Computación y Sistemas Vol. 8 Núm. 2, pp. 119-131
© 2004, CIC-IPN, ISSN 1405-5546, Impreso en México

Routing with Wavelet-Based Self-Similarity Estimation

Ruteo con Estimación de Auto-Similaridad Utilizando Onduletas

Cesar Vargas-Rosales and Luis J. Manzanero
Center for Electronics and Telecommunications
ITESM Campus Monterrey, Monterrey, N.L., C.P. 64849, México,
E-mail: cvargas@itesm.mx, lmanzanero@alestra.com.mx

Article received on June 20, 2002, accepted on August 09, 2004

Abstract

The discovery of self-similar behavior in data traffic has initiated strong research in the area of traffic modeling. However, the way it affects the routing process is a subject not yet studied. This work presents the idea of providing intelligence to routers by estimating the Hurst Parameter using wavelets in a data link and uses this value as part of the routing metric. The motivation is to keep packets from using paths with high values of the Hurst Parameter, since in those paths the traffic tends to be bursty and therefore being more probable to drop packets and increase delay due to congestion. The algorithm is implemented for the Open Shortest Path First (OSPF) routing protocol in a simulation environment. Numerical results show that this technique is useful to provide traffic with low values of end-to-end delay and with smaller variance than those obtained when using traditional routing protocols.
Keywords: Self-similarity, Routing, Traffic.

Resumen

El descubrimiento de comportamiento auto-similar en el tráfico de datos, ha iniciado una enorme investigación en el área de modelado de tráfico. Sin embargo, la forma en la que afecta al ruteo en redes es un tema que todavía no se ha estudiado. Este trabajo presenta la idea de proporcionar inteligencia a los ruteadores por medio de la estimación del parámetro de Hurst utilizando onduletas en un enlace de datos, y utilizando ésta en la métrica de ruteo. La motivación es que los paquetes no utilicen trayectorias con altos valores del parámetro de Hurst, puesto que en esas trayectorias el tráfico tiende a contener ráfagas y a ser más probable que paquetes se pierdan y a que se incremente el retardo por congestión. El algoritmo es implementado para el protocolo de ruteo Open Shortest Path First (OSPF) en un ambiente se simulación. Resultados numéricos muestran que esta técnica es útil ya que proporciona tráfico con bajos niveles de retardo de extremo-a-extremo y con menor variación que aquellos valores obtenidos con el protocolo tradicional.
Palabras Clave: Auto-similaridad, Ruteo, Tráfico**.**

1 Introduction

During the last years, there has been a revolution in the mathematical modeling used to represent the behavior of data traffic in computer networks. The most used models in the *classical* traffic modeling theory have been based on the Poisson distribution which has a low correlation between two time intervals. This means that the traffic process has *low memory*, i.e., the traffic conditions at a specific time instant are not affected by the conditions at any other past time, and for practical terms this implies that the future behavior of the traffic can be estimated using only the present conditions, (Bertsekas, D., and R. Gallager, 1992). In the early 1990s, (Leland, *et. al.,* 1994), a traffic study showed that the traffic's behavior can better be modeled using self-similar distributions of probability. The parameter most commonly used to characterize the self-similarity is called the Hurst Parameter, and indicates how *strong* the traffic tends to appear in bursts.

In Leland, *et. al.,* (1994), the statistical analysis of Ethernet traffic measures during a four-year period is reported. They arrived to the conclusion that Ethernet traffic is statistically self-similar. Also, they proposed a

stochastic model by means of a renewal reward process through the aggregation of a sequence of independent identically distributed (iid) heavy-tailed random variables. In Willinger *et. al.,* (1997) and Willinger *et. al.,* (1998), the work is extended analyzing the traffic generated by individual sources or source-destination pairs of the ON/OFF type. They show that the aggregation of ON/OFF heavy-tailed sources produces *long-range dependence* and that the self-similarity observed in Ethernet traffic is governed by the ON or OFF period with the *heaviest* tailed distribution.

The implications of the self-similarity have not been completely appreciated, and there are areas where this aspect is not considered relevant when making a new model, an example is the routing process. Although there has been research about instabilities produced by several other factors (synchronization in the delivery of routing traffic as in Van Jacobson and Field 1994, software and hardware errors in the implementations of routing protocols Labovitz, *et. al.,* 1998) there have not been studies of the impact of self-similarity in the routing processes. As the network use grows, emerging applications such as e-business, real time video broadcasting and IP telephony require the implementation of new mechanisms to offer a better Quality of Service (QoS) to the end user. It is in this area, were the research of how the self-similarity affects the existing routing protocols can be useful to offer an adequate service.

It is now recognized that the self-similarity and the Long-Range Dependence (LRD) are intrinsic characteristics of data traffic, and that are generated by the network dynamics (like in TCP retransmission and flow-control procedures) or by the nature of the application (as in the video transmission).

Long-tailed distributions were found in many aspects related to computer networks. Crovella *et. al.*, (1997) reported that the distribution of user times followed long-tailed distributed periods of activity-inactivity when browsing on the web, and that the distributions of the size of the files being served typically were heavy-tailed; they also made comments on the results of a study where it was found that the file sizes of several Unix file systems were also heavy tailed. In Garrett and Willinger (1994) the authors arrived to the conclusion that the call holding time was better represented by a heavy tail distribution.

This work presents the idea of using the Hurst parameter as a routing metric to determine the cost of a link, i.e., lower values of the Hurst parameter (meaning lower levels of self-similarity and long-range dependence), yield a lower cost for the link, which translates into an increase usage of those portions of the network with low self-similarity. The technique is applied by measuring the self-similarity at the link layer level. Even though the estimation is carried out at the link layer and the estimated values are utilized at the network layer (where routing algorithms reside), we show that the benefits are quantified at the transport layer by TCP end-to-end delay.

