A Study on Web-Log Using Wavelet

Antonia Zhai, Pu Huang, Tim Jia-Yu Pan
Dept. of Computer Science
Carnegie Mellon University
{zhaia, phuang, jypan}@cs.cmu.edu

May 3, 2001

Abstract

Understanding the characteristics of web server workload is crucial for web server management and future capacity planning. In this project we propose to use wavelet-based techniques to analyze the workload collected from busy web servers. We will focus on finding the temporal characteristics of the web server web-log, which contains workload information, and predicting the trend it evolves. Besides, to evaluate the characteristics we get from the wavelet analysis, we will build synthesizers based on the extracted characteristics, and compare the statistics properties of the synthetic workload and its behavior on a simulator with the corresponding counterparts of real-world workload. The closer the behavior our synthetic workload has to abstract that of real-world workload, the better the characteristics we extracted from the wavelet analysis. The techniques we develop here can be also used in other workload or time series analyzing tasks, such as analyzing disk I/O workload or stock quote sequences.

1 Introduction

Today, people commonly get information from WWW, both for work and for fun, and require better services from the web servers while impose them with much more workload. To provide good services to web surfers, either just to save their time or to retain their patronization, it is important for the website managers and the system designers to understand the current characteristics and the evolving trend of the workload on web servers, so that they can make good use of their current resources and also plan in advance for the future.

Workload of a web server may be affected by many factors, ranging from issues on networking and server side configuration, to the client side request sending behavior. Networking issues include the protocol being used, the deployment of web caches (or proxies), and the nature of the network traffic (e.g. burstiness) could definitely influence web servers' workload, since requests are sent through the network before reaching them. Workload of a web server also depends on the content the web server has. Contents consisting of different ratios of image files versus text files, or having different proportions on static and dynamic materials, will have different kinds of workload. User behavior will of course causes impact on the web servers' workload. Possible influences would come from the number of requests sent per session, or the current time and day of the week when the workload is collected.

In this project, we will focus on analyzing the temporal characteristics of the web server workload (web-log), in other words, how does the workload change as time goes by. The main analyzing tool will be the wavelet transformation, due to its convenience to detect scalable features of multidimensional data.

The data set we are using was collected from the official web site of the 16th FIFA (Federation Internationale de Football Association) World Cup during June 10th through July 12th, 1998, when the tournament was proceeding. We choose to use this data set because the World Cup website is one of the most popular, and busiest, websites at that time, and it is the characteristics of the workload of a really heavy-loaded web server that we really care about.

Based on the characteristics we observed from the wavelet analysis, we would try to do workload prediction, more specifically, predicting future workload of a busy web server. Having this kind of knowledge about the workload enables designers and administrators of a busy web site to do better web server management and future capacity planning to provide high performance services.

To evaluate the effectiveness of the characteristics we get from the wavelet analysis, we plan to build workload synthesizer to generate artificial workload, and observe the differences between a real workload and a synthetic one when they are run on some simulators. Statistical measurements on the real workload and the synthetic workload will be calculated as well, however, it has been suggested [1] that validating extracted workload characteristics by simulating the synthetic workload on domain specific measurements, such as cache hit ratio, requests inter-arrival time, and queuing behavior, is more appropriate and reasonable.

This project proposal is organized as follows: Section 2 contains surveys on several areas related to our project, including wavelet transformation (2.1), sampling theorem (2.2), self-similarity (2.3), and workload prediction (2.4) and synthesization (2.5). We describe the theory of wavelet analysis of time series and our observations obtained by applying wavelet analysis on the web-log data in section 3. Section 4 explains the proposed methods on doing realistic web-log synthesis and prediction. Current experiment results are given in section 5, followed by an explanation of a simulation model that will be used to evaluate the correctness of our analysis and synthesized web-log.

2 Survey

2.1 Wavelet Transformation

There are a variety of ways explaining wavelet transform in literature, such as, Short Window Fourier Transform [2], filter bank [3] etc. Here I use a slightly different way – starting with the Shannon Sampling Theorem, then following by Short Time Window Fourier Transform. If we allow the window size to change, we can readily explain the scaling feature of wavelet transform. However, the whole story can not be fully explained for the general theory of wavelet transform but we make no further attempt to bridge the gap here. Such a treatment does not hinder our purpose of this project.

2.2 Sampling Theorem

Continuous waveform, such as, $f(t)$, where t is time variable and $f(t)$ is a real value function, can not be processed by computer directly. Some form of transformation from continuous time domain to discrete domain is necessary – Shannon Sampling Theorem provides such a tool.

Theorem 1 *For a band-limited waveform $f(t)$, we can completely recover $f(t)$ from the discrete sequence $f(nT_{s)}$ for $n = \left\{ \begin{array}{ccccc} \cdots & -1 & 0 & 1 & \cdots \end{array} \right\}$ if $T_s \leq \frac{1}{2B}$ where B is the bandwidth of $f(t)$.*

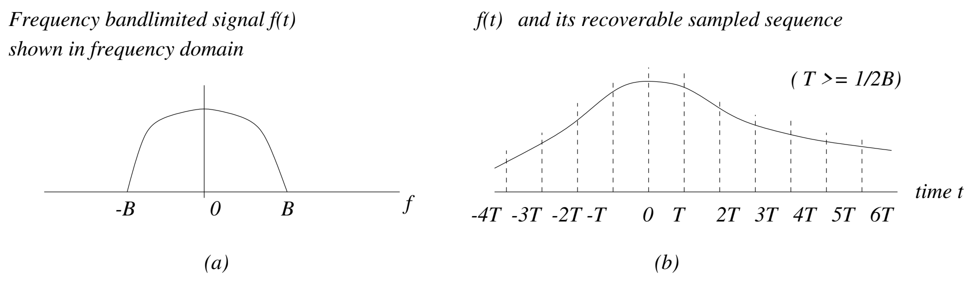


Figure 1: Band limited Signal and Sampling Theorem

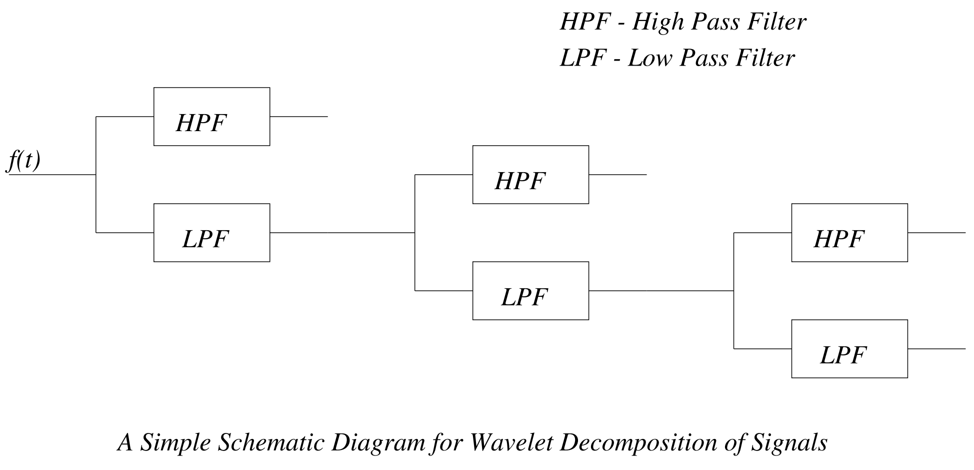


Figure 2: A Simple Wavelet Diagram

By Shannon Sampling Theorem, we conclude that to recover a continuous waveform we have to sample it at least twice as fast as the highest frequency of the original waveform. Otherwise we introduce distortion (aliasing). Furthermore, the narrower the waveform bandwidth the fewer the number of samples we need. If we break down signal bandwidth by half, then we can sample the resulting waveforms at half the sampling rate of the original signal. This is the rationale behind wavelet transform.

For a band-limited signal, $f(t)$, as shown in Fig. 2.2, we take the sampling sequence $f(nT_s)$ and $\frac{1}{T_s}$ is the Nyquist sampling frequency. If we pass the $f(t)$ through a pair of filters h_l and h_h as shown in Fig. 2.2 the resulting low pass signal s_l and s_h will be orthogonal and complete decomposition of the original signal $f(t)$ but with half the bandwidth. The Nyquist sampling frequency for the sub-signals s_l and s_h will be half of that of $f(t)$. This procedure is shown in Fig. 2.2 and this is exactly a one scale wavelet decomposition or wavelet transform of signal $f(t)$. Of course we can apply this procedure to the sub-signals recursively –we have a multi-scale wavelet transform. The scale concept will be more rigorously stated after we introduce Short Time Window Fourier Transform in next section.

2.3 Self-Similar Web traffic

Pioneer work of self-similar network traffic begins with Leland et al. [4], where they show that the observed Ethernet LAN traffic exhibits long-range dependence and large time-scale self-similarity. Subsequent works provide convincing empirical evidence that self-similar traffic patterns are prevalent in today's network infrastructure [5][6][7]. Willinger et al. [8] provide more insights into the reason why this is the case by showing that the superposition of traffic between many source/destination pairs exhibits self-similar pattern if the traffic in each pair is generated by a ON/OFF source (the source alternatively generates traffic and falls to sleep) and the time lengths of the ON-period and the OFF-period follow heavy-tailed distributions. A bibliographical guide to self-similar traffic research can be found in Willinger et al. [9], where they provide hundreds of references. With the explosive growth of World Wide Web, Web traffic begins to catch researchers' attention in recent years. Crovella and Bestavros [10] show that Web traffic does exhibit self-similar pattern when the workload is high[1]. Using the model developed by Willinger et al. [8]. They also empirically show that the self-similar Web traffic can be well attributed to two factors: heavy-tailed Web document distribution and heavy-tailed user thinking time distribution (also see [11] [12] [13]). Mahanti et al. examine Web traffic in a proxy hierarchy and finds similar results[14]. Server side traffic of a very busy Web site (MSNBC) is examined by

[1] They do not observe self-similar Web traffic in less-busy hours.

