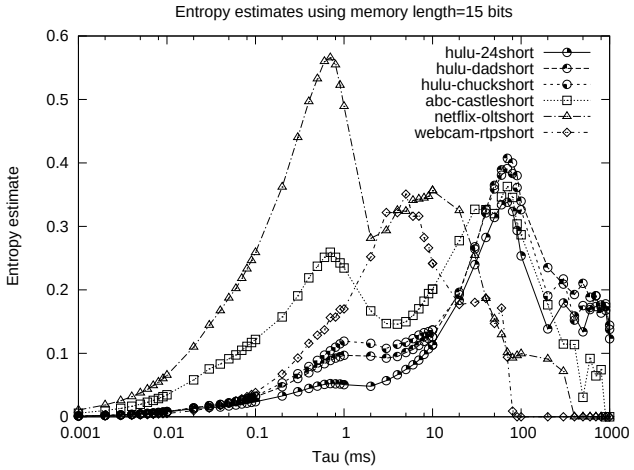


Fig. 6. PPTEn estimates of media streaming flows using $m = 15$.

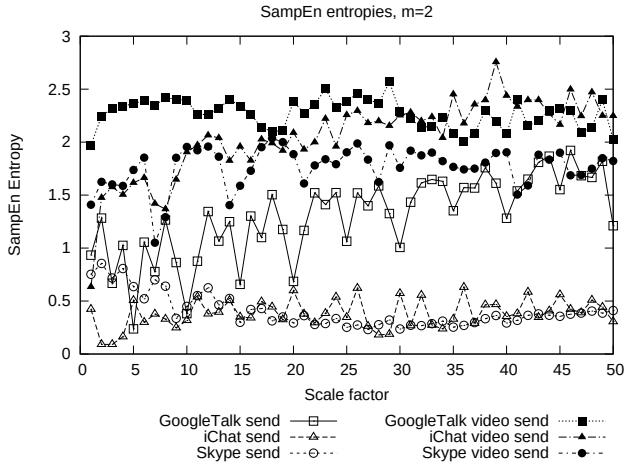


Fig. 7. SampEn estimates of real-time flows with SampEn parameter $m = 2$

The Netflix flow has the highest entropy as expected. Although the webcam flow was expected to have the lowest entropy, it is higher than one of the Hulu flows. Furthermore, a comparison between Figures 5 and 6 shows the peak entropies of the media streaming flows are lower than the peak entropies of the video conferencing flows and on par with the VoIP flows. It might be expected that the media richness of the media streaming flows should make them more complex than the VoIP flows, but that does not account for the buffered nature of one versus the real-time nature of the other. By using the coverage adjusted entropy estimator approach mentioned in Section III-B we can improve the sensitivity of the estimator and may be able to improve the strength of this result.

In our discussion of the media streaming dataset in Section II-B we mention that by just looking at the CDFs in Figure 1(b) it is difficult to distinguish between the Hulu flows and the ABC and Netflix flows. Furthermore, though the flow snapshots in Figure 2(c) show that Netflix has significantly different behavior than the ABC and Hulu flows, distinguishing the Hulu and ABC flows is still impossible. An inspection of the PPTEn estimates, however, shows that the ABC flow's behavior is quite different from that of the Hulu flows.

Not only do the peak entropies allow us to rank the complexity of application network traffic, but the location of the peaks also gives us valuable information about the scale at which we can expect the traffic to be most unpredictable. The location and height of the peaks forms a fingerprint for each application. It is left as future work to see how sensitive to the media content, such as speaker's voice or conversation content in VoIP or which television show is being streamed by a media streaming application, this fingerprint will be.

B. SampEn estimator results

We ran the SampEn estimator on the time series of packet inter-arrival times extracted from each application flow. The multiscale SampEn results are shown in Figure 7 for the real-time flows and in Figure 8 for the media streaming flows. A scale of 1 on the x-axis corresponds to the original time series, a scale of 2 means that every pair of samples in the original series has been averaged together to make up one sample, and so forth. This scaling approach, shown by Rihiijarvi et al [3] allows entropy estimates over larger timescales.

Unlike our PPTEn estimates, the multi scale SampEn estimates do not exhibit telltale peaks. It isn't clear what the effect of increasing the time scale is on the operation of the SampEn estimator, as it was for PPTEn.