In Section 2, we introduce the proposed technique using the Hurst parameter in the routing scheme, as well as the cost function used for each link in the shortest path algorithm that carries out OSPF. In Section 3, we discuss briefly the implementation of the technique as well as the Abry-Veitch estimator. In Section 4 the numerical results are presented and the conclusions afterwards.

2 The Hurst Parameter in the Link Metric

Most of the methods used to calculate the self-similarity parameters (specially the Hurst parameter) are adequate only to work in batch mode, meaning that it is necessary to capture a large set of data and process it offline to obtain the values of the parameters, (Beran 1994). Real time estimation of the self-similarity parameters is necessary in order to have effective tools to monitor and manage the networks at the same rate as the traffic is measured and also to allow making this estimation in network devices without the need to store in memory large amounts of data. A method which can be used is the Abry-Veitch (AV) estimator; see Abry and Veitch (1998), Roughan *et. al.,* (1998) and Veitch *et. al.,* (1999).

The AV estimator analyzes a set of traffic using wavelet transforms (Burrus *et. al.,* 1998; Graps, 1995), it specifically uses the discrete wavelet transform of scaling processes, (Abry *et. al.,* 2000). The signal (in this case, the measure of the traffic) is translated into a time-scale wavelet domain. Thus, the wavelet transform can be understood as a method for simultaneously observing a time series at a full range on different scales.

The family of wavelets to be selected needs to have certain conditions (described in detail in Abry and Veitch, 1998 and Veitch and Abry, 1999) and the most commonly used are called the Daubechies wavelets, (see Daubechies, 1992). The method calculates the inner product of the signal against the wavelet functions and samples the results at several points (known as the dyadic grid) after this; it makes a minimum square interpolation to obtain the slope of the data, α . Finally, the estimation of the Hurst parameter $\hat{H}$ is obtained by the relation $\hat{H} = (1+\alpha)/2$. The detailed description of the method is in the Appendix.

Erramilli *et. al.,* (1996), performed experimental analysis to estimate the impact of the self-similarity in the queue length. They fed several of the packet traces obtained in previous studies to a queue simulator and observed that the heavy tailness of the inter-arrival times produced longer delays that the predicted with the classical distributions even with low utilization levels. To show that this is a result of the long-range dependence in the traces, the researchers scrambled at random the arrival times splitting the sequence in blocks and performing an external shuffle (changing the order of the blocks preserving the internal order of the packets thus obtaining that the long-range dependencies are destroyed and short-range dependencies, SRD, are conserved), and also performing an internal shuffle (the blocks remain in their place and the inner packets are interchanged at random, where the SRD is gone and the LRD remains). The results for the external shuffle were similar for a queue with exponential inter-arrivals, while the internal shuffle showed very little difference with the original data.

The buffering requirements for a link under self-similar traffic are much higher than the requirements when the traffic is Poisson, and although it could be thought that the solution is just to increase the buffer size of the routers, this is not correct because the packets are still exposed to high queueing delays producing time-outs at the transport layer in the hosts and increasing the congestion in the network by retransmissions. Some authors (Erramilli *et. al.,* 1996; Willinger *et. al.,* 1997), consider that the emphasis to handle this performance problem should be in decreasing the network load instead of increasing the buffer capacity.

In a packet of a transport session, to be routed among several nodes, the situation could be more dramatic since along the route it will have to enter several queues, find some available buffer area and be transmitted and acknowledged in a time such that the Round Trip Time (RTT) is short enough to avoid unnecessary retransmissions. Therefore, a packet being routed among links with low self-similarity has a lower probability of being dropped and also a smaller RTT, since the routers in its path will have less occupancy in their queues. In this way the network layer can contribute to reduce the performance problems generated by the LRD, independently of any strategy implemented in higher layers (for example, LRD-aware flow control).

2.1 Proposed routing technique

The protocol at the routing layer is a standard routing protocol (OSPF, RIP, PNNI), see Huitema, (2000), modified to take the self-similarity information from the link layer and to calculate a cost for each link. This cost should be such that if the traffic tends to be *almost* Poisson $H \approx 0.5$ the routing process should then use the normal determination of the route (e.g., minimum number of hops). To measure the Hurst Parameter, the link outgoing traffic is used to make the estimation of the cost; this is because the packets to be sent will contend for buffer space with other packets going in the same direction. The estimation of the Hurst parameter is performed using the Abry-Veitch algorithm.

The cost of the links is updated in a regular interval to reflect the self-similarity. The link cost update interval is set to 60 seconds which is short enough to reflect the dynamic behavior of the network but long enough to avoid instabilities in the routing process. The link metric or cost to be used in OSPF, is given by

$$C = C_1 + 2(H - 0.5)C_2, \qquad 1 \le C \le 65{,}535. \tag{1}$$

If another protocol is used, a different cost function should be considered. Each term in Equation (1) represents a component of the cost; the first one, the C_1 term produces a hop by hop count to be obtained by the shortest path algorithm. The second term, $2(H - 0.5)C_2$, is the weight produced by the self-similarity. Therefore, we can control the influence of the Hurst Parameter in the weight of the link.

A Hurst parameter of *H*=0.5 (which means Poisson traffic) yields a cost of C_1, and a value of *H*=1.0 produces a cost of $C_1 + C_2$ with a maximum of 65,535; ($2^{16} - 1$ is the highest cost for a link in OSPF). Each router in the network updates its outgoing link cost computing (1). After calculating the costs of all the links, a Router Link State Advertisement (LSA) is generated and flooded into the area, and the shortest path is calculated using Dijkstra's algorithm. The value of C_2 is chosen by the network administrator. In Section 4, we present results for some cases.