Padmanabhan and Qiu [15], where they confirm that the Zipf's law of Web request holds.

2.4 Wavelet-based Web traffic Prediction

Given a time series, the prediction problem is to forecast the future by finding patterns in the past and extrapolating these patterns to the future. Time series prediction problems arise from various scientific disciplines: physics, economics, social science, biology, just name a few. And hence a lot of prediction methods have been proposed in various contexts. A complete survey of all these methods is far beyond the scope of this paper. However, based on the underearth philosophy of all these prediction methods, they can roughly divided them into two categories: statistical methods and non-statistical methods. Statistical methods view a time series as a realization of a certain random process and try to estimate the structure and parameters of the random process (build the model), then use the estimated model to forecast the future. Classic methods include linear regression, Bayesian inference, Maximum-likelihood estimation, etc. All these methods can be found in any classic statistics books. Tanner[16], Cox and Hinkely[17] are among the best of them. The non-statistical methods are more popular in the computer science community. These methods include neural networks, genetic algorithms, reinforced learning, etc. Mitchell[18] is an excellent introduction to most of these methods.

2.5 Workload Synthesizer

Different approaches have been pursued to generate realistic workload. We survey several of them, which are SURGE [19], b-model [20] and a approach based on wavelet-based Hurst parameter estimation [21].

SURGE (Scalable URL Reference Generator) is a WWW workload generator framework which combines multiple analytical models of WWW use [19]. The work based on the observation that web server workload is generated by client-side request sending behavior, which is assumed to be an ON/OFF process per client in their work.

An ON/OFF process of a client on sending URL requests depends on a series of analytical models of how WWW is used. The following list shows the aspects of WWW use whose analytical models were suggested to be taken into consideration by [19] :

1. Server-side file size distribution.
2. Popularity of the files, i.e. distribution of the number of requests on the files.
3. The sequence of URL requests sent to the server.

4. Distribution of the number of requests sent per ON period.

5. Distribution of the length of Active-OFF periods (time to wait to send the next request, while in the ON period).

6. Distribution of the Inactive-OFF periods (time for the client to stay idle between two consecutive ON periods).

The advantage of this workload generation framework is its flexibility on adding/removing web usage models for studying how does a particular factor affect the web server workload. The disadvantage is that too many models have to be considered, let alone the large amount of parameters.

The "b-model" approach takes in three parameters (total amount of workload, the length of the output workload sequence 2^n, and the workload distributing factor b) and generates workload by distributing the total traffic into a sequence of 2^n equally separated time points. The workload is generated by a top-down two-halves traffic distributing process: a workload of 2^i points is the concatenation of 2 workloads of 2^{i-1} points, with one constitutes b times of the total workload, and the other has the rest.

In the paper, the authors proposed three methods to estimate the b value: Entropy Plot, R/S plot and Variance Plot. The b value can be directly calculated from the entropy plot, however, if the R/S plot or the variant plot are used, Hurst parameter is first estimated, followed by an approximation of the b value from the value of the estimated Hurst parameter.

The previous two workload generation methods are somewhat not directly related to the wavelet concept. However, the approximation of the b value from the Hurst parameter gives us a way to combine wavelet analysis technique into the workload generation effort. In [21], a wavelet based method for the estimation of the Hurst parameter is proposed, which could be used to generate synthetic workload using the "b-model" method proposed by [20]. Their proposed estimating method of the Hurst parameter in the wavelet domain derives directly from the following two statements:

1. the wavelet decomposition discriminates the different frequential contributions, in other words, the wavelet coefficients at a particular level is associated to the mean amount of energy of the analyzed signal around some specific frequency.

2. self-similar processes are characterized (at least asymptotically) by a $\frac{1}{f^{\gamma}}$ decay of the power spectral density, and if the signal is of the form of fractional Gaussian noise, the Hurst parameter H can be calculated as $H = \frac{(\gamma+1)}{2}$.

Combining the method from [21] for estimating Hurst parameter and the method to generate synthetic workload using estimated Hurst parameter from [20], we can have a workload generating method based on wavelet-based analysis.

3 Wavelet Analysis: Theory and Observations

3.1 Short Time Fourier Transform(STFT)

Wavelet transform is a signal analysis tool with the capacity of variable time-frequency localization. Short time Fourier transform is fixed localization. We are interested in the multiresolution aspect of the compactly supported wavelet.

Standard Fourier transform decomposes signal into frequency components and determines the relative strength of each component. It says nothing about when that frequency component occurs. One idea to resolve this dilemma is to use a fixed time duration window sweeping over the time domain of the signal to extract the "time-localized" frequency components of the signal confined within this time window. This is referred to as the Short-Time Fourier Transform (STFT) or Gabor transform.

Mathematically, we can formulate STFT as following. For a infinite time duration signal $f(t)$, we propose a fixed time duration window signal $g(t)$ by design certain criterion. $f(t)$ and $g(t)$ are shown in 3.1 – we choose $g(t)$ as a rectangular pulse function here.

$$g(t) = \begin{cases} 1 & \text{for } \|t\| \leq T/2, \\ 0 & \text{for } \|t\| > T/2 \end{cases} \tag{1}$$

where T is the window size.

To express the sweeping procedure, we need another variable to denote the sweeping position of the window function $g(t)$ along the time axis, say τ. Then $g(t-\tau)$ moves $g(t)$ to the right of $g(t)$ by a distance τ, or we can say that $g(t)$ sweeps to the time instance τ. Multiplying $g(t-\tau)$ with $f(t)$ and taking the regular Fourier Transform, we have

$$STFT_\tau(f) = \int_{-\infty}^{+\infty} f(t)g(t-\tau)e^{-j2\pi ft}dt = \int_{\tau-T/2}^{\tau+T/2} f(t)e^{-j2\pi ft}dt = e^{-j2\pi f\tau}\int_{-T/2}^{T/2} f(t+\tau)e^{-j2\pi ft}dt. \tag{2}$$

This STFT gives us the frequency components of the signal around the time localized to τ. It is obvious that $STFT_\tau(f)$ has nothing to do with the all the signal values $f(t)$ if t is not in the neighborhood of τ, and hence, $STFT_\tau(f)$ is a localized feature of the signal $f(t)$ around $t = \tau$. If we move the variable τ from $-\infty$ to $+\infty$ in the time domain, we would capture all the frequency features localized to every time instance. Now let us focus on a single time instance τ to see the decomposition. Recalling the Fourier Series Expansion of a time limited signal, say $g(t)$, with a time duration support width T, or a periodic signal

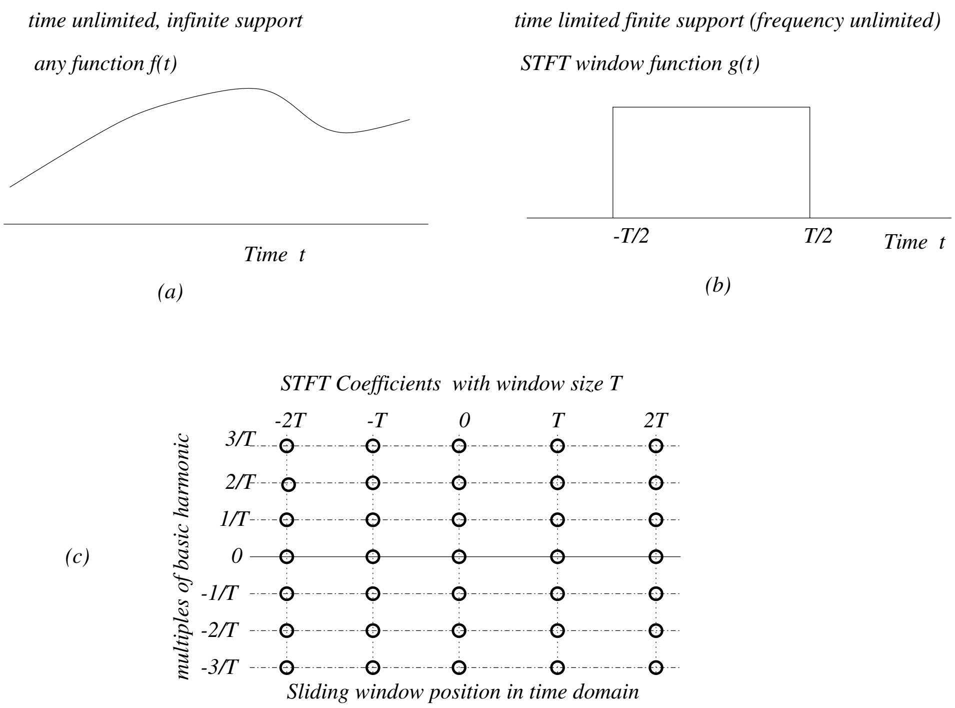


Figure 3: STFT Transform

with a period T, we know that its frequency component components (spectrum) different from zero at only the harmonics of the base frequency $1/T$. Therefore, for each fixed τ, $STFT_\tau(f) \neq 0$ only at multiples of $f_0 = 1/T$ or $f = n/T = nf_0$. We can express $STFT_\tau(f)$ as a sequence with respect to f,

$$STFT_\tau(f) = \sum_{i=-\infty}^{+\infty} c_\tau(i)\delta(f - i \times f_0) \tag{3}$$

where $c_\tau(i)$ is the discrete frequency components localized to time τ. To recover the localized signal, we use the Inverse Fourier Transform or Inverse Fourier Expansion,

$$f_\tau(t) = \frac{1}{T}\sum_{i=-\infty}^{+\infty} c_\tau(i)e^{j2\pi it/T} \tag{4}$$

where we define the signal $f_\tau(t)$ as time limited with support time duration width T,

$$f_\tau(t) = \begin{cases} f(t-\tau) & \text{for } \|t-\tau\| \leq T/2, \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

STFT transforms an arbitrary signal $f(t)$ into a 2-dimensionally indexed coefficient set as shown in Fig. 3.1. To recover the original signal over all the time domain, we have

$$f(t) = \sum_{j=-\infty}^{+\infty} f_{\tau=j\times T}(t) = \frac{1}{T}\sum_{j=-\infty}^{+\infty}\sum_{i=-\infty}^{+\infty} c_{j\times T}(i)e^{j2\pi it/T} \tag{6}$$

by simplifying $c_{j\times T}(i)$ into $c_j(i)$ or c_{ij}, we have the "wavelet coefficients" a wavelet transform derived from the rectangular time window function $g(t)$. c_{ij} and $f(t)$ are mutually exclusively determined.

