Effect of system load on video service metrics

Ruairí de Fréin† ††

†KTH - Royal Institute of Technology, Stockholm, Sweden

††Waterford Institute of Technology, Ireland

web: https://robustandscalable.wordpress.com

in: Signals and Systems Conference (ISSC), 2015 26th Irish. See also BIBTEX entry below.

BIBTEX:

```
@article{deFrein15Effect,
author={Ruair\'{i} de Fr\'{e}in$^\dagger$ $^{\dagger\dagger}$},
journal={Signals and Systems Conference (ISSC), 2015 26th Irish},
title={Effect of system load on video service metrics},
year={2015},
pages={1-6},
keywords={client-server systems;computational complexity;frequency-domain analysis;
regression analysis; video signal processing;LR approach;client machine;
client-server service modeling literature;frequency domain;hierarchical model;
kernel-level metrics;linear regression;load signal;low computational complexity;
machine kernel metrics;model selection;service quality metric;service-level metrics;
system load effect;video service metrics;Delays;Frequency-domain analysis;
Histograms;Kernel;Load modeling;Servers},
doi={10.1109/ISSC.2015.7163768},
note = {\url{http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7163768&isnumber=7163737}},
month={June},}
```


document created on: November 5, 2015
created from file: rdefreinISSC_online.tex
cover page automatically created with CoverPage.sty
(available at your favourite CTAN mirror)

Effect of System Load on Video Service Metrics

Ruairí de Fréin
KTH Royal Institute of Technology, Sweden

***Abstract*—Model selection, in order to learn the mapping between the kernel metrics of a machine in a server cluster and a service quality metric on a client's machine, has been addressed by directly applying Linear Regression (LR) to the observations. The popularity of the LR approach is due to: 1) its implementation efficiency; 2) its low computational complexity; and finally, 3) it generally captures the data relatively accurately. LR, can however, produce misleading results if the LR model does not characterize the system: this deception is due in part to its accuracy. In the client-server service modeling literature LR is applied to the server and client metrics without treating the load on the system as the cause for the excitation of the system. By contrast, in this paper, we propose a generative model for the server and client metrics and a hierarchical model to explain the mapping between them, which is cognizant of the effects of the load on the system. Evaluations using real traces support the following conclusions: The system load accounts for $\geq 50\%$ of the energy of a high proportion of the client and server metric traces –modeling the load is crucial; the load signal is localized in the frequency domain: we can remove the load by deconvolution; There is a significant *phase* shift between both the kernel and the service-level metrics, which, coupled with the load, heavily biases the results obtained from out-of-the-box LR without any system identification pre-processing.**

I. Introduction

Cisco [1] predicts that network traffic volumes in the order of tens of exabytes are not that far off [2]; given that 90% of the bits transmitted on the Internet will be video related and that the number of consumers of these bits will soon exceed 1bn, monitoring customer experience will be crucial to safeguard revenues. The authors addressed the monitoring problem in [3]; an important open problem is how to predict the quality of video on a client's machine given the operational characteristics of the server delivering it. The first step is to develop a model that characterizes the system.

Related work: The authors of [4] made publicly available a series of datasets for evaluating techniques for predicting real-time service-level metrics from device statistics, which form the basis for our study. There are two notable differences between [4] and the present paper that warrant reporting: 1) They did not explicitly model the affect of the load on the service or kernel metrics they observed. In words, they assumed that the statistics of the system were exactly the same irrespective of the number of users requesting video. 2) They presented their results in terms of a Normalized Mean Absolute Error (NMAE), $\frac{1}{\bar{y}}\sum_i |y_i - \hat{y}_i|$, where $\bar{y}$ is the mean of the signal to be predicted, and $\hat{y}_i$ is the signal prediction. If the signal to be estimated y_i is zero mean, the NMAE is infinity, which is problematic. For non-zero mean signals, $a = \bar{y}$ and $b = \bar{\hat{y}}$, the NMAE may give arbitrarily good or bad scores for prediction performance. For example if the mean of $a = b = \bar{y} = \bar{\hat{y}} = 20$ and the absolute errors are $|y_i - \hat{y}_i| \approx 2$ the NMAE is generally $\approx 10\%$; however, if $a = b = \bar{y} = \bar{\hat{y}} = 10$ the NMAE is $\approx 20\%$ –which is twice as bad. Finally, if the means a, b vary across samples they may at times dominate the absolute errors, and at other times be dwarfed by the absolute errors. We address this problem.

We take a first step toward solving these problems using a standard Signal Processing approach: we ensure that 1) all signals are zero-mean and compute the Signal-to-Noise-Ratio between estimates and the ground-truth; 2) we propose a form of *supervised deconvolution*, [5], to remove the time varying load component in the prediction step. This approach is novel: in related works, the prediction of quality-of-service metrics for IPTV using decision trees was discussed in [6] using the aid of a domain expert to select relevant features. Network-level metrics, e.g., delay, loss, and jitter measurements, were used in [7] to estimate the quality-of-service metrics for IPTV streaming clients. The primary purpose of this paper is not prediction, but to characterize the system correctly and to describe the effects of disregarding the load on the system.

LR Set-up: We consider a standard LR problem: modeling the relationship between a scalar, dependent variable y_i and a number of explanatory variables (independent variables) which are denoted in vector form $\mathbf{x}_i$. The authors of [4] consider solvers other than LR; we focus on LR as it provides a well understood base-line technique to present ideas. If $y_i \in \mathbb{R}$ is a service-level metric on a client's machine in a networking scenario, $\mathbf{x}_i \in \mathbb{R}^N$ is a set of kernel metrics taken from a server machine, i is a time index in discrete time, the goal of this paper is to propose a model that correctly characterizes the relationship between the pairs $\{y_i, \mathbf{x}_i\}$ at previous measurement time indices. If the service is video, irrespective of the number of users on the system, the model should allow us to characterize the load's effect on the client's video quality if we have knowledge of the kernel metrics of the server delivering the video.

