# State Acquisition in Computer Networks

Ruairí de Fréin
School of Electrical and Electronic Engineering
Dublin Institute of Technology, Ireland

***Abstract*—We establish that State Acquisition should be performed in networks at a rate which is consistent with the rate-of-change of the element or service being observed. We demonstrate that many existing monitoring and service-level prediction tools do not acquire network state in an appropriate manner. To address this challenge: (1) we define the rate-of-change of different applications; (2) we use methods for analysis of unevenly spaced time series, specifically, time series arising from video and voice applications, to estimate the rate-of-change of these services; and finally, (3) we demonstrate how to acquire network state accurately for a number of real-world traces using Greedy Acquisition. The accuracy of State Acquisition is improved when it is performed at a rate which is consistent with its rate-of-change. An improvement in representation accuracy of one order of magnitude is achieved for voice and video streaming applications; this improvement does not incur any additional bandwidth or storage cost.**

***Index Terms*—State Acquisition; Network monitoring; Spectral analysis; Period detection.**

## I. Introduction

MANY network management tools use Linux tools for State Acquisition to acquire inputs for higher-level functions such as network monitoring algorithms [1], service level prediction algorithms [2], [3] and resource allocation routines [4]. Some examples of these acquisition tools include the System Activity Report (SAR) [5], Nagios [6] and top [7] (in association with tools such as netstat and dropwatch). Monitoring tools that use these acquisition methods promise to deliver notifications to the user if the aggregate of some metric is above a threshold across an entire infrastructure or on a per machine basis [6]. With respect to a cluster of machines, aggregates in terms of the average, maximum, minimum or sum are computed [1] as a function of time and/or across machines or instances [8]. These types of notifications are given for a number of different metrics, for example, revenue or data center temperature. Statistics are then absorbed by tools such as StatsD [9] and representative charts are generated every 10s for example (using tools such as Graphite). This paper considers the validity of current State Acquisition approaches, which acquire the data for higher-level functions such as monitoring, learning, graphing and problem diagnosis. In particular, we examine methods for acquiring and aggregating network metrics in voice and video applications [10].

To fix ideas, State Acquisition is defined as a function that measures the state of a network or service and converts this state into a numeric value; its role is different to monitoring, which is a means for making the acquired state available to the network manager. Fig. 1 demonstrates a generic acquisition-monitoring set-up, which is representative of many scenarios. An acquisition function observes an element's state every $T_a$ seconds; a monitoring agent submits a report to the network manager every $T_m$ seconds based on the acquisition agent's observations. In many applications, the acquisition and monitoring functions use the same time-step, $T_a = T_m$; the acquisition agent and monitoring agent are one and the same.

The design of network monitoring algorithms (such as [1]) does not focus on the role of metric acquisition. Linux comes with widely used metric acquisition routines. The assumption is that existing tools can be relied upon to acquire metrics; the role of the monitoring protocol is to determine when to aggregate the metrics and to deliver them to the network manager. The confidence the monitoring and learning communities have in these acquisition routines is unwarranted – these routines may not be sufficiently accurate for modern network monitoring and learning protocols [11]. The performance of a monitoring function depends on the quality of the data that it consumes. The increase in dynamicity and heterogeneity of modern networks [12], which makes networks harder to manage, exacerbates this problem.

The problem with off-the-shelf metric acquisition routines is that many of them are periodic with a default resolution of 1 second, for example SAR [5]. Consequently, many network monitoring routines monitor the network with this temporal precision limitation [13]. This was one of the reasons (scalability issues contributed also) why SoundCloud developers developed their own monitoring solution, Prometheus [8]. Similar in spirit to our approach, Prometheus stores all data that is periodically pulled from SoundCloud's [14] instrumented microservices architecture as time series, where time-stamps have a millisecond resolution and values are 64-bit floats. In order to save disk space, Nagios XI stores performance data in a Round Robin Database, which consists of performance metrics periodically averaged over $1, 5, 30$ and $360$ minute time-steps. We examine if these periodic acquisition approaches yield sufficient accuracy.

In the case of event-based monitoring and learning routines using SAR [5], there may be a lag of up to 1 second between the occurrence of an event and the time a monitoring report is sent in response to it [3]. The descriptor, "event-based", is inappropriate. The authors of [2] and [3] use SAR to acquire kernel metrics from a video server once per second in order to perform client service level prediction, dealing with load-

ISBN 978-3-903176-08-9 

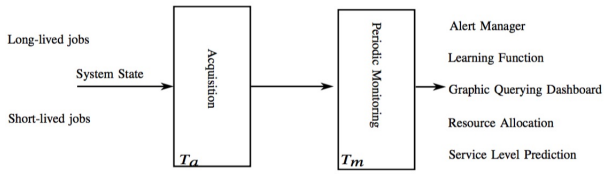


Fig. 1. The role of state acquisition in the monitoring ecosystem: metrics are pushed every $T_a$ seconds to a periodic monitoring gateway, with period $T_m$, from instrumented systems delivering either long or short lived jobs. Periodically acquired observations are stored locally, rules are applied to them to generate new time series; alerts are triggered; they contribute to monitoring or querying dashboards; or they are supplied to learning APIs. The performance of these higher-order functions depends on quality of the acquired data.

effects in particular in [15]. For a video and voice resource allocation application the authors of [4] acquired measurements of resource parameters (CPU, RAM, latency and call drops) every 15s from a Clearwater cloud ISM test-bed, using SNMP and Cacti [16] to determine how physical resources could be dynamically allocated to virtualized network functions. More generally, RTCP [17] is commonly used to provide out-of-band statistics for RTP sessions by periodically reporting on packet counts, packet loss, packet delay variation, and round-trip delay time to participants in a streaming multimedia session [18]. The recommended minimum RTCP report interval per station is 5 seconds. Nagios' Remote Plugin Executor periodically polls the agent on the remote system for disk usage and system-load statistics. Now that we have established the central role periodic acquisition and monitoring plays in networks, we examine the efficacy of periodic State Acquisition.

We query the acquisition resolution required to monitor the quality of service received by the client in a video and in a voice session. Minimizing network bandwidth usage plays a role in determining the acquisition periods of $T_a = 1, 5$ and 15 seconds used in the approaches above. Consuming bandwidth with monitoring reports is undesirable; however, reporting the system state inaccurately is perhaps even more undesirable than not reporting at all. We examine this trade-off between accuracy and bandwidth usage, but also consider whether crucial characteristics of each trace have been preserved by State Acquisition at different points of this trade-off.