3.2 Wavelet Transform

For the purpose of multiresolution analysis, orthonormal wavelets with finite support are most often used for their capability of variable time-frequency localization. To better understand the multiresolution aspect of wavelet, we show in the section the one-to-one correspondence between the coefficients in the wavelet transform and the dyadic subband analysis.

Wavelet transform is the next logical step from STFT: using variable size window instead of fixed window size. The variable window size gives us variable time-frequency localization comparing with fixed localization in STFT. Our prime interest is the multiresolution aspect of wavelet analysis.

Two concepts lays the foundation of wavelet transform: signal translation and scaling. A translation of a signal $f(t)$ by a constant time b is a new time signal $f(t-a)$ while a scaling is $f(t/a)$ where a is a positive real number. Assuming $f(t)$ has finite support in time, we can visualize that scaling contracts or expands the finite support of $f(t)$ as shown in Fig. 4 according to the values of a. If we take the Fourier transform, we can see the spectrum of $f(t)$ expands or contracts as well, but opposite to the behavior of $f(t)$ as shown in the figure. The left columns of figures are $f(t) = (1-t^2)exp(-t^2/2)$, $f(2t)$ and $f(0.5t)$. We can see the support expansion and contraction with $a > 1$ or $a < 1$. The right column of figures are the corresponding Fourier transform. More importantly, $f(t)$ is localized both in time and in frequency.

Assume that $\psi(t)$ is a function with the time-frequency localization property as $f(t)$. We can make a set of functions by translation and scaling $\psi(t)$ into a orthonormal basis of the Hilbert space of $L^2(R)$. Let us denote such a basis as

$$a = a_0^m \quad b = nb_0a_0^m \tag{7}$$

so that

$$\psi_{mn}(t) = a_0^{-m/2}\psi(a_0^{-m}t - nb_0) \tag{8}$$

where m and n are integers. For any function $f(t)$, we can have

$$f_(t) = \sum_m \sum_n d_{mn}\psi_{mn}(t) \tag{9}$$

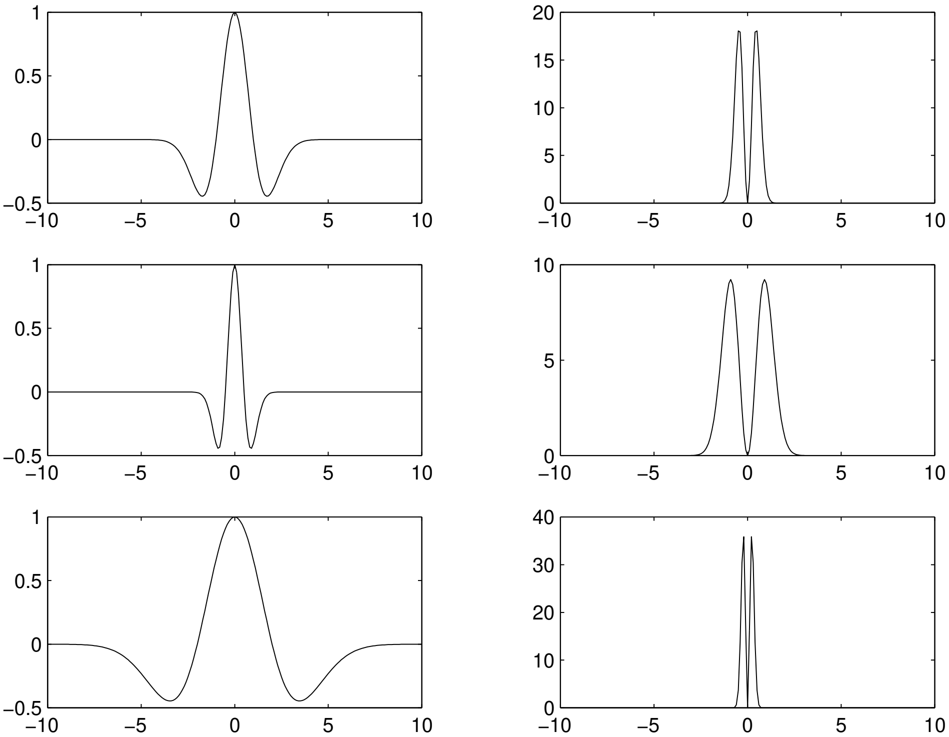


Figure 4: Wavelet Scaling

where d_{mn} are the wavelet coefficients which are

$$
\begin{aligned}
d_{mn} &= < f(t)\ ,\ \psi_{mn}(t) > \\
&= \frac{1}{a_0^{m/2}} \int f(t)\psi(a_0^{-m} t - n b_0) dt
\end{aligned}
\tag{10}
$$

Wavelet transform is a process of breaking a time signal into linear combinations of a chosen basis defined as above. Due to the locality of the basis signal, wavelet transform is better suited for representing signals with abrupt changes. To understand the meaning of wavelet coefficientsΦiΦ, we can consider the following procedure. Choosing $\psi(t)$ as shown in the above figure as the mother wavelet, we translate it to a fixed time τ, we have a new wavelet $\psi(t-\tau)$. Comparing $\psi(t-\tau)$ with the signal to be analyzed $f(t)$ over the finite support of $\psi(t-\tau)$ (by inner product), we use a real number as a measure of the closeness of $f(t)$ and $\psi(t-\tau)$, which we call the wavelet coefficient at time τ at scale 1. If we change the scale factor to 2, but with the same time translation τ, we have a new wavelet function $\psi(2(t-\tau))$ and compare it with $f(t)$ to obtain a new wavelet coefficient. The two coefficients are all taken at time τ but at different scale level. In the wavelet figure, they are aligned vertically at time τ. We know $\psi(2(t-\tau))$ and $\psi(t)$ are thiner or fatter copies of each other – they have different time axis support. If the two coefficients are the same, we can conclude that at time τ, our signal $f(t)$ is similar to $\psi(t)$ in a larger scale while similar to $\psi(2(t-\tau))$ in a smaller scale. If the two similarity (wavelet) coefficients are the same, they are close in exactly the same degree. This is exactly the definition of self-similarity of a signal. Therefore, to check if a signal possesses self-similarity properties we can check if there are vertical lines in the wavelet frequency-time figure. Preliminary processing of the web trace date shows obvious vertical lines in the figures. Details will be discussed later.

3.3 Preliminary Wavelet Analysis Data

The 16^{th} Federation Interationale de Football Association (FIFA) world cup was hold in France from June 10^{th} through July 12^{th}, 1998. The web site for France '98, `http://www.france98.com` covered the game continuously and received over 1 billion client requests. Web server data collected over a 92 day period (April 30th, 1998 — July 26th, 1998), produced a trace that allows us to study the workload of Web site during such major event.

The trace that I work with contains the following information:

- timestamp — The time of the request in GMT time which is 2 hours later than the local time.
- clientId — The unique integer that identified the client that made the request. The client might be a proxy that represents a group of user.
- objectId — The object been request.
- size — Number of bytes in response.
- method — The method that is contained in the request.
- status — The HTTP version indicated in the client's request and the response status code
- type — The type of file requested.
- server - The server that handled the request.

The entire trace is 250GB uncompress and 9GB compressed. In this project, due to the limited disk space and input size Matlab is capable of handling we will concentrate on a segment of the trace: that is the data for a 2-day period between June 23rd to June 24th inclusive.

Figure 5 shows the number of request per hour for the four day period that was studied. Both graphs peaked at 12:00PM and 6:00PM French time.

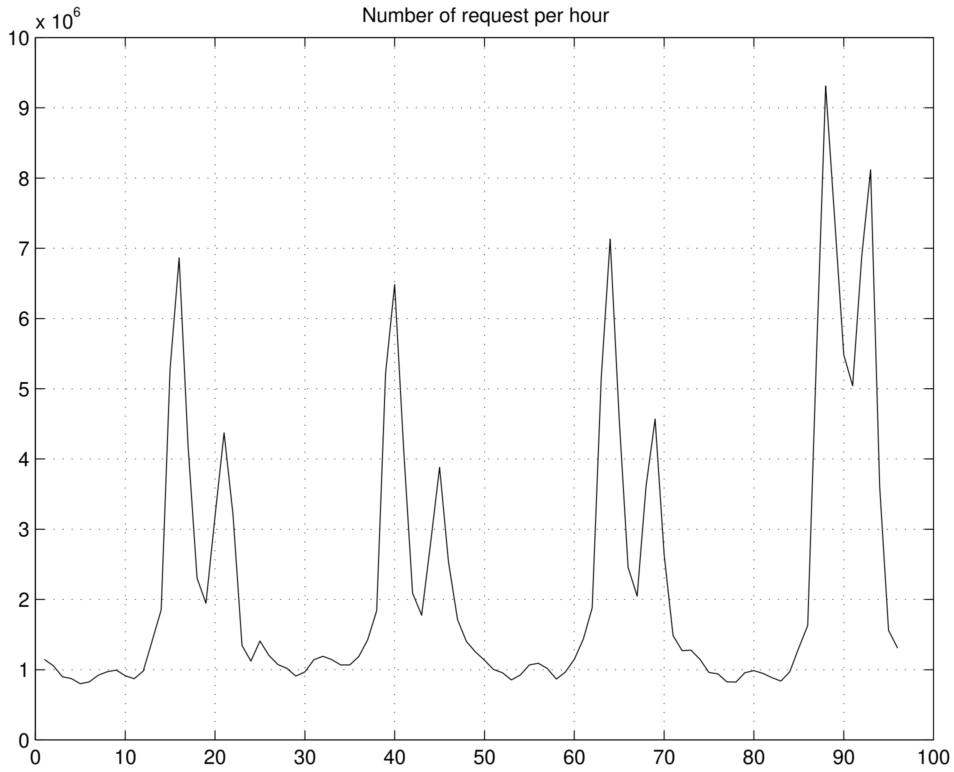


Figure 5: Number of request per hour

We parsed the input stream and collected number of request per unit time. We generated 3 different traces of data:

- number of requests per second
- number of requests per minute
- number of requests per hour

We performed wavelet analysis on different segment of the data with different scales. We have found:

- No obvious self-similarity pattern was observed with number of request per hour. The results are shown in Figure 7.
- With number of requests per second, when scale is low, no self-similarity is observed.
- With number of requests per second, when scale reached 48, self-similarity pattern become obvious. The results are shown in Figure 8.