Contribution: To simplify our initial approach we assume that the network is sufficiently well resourced to safely disregard the effects of the network in this mapping [4]. LR is a popular model for fitting a predictive model to an observed dataset $\{y_i, \mathbf{x}_i\}$. Our objective in this paper is not to design a new LR solver but to demonstrate that *off-the-shelf* application of standard regression toolboxes, used in isolation, may lead to misleading results. Our first contribution is to propose a new generative model for service and kernel-level metrics in the Networking domain. We call this model/processing approach Rapsode: [R]egression [A]nd [P]rediction from [S]ynchr[o]nized [De]convolution. Our second contribution is to show that the application of supervised deconvolution [8], [5], source separation [9] and spectral subtraction-like techniques [10] allow us to perform prediction of the effects of the system load on y, by using a model that characterizes the system. To verify the efficacy of this model on a real-world dataset and to determine the suitability of our

model, we propose to consider the following experiment. If a previously unobserved vector of kernel measurements, $\mathbf{x}_{i+1}$, is drawn on the server machine, and we have no knowledge of its accompanying y_{i+1}, does our proposed model give a good prediction of the effect of the load on y_{i+1}?

Organization: In Section II we propose a first hierarchical model for relationship between the load on the system and the kernel and service metrics. In Section III we illustrate the effects of the load in the learning problem. In Section IV we provide evidence for the existence of the load and propose estimators to characterize it. We provide empirical evidence to support our results using video traces in Section V.

II. Problem Statement: Rapsode

We assume that the system is operating under a light to medium load. The relationship between the kernel metrics and the service metrics is well represented by a linear model. The response of the server, with respect to kernel metric n, to one request for video at time i is expressed as:

$$\mathbf{x}_i[n] = \hat{\mathbf{u}}_i[n] + \boldsymbol{\epsilon}_i[n], \qquad \text{where } i \in \mathbb{Z}, \mathbf{x}_i[n], \hat{\mathbf{u}}_i[n] \in \mathbb{R}. \quad (1)$$

The expression $\mathbf{x}_i[n]$ denotes the n-th element of the i-th time vector. The signal $\hat{\mathbf{u}}_i[n]$ could be a constant signal which corresponds to an increase in the CPU workload, for example, an extra X units per additional user for the duration of the video requested by the user. In a more general set-up, the vector $\hat{\mathbf{u}}$ is the sum of shifted Heaviside functions. We assume that the Heaviside functions are scaled in order to account for the sensitivity of a given feature to the effect of adding a new user. Without loss of generality, we assume that this scaling is specific to each service and machine. The noise signal $\boldsymbol{\epsilon}_i[n]$ captures deviations from the expected performance. It is convenient to assume that this deviations signal is normally distributed with 0 mean and variance σ^2, and that the deviation signals are uncorrelated. For two simultaneous video requests

$$\mathbf{x}_i[n] = 2\hat{\mathbf{u}}_i[n] + \boldsymbol{\epsilon}_i[n,1] + \boldsymbol{\epsilon}_i[n,2], \quad (2)$$

where the deviation from the expected performance due to the second user is $\boldsymbol{\epsilon}_i[n,2]$. When users start and stop watching video arbitrarily, $K(i)$ is the number of users at time i.

$$\mathbf{x}_i[n] = K(i)X + \sum_{k=1}^{K(i)} \boldsymbol{\epsilon}_i[n,k]. \quad (3)$$

We call the component signal, $\mathbf{l}_i[n] = K(i)X$, the load signal. In this paper, we consider the case of a sinusoidal variation in the load. This choices makes it easier to demonstrate the effect of the load in closed form. In addition, real traces are available from an independent study in which the traces have a sinusoidal component [4].

$$l_i[n] = K(i)X \approx a[n]\cos(\omega_n i + \phi_n). \quad (4)$$

The observed n-th kernel metric is the linear combination:

$$\mathbf{x}_i[n] = a[n]\cos(\omega_n i + \phi_n) + \hat{\mathbf{x}}_i[n]. \quad (5)$$

Here the real-valued scalars, $a[n] \in \mathbb{R}$, $\omega_n \in \mathbb{R}$ and $\phi_n \in \mathbb{R}$, are the amplitude, radial frequency, and phase of the load pattern –they collectively describe the behaviour of the ensemble of users. We make the simplifying assumption that they are constant. We also simplify our notation by introducing the notation $\hat{\mathbf{x}}_i[n] = \sum_{k=1}^{K(i)} \boldsymbol{\epsilon}_i[n,k]$, for the aggregate deviation. The deviation signal $\hat{\mathbf{x}}_i[n]$ is not explicitly delayed or attenuated.

This model is general: The amplitude scales the load to give it a response in the correct range for the n-th feature; if the n-th feature is not a function of the load, $a[n] = 0$. The phase ϕ_n may capture the network and machine delay between when the request is made and the response given. This model is further generalized by considering loads which are sums of sinusoids, or other types of parametric signals and stochastic processes. The service-level metric is a linear function of the set of kernel metrics. The service-level metric is modelled by:

$$y_i = \sum_n \mathbf{w}[n] \left\{ a[n]\cos(\omega_n i + \phi_n + \varphi_n) + \sum_{k=1}^{K(i)} \boldsymbol{\epsilon}_i[n,k] \right\} \quad (6)$$

The additional phase terms $\varphi_n, \forall n$ capture the delay in the effect of the load due to client requests on the server machine, and network and server delays.

System Characterization: We have introduced a deviations signal for each kernel feature, $\hat{\mathbf{x}}_i[n]$, to explain the deviation of each kernel metric from its ideal performance for a given load. A deviations signal is also required for the service-level metric, $\hat{y}_i$. We explain the effect of each user's request for video on the kernel-level metrics, and then on his service-level metric. In other words, we want to explain the signal that captures the deviation of the service-level metric from the ideal performance, as a function of the effect of the user requests on each of the video server's kernel metrics (the deviation of each of the kernel metrics from their typical performance). We propose the model,

$$\hat{y}_i = \mathbf{w}^T \hat{\mathbf{x}}_i, \quad (7)$$

and not the generative model we described above,

$$y_i = \mathbf{w}^T(\mathbf{l}_{i-d} + \hat{\mathbf{x}}_{i-d}). \quad (8)$$

The problem is that the signals that we can measure, the pairs $\{\mathbf{x}_i, y_i\}$, are mixtures of what we want and a high energy load component, which we do not want. The reason why we distinguish between these two problems is that the load may potentially drown-out the deviation signals, $\hat{y}_i$, $\hat{\mathbf{x}}_i[n], \forall n$. In general a good approximation of the load is known. There is little point in approximating it. The ability to estimate and predict deviations from ideal performance is the crucial problem. The approximation of the load comes from the TCPSCK field of the UNIX SAR command [11] which is run every second on the video server. The SAR command writes the contents of selected cumulative activity counters in the operating system to standard output. The number of TCP sockets currently in use gives a good indication of the load on the kernel. The service-level metric, RTP Packet Count, is obtained using a specially instrumented version of VLC Media player that writes the RTP Packet Count to file every second.

