Application of Multifractals in the Characterization of WWW Traffic

Abdullah Balamash and Marwan Krunz
Department of Electrical & Computer Engineering
University of Arizona
Tucson, AZ 85721

Abstract— **In this paper, we explore the viability of multifractal analysis in modeling the traffic generation process at a WWW server. In principle, a WWW traffic model can be used for generating representative WWW traces and in designing prefetching and cache replacement policies. Multifractal processes constitute a superset of monofractal (self-similar) processes. They are characterized by a time-dependent scaling law, which provides flexibility in describing irregularities that are localized in time. Riedi et al. [11] presented a multifractal process that can be fitted to empirical time series with an arbitrary autocorrelation function (ACF) and with an approximately lognormal marginal distribution. We use this model to *simultaneously* capture the temporal and spatial localities of WWW traffic. Furthermore, the popularity profile is captured by construction using the LRU (least recently used) stack and the popularity profiles of each file in the real trace. We classify files into several classes according to their popularity profile and model the stack distance of each class separately. Trace-driven simulations are used to study the performance of our model and contrast it with a previously proposed model.**

I. Introduction

The accelerating growth of World Wide Web (WWW) traffic over the Internet will soon stress to limit the switching and transmission capacities of the current networking infrastructure, leading to excessive response times for client requests. While the transmission bottleneck can be addressed through the deployment of new, ultra-speed fiber-based technology, the switching capacity will continue to be a major cause for network congestion. One approach to reduce the traffic volume is to deploy WWW caching within the network and at its peripherals (WWW servers and client browsers). To be of any practical benefit, the cache should only store the "most important" files. Determining which files are important and which ones should be flushed out in case of cache saturation is the primary function of a cache replacement policy.

Several caching policies have been used for WWW traffic [5], including the *least recently used* (LRU) and the *least frequently used* (LFU) policies. LRU evicts the least recently requested file. To some degree, this policy accounts for the *temporal locality* of WWW traffic. In contrast, LFU *partially* accounts for the popularity profile of the traffic by flushing out the least frequently used files (i.e., the least popular files). It is clear from this context that the performance of a cache replacement policy depends on its ability to exploit the intrinsic characteristics of WWW traffic, including the temporal locality and the popularity of the requested documents. However, in order to design such an ideal policy, one first needs to understand the characteristics of WWW traffic and incorporate them in a stochastic model that can be used for synthetic trace generation and for designing and testing cache replacement policies. A WWW traffic model can also be used in designing *prefetching* policies and in evaluating the delay and loss performance over a network.

This work was supported by the National Science Foundation under grants ANI 9733143 and CCR 9979310 and ANI 0095626.

In this paper, we apply the multifractal analysis due to Riedi [11] in modeling WWW traffic. The presented model captures the essential characteristics of WWW traffic, including the temporal and spatial localities as well as the popularity profile. For our purposes, we view the traffic as a stream of file objects (or documents) that is generated by a WWW (or proxy) server in response to clients requests. Note that a user request (a click on a URL pointer) could trigger the fetching of multiple objects that contain images, video clips, etc. Thus, one should expect the presence of strong spatial localities (cross-correlations between documents) within WWW traffic.

The modeling of WWW traffic has been previously addressed in several papers [1], [3], [6]. All of these works have suggested the use of the stack distance string, which is an equivalent form of representation of the reference string (stream of file objects). In [1], the authors use client-based traces to demonstrate the presence of long-range dependence (LRD) in WWW traffic. They provide a fractional ARIMA (F-ARIMA) LRD model for the stack-distance string of a client reference stream. Their model exhibits self-similarity, which is a characteristic of monofractal processes [9]. A disadvantage of this approach is that is requires transforming the Gaussian marginal distribution of the F-ARIMA model into a more appropriate one (e.g., lognormal). As explained in the next section, such a transformation distorts the general structure of the autocorrelation function of the modeled data (the spatial locality), although it may still retain the value of the H parameter. In [6] the authors relied on a measure called the *scaled stack distance* to capture the impact of short-term correlation. Their model incorporates the popularity profile by construction, but it does not capture the spatial locality.

The rest of the paper is organized as follows. Section II describes the WWW traffic properties. In Section III we give a brief overview of wavelet analysis and the multifractal wavelet model [11], which we use to approximate both the marginal distribution and the correlation structure of the empirical data. In Section IV we describe our model and in Section V we test its performance by simulation. Section VI concludes the paper.