We explore the stability in the routing by using the paths with less self-similarity on the links. At each interface, costs are updated in a regular interval and the routes recalculated, this could produce routing instability as traffic flips suddenly from one route to the other. This is avoided by not updating the costs and therefore the routes on real-time, only on intervals of 60 seconds each.

3 Implementation of the Abry-Veitch Estimator

To show the proposed routing technique, the Opnet Modeler system was used, (MIL3 Technologies), which is a development environment for the modeling of communication networks and distributed systems.

The main change to the code was in the process to perform the convolution in real-time. A filter bank such as that shown in Figure 1 was used, (Roughan, *et. al.,* 1998), it is a recursive *fast filter-bank-based* pyramidal algorithm, see Abry, *et. al.,* (2000), which helps us compute the discrete wavelet transform, and it also has a lower computational cost than that of a Fast Fourier Transform (FFT), (Daubechies, 1992). The fast pyramidal algorithm has a complexity of *O*(*n*) for data length *n*, and is simple enough to implement on-line and in real-time estimation in high speed networks, see Abry, *et. al.,* (2000).

Since the output of each stage of the filter bank is decimated, it is more efficient to decimate at the same time as the convolution instead of the conventional strategy to convolve and obtain the full result of the signal and after this decimate the result. Therefore, we avoid to process unnecessary values, which will be later discarded. This is performed using a shift-register; each of the stages requires two of these shift-registers because each stage performs two convolutions with different filters. The process starts initializing the shift registers with zeros and the vectors of the response function with the wavelet coefficients transformed as described in Section 2. After this, the program starts inputting data from the signal to process. The data is convolved with the response function by the mean of a multiplication with carry; this technique is derived from the way of performing a multiplication digit by digit of two numbers of arbitrary precisions in Chapter 20 of Press, *et. al.,* (1992), in which the fact that the convolution of two numbers is equal to the multiplication of these numbers digit by digit is used. Here the reverse operation is done, using the multiplication for performing the convolution sample by sample as it is being input into the filter and avoid the storage of large quantities of data. The result of the convolution of the first stage is taken as input to the next stage. The decimation is performed at the same time, and the counter is updated each time a value is entered to the filter, and only when the counter has an odd value the results are stored in an accumulator for each stage, storing the sum of the squares of the output of each stage. This is further explained in the Appendix.

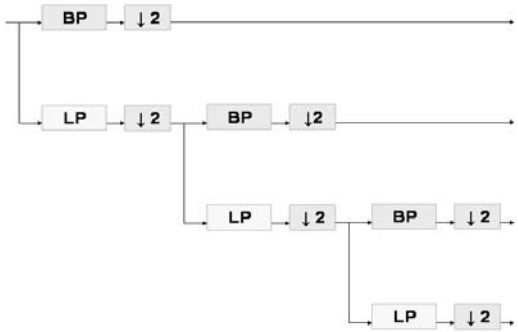

Fig. 1. Filter bank to estimate Hurst parameter

The routing model was implemented under OSPF in the Opnet Modeler working over Ethernet Local Area Networks. To support the isolation and function separation of each of the layers in a protocol stack and to make the implementation independent of the lower layers, we do not implement the self-similarity calculations in the OSPF layer; instead this is delegated to the link layer which puts the value available to the upper layers as a statistic. The estimation was performed to process the size of a packet in the self-similarity filter-bank as it is received. To speed up the simulation execution, the network nodes not involved in the routing process (end-hosts, hubs, layer-2 switches, etc.) implement the standard Ethernet module to avoid the overhead of an unnecessary calculation of the self-similarity, therefore in a running simulation there are two versions of the Ethernet module, the standard module on the non-routing hosts and the modified self-similar aware running on the routers. The modified Ethernet protocol calculates the self-similarity in a time interval controlled by the user and stores the result on a local statistic on each instance of the node.

During the entire simulation, we use time-slots of duration one second to accumulate the amount of bits that each outgoing link on each router transmits. After 60 seconds of keeping track of the traffic in bits per second, the information of this time series is processed as the following algorithm shows:

- **Step 1.** Process the gathered traffic information according to the steps of the AV wavelet estimator algorithm written in the Appendix, and obtain the Hurst parameter.
- **Step 2.** Compute the new link cost according to Equation (1) for every outgoing link in every router and pass this information to the network layer. Generate routing information packets (link state advertisements) and send them to all the neighboring routers.
- **Step 3.** Let Dijkstra's algorithm compute the shortest path routes using the new link costs and generate the new routing tables.
- **Step 4.** Continue normal operation accumulating traffic information and repeat this algorithm after 60 seconds.

4 Numerical Results

To test the model and obtain results, the networks shown in figures 2 and 3 were modeled in Opnet. On all these topologies, the simulation was executed three times, with the routers running standard OSPF, then the modified version of OSPF with $C_1 = 1$ and $C_2 = 65{,}534$ in Equation (1) (this case is named in the result tables as ``Self-Sim only'' - SS) and afterwards $C_1 = 10$ and $C_2 = 10$ (this case is referred as ``Self-Sim Weighted'' - SSW). In the SS scenario, the routers try to avoid those paths with self-similarity by setting a high cost to this characteristic, i.e., we give priority to the shortest path or minimum hop count. In the SSW scenario, the routers give priority treatment to the self-similarity over the shortest path since each link has at least a cost of 10, i.e., ($C_1 = 10$), and those links with low self-similarity ($H \approx 0.5$) will have lower costs. All the links in the network are 10BaseT links and the services simulated are FTP, Remote login and HTTP. The simulation time is of six hours.

The performance metric to be used is the TCP Delay in the servers located in the network diagram, but this could lead to an uneven comparison of two scenarios. The Delay by itself is not a good measure because a smaller delay in a scenario could have been artificially produced by a low throughput in the whole network. To make a fair comparison between two scenarios, we use the ratio of the sessions served by a computer to the average TCP delay of this equipment; in this way a model which either permits more connections or yields a smaller delay is going to produce a higher value of this ratio.