III. Learning Problem: Interfering Load

We want to model the effect of deviations from ideal behaviour for each of the kernel metrics, e.g. $\hat{\mathbf{x}}_i[n]$ on the deviations of the service-level metrics, e.g. $\hat{y}_i$. We do not want to model the load signal as we can measure it directly using the TCPSCK feature of x_i. In the remainder of this paper feature $n = n^\star$ denotes the TCPSCK feature, e.g. $\mathbf{x}_i[n^\star] = \mathbf{l}_i[n^\star]$. We

take the TCPSCK feature as the true load signal; the ideal load is known. With this knowledge we can begin to approximate the parameters ϕ_n, φ_n, a_n. In addition we need to estimate the set of frequencies, θ_l, which capture the load signal's energy.

Simple Learning Model: We consider the case where we do not acknowledge the presence of the load signal and attempt to fit the model in Eqn. 8. Let's initially only consider one feature for simplicity, e.g. $\mathbf{x}_i[n]$ and try to estimate y_i. We drop the time index and the feature index for notational convenience. We let both the dependent and independent variables consist of the sum of two components: the load component, which is denoted by a subscript l; and the deviation component, which is denoted by a subscript d. Therefore $y_i = y_l + y_d$, where $y_i, y_l, y_d \in \mathbb{R}$ and $\mathbf{x}_i[n] = \mathbf{x}_l + \mathbf{x}_d$ and $\mathbf{x}_i[n], \mathbf{x}_l, \mathbf{x}_d \in \mathbb{R}$. We consider one weight w. The sum of the error in the mapping for all times i is denoted:

$$\sum_i c(\mathbf{x}_i[n], y_i) = \sum_i ((y_l + y_d) - w(\mathbf{x}_l + \mathbf{x}_d))^2 \quad (9)$$

We can derive the weight using a Maximum Likelihood Estimation (MLE) argument that minimizes $\sum_i c(\mathbf{x}_i[n], y_i)$ by taking the derivative with respect to w and setting it to zero, and solving for w. Solving for w we have

$$w = \frac{\sum_i y_l \mathbf{x}_l + y_d \mathbf{x}_d + y_l \mathbf{x}_d + y_d \mathrm{x}_l}{\sum_i (\mathbf{x}_l^2 + \mathbf{x}_d^2 + 2\mathbf{x}_d \mathbf{x}_l)}. \quad (10)$$

We consider the relative importance of the load and deviation terms in this update. Consider the (cross)-correlation or energy terms for $y_l = a_n \cos(w_n i + \varphi_n)$, $x_l = a_n \cos(w_n i)$ where we let $\phi_n = 0$ for notational simplicity, and select $\varphi_n = \pi$ based on the results obtained in the empirical evaluation that follows.

Justification: It is important to note that if the service metric is the number of RTP packets received by the client and one of the features is the TCPSCK rate, that as the load increases the number of TCP sockets will increase, but the number of received RTP packets will probably begin to decrease as the system does not have unlimited resources. If the TCPSCK count is a sinusoidal function, for example $\cos(\omega_n i)$ then the number of RTP packets will be $\cos(\omega_n i + \pi) = -\cos(\omega_n i)$.

The cumulative deviations signal is defined as $\mathbf{x}_d = \hat{\mathbf{x}}_i[n] = \sum_{k=1}^{K(i)} \epsilon_i(n,k)$. The variance of each component $\epsilon_i(n,k)$ is σ^2, and each component is zero mean. The upper summation limit $K(i)$ changes with time. The variance of the sum of uncorrelated random variables is the sum of their variances. We make the simplifying assumption that the random deviations are uncorrelated. We denote the variance of the sum of the deviations overall time to be $\sigma_x^2(i) = \mathrm{var}(\hat{\mathbf{x}}_i[n]) = \mathrm{var}(\sum_{k=1}^{K(i)} \epsilon_i(n,k)) = K(i)\sigma^2$. The time variation arises because of the changing number of users. As we assume that $K(i)\sigma^2 \ll a[n]^2, \forall i$ we approximate this variance with a constant $\sigma_x^2 = \sigma^2 \max K(i)$ in the following argument.

Remark: These assumptions are reasonable. In a real system, the user-dependent variance assumption, $K(i)\sigma^2$, implies that the more users requesting video, the more volatile the quality of each user's video will be. The second assumption $K(i)\sigma^2 \ll a[n]^2$ implies that the variance of each kernel metric's behaviour, for a given number of users on the system, will typically be much smaller than the mean effect of that number of users on the kernel-level metric. This assumption is reasonable when the system is light-to-medium loaded, and thus catastrophic system failures may be disregarded.

TABLE I. COMPONENTS OF THE WEIGHT LEARNING RULE

$\frac{1}{T}\sum_i y_l \mathrm{x}_l = -\frac{1}{T}\sum_i a^2[n]\cos^2(\omega_n i) = -\frac{a[n]^2}{2}$
$\frac{1}{T}\sum_i y_d \mathrm{x}_d = \sigma_{xy}^2$
$\frac{1}{T}\sum_i y_l \mathbf{x}_d = \sum_i -\frac{a[n]}{T}\cos(\omega_n i)\mathbf{x}_d$, $\frac{1}{T}\sum_i y_d \mathbf{x}_l = \sum_i \frac{a[n]}{T}\cos(\omega_n i)\mathbf{y}_d$
$\frac{1}{T}\sum_i \mathrm{x}_l \mathrm{x}_l = \frac{1}{T}\sum_i a^2[n]\cos^2(\omega_n i) = \frac{a_n^2}{2}$
$\frac{1}{T}\sum_i \mathrm{x}_d \mathrm{x}_d = \frac{1}{T}\sum_i \mathrm{x}_d \mathrm{x}_d = \sigma_x^2$
$\frac{1}{T}\sum_i \mathbf{x}_d \mathbf{x}_l = \frac{1}{T}\sum_i a[n]\cos(\omega_n i)\mathbf{x}_d$

TABLE II. DETECTING LOAD SUPPORT, θ_l, FROM TCPSCK, $\mathbf{x}_i[n^\star]$.