The trends in Figure 7 indicate that Skype audio has the lowest complexity and iChat video and GoogleTalk video seem to have the highest. Strangely, the entropy of Skype video is lower than that of GoogleTalk audio and for larger time scales as low as that of Skype audio. The trends are not consistent with those shown by our PPTEn estimator or with the expectations that we outline in Section II-A. The effect of increasing m does not affect the overall trends but adds to the variability already evident when we use $m = 2$.

For the media streaming flows in Figure 8, one trend that is immediately apparent is that the SampEn estimator sees the webcam RTP stream as the most complex of the media traces. This is exactly opposite to our expectation and the result from our PPTEn estimator. The SampEn estimator correctly shows the Hulu traces having similar entropies and distinguishes them from the ABC trace, as did the PPTEn estimator. For the media streaming, with the exception of the webcam flow, the SampEn estimator provides the same entropy based ordering as PPTEn. A big drawback of SampEn in this application is that the concept of scale is not intuitively related to the behavior of the application whereas with PPTEn, it can be seen how the τ value of the entropy peaks relates to the network traffic patterns.

V. Applications of entropy estimator

We submit that PPTEn entropy estimation can be applied to traffic scheduling and admission control at the network edges, where traffic patterns have not yet been attenuated by queueing delays. The "entropy fingerprints" that we get from PPTEn summarize many characteristics of each application's network traffic. Not only can we compare applications on the basis of

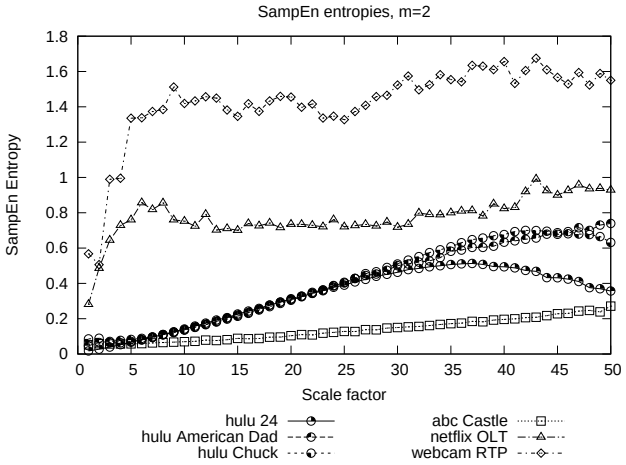


Fig. 8. SampEn estimates of media streaming flows with SampEn parameter $m = 2$

peak entropy, but we can also categorize them according to number of entropy peaks, the τ value of the entropy peaks, and even the range of τ over which the entropy is above some threshold (which is smaller, for example, for real-time applications that media streaming applications). Having all that information about each application will enable informed admission control decisions. Intelligent traffic scheduling, for example channel time allocation, at the medium access control layer can be done on the basis of the entropy based complexity characterization.

VI. Conclusion

In this paper we showed that entropy estimation works well for measuring the complexity of per-application network traffic. We presented results of using two entropy estimation approaches – our own PPTEn estimator and the SampEn estimator – and showed that the output of our PPTEn entropy estimator provides more information on the application behavior and can more readily be used to compare per-flow network traffic complexities.

The PPTEn estimates corresponded almost exactly to the network traffic complexity ordering we came up with based on visual analysis of the network traffic from Skype, GoogleTalk, iChat, Hulu.com, ABC.com, and Netflix.com. In addition, the PPTEn estimates over a range τ highlight many application characteristics, some of which are even too subtle for visual observation. We refer to the entropy estimates as "entropy fingerprints" because of how closely related to the flow characteristics they are.

The fact that so many application characteristics are reflected in its estimates makes PPTEn well suited for use in the realm of traffic scheduling and admission control.

Acknowledgment

This work has been partially supported by NSF grant CCF-091694 and a US Army-ARO MURI grant.

References

[1] L. Roberts, "A radical new router," *IEEE Spectrum*, July 2009.

[2] C. E. Shannon, "A mathematical theory of communication," bell sys," *Tech. Journal*, no. 27, pp. 379–423, 1948.

[3] J. Riihijarvi, M. Wellens, and P. Mahonen, "Measuring complexity and predictability in networks with multiscale entropy analysis," in *INFOCOM 2009, IEEE*, April 2009, pp. 1107 –1115.

