Switch Performance with Fractal Traffic

Anqing Xing

Supervisors: W. Grassmann and C. McCrosky

Abstract

Poisson and Bernoulli models are often used to simulate network traffic for analytic simplicity, even though several studies have shown that some real network traffic is fractal (self-similar). This research describes an algorithm for generating fractal traffic. The generated fractal series are then used as load in simulations of two types of switches (i.e., the hypercube switch and the omega switch). Simulations are performed for both infinite and fixed buffer sizes. The performance of two types of switches are evaluated under both fractal and non-fractal network traffic. Simulation results show that fractal network traffic results in more cell loss than non-fractal traffic and that conventional network models (e.g., Poisson arrival model) typically underestimate the required buffer size for fractal traffic. It turns out that the hypercube switch performs better (in terms of less cell loss and low latency) than the omega switch. It is also found that, with a fractal traffic input to an omega switch, the aggregated output traffic remains fractal but with a reduced degree of self-similarity. The degree of self-similarity of the output traffic can be significantly reduced if the output port of each cell is randomly selected.

1 Introduction

When modeling network traffic, cell arrivals are often assumed to be Poisson or Bernoulli processes because such processes have attractive theoretical properties. Poisson and Bernoulli processes are quite limited in their burstiness, especially when multiplexed to a high degree. Recent studies show that network traffic is statistically self-similar and none of the commonly used traffic models is able to capture this fractal behavior. Real network traffic is much burstier than the Poisson models predict, over many time scales. As reported in [1], a strong self-similarity of Ethernet traffic is observed. Aggregating such traffic intensifies the self-similarity (or burstiness) instead of smoothing it. Similar results were reported on wide area network traffic [2]. In communications, telephone call holding times and frame sizes for variable-bit-rate (VBR) video can be better modeled using heavy-tailed distributions [3]. Fractal traffic behavior has serious implications for the design, control and analysis of high speed networks. Further understanding of switch performance under fractal models is essential.

In this study, two types of switches are simulated using both fractal and non-fractal traffic. Since a Poisson model can be approximated by a Bernoulli model, the latter will be used to generate non-fractal traffic. The next section describes the algorithm we use for generating fractal series. Three commonly used methods for testing the degree of self-similarity of a given series are briefly discussed. The traffic generated by the adopted algorithm is tested using these methods. Section 3 uses both fractal and non-fractal traffic as input for simulations of an omega switch. With Bernoulli traffic, the cell loss probability of a bufferless omega switch is obtained analytically. The analytical result is in good agreement with the simulation results. The performance results of a hypercube switch with fractal and non-fractal traffic is reported in Section 4. Conclusions are given in Section 5.

2 Fractal Traffic Generation

There are several methods for generating fractal series that can account for self-similarity in real network traffic. Among them is a process called fractional differencing. Fractional differencing is a generalization of an ordinary difference operator. In general, the ordinary difference operator is defined as

$$\nabla^n X_k = \sum_{i=0}^{n} \begin{pmatrix} n \\ i \end{pmatrix} (-1)^i X_{k-i}, \qquad n = 1, 2, ... \tag{1}$$

For example, when $n = 1$ and 2, $\nabla^1 X_k = (X_k - X_{k-1})$ and $\nabla^2 X_k = (X_k - X_{k-1}) - (X_{k-1} - X_{k-2})$ $= X_k - 2X_{k-1} + X_{k-2}$. If we allow n to be fractional, the generalized fractional differencing operator has the form

$$\nabla^d X_k = \sum_{i=0}^{\infty} \begin{pmatrix} d \\ i \end{pmatrix} (-1)^i X_{k-i}, \qquad -0.5 < d < 0.5 \tag{2}$$

The fractional binomial coefficient is related to the Gamma function as follows.

$$\begin{pmatrix} d \\ i \end{pmatrix} (-1)^i = \frac{\Gamma(-d+i)}{\Gamma(-d)\Gamma(i+1)} \tag{3}$$