Sort from largest to smallest: $\lvert\mathbf{X}_{n^\star}(s)\rvert = \mathrm{sort}(\lvert T\{\mathbf{x}_i[n^\star]\}(\omega)\rvert)$
Compute the definite summation: $I(b) = \sum_{s=1}^{b} \mathbf{X}_{n^\star}(s)$
Find the transition point: $b^\star = \mathrm{locate}\{I(b) > T\}$
Generate the load set: $\theta_l = \{\omega = \mathrm{unpermute}\{s\} \lvert 1 < s < b^\star\}$.

Analysis: We scale the numerator and denominator terms (in Eqn. 10) by $\frac{1}{T}$ for notational convenience and compute the summations over an appropriate range of samples in Table I. When the load signal dominates the residual signal, $a^2[n] \gg \sigma_x^2, \sigma_{xy}^2$, $a^2[n] \gg a[n]$, the derived weights depend solely on the phase difference between the server and client's observations of the load

$$w = \frac{-\frac{a[n]^2}{2} + \frac{a[n]}{T}\sum_i \cos(\omega_n i)\,(\mathbf{y}_d - \mathbf{x}_d) + \sigma_{xy}^2}{\frac{a[n]^2}{2} + \frac{1}{T}\sum_i a[n]\cos(\omega_n i)\mathbf{x}_d + \sigma_x^2} \approx -1. \quad (11)$$

This implies that if the load is the dominant signal component, when the load increases on the system, the RTP Packet Count decreases. On the other hand, when there is no load signal present the update rule reduces to the ratio of 1) the covariance of the deviations of the kernel deviations signal with the service deviations signal with 2) the auto-correlation of the kernel deviations signal $w = \sigma_{xy}^2/\sigma_x^2$, which is precisely what we would like to understand better. This argument is easily extended to the case where the service-level metric is modeled by the sum of a number of features, some of which have the load component, $a_i[n] \neq 0$, and the terms that do have the load component are in phase. In this case, the residual does not play a significant role in the update for the corresponding component of $\mathbf{w}[n]$ as the load dominates –this is undesirable when the purpose of prediction is to be able to forecast failure.

Property: Learning a LR model when the load is mixed with the deviations in the manner above yields a weight which captures the relative attenuation and phase shift between the load measured on the server's kernel and the client's machine, as opposed to the deviations terms –which is what we are really interested in. The purpose of LR is not to learn the relative phase shift between the kernel and service-level metrics.

Time-varying Mean Component: An alternative way of modeling this process is by modeling the observed client service-level metric y_i as a function of a varying mean – corresponding to the system load– and the variation about it,

$$y_i = \mu(i) + \sum_n \mathbf{w}[n] \left\{ \sum_{k=1}^{K(i)} \epsilon_i[n,k] \right\} \quad (12)$$

This is clearly not a linear regression problem in the standard form. How can we learn a mapping which is independent of the load? How can we subtract the load component from each signal? How can we estimate the load signal?

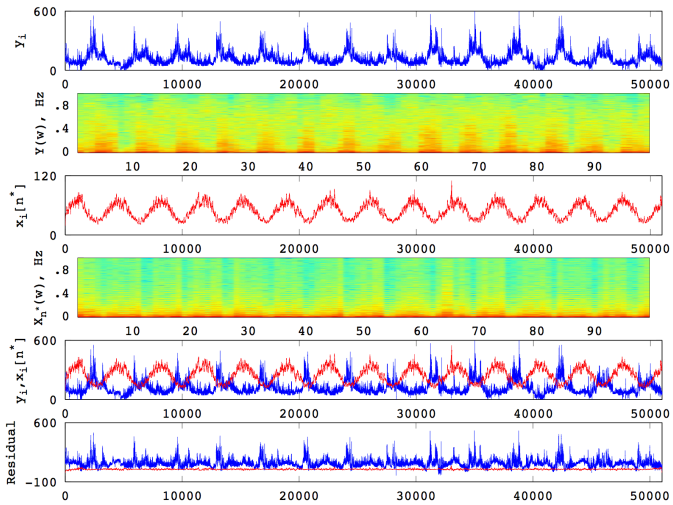

Fig. 1. RTP Packet Count, y_i recorded over a $\approx$ 50k second interval (row 1); Its time-frequency magnitude spectrogram (row 2); TCPSCK kernel parameter (row 3), $\mathbf{x}_i[n^\star]$, which is a good proxy for the load; Its time-frequency magnitude spectrogram (row 4). The RTP Packet Count has the same period as the TCPSCK parameter. Row 5 overlays an amplitude scaled version of the TCPSCK feature with the RTP Packet Count. Row 6 plots the filtered RTP Packet Count $\hat{y}_i$ and the TCPSCK feature $\hat{\mathbf{x}}_i[n^\star]$.

IV. Load Detection

We begin by providing evidence that the load plays a significant role in the performance of the service-level metrics and the kernel metrics. We plot the RTP Packet Count recorded over a $\approx$ 50k second interval, its time-frequency magnitude spectrogram and the TCPSCK kernel parameter, which is a good proxy for the load, along with its time-frequency magnitude spectrogram in Fig. 1. From visual inspection, it is clear that the RTP Packet Count has the same period as the TCPSCK parameter. There is a strong sinusoidal component in both signals which is evident in the lower frequency ranges of the time-frequency magnitude spectrograms in Fig. 1. The response of the RTP packet count, y_i, to the load, $\mathbf{x}_i[n^\star]$, captured here by the SAR TCPSCK feature, is $\varphi \approx \pi$ degrees out of phase with it. It is reasonable to model both signals as: $y_i = wa[n^\star]\cos(\omega_n i + \frac{\pi}{2}) + \hat{y}_i$, and $\mathbf{x}_i[n^\star] = a[n^\star]\cos(\omega_n i - \frac{\pi}{2}) + \hat{x}_i[n^\star]$. If we learn a mapping from the kernel parameters, which has a term $\cos(\omega_n i + \frac{\pi}{2})$, to the service-level metric, which has a term $\cos(\omega_n i - \frac{\pi}{2})$, the dominant component of the kernel-level metric is π rads out of phase and the weight, w, attempts to compensate for the phase difference in the different signals. The SAR feature corresponding to the TCPSCK feature dominates the RTP Packet Count[1].