II. WWW Traffic Properties

In [7], [13], [1], [3], the authors identified some of the important properties of WWW traffic that should be incorporated in a WWW model. These properties include temporal locality, spatial locality, and popularity.

Temporal locality measures the closeness in time between requests to the same file. In [13], [1], [3], it was suggested that this property can be captured (at least, in part) through the marginal distribution of the stack distance string. The stack distance string is a transformation of the reference string using the LRU stack. It is obtained as follows. Let $REF_t = \{r_1, r_2, ..., r_t\}$ be the reference string up to time t, where r_j is the name of the object (file) requested at time j, $j = 0, 1, 2, \ldots$. Note that REF_t may contain multiple instances of the same file. Suppose that there are n unique objects at the server. The LRU stack at time t is defined by the ordered sequence $LRU_t = (Obj_1, Obj_2, Obj_3, ..., Obj_n)$, where Obj_1 is the most recently requested object, Obj_2 is the second most recently requested object, and so on. If a request is subsequently made for an object, say Obj_i, the LRU stack is updated by moving Obj_i from its location in LRU_t to the top of the stack, i.e., $LRU_{t+1} = \{Obj_i, Obj_1, Obj_2, \ldots, Obj_{i-1}, Obj_{i+1}, \ldots, Obj_n\}$. The stack distance is defined as the distance (in number of objects) between the top of the stack and the original location of the object that has just been moved to the top of the stack.

Spatial locality measures the correlation between requests to different objects (e.g., if object A is requested, then there is a good chance that object B will be requested in the near future). Modeling this property can help in designing cache replacement policies that employ prefetching, which improves the hit ratio of the cache. Previous work [1] has shown that spatial locality can be captured (at least, in part) through the autocorrelation structure of the stack distance string. In [1] it was argued that the correlation structure of the stack distance string exhibits LRD behavior. To simultaneously model the marginal distribution (temporal locality) and the correlation structure (spatial locality) of the stack-distance string, the authors in [1] relied on the work in [10] that proves the invariance of the Hurst parameter to transformations of the marginal distribution of an LRD process. However, the proof is valid asymptotically (i.e., does not apply to the finite-lag autocorrelations) and only for Gaussian processes (e.g., fractional ARIMA).

Popularity refers to the overall likelihood of requesting a particular object. Previous work [4] has shown that the popularity of WWW objects follows a Zipf-like law, which states that the relative probability of requesting the ith most popular object is proportional to $1/i^{\alpha}$, where $0 < \alpha < 1$. The exponent α varies from trace to another. If $\alpha = 1$, then we say that the popularity profile follows Zipf's law in the strict sense [4].

III. Multifractal Model of Riedi et al.

We now describe Riedi et al.'s model [11], which is subsequently employed in characterizing the temporal and spatial localities of a WWW server traffic. More details about multifractals can be found in [12], [8]. Riedi et al.'s model relies heavily on the discrete wavelet transform. The idea behind the wavelet transform is to express a signal (time function) $X(t)$ by an approximated (smoothed) version and a *detail*. The approximation process is repeated at various levels (scales) by expressing the approximated signal at a given level j, say X_j, by a coarser approximation at level $j+1$, say X_{j+1}, and a detail D_{j+1}. At each scale, the approximation is performed through a *scaling function* $\phi(t)$, while the detail is obtained through a *wavelet function* $\psi(t)$. More formally, a wavelet expansion of the signal $X(t)$ is given by:

$$X(t) = \sum_k U_{J,k}\phi_{J,k}(t) + \sum_{j=J}^{\infty}\sum_k W_{j,k}\psi_{j,k}(t) \qquad (1)$$

where

$$W_{j,k} \stackrel{\text{def}}{=} \int_{-\infty}^{\infty} X(t)\psi_{j,k}(t)dt \qquad (2)$$

$$U_{j,k} \stackrel{\text{def}}{=} \int_{-\infty}^{\infty} X(t)\phi_{j,k}(t)dt \qquad (3)$$

and $\psi_{j,k}$ and $\phi_{j,k}$, $j, k = 0, 1, 2, \ldots$, are *shifted* and *translated* versions of the wavelet and scaling functions $\psi(t)$ and $\phi(t)$, respectively, and are given here by:

$$\psi_{j,k}(t) \stackrel{\text{def}}{=} 2^{-j/2}\psi(2^{-j}t - k) \qquad (4)$$

$$\phi_{j,k}(t) \stackrel{\text{def}}{=} 2^{-j/2}\phi(2^{-j}t - k). \qquad (5)$$

In (1), the index J indicates the coarsest scale (the lowest in detail). The coefficients $W_{j,k}$ and $U_{j,k}$ are called the wavelet and scale coefficients at scale j and time $2^j k$. Together, they define the discrete wavelet transform of the signal $X(t)$ (assuming that $\phi(t)$ and $\psi(t)$ are specified).