This paper is organized as follows. In Section II, we provide a framework for describing state acquisition methods. In Section III, we consider periodic acquisition and the effects of performing faster acquisition empirically and motivate Greedy Acquisition. In Section IV, we discuss rate of change estimation. In Section V, we describe a Greedy Acquisition algorithm which performs acquisition in a manner which is consistent with the rate of change of the observed process. In Section VI, we perform a numerical evaluation of current acquisition methods, and the Greedy Acquisition method.

## II. State Acquisition: Unevenly spaced samples

The time series observed in networks are unevenly spaced. They consist of a sequence of observation time and value pairs $(t_n, x_n)$ with strictly increasing observation times. When audio or video frames (that are streamed from a SoundCloud or Youtube server) are observed at a client machine, the time-stamps of frame arrivals and the frame-sizes, $(t_n, x_n)$, can be useful for service outage detection. Fig. 2 illustrates a frame-arrival time-series at a client machine when a podcast is streamed. When the network is sufficiently well provisioned, this process has a periodic component. The server periodically sends the client a segment of data, which the client then plays-out to the user at the sampling rate of the original recording [19]. Important statistics such as jitter and packet loss rates may be computed from this time-series [18]; however, network anomaly diagnosis may be performed, or service-level prediction [3], if these time-series are gathered from multiple network elements at one monitoring/learning gateway [10]. This motivates the question: do the acquisition methods that currently operate at clients allow us to determine (1) the rate of change of the observed process and (2) the period of this process illustrated in Fig. 2? Knowledge of these parameters is crucial for building a learning function that can predict future network behaviour (cf. [3]).

We define a framework for analyzing unevenly spaced time series to answer this question. For $N \geq 1$, the space of strictly increasing time sequences of length $N$ is denoted: $\mathbb{T}_N = \{(t_1 < t_2 < \ldots < t_N) : t_n \in \mathbb{R}, 1 \leq n \leq N\}$. More generally, the space of strictly increasing time sequences is denoted, $\mathbb{T} = \cup_{N=1}^{\infty} \mathbb{T}_N$. The observation values, $x$, in computer networks are real-valued, $\mathbb{R}^N$. Bringing these ideas together, the space of real-valued, unevenly spaced time series of length $N$, is $\mathcal{T}_N = \mathbb{T}_N \times \mathbb{R}^N$. Finally, the space of real-valued unevenly spaced time series is $\mathcal{T} = \cup_{N=1}^{\infty} \mathcal{T}_N$. We often need to quantify the number of observations in some sequence $x$, e.g. $N = |x|$. The sequence of observation times is denoted, $T(x) = (t_1, \ldots, t_N)$, and the sequence of observation values of $x$ is $V(x) = (x_1, \ldots, x_N)$.

Time series acquisition methods (used by [5],[6] and [7]) are used to summarize the performance of network entities, so that at times $t = nT_a$, performance can be quantified. Here $n$ is an integer that denotes the acquisition time index. There are a number of different acquisition methods. The extracted metrics are passed to a monitoring protocol. Many methods do not yield data which is of sufficient quality. In this paper we address the problem of how to extract a suitable metric from unevenly spaced system events. In short, if we observe some process at times $T(x)$, we investigate what value we should use to represent this process at time $t \notin T(x)$, which is typically not an observation time.

**Observation methods:** Firstly, we take the previous observation as our estimate. Secondly, we take the next observation as our estimate; and thirdly, we interpolate between the two. For a time series $x \in \mathcal{T}$ and a point in time $t \in \mathbb{R}$, typically not an observation time, the most recent observation time is

$$p(t) =: \begin{cases} \max(s : s < t, s \in T(x)), & \text{if } t \geq \min(T(x)) \\ \min(T(x)), & \text{otherwise.} \end{cases} \quad (1)$$

The next available observation time is

$$n(t) =: \begin{cases} \min(s : s \geq t, s \in T(x), & \text{if } t \leq \max(T(x)), \\ +\infty, & \text{otherwise.} \end{cases} \quad (2)$$

For $x \in \mathcal{T}$ and $t \in \mathbb{R}$, $x(p(t))$ is the previous observation value of $x$ at time $t$, $x(n(t))$ is the next observation value of $x$ at $t$, and $x_l(t) = (1 - \omega(t))x(p(t)) + \omega(t)x(n(t))$ where

$$\omega(t) = \begin{cases} \frac{t-p(t)}{n(t)-p(t)}, & \text{if } 0 < n(t) - p(t) < \infty \\ 0, & \text{otherwise,} \end{cases} \quad (3)$$

is the linearly interpolated value of $x$ at time $t$. These acquisition schemes are called last-observation, next-observation and linear interpolation, respectively. We adopt the convention that $x(t) = x(n(t)) = x_l(t)$ when $t \in T(x)$. The interpolated signal $x_l(t)$ is a continuous piece-wise-linear function.

Local-in-time statistics: It is convenient to consider **short-time** observations of these time series in order to generate local-in-time statistics of network behaviour. The time series generated in networks over a closed interval, which starts at time $s$ and ends at time $t$, $[s, t]$, where $s < t$ is

$$x\{s, t\} = ((t_n, x_n) : s \leq t_n < t, 1 \leq n \leq N). \quad (4)$$

One acquisition approach (cf. [1]) is to apply the max operator to the values in a closed interval $[s, t]$ and to slide this interval, by some step-size, over the entire signal. The maximum signal value in the closed interval $[s, t]$ is denoted

$$a_s(t) = \max V(x\{s, t\}) \quad (5)$$

The minimum value in this interval

$$m_s(t) = \min V(x\{s, t\}) \quad (6)$$

The average value in a closed interval is commonly used to summarize system performance [5],[6] and [7].

$$\mu_s(t) = \frac{1}{|X\{s, t\}|} \sum_{n=n(s)}^{p(t)} x_n \quad (7)$$

The maximum, minimum and average statistics are reported by [20]. Other higher-order statistics are computed in a similar manner, and may be of use to network monitoring applications. The time duration between consecutive observations of $x$ is useful for determining the rate of change of the time series

$$\Delta t(x) = ((t_{n+1}, t_{n+1} - t_n) : 1 \leq n \leq N - 1). \quad (8)$$

## III. Acquire Faster: Periodic Acquisition

How representative are $x(p(t)), x_l(t)$ and $\mu_x(t)$ of the data they summarize? The acquisition methods used in [1], [2], [3] and [4], are periodic, and use one of the approaches above, at a rate of 1Hz or greater to quantify network behaviour. Given the increased dynamicity and complexity of modern networks this warrants further inspection. Is a periodic approach good enough? Is an acquisition rate of 1Hz appropriate? To determine the rate of change of unevenly spaced signals, we examine their spectral content by computing a Power Spectral Density (PSD) estimate of $x$.