The network consists of four routers and a “medium load” traffic given by the number of stations and servers in it (as shown in Figure 2). The bottom router has a hub with three servers providing FTP, Remote Login and HTTP services each. Each of the other routers has an Ethernet hub connecting ten stations, which at random select a service and one of the servers and start a session to it.

When running the simulation, the model that gave the lowest delays across the servers was the SSW model. Besides this, the delay had a very low variance as it can be appreciated in figures 4, 5 and 6 where the graphics of the TCP delay for the SSW model is a smooth line instead of the spiked lines of the other two

models. The possible reason to the best performance of the SSW model against the others is due to the simple topology with few alternate paths.

In Figure 7, we can see the Hurst parameter estimated in router 45 of the network in Figure 2, as a function of the simulation time. We can see that only the link corresponding to port 4 has a Hurst parameter close to 0.5 at the end of the simulation, meaning that high amounts of traffic will be released over this link due to a lower self-similarity behavior and as a consequence a lower cost.

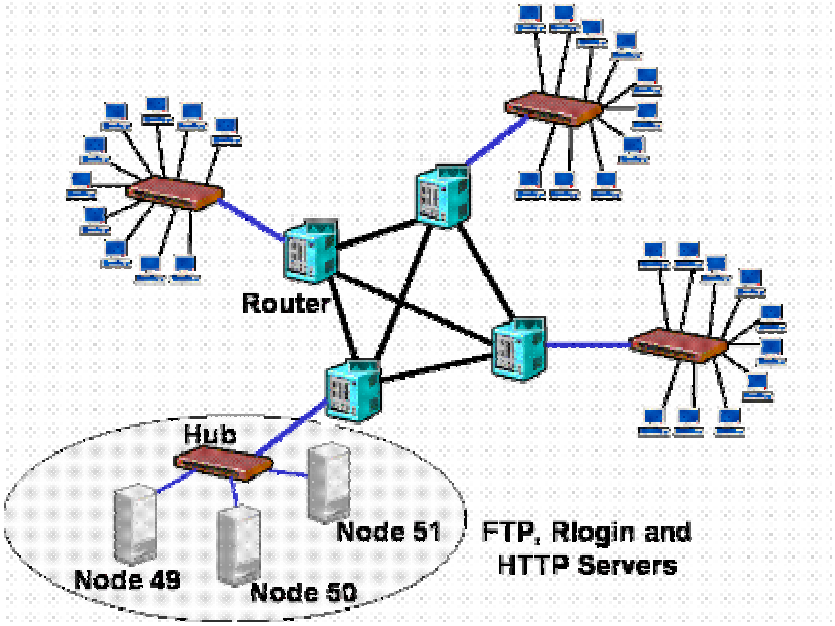

Fig. 2. Medium loaded network, topology one

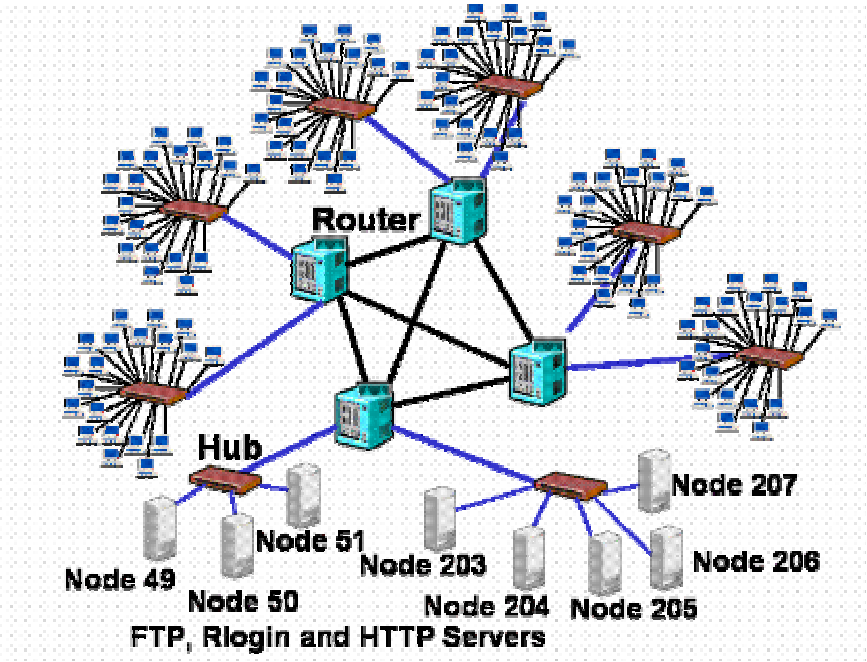

Fig. 3. Highly loaded network, full router mesh

We also run the simulations in a highly loaded network, containing four routers, eight servers and six Ethernet hubs each one containing twenty stations (Figure 3). In this case the SSW model also provided the best overall results. However, the results are not as definitive as in the previous case, therefore a future research could explore better values for the coefficients C_1 and C_2, and possible modifications for links under high load. In Table 1, we have the results for the TCP delay and number of sessions of the services simulated. As we can see, the TCP delay variance is smaller for SSW than for the other two methods. We can also see that for most of the cases, SSW outperforms the other methods. In Table 2, we can still see SSW with good performance, but now it is not as clear as for the case in Table 1. This is due only to the increase in traffic loading that makes most of the links have almost the same self-similarity and hence making the routing algorithm stay with one route.