Several wavelet and scale functions have been used in the literature. In our work, we use the wavelet and scale functions of the Haar wavelet transform. As shown in [11], the Haar wavelet transform (specified by the coefficients $W_{j,k}$ and $U_{j,k}$ for all j and k) can be obtained recursively as follows (we adopt the same convention of [11] in which the higher the value of j, the better the approximation of the original signal):

$$U_{j,k} = \frac{U_{j+1,2k} + U_{j+1,2k+1}}{\sqrt{2}} \qquad (6)$$

$$W_{j,k} = \frac{U_{j+1,2k} - U_{j+1,2k+1}}{\sqrt{2}} \qquad (7)$$

To initialize the recursion, the values of $U_{j,k}$, $k = 0, 1, \ldots,$ at the highest value of j are taken as the empirical trace to be modeled. Figure 1 depicts the generation process of the scale coefficients (from top to bottom).

In order to generate synthetic traces with a given autocorrelation structure, the Haar transform is reversed by rewriting (6) and (7) as:

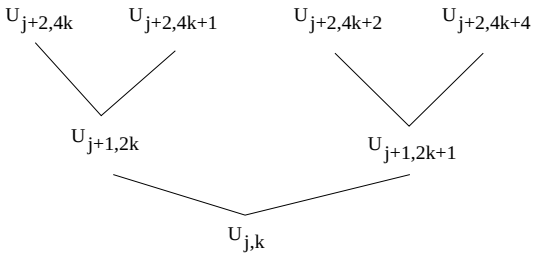

Fig. 1. Scaling coefficients from fine to coarse scales.

$$U_{j+1,2k} = \frac{U_{j,k} + W_{j,k}}{\sqrt{2}} \tag{8}$$

$$U_{j+1,2k+1} = \frac{U_{j,k} - W_{j,k}}{\sqrt{2}} \tag{9}$$

Now to generate nonnegative data, which represent stack distances, we need to have $|W_{j,k}| \leq U_{j,k}$. To satisfy this constraint, the wavelet coefficients can be defined as:

$$W_{j,k} = A_{j,k} U_{j,k} \tag{10}$$

where $A_{j,k}$ is a random variable (rv) defined on $[-1, 1]$. Using (8), (9), and (10), we arrive at the following recursion for synthesizing the scale coefficients:

$$U_{j+1,2k} = (\frac{1 + A_{j,k}}{\sqrt{2}}) U_{j,k} \tag{11}$$

$$U_{j+1,2k+1} = (\frac{1 - A_{j,k}}{\sqrt{2}}) U_{j,k} \tag{12}$$

The $A_{j,k}$ must also satisfy the following additional constraints:

1. The rvs $A_{j,k}, k = 0, 1, \ldots, 2^{j-1}$ are *i.i.d.*
2. Let A_j be a generic rv that has the same distribution as $A_{j,k}, k = 0, 1, ..., 2^{j-1}$. Then, for each j, A_j is symmetric with zero mean.
3. A_j is independent of A_l for $l > j$ and is also independent of $U_{0,0}$.

The wavelet energy at a given scale is defined as the variance of the wavelet coefficients at that scale. It has been shown that the correlation structure of the signal can be approximately captured by controlling the wavelet coefficient energy. The ratio of the energy at scale $j - 1$ to the one at scale j (j is finer than $j - 1$) was found to be [11]:

$$\eta_j = 2 \frac{E[A_{j-1}^2]}{E[A_j^2](1 - E[A_{j-1}^2])} \tag{13}$$

Equation (13) can be used to solve for $E[A_j^2]$, $j = 1, 2, \ldots$, given the initial value $E[A_0^2] = \frac{E[W_{0,0}^2]}{E[U_{0,0}^2]}$, where the values for $W_{0,0}$ and $U_{0,0}$ are obtained from the analysis part described earlier (Equations (6) and (7)). The marginal distribution for A_j is taken as beta with equal shape and scale parameters (other distributions can also be used). Thus, the random variable A_j is completely specified by $E[A_j^2]$.