In order to estimate the relative attenuation and delay parameters (α_ω and δ_ω in frequency bin ω) between any two signals we call on the linear transform $\mathrm{T}\{y_i\}(\omega) : y_i \mapsto Y(\omega)$ and $\mathrm{T}\{\mathbf{x}_i[n]\}(\omega) : \mathbf{x}_i[n] \mapsto \mathbf{X}_n(\omega)$. In the discrete frequency domain, e.g. when $T\{\cdot\}()$ is the DFT, ω denotes the frequency bin index, and $T\{y_i\}(\omega)$ and $T^{-1}\{Y(\omega)\}(i)$ are the DFT and IDFT respectively. The magnitude spectrogram is defined as: $|T\{y_i\}(\omega)|$. It is a property of the DFT that a delay in the discrete time domain is an phase shift in the discrete frequency domain. The relative attenuation, for two discrete frequency domain signals, $X_1(\omega)$ and $X_2(\omega)$, is defined as well.

$$\delta_\omega = -\frac{1}{\omega}\angle\left\{\frac{\mathbf{X}_1(\omega)}{\mathbf{X}_2(\omega)}\right\},\ \alpha_\omega = \left|\frac{\mathbf{X}_1(\omega)}{\mathbf{X}_2(\omega)}\right| \tag{13}$$

Let's consider how dominant the load component is in the service-level metric (RTP Packet Count). We evaluate the following ratio to find the content of the signal resulting from the load component. The entire set of frequency bins is θ. The set of frequency bins corresponding to the load is θ_l.

$$E(y_i|\theta_l) = 100\frac{\sum_{\omega\in\theta_l}|T\{y_i\}(\omega)|^2}{\sum_{\omega\in\theta}|T\{y_i\}(\omega)|^2}, \tag{14}$$

The set θ_l is determined from the TCPSCK feature using the algorithm in Table. II. The index variable s denotes the index into the sorted vector $|\mathbf{X}_{n^\star}(s)|$. The operator $\omega = \text{unpermute}\{s\}$ gives the ω corresponding to a given s. The summation takes as its lower bound the first non-zero frequency bin (DC is excluded). This computation illustrates that 50.846% of the RTP Packet signal lies in 0.07% of the frequency bins, the set θ_l and that 86% of the energy of the TCPSCK signal lies in the same frequency bins, θ_l. The set of load bins, θ_l, is selected by ordering, by magnitude, the DFT coefficients and choosing the frequency bins that account for 86% of the signal. Somewhat surprisingly the load signal is well localized. We conclude that: 1) The load signal has narrow support in the frequency domain of the TCPSCK signal; 2) The TCPSCK signal has a narrow bandwidth, which is indicative of a sinusoidal signal; 3) The dominant component in the RTP Packet Count is the TCPSCK signal, which is a proxy for the load. Many of these properties are exploited by the Blind Source Separation community in a different setting [5], [8], [9]. We posit that we can remove the sinusoidal component from both the RTP Packet Count and the TCPSCK signal using the binary mask (and θ_l computed from Table II):

$$M_{\theta_l}(\omega) = \begin{cases} 1, & \omega = 0, \text{ or } \omega \notin \theta_l \\ 0, & \omega \in \theta_l \end{cases} \tag{15}$$

and resynthesize the residual signal, $y_d \approx \mathrm{Re}\{T^{-1}\{M_{\theta_l}(\omega)\mathbf{X}_n(\omega)\}(i)\}$, which is the signal that measures the deviation of the service-level from its typical level. The operator $\mathrm{Re}\{\cdot\}$ returns the real values of its argument. We postulate that the remaining signal is the deviation signal of the service-level metric, which a function of the deviation signal of the kernel metrics. Fig. 1 illustrates the residual signal obtained from this procedure.

We use a form of power-weighted estimator, similar to the estimators proposed by Rickard and Yilmaz [9]. They are motivated by the idea that y_i and $\hat{x}_i[n^\star]$ consist of a load component and a deviation component. The frequency bins –where the component we desire resides– should have a weighting corresponding to the cross-energy of that component (across both the service and the kernel metric), as

TABLE III. Relative α and δ between y_i and $\mathbf{x}_i[n^\star]$.

	α	δ
Power Weighted Estimators	3.0038	-1784.4 samples

[1] A phase shift does not always imply a time shift [8]. We determine later that the observed phase shift corresponds to a reflection through the mean of the signal, an observation that agrees with the update rule for the weights calculated previously. Secondly, we have assumed the TCPSCK feature is a good proxy for the load and therefore its associated deviations signal is zero, $\hat{\mathbf{x}}_i[n^\star] = 0$. This is not true for the other features, $\hat{\mathbf{x}}_i[n] \neq 0, \forall n \neq n^\star$.

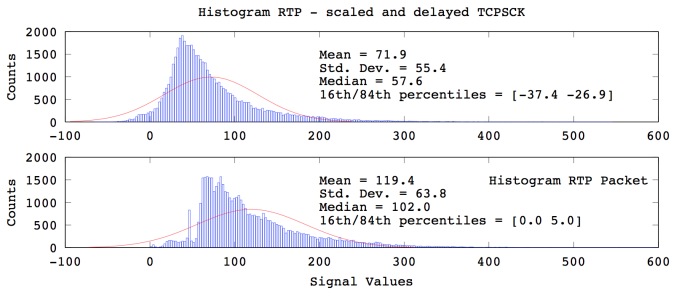

Fig. 2. Histogram of RTP Packets: Residual signal $\hat{y}_i$ (in row 1) is unimodal but skewed; y_i (in row 2) is tri-modal and skewed.