The Lomb-Scargle method [21] generates a PSD estimate of unevenly spaced time series, $P : x_n \in \mathbb{R} \mapsto \hat{x}(\omega) \in \mathbb{R}_+$, without the need to invent otherwise non-existent, evenly spaced data. The underpinning assumption is that the signals are periodic. The approximate periodicity of unevenly spaced time series assumption is realistic when we consider the rate of arrival of audio or video frames during a streaming session (Fig. 2).

To reduce notation we subtract the mean from the signal $x$, and then determine the normalized spectral content of $x \in \mathcal{T}$

$$\hat{x}(\omega) = P(x)\{\omega\} = \frac{1}{2\sigma^2} \times \left\{ \frac{\left[\sum_n x_n \cos(\omega(t_n - \tau))\right]^2}{\sum_n \cos^2(\omega(t_n - \tau))} + \frac{\left[\sum_n x_n \sin(\omega(t_n - \tau))\right]^2}{\sum_n \sin^2(\omega(t_n - \tau))} \right\} \quad (9)$$

at the frequencies $\omega$, where $\sigma^2 = \frac{1}{N-1} \sum_{n=1}^{N} (x_n - \mu)^2$. The following time offset is used to guarantee the time in-variance of the computed spectrum

$$\tan(2\omega\tau) = \frac{\sum_{n=1}^{N} \sin(2\omega t_n)}{\sum_{n=1}^{N} \cos(2\omega t_n)}. \quad (10)$$

We assume that network-generated time-series are base-band or low-pass signals. Empirical evidence in Section VI supports this assumption. To fix ideas, we stream a podcast from SoundCloud [14]. The average received bit-rate of the podcast is 104, 991kbps. We capture an unevenly spaced time series which consists of the time-stamps and sizes (in bits) of each frame, $(t_n, x_n)$ respectively, in the stream using TCPDUMP [22]. We filter out the time-stamp and frame sizes and plot this unevenly spaced time series in Fig. 2. This trace has a periodic component – every approximately 10 seconds, a number of 12kbits frames are delivered.

In Fig. 3 we plot the PSD estimate of this unevenly spaced time series. The component with the highest PSD is $\approx$ .7Hz. By inspection of Fig. 2, we determine that the fundamental frequency of this trace is < 1Hz. The upper envelope of the PSD falls to 0dB/Hz at $\approx$ 100Hz. There are other higher-frequency components in Fig. 2. Consequently, in Fig. 3 we illustrate where the PSD of the trace has fallen by 50dB from its peak value – this occurs at $\approx$ 108 Hz – to demonstrate a range of frequencies of interest to monitoring applications.

Do current time series acquisition approaches preserve this important information about the rate of change and the period of frame delivery? To answer this question, we acquire state values every $T_a$ = 1s, at times $t \in \mathbb{N}$ and examine the PSD of the resulting traces. The time period $T_a$ = 1s is representative of the period used in [1], [2], [3] and [4]. The set of non-negative integers is $\mathbb{N}$. The underlying process we want to acquire an accurate representation for, is acquired by taking (1) the last sample closest to some sampling time, $x(p(t)) \mid t \in \mathbb{N}$; (2) the average value over the last sampling period, $\mu_{t-1}(t) \mid t \in \mathbb{N} \setminus 0$, where $t$ is in the set of positive integers; and finally, (3) the linear interpolated value $x_l(t) \mid t \in \mathbb{N}$. We use a stem to denote the position and height

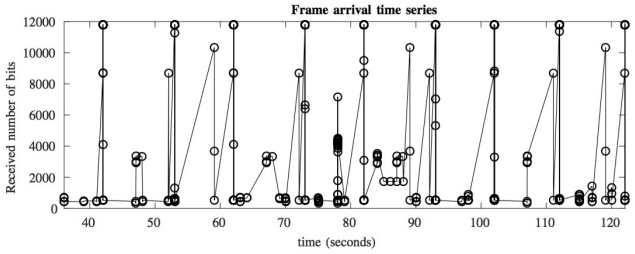


Fig. 2. Unevenly spaced time series observed when streaming audio to a client. The time and value pairs illustrated consist of frame arrival time-stamps and frame sizes. Every $\approx$ 10s a number of frames of size $\approx$ 12kbits are received. This time-series has a clear periodic component.

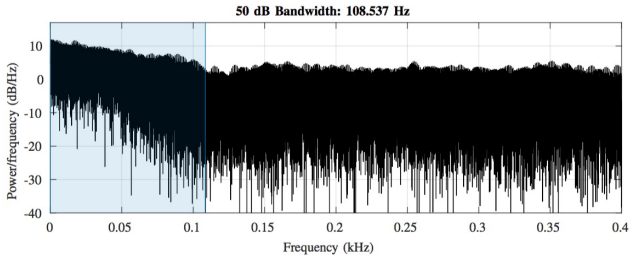


Fig. 3. PSD of unevenly spaced time-series observed when receiving a streamed podcast from SoundCloud. The 50dB bandwidth is 108Hz. The upper envelope of the PSD falls to 0dB at 100Hz.

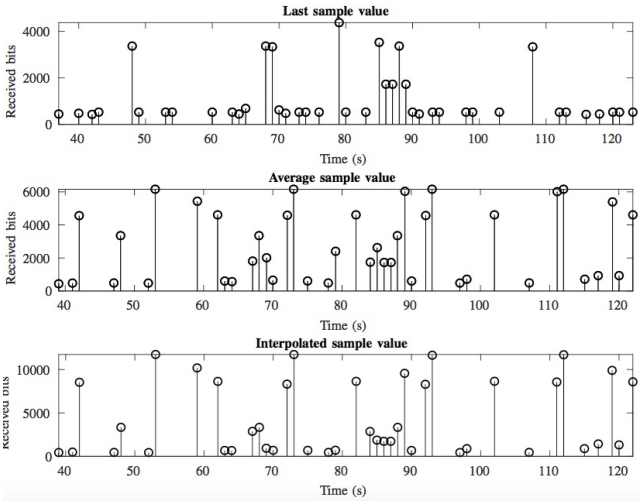


Fig. 4. Using periodic acquisition (time-step of 1 second), we illustrate the effect of using the last sample, average value and interpolated values at acquisition times $t \in \mathbb{N}$.

of each state acquisition value in Fig. 4; each trace should be compared with the original time series in Fig. 2.