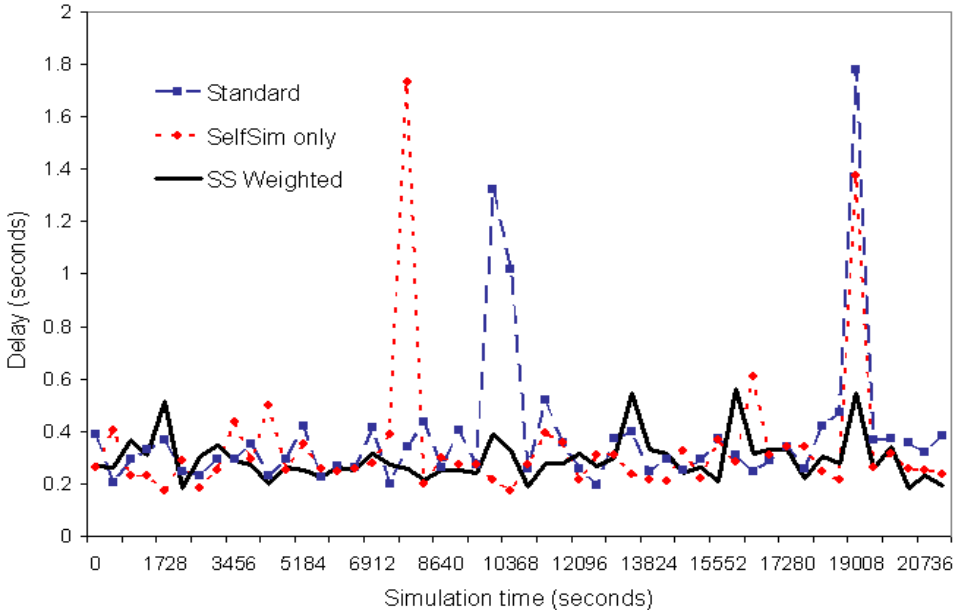

Fig. 4. TCP Delay for Node 49

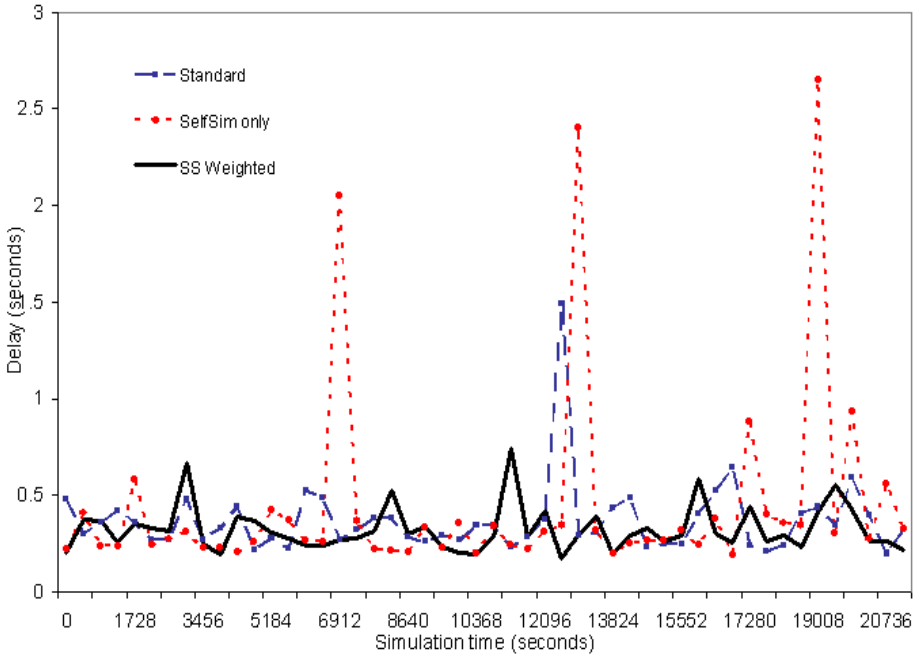

Fig. 5. TCP Delay for Node 50

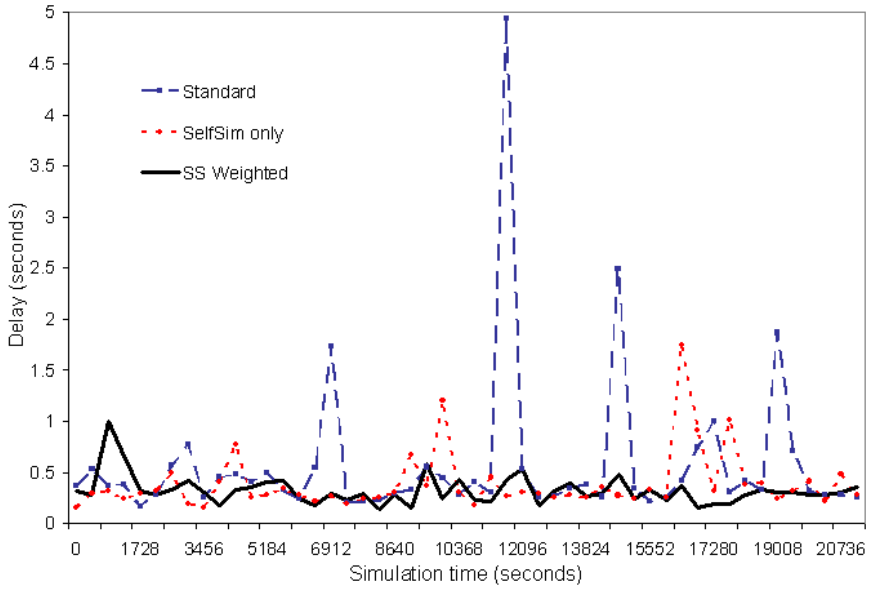

Fig. 6. TCP Delay for Node 51

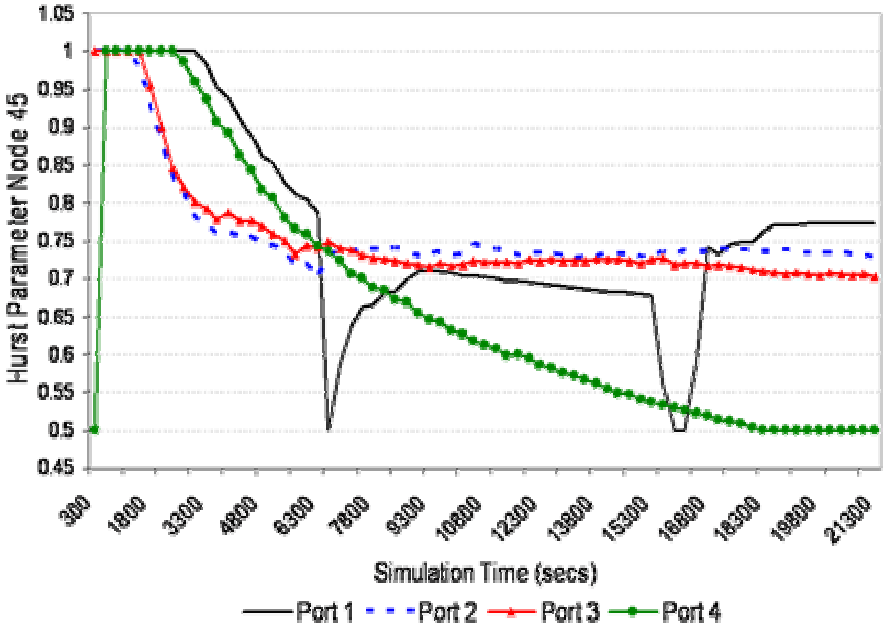

Fig. 7. Hurst parameter of the output ports of router node 45

5 Conclusions

We have introduced the Hurst parameter to estimate self-similarity of the traffic offered to routers. The estimation helps in the decision process of choosing the best shortest path based on self-similarity, since this is related to burstiness and eventually to effects on buffer occupancy. We introduced the Abry-Veitch algorithm in the outgoing links as an additional procedure within routers working with OSPF to estimate the Hurst parameter affecting the cost assigned to those links and as a consequence, the routing decisions. These decisions are benefited by the estimation of the Hurst parameter of the traffic that traverses the links of the routers and change link costs based on likelihood of traffic burstiness.

The method works well and the impact has been measured on TCP delay. We also introduced a measure that relates the number of sessions and the TCP delay to see that the routing of OSPF using Hurst parameter

estimation accommodates more sessions than the network working under standard OSPF. There are more issues to be studied such as performance when links fail, how to accommodate Quality of Service, how to introduce in the link metrics other measures that have self-similarity such as jitter, delay, packet loss, etc. Other issues would be to implement this method in ATM networks, to see how this could help MPLS, IGRP or RSVP and how it can work with traffic dispersion, multipath routing and multicasting.

Appendix. AV Estimator Algorithm

In this Appendix, we provide a brief description of the on-line AV wavelet estimator algorithm in Abry and Veitch (1998) and Roughan *et. al.,* (1998). Let $x(t)$ be the self-similar signal from which the parameters are to be estimated. In this case, the outgoing traffic in bits per second.