opposed to giving equal weightings to frequency bins which correspond to spurious noise. This is achieved by setting $C(\omega) = T\{y_i\}(\omega)T\{\mathbf{x}_i[n^\star]\}(\omega)$

$$\alpha = \frac{\sum_{\omega\in\theta_l} |C(\omega)|^2\alpha_\omega}{\sum_{\omega\in\theta_l} |C(\omega)|^2}, \quad \delta = \frac{\sum_{\omega\in\theta_l} |C(\omega)|^2\delta_\omega}{\sum_{\omega\in\theta_l} |C(\omega)|^2} \tag{16}$$

where α_ω and δ_ω are the instantaneous relative amplitude and delay experienced by the RTP Packet Count given the TCPSCK signal. The application of these estimators is justified: 1) Not all frequency bins are as important; 2) The load component is localized in a 0.07% of the frequency bins in both signals; 3) The cross product between the RTP Packet Count and TCPSCK acts to cancel out interfering terms, emphasizing the common component in both signals, i.e. the load; 4) The load dominates both y_i and $\mathbf{x}_i[n^\star]$.

V. Empirical Evaluation of the Rapsode Model

In this section we: 1) fit the TCPSCK feature to the RTP Packet count (using α, δ) to estimate the deviations signal $\hat{y}$; 2) evaluate the presence of the load in all the features to see if it is as pervasive as our analysis suggests; and finally, 3) evaluate how well the load predicts the RTP Packet count.

1) One-feature estimation: Given the dependence of the service-level metric on the SAR TCPSCK feature, we attempt to estimate the service-level metric using the SAR TCPSCK. The relative attenuation and delay between the TCPSCK feature and the RTP Packet Count signal are estimated (cf. Eqn. 6) in Table III using 50k samples. We plot a histogram of the RTP Packet count residual signal estimate, $\hat{y}_i = y_i - \alpha\mathbf{x}_{i-\delta}[n^\star]$, along with the original RTP Packet count signal in Fig. 2. Note that the RTP Packet Count residual, $\hat{y}$, has only one mode compared to the original RTP Packet Count. This is another benefit of removing the load component from the service-level feature when we are looking to predict the future service-levels, as LR models typically model the noise as a unimodal (Gaussian) distribution. The deviation histogram is still skewed; we will address this in future work by examining the quality of the procedure for estimating the set θ_l.

2) Universality of the load: We discard the non-numeric kernel metrics and the kernel metrics which have a constant value. Approximately 231 out of the 854 kernel metric signals are left, e.g. 27%. These 231 features are used for estimation and prediction (cf. Table IV, rows 1-2). We examine the effect of the load signal on these 231 kernel metrics to determine if $a[n]$ is non-zero in (Eqn. 6). We evaluate the energy of the signal present in the frequency bins (of the kernel metrics) corresponding to 86% of the energy of the TCPSCK kernel metric, the set θ_l. Row 4 in Table IV states that approximately 20% of the kernel features have greater than 50% of their energy in the same set of frequency bins, θ_l, as the load. Row 3 and 5 give the percentage for a percentage energy content greater than equal to 90% and 40%. The load is captured by 0.07% of the frequency bins of $\hat{\mathbf{x}}_i[n^\star]$. This statistic motivates the assertion that the discrete frequency domain support of many of the kernel features is approximately the same as the support of the TCPSCK signal. If the energy in these frequency bins is due to the load, the relative delay between y_i and the 231 other features, $\hat{\mathbf{x}}_i[n]$, e.g. δ_{yx} should be approximately the same. However, if there is significant interference due to the response of the kernel to the load in these same frequency bins, the relative parameter estimators will perform poorly.

TABLE IV. Analysis of the Kernel Metrics

1	Total No Kernel Metrics, y_i	854
2	"Useable" Kernel Metrics, $\mathbf{x}_i[n^\star]$	231
3	$E(\mathbf{x}_i[n]\|\theta_l) > 90, \forall n$ in .07% of the θ bins	$\approx 10\%$
4	$E(\mathbf{x}_i[n]\|\theta_l) > 50, \forall n$ in .07% of the θ bins	$\approx 20\%$
5	$E(\mathbf{x}_i[n]\|\theta_l) > 40, \forall n$ in .07% of the θ bins	$\approx 42\%$
6	$-20 \leq \delta_{xx} \leq 20$	for $\approx 25\%$ of the features.
7	$\delta_{xx} = 1760 \pm 20$	for $\approx 11\%$ of the features
8	$\delta_{yx} = 1760 \pm 20$	for $\approx 16\%$ of the features.
9	$\delta_{yx} = 1760 \pm 20$ or $-50 < \delta_{xy} \leq 0$	for $\approx 37\%$ of the features.

Relative Delay Estimates: Firstly, we compute the relative phase shift between all of the features and the TCPSCK feature, namely the feature-to-feature estimates, δ_{xx}. Secondly, we compute the relative phase shift between all of the features and the RTP Packet Count, δ_{yx}. The histogram bins are 20 samples apart in Fig. 3. We consider relative delays between the features in the range $-6000 \leq \delta_{xx} \leq 6000$ for all 50k samples. We claim that the estimates δ_{xx} are clustered around 0 (cf. Table IV row 6). The relative delay δ_{xx} between 25% of the features, e.g. 61, and the load is ≈ 0 samples. Examining δ_{yx}, 33 of the estimates yield a delay of 1760 ± 20 samples (cf. Table IV row 8). In other words, approximately 16% of features have a relative delay with the load that is the same. The peaks at the relative delays ± 6000 mean the relative delays δ_{yx} –for ≈ 20 of the feature vectors– are greater/less than ± 6000 samples. Given that the sinusoidal load oscillation is ≤ 6000 samples we rule these delays out of being due to the load (unless they are integer multiples of the period of the load). Three of the δ_{yx} histogram bins, indicated $33, 27, 25$, account for 85 of the features; all of the other histogram bins have less than 6 features in them. The bins corresponding to the count labels $27, 25$ correspond to a δ_{yx} which is approximately zero. In our simplified model for the affect of the load, we modelled it as a cosine: $\cos(\omega_n i + \varphi_n)$, assuming $\phi_n = 0$.