**Analysis:** The PSD allows us to estimate the rate of change of the network/service – a parameter which is of crucial interest to service providers. For example, unusually high rates of change may indicate anomalies; periodic components may indicate that the network is healthy. The acquisition methods in Fig. 4 do not allow the network manager to estimate the

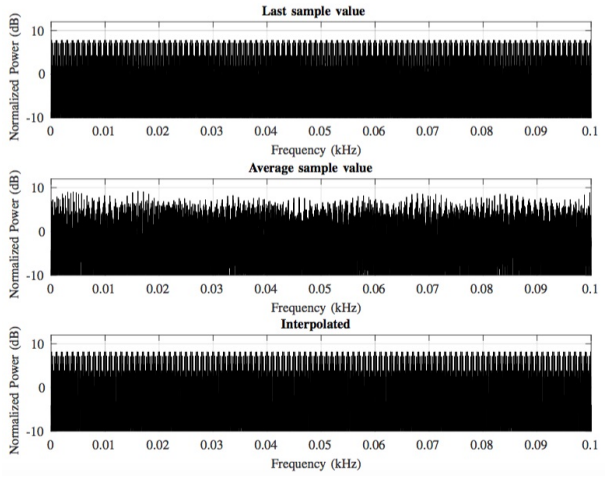


Fig. 5. PSD of current acquisition approaches: $x(p(t))$, $\mu_{t-1}(t)$ and $x_l(t)$. Important information such as the rate of change of the underlying time series and the period is lost. The flat PSDs illustrated demonstrate that higher frequency information has been lost.

maximum rate of change of the network. The information in the frequency band $0 < f < 108$Hz is lost by the acquisition methods used in Fig. 4. One reason for this is that they create an estimate for data that does not exist.

**(1) Information about the period of the trace is lost.** This is a serious shortcoming. For example an artifact of $x(p(t))$ is that its period is approximately 20 seconds and not approximately 10 seconds. Period calculations for each trace will be inaccurate as the times of frame arrivals in the original trace are never in the set of times $t \in \mathbb{N}$. For example the 12kb frame arrival times occur just after $40, 50, 60, \ldots$ seconds. The large frame arrivals occur just before $50, 70, \ldots$ seconds in $x(p(t))$. In the average acquisition trace $\mu_{t-1}(t)$, the pulse amplitudes (frame sizes) vary in height, which makes period detection hard. Finally the interpolated acquisition trace $x_l(t)$ exhibits pulse amplitudes which vary each time they appear with a period of 20 seconds. Estimates of the period tell us when to expect the next burst of podcast data. The ability to detect if this burst of data has arrived, or has arrived late, is lost. This is because the exact times when bursts of data arrive have been lost. Secondly, the sizes of the burst have been lost; this is due to averaging or interpolation, or because the previous observation was not part of the periodic train of frames. The effect of using the last, average or interpolated value over the previous second causes the period information to be obscured. Averaging unevenly spaced time series introduces ambiguity.

**(2) Higher frequency information in the range up to 108Hz and above is lost in Fig 5.** The acquisition techniques, $x(p(t))$, $\mu_{t-1}(t)$ and $x_l(t)$ low-pass filter the original unevenly spaced time-series. We posit that high-frequency variations in this trace may help us detect sub-optimal network performance. Choosing a low-pass filter in the acquisition step, without reference to the rate of change of the time series, may

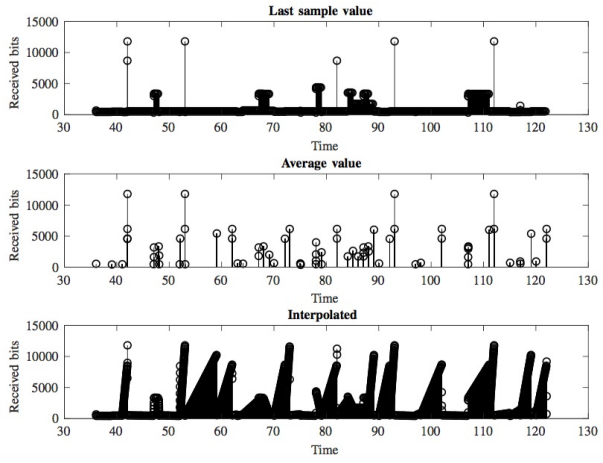


Fig. 6. High-rate periodic acquisition: Acquisition at 200 Hz does not significantly improve the ability to estimate the period and rate of change of the original data from the acquired time-series.

remove crucial monitoring and problem diagnosis information. **Remark:** the amplitude and location in time of the acquired stems (in Fig. 4) depend on the relative position of each of the events in $x$ relative to the acquisition times $t \in \mathbb{N}$. Shifting the acquisition times relative to the events in $x$, can greatly increase or decrease the efficacy of acquisition and monitoring. We have shown that for an arbitrary periodic acquisition starting time that periodic acquisition is harmful for higher-order functions such as monitoring and learning.

**Higher acquisition rate:** The periodic acquisition methods above, with a period of 1Hz, remove crucially important information about the fundamental frequency and the rate of change of frame delivery in Fig 2. Loss of this information will reduce the effectiveness of higher-order functions such as monitoring and learning that consume the acquired performance data. The PSD of the original unevenly spaced time-series experiences a drop of 50dB at approximately 100Hz in Fig. 3. We consider the effect of acquiring this time series using a significantly higher acquisition rate of 200Hz to determine if this facilitates the capture of crucial parameters such as the period and rate of change of the underlying trace. In effect we are assuming that the Nyquist rate of this unevenly spaced time series is $\approx$ 200Hz.

Increasing the rate of State Acquisition by a factor of 200 increases the bandwidth of the monitoring protocol that consumes these metrics. Fig. 6 demonstrates that increasing the acquisition rate 200-fold does not improve our ability to estimate the period and rate of change of the time series. The acquisition methods $x(p(t)), \mu_s(t)$ and $x_l(t)$ suffer to similar problems as before; but they now consume more bandwidth and storage.

## IV. Rate of Change Estimation

In this section we determine the appropriate way to acquire periodic performance traces from audio and video streams by estimating the rate of change, $c$, of the trace. Even if the resulting rate of acquisition is unfeasibly high, knowledge of this parameter can facilitate better design decisions. For example, we can low-pass filter the signal and use this filtered time series in learning algorithms for example, cognizant of the fact that we cannot make predictions outside of a certain range of frequencies. We appeal to classical evenly spaced sampling theory for guidance with the challenge of rate of change estimation for unevenly spaced time series.