- Pass the input data by the filter bank shown in Figure 1. The number of stages of the filter is known as the number of octaves. The length of the response function of the bandpass function is the same as that of the low pass filter and it is known as the number of moments; the values of this vector are obtained from the coefficients of the wavelet family used, and for the band pass filter the coefficients are those of the wavelet with the sign of the elements in the odd position exchanged, while for the low pass filter the elements are the original wavelet coefficients transformed by flipping the original vector (i.e., swap the element in the *n*-th position with that at position one, the element in the (*n*-2)-nd position with that at position two and so on).
- The result of the convolution of the bandpass function and the signal is down-sampled (that is, the elements in the even positions are discarded) and the resulting vector is called the detail of the octave j, $d_x(j,i)$ with length len_j. Make $S_j \leftarrow S_j + d_x(j,i)^2$ where $i = 1,\ldots,len_j$. Make $n_j \leftarrow n_j + len_j$. Continue introducing samples from the signal. The previous steps are performed on the arrival of any sample of the signal, while the next are only performed on a longer time scale, when the Hurst parameter wants to be estimated, in our case 60 seconds.
- Calculate for all the stages the following expressions

$$\mu_j = \frac{1}{n_j}\sum_k d_j^2[k], \tag{2}$$

$$\sigma_j^2 = \varsigma\left(\frac{n_j}{2}\right), \tag{3}$$

$$g_j = \frac{\psi(n_j/2)}{\ln(2)} - \frac{\log_2(n_j)}{2}, \tag{4}$$

$$y_j = \log_2(\mu_j) - g_j, \tag{5}$$

where $\varsigma(x)$ is the Riemann-Zeta function $\varsigma(x) = \sum_{n=1}^{\infty}(1/n^2)$, see Zwillinger, (1987); $\psi(x) = \Gamma'(x)/\Gamma(x)$ is the Psi function (also called the digamma function) see Korn, page 823, (1968) and $\Gamma(x) = \int_0^\infty t^{x-1}e^{-t}dt$ is the gamma function with derivate $\Gamma'(x)$.