The algorithm used here for generating fractal series $\{X_k\}$ was first proposed by Hosking [4] and can be outlined as follows. X_0 is chosen from the normal distribution $N(0, v_0)$. Set $N_0 = 0$ and $D_0 = 1$. Then generate n points by iterating the following for $k = 1, 2, ..., n$.

$$N_k = \rho_k - \sum_{j=1}^{k-1} \phi_{k-1,j}\rho_{k-j} \tag{4}$$

$$D_k = D_{k-1} - N_{k-1}^2 / D_{k-1} \tag{5}$$

$$\phi_{kk} = N_k / D_k \tag{6}$$

$$\phi_{kj} = \phi_{k-1,j} - \phi_{kk}\phi_{k-1,k-j} \qquad j = 1, 2, ..., k-1 \tag{7}$$

$$m_k = \sum_{j=1}^{k} \phi_{kj} X_{k-j} \tag{8}$$

$$v_k = (1 - \phi_{kk}^2) v_{k-1} \tag{9}$$

Choose each X_k from the normal distribution $N(m_k, v_k)$. The autocorrelation function has an asymptotically hyperbolic shape, and is given by

$$\rho_k = \frac{d(1+d)...(k-1+d)}{(1-d)(2-d)...(k-d)} = \frac{\Gamma(1-d)\Gamma(k+d)}{\Gamma(d)\Gamma(k+1-d)} \tag{10}$$

As $k \to \infty$, the above autocorrelation function can be approximated by the following hyperbolic function of k and d.

$$\rho_k \approx \frac{\Gamma(1-d)}{\Gamma(d)} k^{2d-1} \tag{11}$$

This algorithm requires $O(n^2)$ computation time. It algorithm takes about 10 hours on a current workstation to produce 200,000 points. A sample fractal series of 10,000 points obtained using the above algorithm is shown in Figure 1.

There are three commonly used methods to test the degree of self-similarity. The first one is the auto-correlation test. For a given series, X_k $(k = 1, 2, ...)$ with a mean value of $\overline{x}$, the auto-correlation for lag k, ρ_k is defined as $\rho_k = E(X_i - \overline{X})(X_{i+k} - \overline{X})/S^2$ (where S^2 is the variance of the series). The auto-correlation of a series is the measure of the long range dependence between values in a the series at different lags. Clearly, if $k = 0$, $\rho_0 = 0$. A Poisson series exhibits very little dependence between

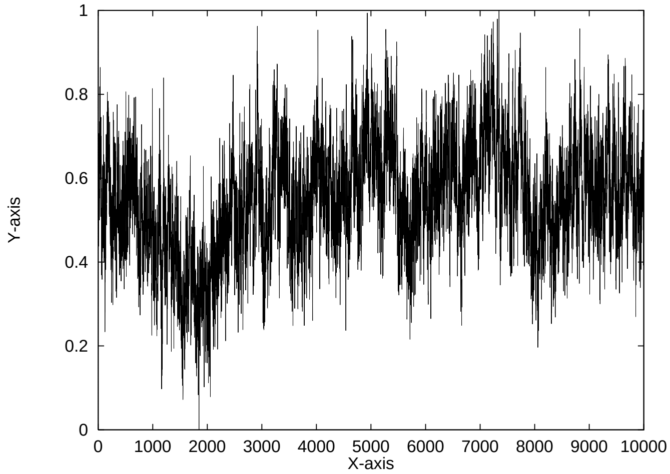

Figure 1: A Generated Fractal Series

values and, therefore, its auto-correlations decrease exponentially as k increases. A fractal series has auto-correlations which decay as a hyperbolic function (i.e., as $k^{-\alpha}$, as $k \rightarrow \infty$, $0 < \alpha < 1$) rather than a negative exponential (i.e., as ρ^k, as $k \rightarrow \infty$, $0 < \rho < 1$). The auto-correlation of the fractal series shown in Fig. 1 is plotted in Fig. 2 which shows a strong long range dependence.