**The problem with unevenly spaced time series:** The periodic time-series we observe for video and audio stream applications, $x$, consist of a sum of weighted and delayed Dirac pulses, $\delta(t)$, e.g.

$$\sum_{n=1}^{N} x(t_n)\delta(t - t_n). \tag{11}$$

When these pulses form an infinitely long sequence of evenly spaced pulses, spaced $T$ seconds apart, and the pulses have identical heights, this is called a Dirac comb [23]. It is a known result in Signal Processing that the Fourier transform of a Dirac comb with period $T$s produces another Dirac comb in the frequency domain with period $\frac{1}{T}$Hz. As these Dirac pulses are spaced by $\frac{1}{T}$Hz in the frequency domain, to avoid aliasing, we must ensure that the spectral content of the signal we wish to acquire is confined to a region of $c = \frac{1}{2T}$Hz, which is the maximum permissible rate-of-change of this evenly spaced time series.

Unlike the uniform case, the Fourier transform of the unevenly spaced time series in Eqn. 11 will generally not be a Dirac Comb, that is, a sequence of uniformly spaced delta functions; the symmetry in the Dirac comb is broken by uneven spacing of the pulses, which leads to information rich transform we observe in Fig. 3. This is because in the Fourier domain, the locations and heights of the Dirac delta functions are related to the intervals between the time domain observations. Randomizing the observation times, randomizes the locations and heights of the Fourier domain peaks and heights.

To leverage the results of classical sampling [23] to estimate $c$ for an unevenly spaced time series, we express a unevenly spaced time series as a evenly spaced time series. To this end, we determine the Dirac comb which has Dirac deltas that align exactly with each of the pulses in the unevenly spaced time series. In other words, we need to determine the largest $T_a$ such that each $t_n \in T(x)$ can be written as $t_n = \alpha + nT_a$, for integers $n$ and an arbitrary time offset $\alpha$.

**Definition (1) Accurate acquisition:** To accurately acquire an unevenly spaced time series the acquisition period $T_a$ should satisfy the condition

$$\Delta t(x) \mod T_a = 0, \forall t_n \in \mathbb{T}_N \tag{12}$$

**Definition (2) Rate-of-Change:** The rate of change of the unevenly spaced time series $x$ is $c = \frac{1}{2T_a}$ if $T_a$ satisfies the condition

$$\Delta t(x) \mod T_a = 0, \forall t_n \in \mathbb{T}_N \tag{13}$$

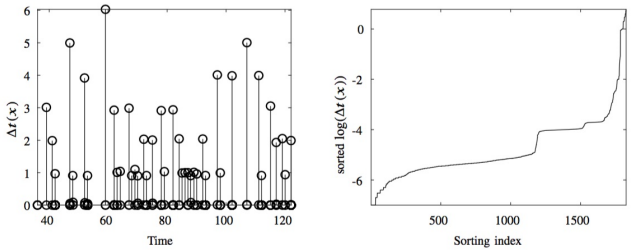


Fig. 7. The time durations between consecutive frame arrivals $\Delta t(x)$ are plotted (stem height) on the LHS versus experiment time (x-axis). These durations are sorted from smallest to largest on the RHS. Very few of the consecutive arrival times exhibit a large delay.

Finding the value of $T_a$ that ensures that acquisition is being performed at a sufficiently high rate, leads to acquisitions rates which are impractical for real-world applications. If observation times are recorded to $d$ decimal places, the largest rate of change is determined by machine precision.

To make progress, we use a rough upper bound on $T_a$, which consists of examining the minimum separation between consecutive frame arrivals:

$$\min \Delta t(x). \tag{14}$$

As the traces we will consider are generally base-band signals, we desire a value of $T_a$ which captures most of the spectral energy of the trace $x$.

**Analysis:** The smallest difference between two time-stamps, $\min \Delta t(x)$ in Fig. 2 is $10^{-7}$ seconds. This would necessitate acquiring the signal at an unfeasibly high rate, $> 10^7$ Hz, which is impractical most networking applications. We propose a Greedy Acquisition solution to solve this problem and examine a trade-off between the accuracy of the acquired time-series, the bandwidth of different versions of the acquired time-series.

## V. Greedy Acquisition

A Greedy Acquisition algorithm for unevenly spaced time-series is presented that acquires observations at a rate which is consistent with the rate of change of the original time series, $c$.

Fig. 7 demonstrates the number of time intervals between consecutive values in $x$ (the audio trace) that are greater than 1 second (on the RHS). In summary, only 1.5% of these time intervals are greater than 1s; the majority of the time spacings between events are much smaller. A useful rule of thumb for periodic acquisition is that if we increase the time between acquisitions in the acquisition routine, we decrease the range of frequencies that we can observe in the trace [23]. It is challenging to perform accurate acquisition using a periodic acquisition scheme because the time-step must be sufficiently small to capture a range of rates of change. Using a fixed, periodic acquisition rate which is of the order of 1Hz (in [5] and [7]) or even one kilohertz (in [6]) will not yield an accurate representation of the performance of $x$. For example, for the audio streaming trace in Fig. 2, an acquisition rate of 1Hz is significantly too low an acquisition rate for 98.5% of the frame arrivals in this time-series. Averaging these values over a one-second interval, removes all frequency components above $\approx$ 1Hz, which make subsequent functions such as problem diagnosis difficult. Similarly in the case of fixed periodic acquisition rate of 1kHz (which is the time-step used by Prometheus [8]) this rate is too low for 94.25% of the frame arrivals in Fig. 2.

From inspection of Fig. 7 it is clear that locally in time, the value of $\min \Delta t(x)$ may be much higher than it would for the entire time series. This means that an acquisition performed at a rate that is consistent with the local rate of change of the time series, could be expected to significantly reduce the bandwidth required to accurately represent the underlying event steam $x$, over a periodic acquisition scheme with the same level of fidelity. We investigate an adaptive acquisition protocol, where the value of $T_a$ changes with time, and posit that it should give a more accurate time series representation, using less bandwidth than before.

A greedy approximation for the time series $x$ is generated by summing a finite number of functions $g_i$ taken from a dictionary of such functions $D$. For example, the function $g_1$ consists of a sum of Diracs which are weighted by the largest value of the time series $x$, e.g. $m_1 = \max(V(x))$, which gives the function

$$g_1 = m_1 \sum_k \delta(t - t_1(k)) \tag{15}$$

where $t_1$ is the set of times where $x = m_1$ and there are $k$ elements in this set.