- Self-similarity pattern was observed very clearly on number of requests per minute. The result are shown Figure 9, Figure 10.

Figure 6 shows the variance plot of the number of request per hor. It shows that the self-similarity pattern does not exist until scale 16 (2^4), which is consistent with the results obtained from continuous wavelet transform. The plot shows that there is a self-similarity pattern between scale 16 and 64, with Hurst parameter around 0.896 (slope$=-0.20799$).

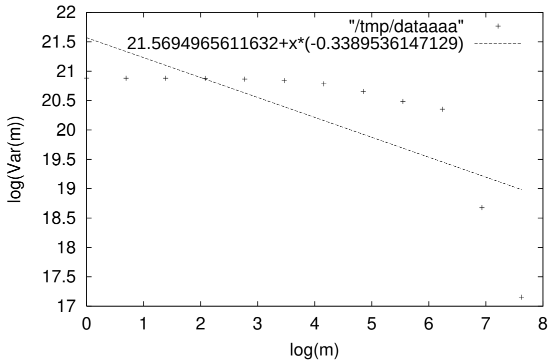


Figure 6: Variance Plot of Number of Request Per Hour

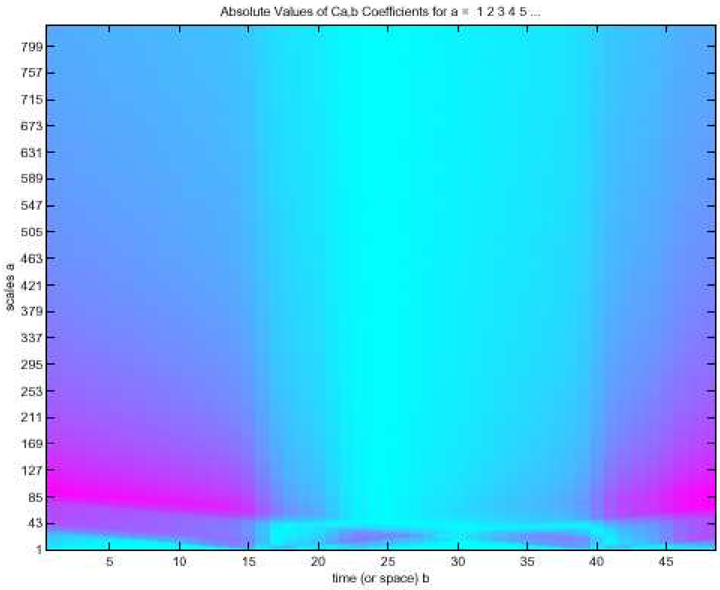


Figure 7: Wavelet Analysis on Number of Request Per Hour with Scale 1-800

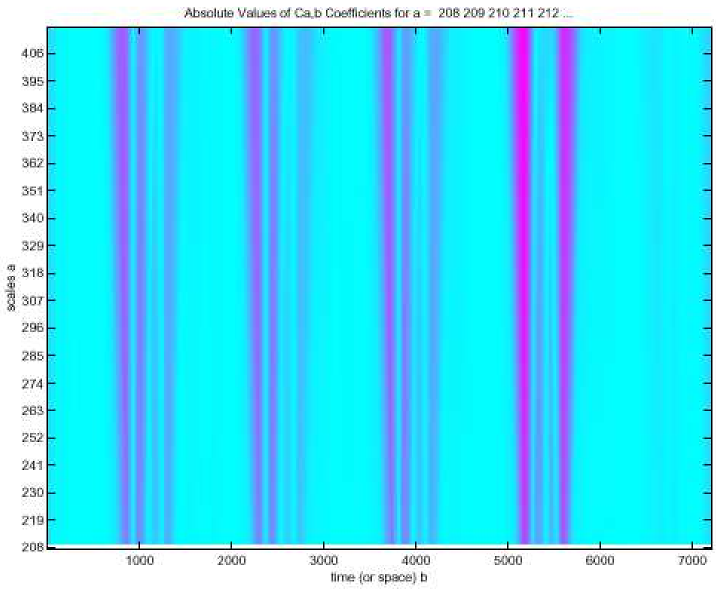


Figure 8: Wavelet Analysis on Number of Request Per Second with Scale 208-416

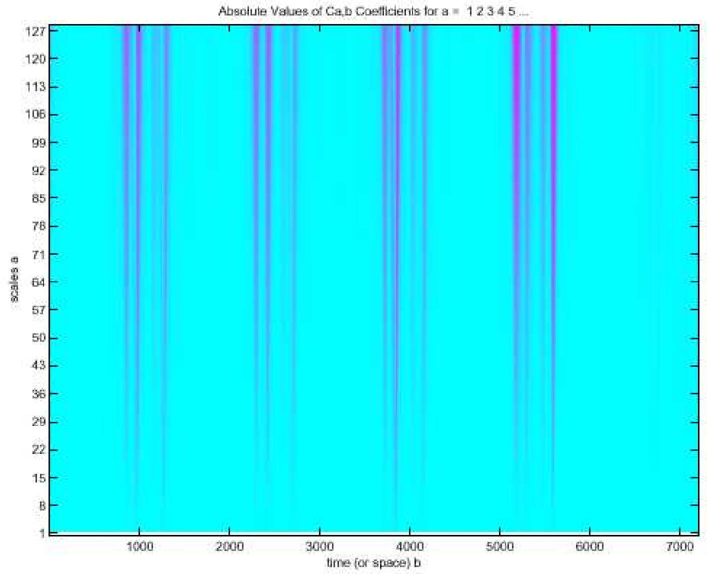


Figure 9: Wavelet Analysis on Number of Request Per Minutes with scale 1-128

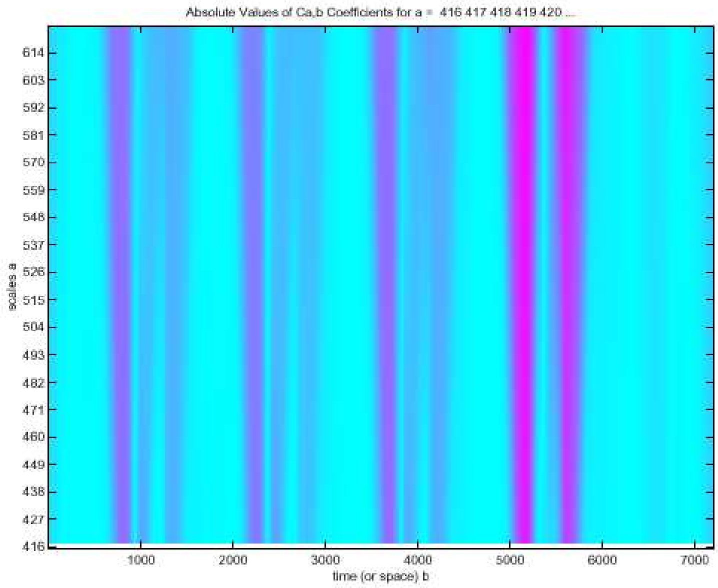


Figure 10: Wavelet Analysis on Number of Request Per Minutes with scale 416-624

4 Methods

4.1 Workload Synthesizer

The methodology of our workload synthesizer consisted of two main steps: *workload model fitting* and *workload generation*. The workload generation process is based on the model deduced from the workload model fitting process. Different workload models require different parameters. Therefore, appropriate parameters must be measured and given to a workload generation process before it can be used.

4.1.1 Workload generation

Many different models with different parameters had been proposed to characterize a workload, and in our work, we chose two classes of them: *b-model* [20] and *Multifractal wavelet model (MWM)* [22]. B-model requires three parameters to generate a workload having the corresponding characteristics. The parameters it needs are (1) total amount of workload, (2) the length of the output workload sequence 2^n, and (3) the workload distributing factor b, or the bias. The idea of its synthesization process is distributing the total traffic into a sequence of 2^n equally separated time points, according to the given bias b. More specifically, the workload is generated by a top-down two-halves traffic distributing process: a workload of 2^i points is the concatenation of 2 workloads of 2^{i-1} points, with one constitutes b times of the total workload, and the other has the rest. The workload generation is done directly in the time domain.