Discussion: We posit that half a cosine oscillation corresponds to ≈ 1760 samples. Recall that $\cos(\omega_n i + \pi) = -\cos(\omega_n i)$. The peaks in the histograms δ_{xx} and δ_{yx} are highly significant because they correspond to the case where the negative load, $-\cos(\omega_n i)$ and the positive load $\cos(\omega_n i)$ appear in the kernel features and the service-level metric, and the peaks in the relative delay histograms are accounted for in this way. This means that the relative delay of ≈ 1760 samples between RTP Packet Count and 33 of the features is due to the negative load $-\cos(\omega_n i)$. Approximately zero delay is measured between the RTP Packet Count and another $25 + 27$ of the features,

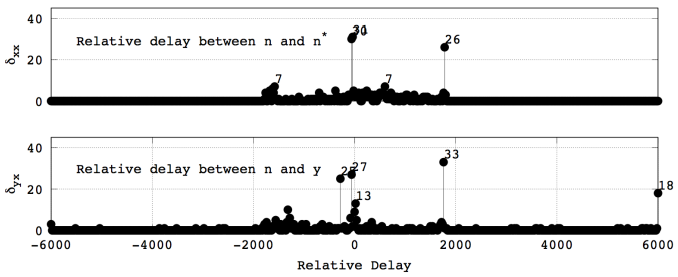

Fig. 3. δ_{xx} and δ_{yx} estimation.

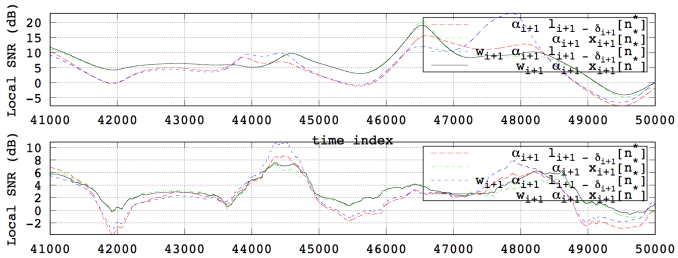

Fig. 4. The upper figure illustrates how well the load predicts the entire RTP Packet Count. The lower figure illustrates how well the load predicts the load.

e.g. the load is $\cos(\omega_n i)$. In the δ_{xx} histogram the relative delay of ≈ 1760 samples between TCPSCK and 26 of the other features is due to the negative load $-\cos(\omega_n i)$. Approximately zero delay is measured between the TCPSCK and 61 of the other features due to the load being $\cos(\omega_n i)$.

In summary, $\approx 42\%$ of the features have greater than 40% of their energy in the same frequency bins as the load. The load has a frequency domain support of 0.07% of the frequency bins. Approximately 37% of the usable kernel features have a relative delay with the service-level metric and TCPSCK which is consistent with the load signal being modelled as $\cos(\omega_n i)$ or $-\cos(\omega_n i)$. We conclude that the load plays a big role in characterizing the kernel features and the RTP Packet Count.

Local Load Predictions: Given that the load is highly prevalent in $30-40\%$ of the kernel features we evaluate how well the load, $\mathbf{x}_i[n^\star]$ predicts the load component of the RTP Packet Rate, the frequency bins θ_l, e.g. $y_i|\omega \in \theta_l$. Then, we evaluate how well the load $\mathbf{x}_i[n^\star]$ predicts both the deviations and the load of the RTP Packet Rate, the signal y_i. The purpose of these experiments is not to perform perfect prediction, but to 1) demonstrate that the load is prevalent and helps perform prediction; 2) justify our characterization of the model; and finally, 3) evaluate the combination of our relative attenuation, delay and MLE weight estimators. We believe that these prediction algorithms provide a good base-line for future algorithms that exploit all of the features –the purpose of the predictions in this paper is to establish how well the Rapsode model characterizes the system. For the two experiments above, we vary the complexity of the predicted signal to demonstrate the improvement gained by introducing more parameters. Each prediction algorithm estimates 1) the set of frequency bins θ_l; and 2) the relative attenuation and delay α, δ. To increase the complexity of the model we then consider predictions which incorporate the MLE weight estimate w – using the sets of parameters $\{\theta_l, \alpha, \delta\}$ and $\{\theta_l, \alpha, \delta, w\}$.

TABLE V. PREDICTOR ALGORITHMS

Estimate δ_{xy} and α_{xy} from y_i and $\mathbf{x}_i[n^\star]$ given θ_l for the last 30k samples
Evaluate the MLE of w using the last 30k samples
Generate the predictions $\hat{y}_{i+1} = w\alpha\mathbf{l}_{i+1-\delta}[n^\star]$, $\hat{y}_{i+1} = \alpha\mathbf{l}_{i+1-\delta}[n^\star]$, $\hat{y}_{i+1} = -w\alpha\mathbf{l}_{i+1}[n^\star]$, $\hat{y}_{i+1} = -w\alpha\mathbf{l}_{i+1}[n^\star]$.
Evaluate the local SNR measure S_{i+1} between $\hat{y}_{i+1}$ and y_{i+1}.
Evaluate the local SNR measure S_{i+1} between $\hat{y}_{i+1}$ and y_{i+1} in the bins θ_l.

Each prediction algorithm is a two-step process summarized in Table V. First, we estimate the load signal $\mathbf{l}_i[n^\star] = \mathbf{x}_i[n^\star]|_{\omega\in\theta_l}$ using the frequency bins θ_l. We use this load signal estimate, and y_i, the service-level metric (RTP Packet count), to learn estimates of α, δ for some training period (samples $r = 10000 \ldots 40000$). We center the signals corresponding to the previous 30000 samples. We compute the Maximum Likelihood Estimate for the weight parameter given fixed parameters α and δ, $w = \left(\sum_{i\in r} \alpha\mathbf{l}_{i-\delta}[n^\star]y_i\right) / \left(\sum_{i\in r}(\alpha\mathbf{l}_{i-\delta}[n^\star])^2\right)$. We then compute the predictions $\hat{y}_{i+1} = w\alpha\mathbf{l}_{i+1-\delta}[n^\star]$, $\hat{y}_{i+1} = \alpha\mathbf{l}_{i+1-\delta}[n^\star]$, $\hat{y}_{i+1} = -w\alpha\mathbf{l}_{i+1}[n^\star]$ and $\hat{y}_{i+1} = -w\alpha\mathbf{l}_{i+1}[n^\star]$. We use the Signal-to-Noise-Ratio (SNR) of the service-level metric-to-(Service-level metric minus prediction)-Ratio as our indicator of performance. We consider the local SNR, e.g. between the last 1000 predictions and true values.

$$S_{i+1} = 10\log_{10}\left(\sum_i y_{i+1}^2\right) / \left(\sum_i (y_{i+1} - \hat{y}_{i+1})^2\right) \qquad (17)$$

We then re-position our training set window over the next range of samples $10001 \le i \le 40001$; center the signal; recompute the parameters α, δ for this new window of samples; learn a MLE for the weight w; and generate a new prediction. We continue this procedure for 10000 samples so that we can evaluate the effect of locally obtained parameters α, δ and the weight w on the prediction process 10000 times, in a manner similar to a real-prediction engine.