The next function in the dictionary $D$ is $g_2$. It consists of a sum of Diracs which are weighted by the second largest value of the time series $x$, $m_2 = \max(V(x) \setminus m_1)$. In words, we remove all of the instances of the maximum value of $x$ from $x$ and then find the next largest value, $m_2$. We then construct the function $g_2$ which has Dirac pulses of height $m_2$ every time $x = m_2$,

$$g_2 = m_2 \sum_k \delta(t - t_2(k)). \tag{16}$$

Similar to the previous case, $t_2$ is the set of times where $x = m_2$; there are $k$ elements in this set. Continuing this process, we construct the entire dictionary $D = \{g_i\}$.

A $j$-th order approximation of $x$, which is denoted $y_j$, is obtained by summing up the first $j$ elements of the dictionary

$$y_j := \sum_{i=1}^{j} g_i. \tag{17}$$

The accuracy of the $j$th order approximation is computed using the Frobenius norm,

$$\epsilon_j = \sqrt{\int (x - y_j)^2 dt}, \tag{18}$$

where $y_j(t_n) = 0$ if we have not acquired a value at time $t = t_n$ for the $j$th approximation.

The set of values $m_1, m_2, \ldots$ are the values of the frame sizes sorted from largest to smallest. Because these frames are generated as part of well-defined protocols, which have

**Data:** input: real-time process for acquisition from $x$ and $\beta$.
**Result:** acquired time series $(t_n, y_n)$.
initialization: sorted list of frame sizes $m_i$ from historical data;
**while** *still receiving stream* **do**
  read current value;
  **if** $x > \beta$ **then**
    acquire time $t_n = t$;
    acquire the value $y_n = x_n$;
  **else**
    do not acquire value or time;
  **end**
**end**

**Algorithm 1:** Online Greedy Acquisition Algorithm

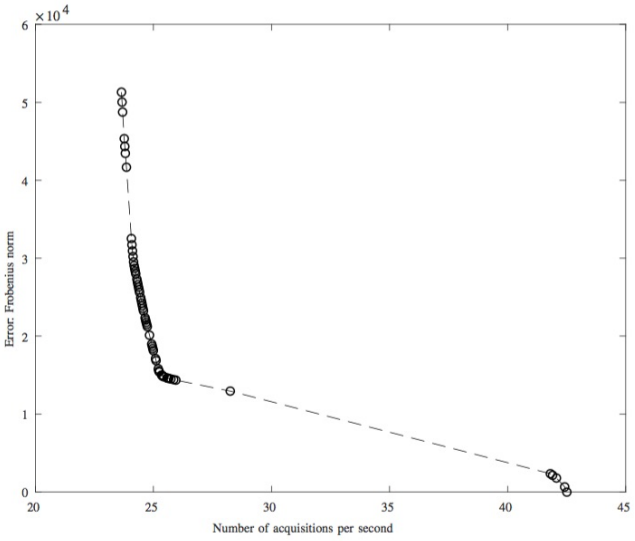


Fig. 8. Frame arrival process statistic acquisition for the SoundCloud trace using Greedy Acquisition: The error in the acquisition time-series decreases as the number of acquisitions per second increases. We examine the first through to the sixtieth order approximation of $x$. The error achieved when 25 measurements are acquired on average per second is impressively low, given that we are computing the Frobenius norm of frame sizes of up to 12kbits.

predetermined frame sizes for an array of scenarios, the number of different values of $m_i$ that we can expect is finite, and generally, quite small. It is possible to determine the set $\{m_i\}_i$ a-priori for any given service.

An online real-time version of the Greedy Acquisition algorithm presented above, is presented in Alg. 1. It consists of determining whether or not the current frame arrival has a frame size which is larger than a user defined threshold, $\beta$. The value of $\beta$ determines the accuracy of the achieved approximation. If an incoming frame size is larger than $\beta$, the frame size and time are recorded as part of the unevenly spaced acquired time series $(t_n, y_n)$. Greedy approximation has the following properties. (1) the accuracy of the approximation increases monotonically as $\beta$ decreases, because each increase in the order of the approximation $j$, reduces the error in

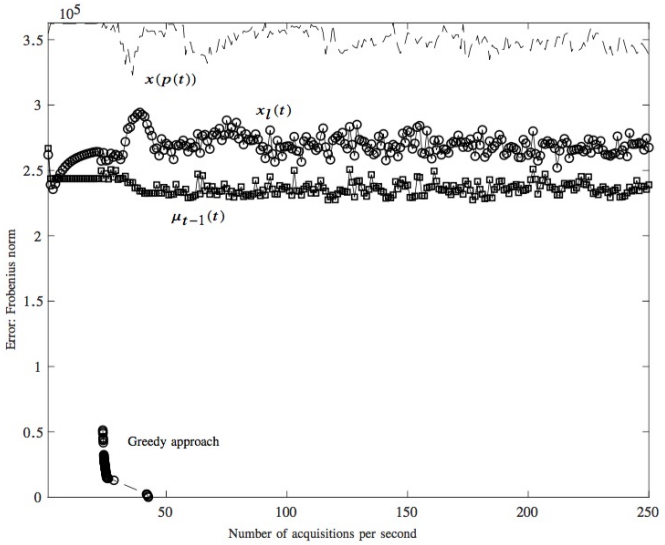


Fig. 9. Comparison of the acquisition accuracy bandwidth trade-off achieved using tradition periodic acquisition methods (last observation, average and interpolation) and using Greedy Acquisition. The greedy approach is significantly better.

the approximation of $x$. (2) The bandwidth requirement of greedy approximation increases as $j$ increases, but it increases efficiently, in line with the rate-of-change of $x$. (3) Because the original frame arrival times are used, the periodic component of the time series is generally captured by $(t_n, y_n)$; moreover, the rate-of-change of the time-series $(t_n, y_n)$ is generally consistent with that of $(t_n, x_n)$, in comparison with a periodic acquisition algorithm, which chooses the acquisition period in an arbitrary way, and is thus, not always suited to the process being observed.

## VI. Numerical Evaluation

We examine the accuracy of different acquisition methods using a simple video and audio use-case. We focus on audio and video because according to Cisco, by 2021, 82% of all consumer Internet traffic will be IP video traffic [24]. The ability to acquire measurements from the traces clients receive will be crucial. Higher quality acquisition data will improve next-generation monitoring protocols.