- Calculate the next sums:

$$Sum_0 = \sum_j \frac{1}{\sigma_j^2}, \tag{6}$$

$$Sum_1 = \sum_j \frac{j}{\sigma_j^2}, \tag{7}$$

$$Sum_2 = \sum_j \frac{j^2}{\sigma_j^2}. \tag{8}$$

- For each value of j calculate

$$w_j = \frac{jSum_0 - Sum_1}{\left(Sum_0 Sum_2 - Sum_1^2\right)\sigma_j^2}. \tag{9}$$

- Calculate

$$\hat{\alpha} = \sum_j w_j y_j \tag{10}$$

- And finally, the estimation of the Hurst Parameter $\hat{H}$ is given by

$$\hat{H} = \frac{1+\hat{\alpha}}{2}. \tag{11}$$

Table 1: Numerical Results for Medium Loaded Network: Topology One

	Average TCP Delay (secs)			**TCP Delay Variance**		
Server	**STD**	**SS**	**SSW**	**STD**	**SS**	**SSW**
49	0.383	0.340	0.295	0.074	0.069	0.008
50	0.369	0.445	0.324	0.036	0.261	0.014
51	0.579	0.386	0.324	0.574	0.081	0.014
	FTP Sessions			**Ratio FTP Sessions/Delay**		
Server	**STD**	**SS**	**SSW**	**STD**	**SS**	**SSW**
49	17.24	13.97	16.52	45.04	41.13	56.03
50	22.27	21.02	20.75	60.40	47.28	63.97
51	21.59	23.39	18.63	37.30	60.63	57.47
	HTTP Sessions			**Ratio HTTP Sessions/Delay**		
Server	**STD**	**SS**	**SSW**	**STD**	**SS**	**SSW**
49	1,151.45	978.30	1,092.39	3,008.62	2,881.02	3,705.63
50	1,593.35	1,505.17	1,418.80	4,321.56	3,384.71	4,373.16
51	1,480.32	1,548.75	1,354.52	2,557.04	4,014.69	4,178.17
	Rlogin Sessions			**Ratio Rlogin Sessions/Delay**		
Server	**STD**	**SS**	**SSW**	**STD**	**SS**	**SSW**
49	N/A	N/A	N/A	N/A	N/A	N/A
50	11.42	10.78	11.16	30.98	24.25	34.40
51	13.41	13.16	11.21	23.17	34.11	34.56

Table 2: Numerical Results for Highly Loaded Network: Topology One

	Average TCP Delay (secs)			TCP Delay Variance		
Server	**STD**	**SS**	**SSW**	**STD**	**SS**	**SSW**
49	**0.473**	**0.572**	**0.493**	**0.257**	**0.382**	**0.309**
50	**0.518**	**0.364**	**0.632**	**0.570**	**0.047**	**0.651**
51	**0.754**	**0.722**	**0.480**	**0.622**	**0.569**	**0.184**
204	**0.727**	**1.079**	**0.564**	**1.324**	**3.191**	**0.137**
205	**0.596**	**0.828**	**0.952**	**0.159**	**0.995**	**1.503**
206	**0.765**	**0.851**	**0.635**	**1.324**	**3.191**	**0.137**
207	**0.662**	**0.860**	**0.662**	**1.758**	**1.251**	**0.317**
208	**0.553**	**0.580**	**0.780**	**0.251**	**0.687**	**1.467**
	FTP Sessions			**Ratio FTP Sessions/Delay**		
Server	**STD**	**SS**	**SSW**	**STD**	**SS**	**SSW**
49	**24.160**	**27.405**	**24.799**	**51.063**	**47.886**	**50.286**
50	**16.259**	**17.636**	**15.951**	**31.417**	**48.446**	**25.252**
51	**27.932**	**29.913**	**30.278**	**37.054**	**41.449**	**63.049**
204	**26.266**	**30.031**	**28.109**	**36.134**	**27.826**	**49.803**
205	**32.626**	**27.564**	**32.626**	**54.776**	**33.307**	**34.287**
206	**27.723**	**27.564**	**32.626**	**36.245**	**32.373**	**51.355**
207	**29.158**	**27.934**	**29.384**	**44.052**	**32.487**	**44.393**
208	**23.929**	**24.305**	**22.286**	**43.253**	**41.876**	**28.576**
	HTTP Sessions			**Ratio HTTP Sessions/Delay**		
Server	**STD**	**SS**	**SSW**	**STD**	**SS**	**SSW**
49	**1,919.94**	**1,639.10**	**1,858.67**	**4,057.89**	**2,864.13**	**3,768.93**
50	**1,318.75**	**1,304.18**	**1,171.33**	**2,548.11**	**3,582.61**	**1,854.33**
51	**1,817.00**	**1,962.81**	**2,535.096**	**2,410.43**	**2,719.78**	**5,279.00**
204	**1,758.84**	**2,147.83**	**1,758.84**	**2,419.66**	**1,990.12**	**3,116.35**
205	**2,184.59**	**1,818.75**	**1,879.96**	**3,667.76**	**2,197.69**	**1,975.69**
206	**2,039.64**	**1,860.29**	**2,074.66**	**2,666.63**	**2,184.82**	**3,265.60**
207	**1,901.11**	**2,246.40**	**1,920.35**	**2,872.19**	**2,612.51**	**2,901.26**
208	**1,826.42**	**1,651.39**	**1,517.02**	**3,301.35**	**2,845.20**	**1,945.14**
	Rlogin Sessions			**Ratio Rlogin Sessions/Delay**		
Server	**STD**	**SS**	**SSW**	**STD**	**SS**	**SSW**
49	**N/A**	**N/A**	**N/A**	**N/A**	**N/A**	**N/A**
50	**2.98**	**3.53**	**3.46**	**5.76**	**9.70**	**5.47**
51	**4.14**	**6.94**	**6.69**	**5.49**	**9.62**	**13.94**
204	**4.37**	**5.68**	**4.37**	**6.01**	**5.26**	**7.75**
205	**6.70**	**6.17**	**4.79**	**11.26**	**7.46**	**5.03**
206	**7.10**	**4.93**	**7.76**	**9.29**	**5.79**	**12.22**
207	**3.74**	**6.81**	**5.63**	**5.65**	**7.92**	**8.50**
208	**4.67**	**4.50**	**3.10**	**8.44**	**7.76**	**3.97**