In contrast to the b-model, which synthesizes workloads directly in the time domain, the Multifractal wavelet model synthesizes workload in the wavelet domain. The idea of its synthesization is to follow the coarse-to-fine sythesization similar to what the inverse wavelet transform does, and generates/estimates the wavelet and scaling coefficients at each scale-level. The finest-scale scaling coefficients will be the desired output signal. The estimation of the coefficients at each scale is done by picking values according to some probability density functions, and different classes of the probability density functions create different classes of models for synthesizations. Several possible models for generating the coefficients are proposed in [22], including *β multifractal wavelet model* and *point-mass distribution model*. A model for modeling positive $1/f$ noise is also proposed in the paper, and the synthesization process is based on a given Hurst parameter and the desired workload's mean, standard deviation, along with the number of wavelet scales to be considered while synthesizing workload. While the β multifractal wavelet model and the point-mass distribution model can model a broader class of multifractal workloads than the strict self-similar $1/f$ noise workload class, we chose to work with the synthesization of the $1/f$ noise workload based on Hurst parameter first, due to its overlapped effort on parameter estimation (estimating Hurst parameter of a real world data) with that for the b-model.

4.1.2 Workload model fitting

Before using the b-model and the MWM model to synthesis realistic workload, appropriate parameters must be derived from the real world data and supplied to the corresponding synthesization model. For b-model, the most important parameter will be the bias, b, and for the MWM model on $1/f$ process, the Hurst parameter H.

Several methods exist to estimate the bias b of a real world workload. It can be derived directly, using the idea of *Entropy Plot* given in [20], or be derived indirectly through the estimation of Hurst parameter and the corresponding b value can be computed from the derived Hurst parameter using the relationship between them given in [20]. Currently, we chose the second approach, since the estimated Hurst parameter can also be used for the MWM model on $1/f$ process. The relationship between the Hurst parameter H and the bias b is

$$H = \frac{1}{2} - \frac{1}{2}\log_2(b^2 + (1-b)^2).$$

R/S method There are many ways to estimate the Hurst parameter of a given workload, ranging from the traditional approaches of *R/S method* and the *Variance-Time plot method* [23], to the wavelet-based *power-scale method* [21] and so on.

The R/S method is one of the oldest and most well known methods for estimating H [24]. For a time series, $X = X_i, i \geq 1$, and defined the partial cumulative sum as $Y(j) = \sum_{i=1}^{j} X_i$. The *R/S-statistic* or *rescaled adjusted range* is defined as

$$R/S = \frac{R(t,k)}{S(t,k)} where$$

$$R(t,k) = \max_{0 \leq i \leq k}[Y_{t+i} - Y_t - \frac{i}{k}(Y_{t+k} - Y_t)] - \min_{0 \leq i \leq k}[Y_{t+i} - Y_t - \frac{i}{k}(Y_{t+k} - Y_t)], and$$

$$S(t,k) = \sqrt{k^{-1}\sum_{i=t+1}^{t+k}(X_i - \bar{X}_{t,k})^2}, where$$

$$\bar{X}_{t,k} = k^{-1}\sum_{i=t+1}^{t+k} X_i.$$

Intuitively, the $R(t,k)$ can be considered as the *adjusted range* or the maximum difference of the partial time series $X_i, t+1 \leq i \leq t+k$, and $S(t,k)$ is by definition the sample variance of that partial time series.

For fractional Guassian noise or fractional ARIMA series, the R/S-statistic is proportional to k^H, where k is length of the partial time series under consideration. In other words, the relationship is

$$R(t,k)/S(t,k) = C_H k^H.$$

With the relationship between R/S-statistic and the corresponding scale, the Hurst parameter can be estimated as follow: first, plot the log-log plot of R/S-statistic versus k by calculating every possible, or a sufficient number of values of t and k. The estimated Hurst parameter will be the slope of the best fitting straight line to the points in the plot.

One of the problem of the R/S method is known to be biased towards $H = 0.7$, that is, it biased upwards for small values of H and downwards for large values of H.

Variance-Time plot The idea of the variance-time plot method is based on one of the characteristics of long-range dependent processes: divided the total time range into nonoverlapped intervals, the variance of the expected number of data points fallen in an interval converges slower to zero than $1/n$ (the reciprocal of the sample size), as the interval width increases.

The relationship between the variance of the expected number of data points fallen in an interval and the interval width n is

$$var(\bar{X}_n) = cn^{2H-2},$$

where $c > 0$. So the process to estimate Hurst parameter H using the variance-time plot method, is first plot the log-log plot of $\bar{X}_n$ versus interval width n, fit a straight line to the points in the plot, and the slope of that best fitting line is approximate to $2H - 2$.

A common question for plotting approaches, such as R/S method and Variance-Time plot method, is how to choose the cut-offs of the data points on the plot to exclude the boundary points that would distract the result of line-fitting.

Wavelet-based power-scale plot method The idea of the power-scale plot method for estimating the Hurst parameter is from the following two statements [21]:

1. the wavelet decomposition discriminates the different frequential contributions, in other words, the wavelet coefficients at a particular level is associated to the mean amount of energy of the analyzed signal around some specific frequencies.
2. self-similar processes are characterized (at least asymptotically) by a $\frac{1}{f^\gamma}$ decay of the power spectral density, and if the signal is of the form of fractional Gaussian noise, the Hurst parameter H can be calculated as $H = \frac{(\gamma+1)}{2}$.

In other words, the powers contributed by the wavelet coefficients on different scales have the following relationship:

$$\Gamma_m = 2^{m\gamma}\Gamma_0,$$

where $\Gamma_m = \frac{1}{n_m}\sum_n |x_{mn}|^2$ is the power contributed by the wavelet coefficients on scale m, and $\gamma = 2H - 1$.

Therefore, the step to estimate the Hurst parameter using power-scale method is first make a plot of $\log(\Gamma_m)$ versus m, and then evaluate the slope, γ, of the best-fitting straight line of the data points on the plot, and the Hurst parameter H is approximate to $\frac{(\gamma+1)}{2}$.

4.1.3 Summary

In summary, our proposed approach to synthesize a realistic workload consists of two steps:

1. Estimate the Hurst parameter of a given real world workload, and
2. Use the workload synthesis methods, such as b-model or MWM for 1/f traffic, to generate a realistic workload. (Given them the Hurst parameter obtained from the first step, and other workload characterizing statistics like mean and standard deviation of the workload.)

4.2 Wavelet-based time series prediction

The wavelet-based time series prediction consists of four steps. In step one, we decompose the original time series into wavelet coefficients; then in step two, we extract statistical characteristics from the wavelet coefficients; in the third step, we extrapolate the wavelet coefficients based on their statistical characteristics; and finally in step four, we transform the extrapolated wavelet coefficients back into time domain to get the predicted time signal.

One stage wavelet decomposition can be viewed as passing a time single through a high-frequency-pass filter and a low-frequency-pass filter, as shown in Figure 2.2. The low-frequency content, denoted by A_1 (A stands for approximations), represents the large time scale trend of the signal; the high-frequency content, denoted by D_1 (D stands for details), is the short time range details of the signal. We can apply this procedure recursively: decompose A_1 to A_2, D_2; decompose A_2 to A_3 and D_3, and so on. The maximum level of stages one can reach is determined by the number of data points available. For discrete wavelet decompositions, one can reach at most $Log(n)$ levels, where n is the number of data points in original signal. However, for prediction purpose, we may not decompose the original signal all the way down to the maximum level. We can stop at, say, level m. Once way stop at level m, we get the wavelet coefficients $A_m, D_m, D_{m-1}, D_{m-2}, ...D_1$.

The second step of wavelet-base time series prediction is analyzing these wavelet coefficients. In this paper, we use normal distributions to model the wavelet coefficients (Other models may also be applicable. However, the normal model is simple and provides satisfactory results, as shown in the next subsection). For each D_i, there is a normal distribution $N_i(\mu_i, \sigma_i)$ associated with it. (The same thing happens to A_m) Suppose the length of D_i is J_i, then we have J_i samples of this normal distribution. Based on these samples, we can estimate μ_i, and σ_i.

Once we get the estimated normal distributions for the wavelet coefficients, it is easy to extrapolate them. The simplest way to do this is to extend the coefficients by the means of corresponding normal distributions. In this paper, we use a probabilistic method. For each D_i, we extend it by using a normal random generator with mean μ_i and σ_i).

The last step of prediction is transforming the extended wavelet coefficients back to time domain. This is done by using inverse wavelet transform.

The reason that we analyze and extrapolate the wavelet coefficients instead of the original signal itself to predict the future is that wavelet transform can effectively de-correlates even highly correlated signals [25]. For a given time series X, the autocorrelation sequence is defined as

$$\rho(\tau) = \frac{\Sigma_{t=0}^{n-\tau-1}(X_t - \overline{X})(X_{t+\tau} - \overline{X})}{\Sigma_{t=0}^{n-1}(X_t - \overline{X})^2} \tag{11}$$

ρ measures the correlation between values of the time series X that are separated by τ units. If a time series is highly correlated, $\rho(\tau)$ will has a long tail, which means that what happens currently will cast its effects on an even far way future. In other words, what happens now is high correlated with what happened a long time ago. This property, called heavy-tailed property, makes the original time series hard to predict. By using wavelet transforms, we decompose the original into wavelet coefficients, which generally have a fast decaying autocorrelation sequences. Therefore, extrapolating on the wavelet coefficients is a much more effective way to predict the time series. In fact, one major advantage of wavelet analysis is that it provides a way of re-expressing correlated time series in terms of some combination of uncorrelated variable, which makes the following statistical analysis easier.

H	R/S method	Variance-Time method	Power-Scale method
0.5	0.5343	0.4881	0.5000
0.55	0.5484	0.5448	0.5714
0.6	0.5968	0.5884	0.6071
0.65	0.6408	0.6316	0.6143
0.7	0.6874	0.6739	0.7
0.75	0.7272	0.7118	0.75
0.8	0.7560	0.7493	0.7857
0.85	0.7995	0.7845	0.8929
0.9	0.8164	0.8168	0.8929
0.95	0.8026	0.8470	0.9464
Error	0.0593	0.0521	0.0197

Table 1: Estimate Hurst parameters of traces generated using b-method

5 Experiment Results

5.1 Workload Synthesis

5.1.1 Generate synthesized workload with specific Hurst parameters

We used the b-model workload synthesizer and the Multifractal Wavelet Model workload synthesizer to generate synthesized workloads with specific Hurst parameters. These "standard" workloads could be used on testing the correctness of the programs we wrote for Hurst parameter estimation. They could also act as a comparison basis to compare the several methods doing Hurst parameter estimation.