Our hypothesis is that Greedy Acquisition yields more accurate estimates of the observed process than traditional periodic monitoring approaches. Moreover, we posit that Greedy Acquisition uses less bandwidth and storage, and that the acquired time-series preserve important statistical features of the observed trace, such as its period. In our comparison we implement acquisition functions which estimate $x(p(t))$, $\mu_s(t)$ and $x_I(t)$. The reason that we do not use off-the-shelf tools such as SAR [5] is that they have a minimum acquisition rate of 1Hz (in the case of SAR), a rate which we would like to significantly increase, in order to fully evaluate the potential of periodic acquisition. We consider a frame arrival process because TCPDUMP [22] provides easy access to high

resolution time-stamps for a widely available unevenly spaced time series, which is representative of what we might observe at many other intermediate points in the network.

**Experimental set-up:** In the first scenario a client streams a podcast from an online audio distribution platform, a SoundCloud instance [14]. In 2014 SoundCloud boasted 75 million unique monthly listeners, which demonstrates the popularity of the service, and motivates the need to be able to accurately acquire measurements from the audio traces. In the second scenario a client streams a video from a video server of an Irish public service broadcaster [25].

During our evaluation of both scenarios, the client requested the service and then TCPDUMP [22] was used to capture a description of the contents of the frames received on the client's network interface. Once the media session had terminated, we converted the captured .pcap file to text format using *tshark*, and we *grep*ed the resulting text file for the frame arrival times, $t_n$, and frame sizes $x_n$, forming the unevenly spaced time series $(t_n, x_n)$. Frame arrival times were captured in terms of hours, minutes, seconds, and fractions of a second since midnight. We recorded the size of received frames using the "bytes on wire" field, in bits. We stored frame arrival times using double-precision according to IEEE Standard 754 for double precision, which bounded the precision of rate-of-change estimates. The frame sizes were integer-valued. For the SoundCloud session, the maximum frame received was 11792 bits, and 60 different frame sizes were observed during this session, which gave us 60 different potential acquisition accuracies. In all cases we streamed data for $\approx 15$ minutes.

**Discussion:** Fig. 8 illustrates the error (Eqn. 18) versus bandwidth trade-off achieved for a SoundCloud session by varying $\beta$ in the online Greedy Acquisition algorithm. As the value of $\beta$ is decreased, the accuracy of the greedy representation improves. This accuracy comes at the cost of 25 acquisitions per second on average.

This result is significant, particularly when it is compared with the periodic, last-observation, averaging and interpolation schemes currently used ($x(p(t)), \mu_s(t)$ and $x_l(t)$). Fig. 9 illustrates the accuracy of $x(p(t))$, $\mu_s(t)$ and $x_l(t)$ as a function of the bandwidth consumed by these acquisition approaches. The bandwidth is increased by increasing the resolution of the acquisition time-step. For the periodic acquisition methods, the trend is for the error to decrease as the rate of acquisition increases. We increase the rate of acquisition from 1Hz up to 250Hz in steps of 1Hz.

In this trace the times of the arrivals of the largest frame sizes tend to determine the periodic component of the time series. Preserving the exact times of events is important. The ability to so, is determined by the rate of acquisition. The Greedy Acquisition algorithm acquires a trace at a rate which is consistent with the rate of change of time series. For the 1st approximation $y_1$, the minimum time-step between consecutive frame arrivals $\min \Delta t(y_1)$ does not change as we increase the order of the approximation. The 1st approximation $y_1$, and all subsequent approximations have the possibility of capturing the fastest changes in the observed trace, depending on their importance (frame size).

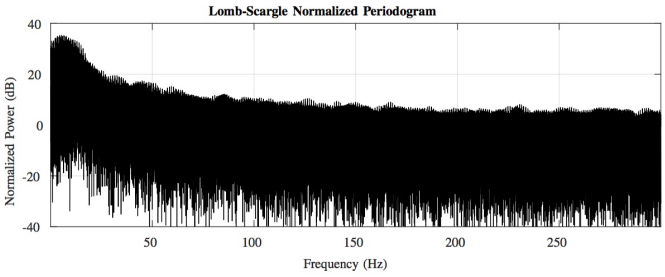


Fig. 10. PSD of frame arrival time series for video streaming session. This trace has a periodic component at 5.4Hz, which implies that acquiring this trace at 1Hz is insufficient.

The traditional periodic approaches have time steps in the range $1/p$s where $p = 1 \ldots 250$Hz in these experiments. These periodic approaches cannot observe the fastest changes in the observed process, or at the maximum rate of change of the process. In conclusion, periodic acquisition approaches yield an error which is 5 times worse than the error of Greedy Acquisition, using 5 times the bandwidth to do so. Finally, important statistics of the observed traces are typically not preserved by periodic approaches.

In the case of the video streaming session, we illustrate the PSD of the time-series in order to provide initial estimates for the period and the rate-of-change of $x$. This trace has a periodic component at 5.4Hz, which implies that acquiring this trace using a periodic acquisition method at 1Hz is insufficiently accurate. Note that even the recommended reporting rate of RTCP (5Hz) is not sufficiently large to capture the period of this trace. In addition, the PSD of this trace exhibits significant power up to 100Hz. Similar to the audio streaming scenario, increasing the acquisition rate of periodic acquisition algorithms does not significantly improve the accuracy of the acquired time series.

With regard to the rate of change of this trace, 0.16% of consecutive frame arrivals have an inter-arrival time of greater than 1s, and 1.3% have an inter-arrival time of greater than $10^{-3}$s. The minimum inter-arrival time is less than $10^{-7}$s. Given the definition of the rate of change of unevenly spaced times series, periodic acquisition at 1Hz is insufficient.

These statistics underline the difficulty of choosing a time-step for periodic acquisition that would yield sufficiently high accuracy. We evaluate Greedy Acquisition, to see if acquiring measurements at a rate which is consistent with the rate of change of the trace, is accurate. Fig. 11 demonstrates that accurate acquisition is achieved by taking 300 acquisitions per second using Greedy Acquisition. The average acquisition rate for video that achieves the same accuracy as audio acquisition is approximately $\times 10$ the acquisition rate for audio. Once again Greedy Acquisition out-performs each of the periodic acquisition approaches – by an order of magnitude drop in the error of the representation – when 300 acquisitions are made per second.