Discussion: Fig. 4 illustrates that the models $\hat{y}_{i+1} = -w\alpha\mathbf{l}_{i+1}[n^\star]$ and $\hat{y}_{i+1} = -w\alpha\mathbf{l}_{i+1}[n^\star]$ predict the entire RTP Packet Count the best, in the sense that the lowest local SNR is almost always positive. Reflecting the load signal through its mean, as opposed to delaying the load by π rads, gives the best predictions. It is remarkable that the load predicts the entire RTP Packet Count with such high fidelity. If we compute the instantaneous SNR of each prediction, e.g. $10\log_{10} y_{i+1}^2/(y_{i+1}-\hat{y}_{i+1})^2$ in isolation, 71% of the 10000 predictions have a positive SNR, and the median instantaneous SNR is $\approx$ 3dB. Similarly in the case where the load $\mathbf{x}[n^\star]$ is used to predict the load component of y_i, predictors $\hat{y}_{i+1} = -w\alpha\mathbf{l}_{i+1}[n^\star]$ and $\hat{y}_{i+1} = -w\alpha\mathbf{l}_{i+1}[n^\star]$ give the best SNRs in the sense that the SNR is never the lowest. The effectiveness of the parameters is summarized as follows: the parameters α and δ scale and align the TCPSCK signal with the load component, y_l in the observed service-level metric trace, and the weight w learns the mapping from the scaled, aligned load signal to the observed service-level metric, which has an additional deviation component, recall $y_i = y_l + y_d$. The weight gives the algorithm a small additional flexibility to compensate for the deviations component, y_d, of the signal y_i; parameters α and δ are not intended to serve this purpose as we assume that the set of frequency bins θ_l in both y_i and $\mathbf{x}_i[n^\star]$ correspond to the load. Typically an evaluation of this type of procedure looks to break the correlation in prediction errors by randomly partitioning the data into a test and training set. In a real system errors in our predictions are potentially correlated. In future work we will consider

the implementation of the predictors using a parallel online learning framework, e.g. [12].

VI. Conclusions

Adopting the correct model for a set of observations from a real-world random process is a crucial first step for developing statistical inference algorithms. We have demonstrated in this paper that adopting the incorrect model causes incorrect, although potentially accurate, predictions to result. Accuracy and correctness are therefore not the same thing if the model is misspecified. In future work we will generalize theses results to the case where there are multiple services.

References

[1] Cisco, "Cisco visual networking index, Global IP Traffic Forecast, 2011-2016.," *white paper*, 2011.

[2] A. Begen, T. Akgul, and M. Baugher, "Watching video over the web part2: Applications, standardization, and open issues.," *IEEE Internet Comp.*, vol. 15, no. 3, pp. 5963, 2011.

[3] Ruairí de Fréin, C. Olariu, Y. Song, R. Brennan, P. McDonagh, A. Hava, C. Thorpe, J. Murphy, L. Murphy, and P. French, "Integration of QoS Metrics, Rules and Semantic Uplift for Advanced IPTV Monitoring," *Journal of Network and Systems Management*, pp. 1–36, 2014.

[4] R. Yanggratoke, J. Ahmed, J. Ardelius, C. Flinta, A. Johnsson, D. Gillblad, and R. Stadler, "Predicting real-time service-level metrics from device statistics," Tech. Rep., KTH, 2014.

[5] Ruairí de Fréin, S.T. Rickard, and B.A. Pearlmutter, "Constructing time-frequency dictionaries for source separation via time-frequency masking and source localisation," in *Independent Component Analysis and Signal Separation*, vol. 5441 of *LNCS*, pp. 573–80. Springer, 2009.

[6] A. Balachandran, V. Sekar, A. Akella, S. Seshan, I. Stoica, and H. Zhang, "Developing a Predictive Model of Quality of Experience for Internet Video," in *Proc. ACM SIGCOMM*, 2013, pp. 339–350.

[7] S. Handurukande, S. Fedor, S. Wallin, and M. Zach, "Magneto approach to QoS monitoring," in *Integ. Net. Man. IFIP/IEEE Int. Symp.*, 2011, pp. 209–16.

[8] Ruairí de Fréin and Scott T. Rickard, "The Synchronized Short-Time-Fourier-Transform: Properties and Definitions for Multichannel Source Separation," *Sig. Proc., IEEE Trans*, vol. 59, no. 1, pp. 91–103, 2011.

[9] O. Yilmaz and S. Rickard, "Blind separation of speech mixtures via time-frequency masking," *Sig. Proc., IEEE Trans*, vol. 52, no. 7, pp. 1830–47, Jul. 2004.

[10] Ruairí de Fréin and S.T. Rickard, "Learning speech features in the presence of noise: Sparse convolutive robust non-negative matrix factorization," in *IEEE Digital Signal Processing, Int. Conf.*, 2009, pp. 1–6.

[11] S. Godard, "Sar, http://linux.die.net/man/1/s," *documentation*.

[12] B. Xu, Ruairí de Fréin, E. Robson, and M. Ó Foghlú, "Distributed Formal Concept Analysis Algorithms Based on an Iterative MapReduce Framework," in *Formal Concept Analysis*, vol. 7278 of *LNCS*, pp. 292–308. Springer, 2012.

This work was supported by an ELEVATE Irish Research Council International Career Development Fellowship co-funded by Marie Cure Actions award: ELEVATEPD/2014/62. The author is affiliated with the TSSG, WIT, Ireland.