It was shown in [11] that the above model generates positive-valued autocorrelated data with an approximately lognormal marginal distribution. In fact, the model can be tuned to produce any desired ACF. Since the stack-distance string is known to have a longnormal-like marginal distribution [7], [13], [1], [3], Riedi's model can be used to capture the temporal locality (marginal distribution of the stack-distance string) and spatial locality (ACF of the stack-distance string) in WWW traffic.

IV. Modeling Approach

In this section, we describe our approach for modeling the stream of file objects generated by a WWW server. Our modeling study was conducted using the WWW *server* trace that was captured at the Computer Science Department, University of Calgary [2]. We use a portion of this trace that consists of HTTP requests made over a period of two months and a half (from October 24, 1994 through December 10, 1994). In our analysis, we only include the requests with 'successful' code, since they are the ones that result in actual data transfer from the server. see [2] for more details of the collected trace.

Let U be the number of unique files (or objects) at the server, and let fr_i be the fraction of times that the ith file, $i = 1, 2, \ldots, U$, appears in the reference string (the popularity profile of file i). The modeling approach proceeds in three steps. First, we extract the stack-distance string from the URL reference string. The multifractal model described in the previous section is then applied to the stack-distance string to capture the temporal and spatial localities of the traffic. Finally, we incorporate the popularity profile of the traffic during the process of generating synthetic reference strings. These main steps are described next.

A. Extracting the Stack Distance String

In [3] stack distances are computed by first assuming an arbitrary initial ordering of the stack. Whenever a file is requested, it gets pushed to the top of the stack, and the distance from the top of the stack and the last location of this file (the stack depth) is recorded. The problem with this approach is that it depends on the initial ordering of the stack, which we have found to affect the marginal distribution and the correlation structure of the stack distance string. Instead, we use a different approach to extract the stack distances, which does not require specifying an initial ordering of files in the stack.

We start with an empty stack and scan the empirical reference string *in the reverse direction, starting with last reference*. During this scanning process, if a file is referenced for the first time (in the reverse direction), it is pushed to the top of the stack *but* its distance is not recorded. Otherwise, if the file has already been referenced before (hence, it is already in the stack), then it gets pushed from its previous location in the stack to the top of the stack, and its depth is recorded as a stack distance. Finally, the resulting trace of stack distances is

reversed to get the correct stack distance string. The following example illustrates the idea. Consider the reference string $[a\ d\ c\ b\ c\ d\ d\ a\ b]$, where each letter indicates the name of a file. If we process this string starting from the end, the first reference is to file b. Since this is the first time file b is being referenced, we push it to the top of the stack without recording any distance. The same procedure is performed for the next two references (for files a and d, respectively). The fourth reference (from the end) is for file d. Since this file has been referenced before, it gets pushed to the top of the stack *and* its stack depth is recorded (in this case, the stack depth for file d is one). The procedure continues until all references are processes (see Figure 2). The end result of this process is the stack distance stream $[4\ 3\ 2\ 4\ 1]$. The corresponding popularity profile is $[(a,3),(b,2),(c,2),(d,2)]$.

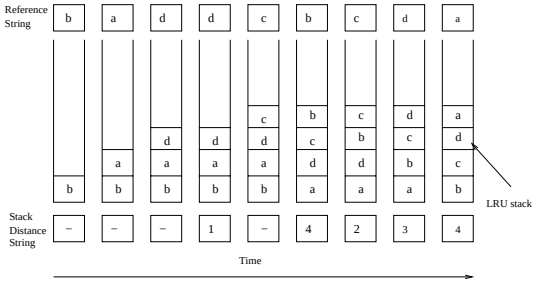

Fig. 2. Example showing the extracting of the stack distance string.