Table 1 and 2 list the results of the three Hurst parameter estimating methods on the synthesized workloads generated by the b-model synthesizer and the MWM synthesizer, respectively. The error values at the bottom row of each table are the square roots of the mean squared error of the estimated Hurst parameters reported by the corresponding method. In general, three of the methods work quite nice for the synthesized workloads, while *power-scale method* has the closest estimations to the desired values.

Figure shows the power-scale plots of the workloads generated using MWM with H=0.75 and 0.85. In general, methods for Hurst parameter estimation work better if the workload has H close to 1.

H	R/S method	Variance-Time method	Power-Scale method
0.5	0.5392	0.6476	0.4576
0.55	0.5786	0.6476	0.5272
0.6	0.6100	0.6641	0.5758
0.65	0.6451	0.6875	0.5970
0.7	0.6934	0.6922	0.6727
0.75	0.7367	0.7372	0.7636
0.8	0.7913	0.8159	0.8060
0.85	0.8380	0.8347	0.8727
0.9	0.8566	0.8518	0.9000
Error	0.0231	0.0666	0.0283

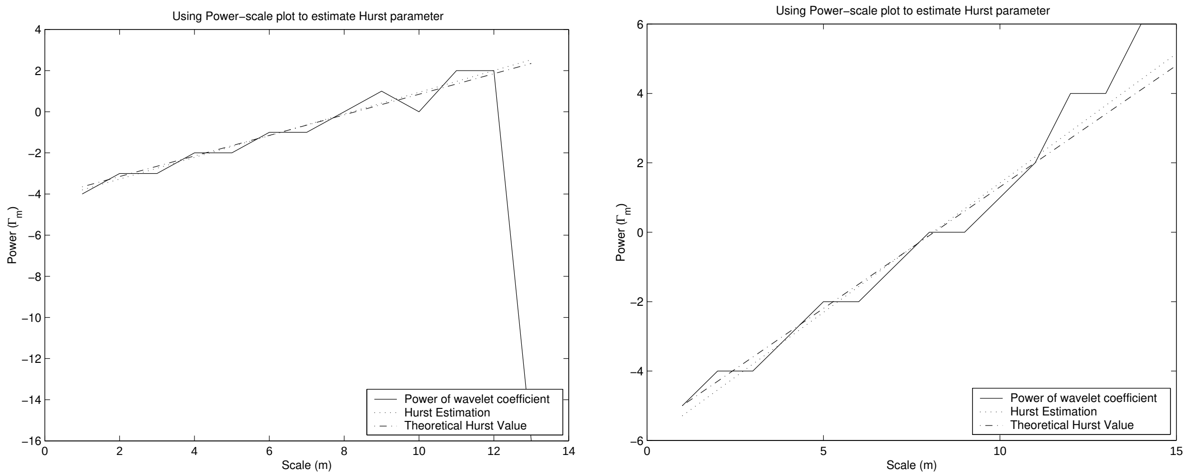
Table 2: Estimate Hurst parameters of traces generated using Multifractal wavelet method

5.1.2 Estimating Hurst parameter of real world workload

Table 3 shows the estimated Hurst parameters of the workloads collected from the World Cup 98 web sites. The power-scale plot method is used to estimated the Hurst parameter. When workloads of 24 hours are examined at once, the estimated Hurst parameters are all bigger than 1, which is a invalid value. This is probably because of the number of requests varies during the day, which violates the stationary assumption of the self-similarity traffic (figure 12).If we separate a 1-day workload into two halves, the first half (requests entered before noon) shows self-similarity since its workload is more stationary, while the second half is not and has estimated Hurst parameter bigger than 1. In [24], the author recommended to recommend at most one hour at a time of the workload at a time. This result is consistent with the result obtained from continuous wavelet transform (figure 8), which shows that self-similarity only exists at certain scales.

5.2 Prediction

In this subsection, we show the performance of our wavelet-based prediction algorithm through two experiments. In both experiments, we use the traffic data collected in the 57th day of the World Cup Tournament. The resolution of these data is one minute, therefore there are $60 \times 24 = 1440$ data points. We use the first 1252 data points as the training set (approximately 0.8 of the whole data). For the 2 minutes forward prediction experiment (Figure 13), we feed the first 1252 training data into our algorithm, and it outputs the predicted traffic data of the next 2 minutes, which correspond to the time point 1253 and



(a) (b)

Figure 11: Hurst parameter estimation using power-scale plot. Haar wavelets are used in this case. Workloads are synthesized using MWM with (a) H=0.75 (b) H=0.85.

1254; then the real data at time point 1253 and 1254 are appended to the first 1252 old training data to form a new training set worth of 1254 data points; this new training set is fed into our algorithm, and it then predicts the traffic at time point 1255 and 1256. Repeat this process until we reach the end to data.

The 10 minutes forward prediction experiment is conducted in the same way except that we exam 10 prediction points instead of 2. The results are shown in Figure 14. In both Figures, the origin of x-axis is the time point 1253, we do not show the first 1252 training data here.

As we can see from Figure 13, our algorithm works very well for the 2 minute forward prediction, the predicted data are almost coincident with the real data. For the 10 minute forward prediction, our algorithm fails to follow the real data only when they have a sudden surge, as shown in the Figure 14. When the traffic does not change so dramatically, our algorithm performs well. So, in general, the performance of our algorithm is good.

As expected, the further the forward prediction we do, the worse the quality of our prediction algorithm has. Figure 15 shows the results of the prediction algorithm on predicting next 20 and next 40 minutes traffic. It can be seen that the predicted curves “lag” behind the real data. This is probably because that our prediction algorithm does the prediction piecewisely, i.e. prediction of an interval is based on the real data seen so far.

It is also interesting to see if our prediction algorithm works well for synthetic signals,

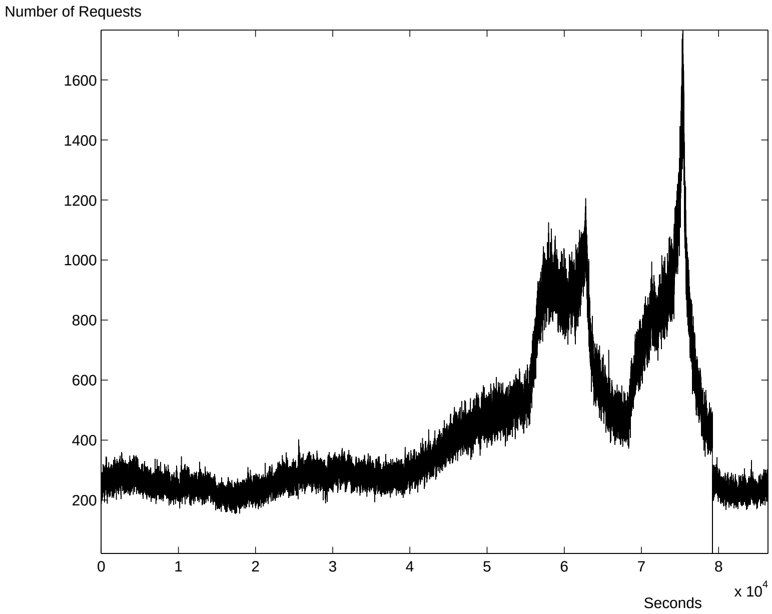


Figure 12: Number of requests per second on June 22, '98. The first half is more stationary, while the second half is not quite.

especially for periodic signals. Figure 16 shows the results of predicting the next 10 values of two periodic synthetic signals: sinusoid and step signals. The prediction algorithm still works nice in these cases. However, if the period of the signal decreases and becomes smaller than twice the time interval which we want to predict, the "lagging" effect of our algorithm make the predicted result bad. As shown in the Figure 17a, the period of the real signal is 10, and we do 10 steps forward prediction. The predicted values always lag behind the real signal 10 steps, which makes the prediction and the real signal have opposite phases. In Figure 17b, the period is 12 and the prediction just can not catch up with the real signal.

Day	Estimated H	Estimated H (before noon)	Estimated H (after noon)
June 21, '98	1.071678	0.886364	1.100000
June 22, '98	1.178322	0.959091	1.227273
June 23, '98	1.400350	0.940909	1.350000
June 24, '98	1.363636	0.986364	1.281818
June 25, '98	1.405594	0.931818	1.390909
June 26, '98	1.361888	0.986364	1.245455
June 27, '98	1.160839	0.968182	1.140909
June 28, '98	1.153846	0.895455	1.104545
June 29, '98	1.388112	0.959091	1.327273

Table 3: Estimate Hurst parameters of real world workload, using power-scale plot. Workloads are the number of requests per second to the World Cup 98 web sites.

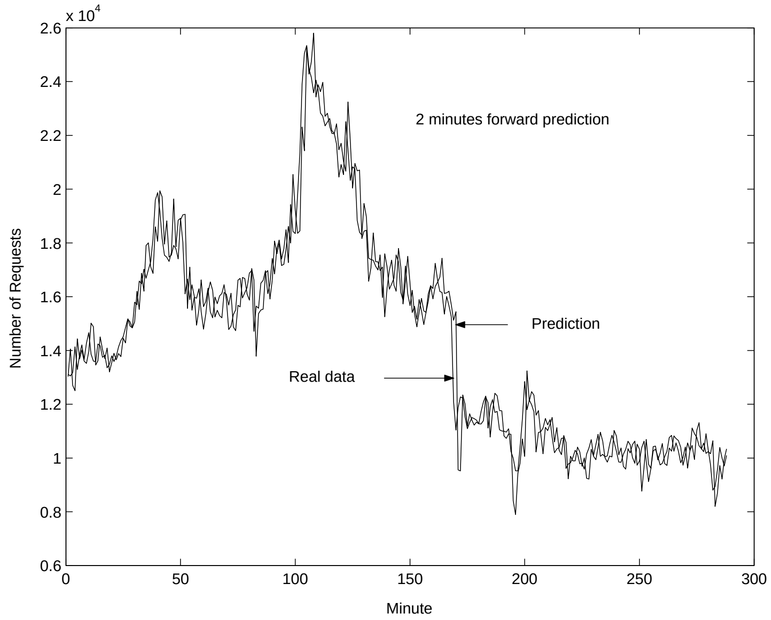


Figure 13: Comparison of the real traffic data and the 2 minutes forward predicted traffic data.