[4] Y. Gao, I. Kontoyiannis, and E. Bienenstock, "From the entropy to the statistical structure of spike trains," in *IEEE International Symposium on Information Theory*, July 2006, pp. 645–649.

[5] J. S. Richman and J. R. Moorman, "Physiological time-series analysis using approximate entropy and sample entropy," *Am J Physiol Heart Circ Physiol*, vol. 278, no. 6, pp. H2039–2049, 2000. [Online]. Available: http://ajpheart.physiology.org/cgi/content/abstract/278/6/H2039

[6] "YouTube Wikipedia entry," [Online] Available at: http://en.wikipedia.org/wiki/Youtube, October 2010.

[7] "Hulu media faq," [Online] Available at: http://www.hulu.com/about/media_faq. [Online]. Available: http://www.hulu.com/about/media_faq

[8] N. Hunt, "Netflix encoding for streaming," [Online] Available at: http://blog.netflix.com/2008/11/encoding-for-streaming.html, November 2008.

[9] D. Bonfiglio, M. Mellia, M. Meo, and D. Rossi, "Detailed analysis of skype traffic," *Multimedia, IEEE Transactions on*, vol. 11, no. 1, pp. 117 –127, jan. 2009.

[10] "iChat in OS X Leopard," [Online] Available at: http://www.apple.com/asia/macosx/leopard/features/ichat.html, October 2010.

[11] "GoogleTalk developer info," [Online] Available at: http://code.google.com/apis/talk/open_communications.html, October 2010.

[12] "iChat Wikipedia entry," [Online] Available at: http://en.wikipedia.org/wiki/Ichat, October 2010.

[13] L. Berkeley, "National laboratory network research. tcpdump: the protocol packet capture and dumper program. http://www.tcpdump.org," in *the Protocol Packet Capture and Dumper Program," 2003*, 2001, p. 164.

[14] M. Perényi and S. Molnár, "Enhanced skype traffic identification," in *ValueTools '07: Proceedings of the 2nd international conference on Performance evaluation methodologies and tools*. ICST, Brussels, Belgium, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2007, pp. 1–9.

[15] D. Rossi, S. Valenti, P. Veglia, D. Bonfiglio, M. Mellia, and M. Meo, "Pictures from the skype," *SIGMETRICS Perform. Eval. Rev.*, vol. 36, no. 2, pp. 83–86, 2008.

[16] T. M. Cover and J. A. Thomas, *Elements of information theory*. New York, NY, USA: Wiley-Interscience, 1991. [Online]. Available: http://portal.acm.org/citation.cfm?id=129837

[17] F. Willems, Y. Shtarkov, and T. Tjalkens, "The context-tree weighting method: basic properties," *Information Theory, IEEE Transactions on*, vol. 41, no. 3, pp. 653–664, May 1995.

[18] G. Basharin, "On a statistical estimate for the entropy of a sequence of independent random variables," *Theory of Probability and its Applications*, vol. 4, p. 333, 1959.

[19] V. Vu, B. Yu, and R. Kass, "Coverage-adjusted entropy estimation," *Statistics in medicine*, vol. 26, no. 21, pp. 4039–4060, 2007.

[20] L. Feinstein, D. Schnackenberg, R. Balupari, and D. Kindred, "Statistical approaches to ddos attack detection and response," in *DARPA Information Survivability Conference and Exposition - Volume I*, vol. 1, April 2003, pp. 303–314 vol.1.

[21] K. Xu, Z.-L. Zhang, and S. Bhattacharyya, "Profiling internet backbone traffic: behavior models and applications," in *SIGCOMM '05: Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications*. New York, NY, USA: ACM, 2005, pp. 169–180.

[22] A. Wagner and B. Plattner, "Entropy based worm and anomaly detection in fast ip networks," in *WETICE '05: Proceedings of the 14th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprise*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 172–177.

[23] A. Lakhina, M. Crovella, and C. Diot, "Mining anomalies using traffic feature distributions," in *SIGCOMM '05: Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications*. New York, NY, USA: ACM, 2005, pp. 217–228.

[24] A. Lall, V. Sekar, M. Ogihara, J. Xu, and H. Zhang, "Data streaming algorithms for estimating entropy of network traffic," in *SIGMETRICS '06/Performance '06: Proceedings of the joint international conference on Measurement and modeling of computer systems*. New York, NY, USA: ACM, 2006, pp. 145–156.