B. Modeling the Stack Distance String

Once the stack distance string is obtained, we fit it using the multifractal model in Section II. Since the marginal distribution of the empirical string is already captured, the fitting is focused on the ACF. Recall that for each scale j, the rv A_j, which is used to generate the scale coefficients, has a symmetric beta distribution with parameter ρ_j. To obtain the value of this parameter, we start from (13). At each scale, we need the variance of the wavelet coefficients, which can be found empirically. The lowest scale should be chosen so that it has enough wavelet coefficients to compute the variance (the zeroth scale has only one data point, which is not enough to compute the variance at that scale). Using the mean and variance of the scaling coefficients at the lowest scale, we can generate the normally distributed scaling coefficients, from which the the scaling coefficients at higher scales are computed using (11) and (12).

C. Generating Synthetic Traces and Capturing the Popularity Profile

Generating a WWW reference string follows the reverse process of extracting the stack distance string from the real data. The synthesis process starts by assuming an initial ordering of the files in the stack. This is done by sampling from a probability distribution that is weighted by the popularity profiles of the various documents (i.e., the more popular a document is, the more likely it will be placed closer to the top of the stack). Suppose that we want to generate an N-long synthetic trace that has a certain popularity profile. From N and the popularity profile, the number of requests to each file is known beforehand. Then, the stack distance string is generated as was described in the previous section. We associate a counter with each unique object. Initially, the counter has a value of N times the popularity index of the object. Every time the object is requested, the corresponding counter is decremented by one. If the counter reaches zero, the file is flushed out of the stack.

Despite its simplicity, the above approach did not yield good results in in terms of approximating the hit ratio at a cache, especially if we have files that are very popular compared to others. To address this problem, we propose a different approach in which objects are first classified into several classes according to their popularity profile. We then model the stack distance string for each class separately. Instead of selecting the file that is at the top of the stack to be the next requested object, we follow the approach in [3], where a small window is used near the top of the stack (e.g., first three objects from the top). The next object in the URL string is selected randomly from that window, according to a probability distribution that is weighted by the outstanding number of requests for each file.

The following example illustrates the trace generation process. Suppose that we have four unique files, a, b, c, and d, with popularity profile [(a,2),(b,2),(c,2),(d,3)]. Using the multifractal model, suppose that we obtained the the stack distance string [4 3 2 4 1]. Assume further that the initial order of the files in the LRU stack is [a b c d] (i.e., file 'a' is at the top of the stack) and that the window size is one. Since the window size is one, file 'a' is the first object in the reference string. Because the next stack distance is 4, file 'a' is pushed to the bottom of the stack. Now file b is at the top of the stack, so it gets added to the reference string. Since the next distance is 3, file b is pushed to the third position in the stack, and so on. When we reach a point where the file at the top of the stack is referenced for the last time (according to its popularity profile), this file is dropped from the stack. The generation process is shown in Figure IV-C.

V. Experimental Results

We evaluate the goodness of the multifractal model and contrast it with a self-similar (monofractal) model that uses transformation of the marginal distribution. Our performance metrics are the file and byte hit ratios for the LRU and LFU cash replacement policies. For each of these policies, the cache is fed by a stream of objects that is generated according to either model. The hit ratio under the real trace is used as a reference. Figures 4 and 5 depicts the results of our simulations for the LRU and LFU policies, respectively. As can be seen from these results, the multifractal-based WWW traffic model gives much better performance than the monofractal-based model.

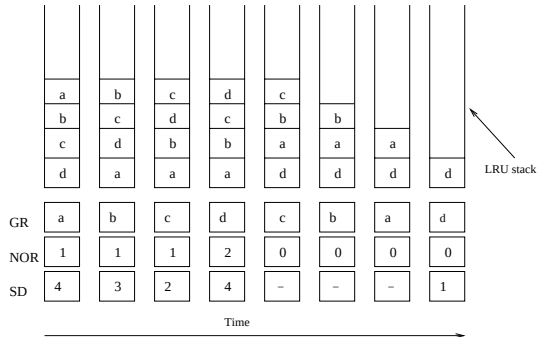

Fig. 3. Example showing the generation of synthetic traffic (GR: Generated Reference, NOR: # of Outstanding Requests, SD: Stack Distance).

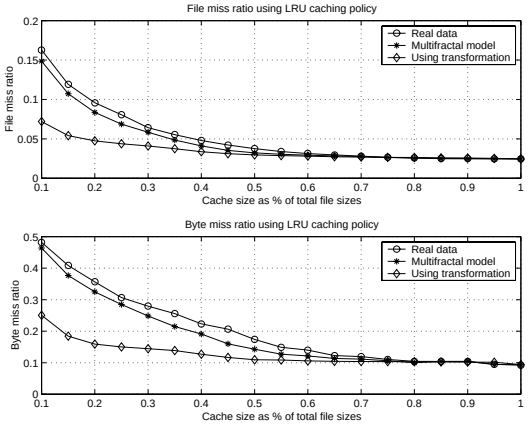

Fig. 4. File and byte miss ratios versus cache size for the LRU policy.