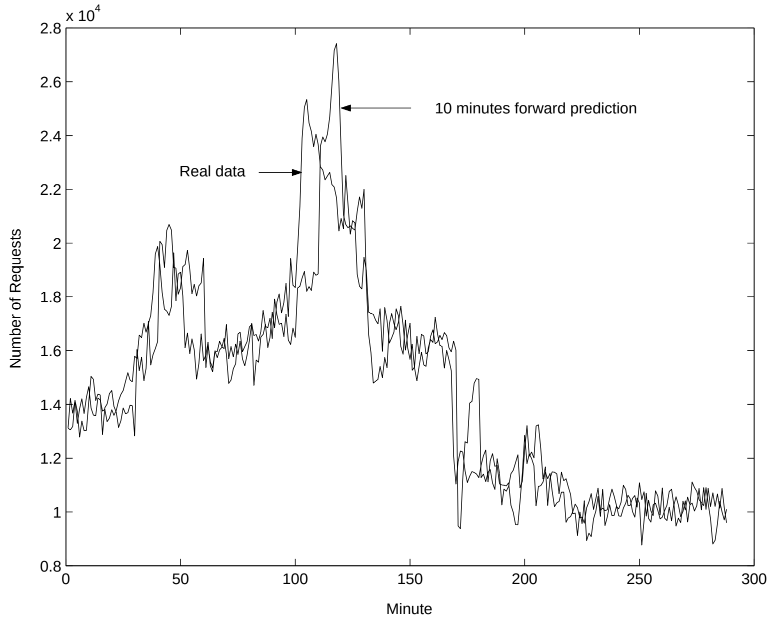


Figure 14: Comparison of the real traffic data and the 10 minutes forward predicted traffic data.

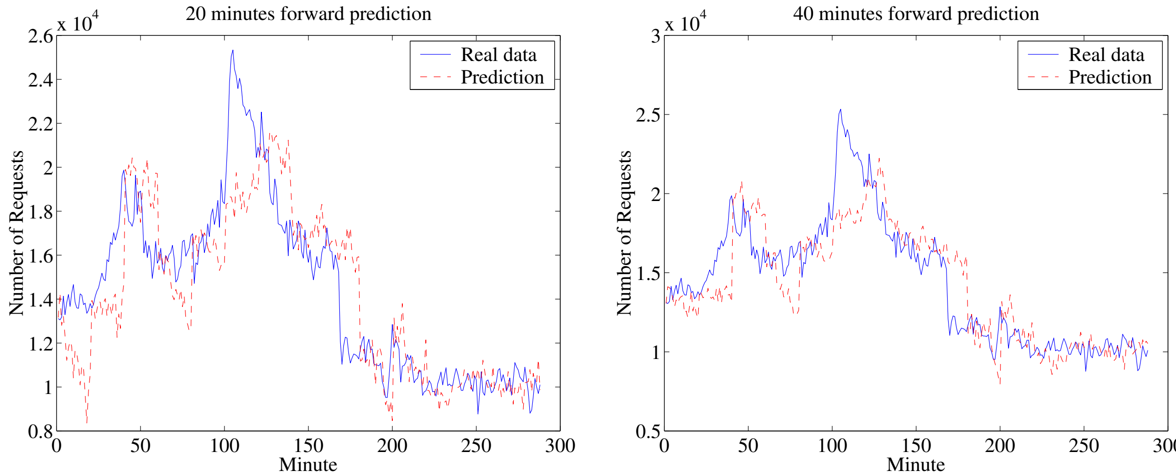


Figure 15: Comparison of the real traffic data and (a) the 20 minutes forward predicted traffic data (b) the 40 minutes forward predicted traffic data.

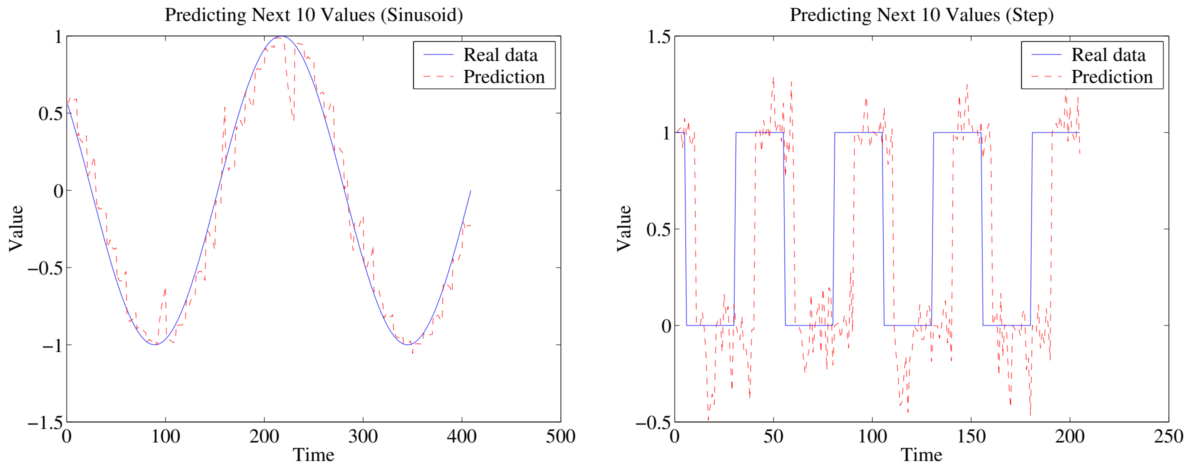


Figure 16: Predict the next 10 points of (a) sinusoid signal $sin(2 * \pi * 8 * x/1024)$, and (b) step signal with period 50.

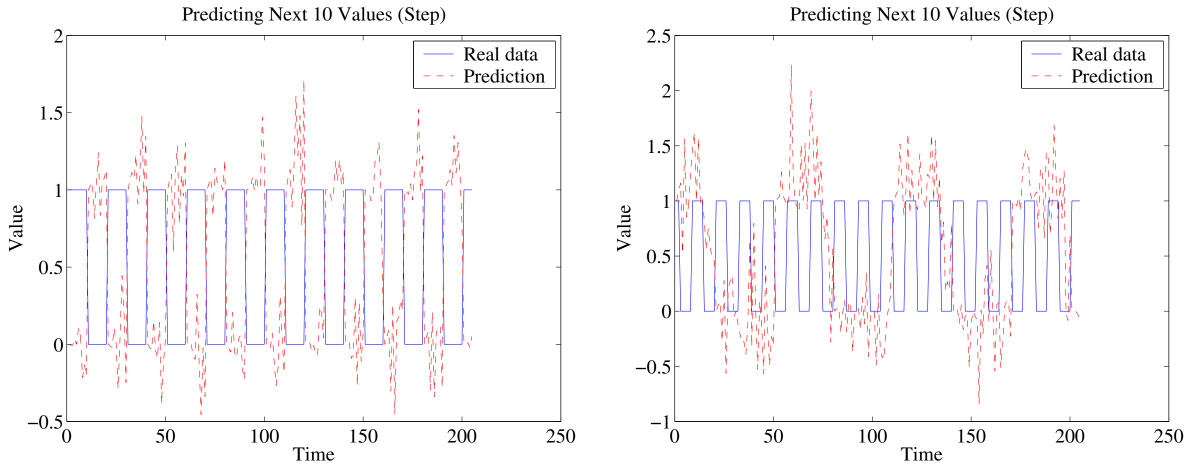


Figure 17: Predict the next 10 points of step signals with (a) period 20, and (b) period 12.

Parameter	High Performace Server	Low Performance Server	Infinite Queue
Max Queue Capacity	95000	195000	990000
Server Processing Power (MBytes/min)	3716	372	372
File Size Distribution: Bounded Pareto			
Lower Bound	Higher Bound	Alpha	Mean
3000	1000000	2.1	5717.7

Table 4: Setting of the Queue Model

Data Trace	Real (High)	Predicted (High)	Real (Low)	Predicted (Low)
Avg. Queue Length	43536	42408	59031	44576
Throughput (MBytes)	248.5	242	237	232
Avg. Time in Queue (min)	1	1	1.419	1.409
Drop Ratio (Requests)	0.0174	0.0234	0.0696	0.0785
Total Requests	12759741	12470825	12759741	12470825
Drop Ratio (Bytes)	0.0173	0.0206	0.0611	0.0692
Total Bytes (MBytes)	72943	71296	72943	71296

Table 5: Results of Queue Model Simulation. Experiments are done under high and low server settings, and the predicted data is obtained from 10-minute prediction.

5.3 Evaluation using Queue Model

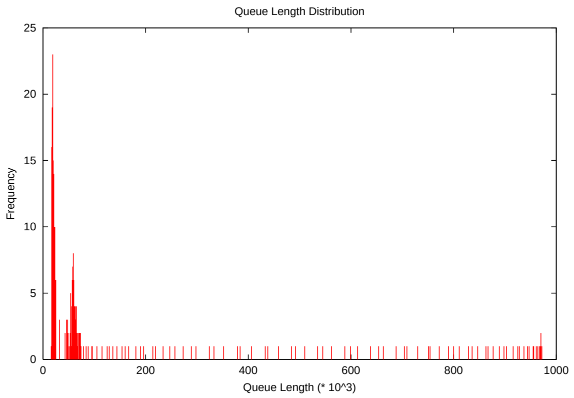
We used a simple queue model to evaluate how well our predicted data resembles the real world data.