References

1. **Abry, P., and Veitch, D.**, "Wavelet Analysis of Long-Range-Dependent Traffic," *IEEE Transactions on Information Theory*, Vol. 44, No. 1, January 1998, pp. 2-15.
2. **Abry, P., Veitch, D., M.S. Taqqu and D. Veitch,** "Wavelet for the Analysis, Estimation, and Synthesis of Scaling Data," *Self-Similar Network Traffic and Performance Evaluation,* Kihong Park and Walter Willinger editors, pp. 39-88, John Wiley and Sons 2000.
3. **Beran, J.**, *Statistics for Long-Memory Processes*, Chapman and Hall, 1994.
4. **Bertsekas, D., and R. Gallager**, *Data Networks*, Prentice-Hall, Second Edition, 1992.
5. **Burrus, C. Sidney, Ramesh A. Gopinath, and Haitao Guo**, *Introduction to Wavelets and Wavelets Transforms, a Primer,* Prentice Hall, 1998.
6. **Crovella, Mark E. and Azer Bestavros**, "Self-Similarity in World Wide Web Traffic: Evidence and Possible Causes," *IEEE/ACM Trans. on Networking*, Vol. 5, No. 6, Dec. 1997, pp. 835-846.
7. **Daubechies, I.**, Ten lectures on Wavelets, SIAM Philadelphia, 1992.
8. **Erramilli, A., Onuttom Narayan, and Walter Willinger**, "Experimental Queueing Analysis with Long-Range Dependent Packet Traffic," *IEEE/ACM Transactions on Networking*, Vol. 4, No. 2, April 1996, pp. 209-223.
9. **Garrett, M. W., and W. Willinger.** "Analysis, Modeling and Generation of Self-Similar VBR Video Traffic," *In Proc., SIGCOMM94 Conference*, pp. 269-280, London, UK, August 1994.
10. **Graps, A.**, "An Introduction to Wavelets," *IEEE Computational Science and Engineering,* Vol. 2, No. 2, 1995, pp. 50-61.
11. **Huitema, C.,** *Routing in the Internet*, Prentice Hall, Second Edition, 2000.
12. **Korn, Granino A. and Theresa M. Korn,** *Mathematical Handbook for Scientists and Engineers: Definitions, Theorems, and Formulas for Reference and Review*, McGraw-Hill, Second Edition, 1968.
13. **Labovitz, C., G. Robert Malan, and Farnam Jahanian**, "Internet Routing Instability," *IEEE/ACM Transactions on Networking*, Vol. 6, No. 5, October 1998, pp. 515-528.
14. **Leland, Will E., Murad S. Taqqu, Walter Willinger, and Daniel V. Wilson,** "On the Self-Similar Nature of Ethernet Traffic (Extended Version)," *IEEE/ACM Transactions on Networking*, Vol. 2, No. 1, February 1994, pp. 1-15.
15. **MIL3 Technologies,** *Opnet Modeling Concepts*, 6th. edition.
16. **Press, William H., Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery,** *Numerical Recipes in C,* 2nd. edition, Cambridge University Press, 1992.
17. **Roughan, M., Darryl Veitch, and Patrice Abry,** "On-Line Estimation of Parameters of Long-Range Dependence," *In Proc. of the GLOBECOM'98*, Sidney, November 1998, pp. 3716-3721.
18. **Van Jacobson and Sally Floyd.** "The Synchronization of Periodic Routing Messages," *IEEE/ACM Transactions con Networking*, Vol. 2, No. 2, April 1994.
19. **Veitch, D. and Abry, P.**, "A Wavelet Based Joint Estimator of the Parameters of Long-Range Dependence," *IEEE Trans. on Information Theory*, Vol. 45, No. 3, April 1999, pp. 878-897.
20. **Willinger, Walter, Murad S. Taqqu, Robert Sherman, and Daniel V. Wilson**, "Self-similarity Through High-Variability: Statistical Analysis of Ethernet LAN Traffic at the Source Level," *IEEE/ACM Transactions on Networking*, Vol. 5, No. 1, February 1997, pp. 71-86.
21. **Willinger, Walter, V. Paxson and Murad S. Taqqu,** "Self-similarity and Heavy Tails: Structural Modeling of Network Traffic," *A Practical Guide to Heavy Tails,* Birkhauser, 1998.
22. **Zwillinger, D.,** editor, *CRC Standard Mathematical Tables and Formulae*, CRC Press, Third Edition, 1987.

César Vargas Rosales. *He received the B.Sc. degree in Mechanical and Electrical Engineering from the National Autonomous University of Mexico (UNAM) in 1988, the M.Sc. and the Ph.D. in Electrical Engineering from Louisiana State University (LSU), Baton Rouge, Louisiana, U.S.A. in 1992 and 1996, respectively. During 1988 – 1990, he was a professor at Universidad Anahuac del Sur in the School of Engineering and the School of Informatics and he worked as a Control Engineer at SIMEX, Integración de Sistemas. From 1992 to 1996, he was a research assistant at LSU in the area of adaptive routing schemes. Since 1996, he has been in the Center for Electronics and Telecommunications at ITESM-Campus Monterrey, Monterrey, Mexico where he is an Associate professor. His main research interests are on Personal Communications Networks, mobility modeling, traffic modeling, multiple accesses and scheduling in wireless networks, routing and QoS in wireless ad-hoc networks. He is a member of the Sistema Nacional de Investigadores in México.*

Luis Jorge Manzanero. *He was born in Mérida, Yucatán, México. He obtained the degree of Computer Systems Engineer from the Instituto Tecnológico de Mérida in 1996 and received the Master of Science in Information Technology from ITESM Campus Monterrey in 2001, where he developed an algorithm to estimate the Hurst parameter in Internet routers using OSPF. He currently works as a network planning engineer at Alestra AT&T, Monterrey, México. His interests are routing, network design and performance, parallel and distributed computing and system and network management.*