VI. Conclusions

In this work, we explored the viability of multifractal processes in modeling WWW traffic. We adapted Riedi et al.'s model [11] to capture the temporal and spatial localities of WWW traffic. The main advantage of this model is that it simultaneously captures the marginal distribution and ACF of the traffic. We provided a novel approach for extracting the stack distance string from the original URL reference string. Our approach does not require specifying an initial ordering for the objects in the stack. To capture the popularity profile, we classified objects into a number of classes and modeled each class independently. We found that this approach yields very good results in terms of the file and byte hit ratios at a cache, for both the LRU and LFU cache replacement policies.

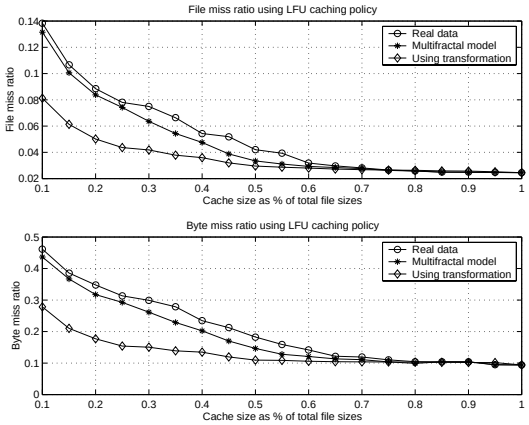

Fig. 5. File and byte miss ratios versus cache size for the LFU policy.

References

[1] V. Almeida, A. Bestavros, M. Crovella, and A. Oliverira. Characterizing reference locality in the WWW. In *Fourth International Conference on PDIS*, pages 92–103, 1996.

[2] M. Arlitt and C. Williamson. Web server workload characterization: The search for invariants. In *Proceeding of the ACM SIGMETRICS Conference*, pages 126–137, 1996.

[3] P. Barford and M. Crovella. Generating representative web workloads for network and server performance evaluation. In *Proceedings of the ACM SIGMETRICS Conference*, pages 151–160, 1998.

[4] L. Breslau, P. Cao, L. Fan, G. Philips, and S. Shenker. Web caching and zipf-like distributions: Evidence and implications. In *Proceedings of the INFOCOM Conference*, pages 126–134, 1999.

[5] P. Cao and S. Irani. Cost-aware WWW proxy caching algorithms. In *Proceedings of the 1997 USENIX Symposium on Internet Technology and System*, pages 193–206, 1997.

[6] L. Cherkasova and G. Ciardo. Characterizating temporal locality and its impact on web server performance. In *Proceedings of the Ninth International Conference on Computer Communication and Networks*, pages 434–441, 2000.

[7] C. Cunha, A. Bestavros, and M. Crovella. Characteristics of WWW client-based traces. *IEEE Trans on Networking*, 1(3):134–233, Jan 1999.

[8] J. Gao and I. Rubin. Multifractal analysis and modeling of long-range dependent traffic. In *IEEE International Conference on Communications*, volume 1, pages 382–386, 1999.

[9] A. Gillbert and W. Willinger. Data networks as cascades: Investigating the multifractal of internet WAN traffic. *IEEE Transactions on Information Theory*, pages 971–991, 1999.

[10] C. Huang, M. Devetsikiotis, I. Lambadaris, and A. R. Kays. Modeling and simulation of self-similar variable bit rate compressed video: A unified approach. In *Proceedings of the ACM SIGCOM Conference*, pages 114–125, 1995.

[11] R. Riedi, M. Crouse, V. Ribeiro, and R. Baraniuk. A multifractal wavelet model with application to network traffic. *IEEE Transaction ON Information Theory*, 45(3):992–1018, April 1999.

[12] R. H. Riedi. Introduction to multifractals. http://www.dsp.rice.edu/publications/.

[13] J. S. and A. Bestavros. Temporal locality in web request streams. In *Proceedings of the ACM SIGMETRICS Conference*, pages 110–111, 2000.