In our experiments, the real world data we used is the number of requests per minute of the World Cup 1998 web site. We assumed that the requested files have size that follow the bounded pareto distribution. Table 4 lists the parameters we used in our experiments, and Table 5 shows the metrics measured in the experiments.

The results show that the predicted data tried to capture the burtiness of the real data. In 10-minute prediction, the algorithm works okay, it even behaves more bursty than the real data. This is shown in Table 5 that the server has better performance (e.g. lower drop rate) when it is simulated using the predicted data. However, if the prediction interval is bigger, say 40 minutes, the predicted data does not capture the burtiness, and the server performance of the simulation done on this data is much better (Table 6).

Data Trace	Real (Low)	Predicted (Low)
Avg. Queue Length	59031	40881
Throughput (MBytes)	237	226
Avg. Time in Queue (min)	1.419	1.031
Drop Ratio (Requests)	0.0696	0.0000
Total Requests	12759741	11421824
Drop Ratio (Bytes)	0.0611	0.0000
Total Bytes (MBytes)	72943	65306

Table 6: Results of Queue Model Simulation. Experiments are done under low performance server settings, and the predicted data is obtained from 40-minute prediction.



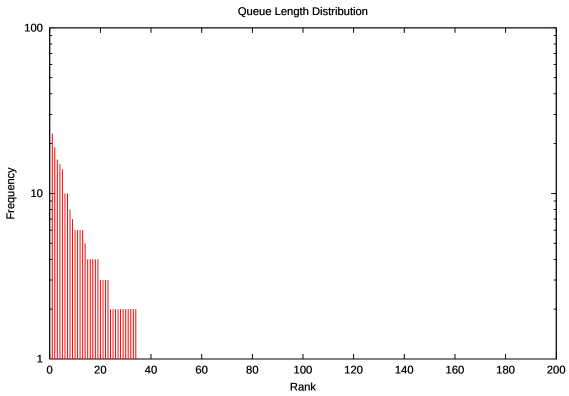


Figure 18: (a) Distribution of queue lengths observed at each time point , plotted in normal scale, and (b) Frequency-Rank plot of the observed queue lengths.

6 Future Work

Due to the limitation in time, we were unable to exploit many design parameters. If possible, we hope to extend our current work in two directions: modifying prediction algorithm for better prediction and improving evaluation techniques for better evaluation of our predicted sequence.

We currently model each different scales of the wavelet coefficients as uncorrelated random variables with normal distribution. To predict the next coefficient, we simple ask the random number generator to generate the next random variable. Our simplified assumptions is obvious not true, because we have observed correlation in the wavelet plot. It would be interesting to exploit correlation we observed and see whether we could generate better prediction by taking this characteristics into account. One possible way is to use the ARIMA model for the wavelet coefficients at each level, and we can even correlate coefficients at different levels.

We have built a simple queuing model to evaluate whether the predicted sequence and the original sequence are statically equivalent. However, due to time constrain, we made many unrealistic simplification, such as we assumed that the service time for all requests are equal. This is obviously a simplified solution, because the users may request different files which are different in size. Transfer time (service time) are proportion to files size. We would like to study the file size as a time sequence, and see if there are self-similarity patterns in this sequence. Further more, we could generate predicted service time having the same algorithm applied to file size sequence.

Last but not least, we would like extend our techniques and use them on other workload or time series analyzing tasks. We only ran our prediction algorithm with one set of data — workload on a web server. But our algorithm can be extend to analyze any time series data such as, disk I/O workload or stock quote sequences.

7 Conclusions

In this project, we tried to understand the temporal characteristics of web server workload using wavelet analysis. We observed self-similarity at certain scale using continuous wavelet plot and variance plot. We observed no self-similarity pattern in the the details of number of request per second. We observed clear self-similarity patterns on the number of requests per minute on the scale of 16 to 128. Beyond scale 128, the self-similarity pattern changed.

By computing Hurst parameters, we realized that there are self-similarity patterns on small sequence of data, such as a 1 hour sequence. However, we are not able to observe self-similarity patterns on longer sequence of data.

To better understand our prediction algorithm, we applied our prediction algorithm on synthetic data, such as sinusoid and step functions. These functions are periodical. Our predicted data follows the original signal approximately, but the generated signals are not smooth. We applied our algorithm to step function of different frequencies. When the frequency is low, we have good prediction. However, when the frequency increases, our prediction algorithm failed to predict the original sequence.

We use queueing model to determine whether our predicted data is statistically equivalent to the real world data, especially, whether our predicted data is able to capture the burstiness of the original data. Correlation between different scales of the wavelet decomposition indicates self-similarity. In our prediction, we were unable to model this correlation, hence, we should expect less obvious self-similarity patterns in the predicted data. Therefore, the predicted data should be less bursty than the real data and perform better on our queue model (low drop rate, small queue length). We used number of requests per minutes as out original sequence. We ran our queuing model on three sequences of data: the real world data, predicting 10 data points ahead, predicting 40 data points ahead. We observed no obvious difference between the real world data and the predicted sequence by predicting 10 data points ahead. However, when we used the sequence predicted by predicting 40 data points ahead, we observed that the predicted data performed much better than the real world data on the queuing model. Average queue length is reduced by 30%. Hence, it is not enough the model the wavelet coefficient of web server workload as random variables with normal distribution. It is necessary to exploit the correlation between different scales of the wavelet coefficients.

References

[1] G. R. Ganger, "Generating representative synthetic workloads: An unsolved problem," *Proceedings of the Computer Measurement Group (CMG) Conference*, December 1995.

[2] A. V. Oppenheim, *Applications of Digital Signal Processing.* Prentice Hall, 1989.

[3] G. Strang and T. Ngugen, *Wavelets and Filter Banks.* Wellesley-Cambridge Press, 1996.

[4] W. Leland, M.S.Taqqu, W. Willinger, and D. Wilson, "On the self-similar nature of ethernet traffic," *IEEE/ACM Transactions on Networking*, vol. 2, 1994.

[5] M. T. J. Beran, R. Sherman and W. Willinger, "Long-range dependence in variable-bit-rate video traffic," *IEEE Trans. on Communications*, vol. 43, 1995.

[6] D. Duffy, A. McIntosh, M. Rosenstein, and W. Willinger, "Statistical analysis of ccsn/ss7 traffic data from working subnetworks," *IEEE J. Select. Areas in Communications*, vol. 12, 1994.

[7] V. Paxson and S. Floyd, "Wide-area traffic: The failure of poisson modeling," *IEEE/ACM Transactions on Networking*, 1995.

[8] W. Willinger, M. S. Taqqu, W. E. Leland, R. Sherman, and D. V. Wilson, "Self-similarity in high-speed packet traffic: Analysis and modeling of ethernet traffic measurements," *IEEE/ACM Transactions on Networking*, vol. 5, 1997.

[9] W. Willinger, M. S. Taqqu, and A. Erramilli, "A bibliographical guide to self-similar traffic and performance modeling for modern high-speed networks," *Stochastic Networks, F. P. Kelly, S. Zachary and I. Ziedins (eds.),Oxford University Press*, 1996.

[10] M. E. Crovella and A. Bestavros, "Self-similarity in world wide web traffic evidence and possible causes," *In Proceedings of the ACM SIGMETRICS 96*, 1996.

[11] S. Deng, "Empirical model of www document arrivals at access link," *In Proc. International Communications Conference – ICC'96, Dallas, Texas, USA*, 1996.

[12] S. D. Gribble, G. S. Manku, D. Roselli, E. A. Brewer, T. J. Gibson, and E. L.Miller, "Self-similarity in file systems," *in Proceedings of ACM SIGMETRICS 98*, 1998.

[13] K. Park, G. Kim, and M. Crovella, "On the effect of traffic self-similarity on network performance," *In Proc. SPIE International Conference on Performance and Control of Network Systems*, 1997.

[14] A. Mahanti, C.Williamson, and D. Eager, "Traffic analysis of a web proxy caching hierarchy," *IEEE Network Magazine: Special Issue of Web Performance*, 2000.

[15] V.N.Padmanabhan and L. Qiui, "The content and access dynamics of a busy web site: Findings and implications," *In the Proceedings of ACM SIGCOMM 2000, Stockholm, Sweden*, 2000.

[16] M. A. Tanner, *Tools for statistical inference, 3rd Edition.* Springer, 1996.

[17] D. Cox and D. Hinkely, *Theoretical Statistics.* Chapman and Hall, London, 1974.

[18] T. Mitchell, *Machine Learning.* McGraw Hill, 1997.

[19] P. Barford and M. Crovella, "An architecture for a www workload generator," *Proceedings of SIGMETRICS*, 1998.

[20] M. Wang and C. Faloutsos, "Datamining meets performance evaluation: Fast algorithms for modeling bursty traffic," *Manuscript*, February 2001.

[21] S. Giordano, S. Miduri, M. Pagano, F. Russo, and S. Tartarelli, "A wavelet-based approach to the estimation of the hurst parameter for self-similar data," *Proceedings of DSP'97*, July 1997.

[22] R. H. Riedi, M. S. Crouse, V. J. Ribeiro, and R. G. Baraniuk, "A multifractal wavelet model with application to network traffic," *IEEE Transactions on Information Theory*, vol. 45, April 1999.

[23] M. S. Taqqu, V. Teverovsky, and W. Willinger, "Estimators for long-range dependence: an empirical study," 1995.

[24] H. Abrahamsson, "Traffic measurement and analysis," *SISC Technical Report T99:05*, September 1999.

[25] D. B. Percival and A. T. Walden, *Wavelet Methods for Time Series Analysis.* Cambridge University Press, 2000.