**Recommendations:** Many current periodic acquisition ap-

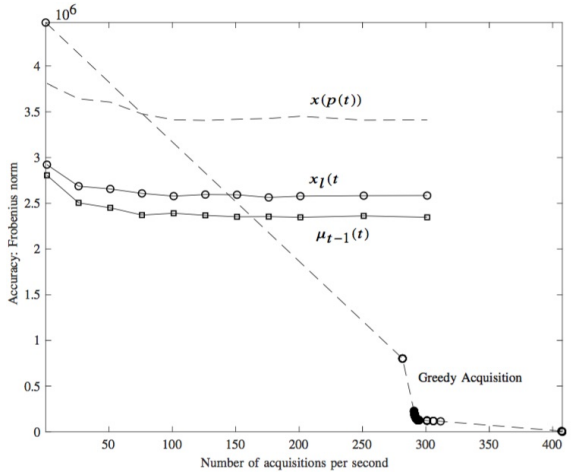


Fig. 11. Frame arrival process statistic acquisition for video trace using Greedy Acquisition: The error in the acquisition time series decreases as the number of acquisitions per second increases. The error achieved when 300 measurements are acquired on average per second is impressively low. The periodic acquisition methods $x(p(t))$, $\mu_{-1}(t)$ and $x_l(t)$ are illustrated for completeness.

proaches acquire time-series without knowledge of the underlying rate of change of the time series under observation. We have argued that knowledge of the rate of change of the process under observation, should drive the process of deciding when to acquire measurements of this process. Many networking time-series exhibit properties such as periodicity. These properties should be preserved by acquisition routines. One point of note from this work is that the times of events $T(x)$ are as important as the values recorded for these events $V(x)$. Periodic acquisition routines such as Nagios, record values with greater precision than times, due the default time resolution of $10^{-3}$s. A second point of note is that the wide array of off-the-shelf acquisition routines means that little research is being done in the area. The received wisdom is that metric acquisition routines exist, and thus, there is little point in re-inventing them. Finally, we have provided evidence that Greedy Acquisition gives improved acquisition performance.

## VII. Conclusions

In this paper we showed that acquiring a system or service's state at a rate which was consistent with the rate of change of the system (or service) provided a high-quality record of the state of the system. Research on capturing system state has lagged behind the growth of networks and the applications that use these networks as a substrate. Today, many monitoring and learning solutions rely on standard periodic State Acquisition solutions, which acquire the system state at a frequency of 1Hz. These solutions do not capture important characteristics of the signals they acquire, for example, periodicity and rate of change. For periodic traces, the ability to estimate the period from an acquired representation of the trace is fundamental. We demonstrated that the rate of change of many applications is much greater than 1 Hz, and then, we demonstrated that present-day acquisition techniques do not capture information which is crucially important for problem diagnosis. Our experiments with real-world voice and video traces demonstrate that high quality State Acquisition is possible if the time-stamps and magnitudes of events are recorded at the rate of change of the application.

## References


[1] D. Jurca and R. Stadler, "H-GAP: estimating histograms of local variables with accuracy objectives for distributed real-time monitoring," *IEEE Trans. Net. and Serv. Man.*, vol. 7, no. 2, pp. 83–95, Jun. 2010.

[2] R. Yanggratoke, J. Ahmed, J. Ardelius, C. Flinta, A. Johnsson, D. Gillblad, and R. Stadler, "Predicting real-time service-level metrics from device statistics," *IFIP/IEEE Int. Sym. Int. Net. Man.*, pp. 1–8, 2015.

[3] R. de Fréin, "Source separation approach to video quality prediction in computer networks," *IEEE Comm Ltr*, vol. 20, no. 7, pp. 1333–1336, Jul. 2016.

[4] R. Mijumbi, S. Hasija, S. Davy, A. Davy, B. Jennings, and R. Boutaba, "Topology-aware prediction of virtual network function resource requirements," *IEEE Trans Net. Serv. Man.*, vol. 14, no. 1, pp. 106–120, 2017.

[5] S. Godard, "SAR," http://linux.die.net/man/1/sar.

[6] E. Galstad. (1999) Nagios. [Online]. Available: https://www.nagios.com/

[7] R. Binns. top. [Online]. Available: https://linux.die.net/man/1/top

[8] J. Volz and B. Rabenstein. (2018) Sound Cloud Developers: Backstage Blog. [Online]. Available: https://developers.soundcloud.com/blog/prometheus-monitoring-at-soundcloud

[9] ETSY. StatsD. [Online]. Available: https://github.com/etsy/statsd

[10] R. 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," *J. Net. Sys. Man.*, vol. 23, no. 3, pp. 673–708, Jul 2015.

[11] R. de Fréin, "The data-centre whisperer: Relative attribute usage estimation for cloud servers," in *24th EUSIPCO*, Aug. 2016, pp. 687–691.

[12] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: state-of-the-art and research challenges," *J. of Int. Serv. and Apps*, vol. 1, no. 1, pp. 7–18, May 2010.

[13] D. Josephsen, *Building a Monitoring Infrastructure with Nagios*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2007.

[14] A. Ljung and E. Wahlforss. (2008) SoundCloud. [Online]. Available: https://soundcloud.com/

[15] R. de Fréin, "Effect of system load on video service metrics," in *IEEE Irish Sig. Sys. Conf. (ISSC)*, 2015, pp. 1–6.

[16] "The Cacti Group Inc." [Noline], 2016, accessed on Dec. 28, 2016. [Online]. Available: http://www.cacti.net/

[17] V. Jacobson, R. Frederick, S. Casner, and H. Schulzrinne, "RTP: A transport protocol for real-time applications," IETF RFC3550, 2003. [Online]. Available: https://tools.ietf.org/html/rfc3550

[18] A. Begen, T. Akgul, and M. Baugher, "Watching video over the web: Part 1: Streaming protocols," *IEEE Int. Comp.*, vol. 15, no. 2, pp. 54–63, Mar. 2011.

[19] ——, "Watching video over the web: Part 2: Applications, standardization, and open issues," *IEEE Int. Comp.*, vol. 15, no. 3, pp. 59–63, 2011.

[20] "Datadog," Online documentation, accessed on Dec. 28, 2016. [Online]. Available: https://www.datadoghq.com/

[21] N. R. Lomb, "Least-squares frequency analysis of unequally spaced data," *Astrophy. and Space Sc.*, vol. 39, no. 2, pp. 447–462, Feb. 1976.

[22] V. Jacobson, C. Leres, and S. McCanne. tcpdump. [Online]. Available: http://www.tcpdump.org/

[23] M. Vetterli, J. Kovacevic, and V. K. Goyal, *Foundations of Signal Processing*. Cambridge: Cambridge Univ. Press, 2014.

[24] "Cisco Visual Networking Index: Forecast and Methodology, 2016-2021." [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.html

[25] (1996) TG4: TG Ceathair. [Online]. Available: http://www.tg4.ie/en/player/home/



This publication has emanated from research conducted with the financial support of Science Foundation Ireland (SFI) under the Grant Number 15/SIRG/3459.