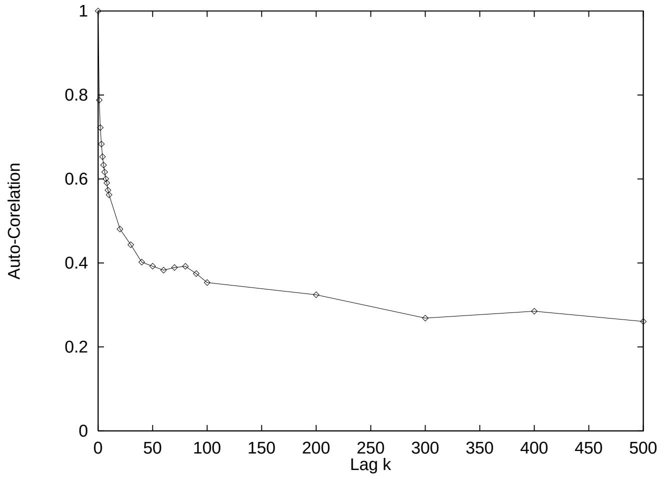

Figure 2: Auto-Correlation vs. Lag

The second method for testing self-similarity is the Hurst effect. Briefly, for a given series, X_k ($k = 1, 2, ..., n$) with sample mean $\overline{X}(n)$ and sample variance $S^2(n)$, the rescaled adjusted range statistic (or R/S statistic) is given by $R(n)/S(n) = 1/S(n)[max(0, W_1, ..., W_n) - min(0, W_1, ..., W_n)]$, with $W_k = (X_1 + ... + X_k) - k\overline{X}(n)$ ($k \geq 1$). Clearly, the R/S statistic is a function of n. A series with short range dependence can be represented by the relation $E[R(n)/S(n)] = cn^H$ with $H = 0.5$, as $n \rightarrow \infty$ (where c is a constant). Fractal series have H values in the range of $0.5 < H < 1$. Figure 3 shows the R/S values of the fractal series shown in Figure 1. The estimated H value is 0.775. Both Fig. 2 and Fig. 3 suggest that the generated series shown in Fig. 1 is self-similar, which will also be graphically proved below.

The third method to measure the degree of self-similarity is by testing the variances of the aggregated series for different aggregation levels. An aggregated series with aggregation level m is obtained by averaging the original series over non-overlapping blocks of size m. For a fractal series, the variance

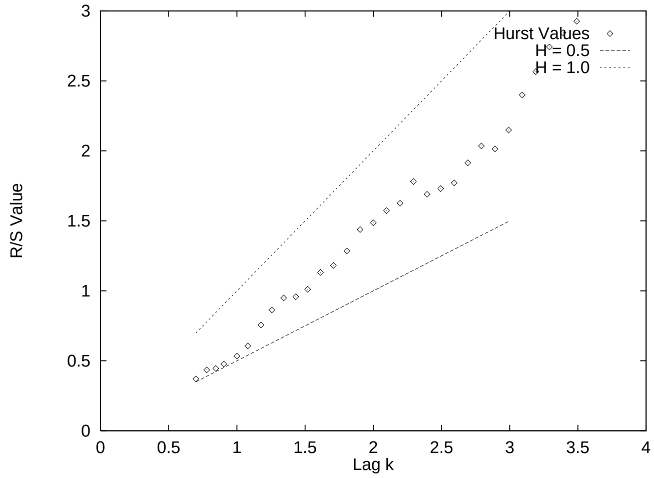

Figure 3: R/S Value vs. Lag (Estimated Hurst value: 0.775)

of the aggregated series decreases as a hyperbolic function ($S_m^2 = S^2 m^\beta$ as $m \to \infty$). Figure 4 shows that the aggregated series (with a block size of 10) remains bursty and has a shape similar to the original series (see Figure 1). This graphically shows that the generated series is self-similar. Furthermore, the aggregated series is also long range dependent (the plot is not shown due to space limitations). The Hurst values of the aggregated series are not calculated due the small number (1000) of points in the series. It is also found (by experiment) that the sum of several independent fractal series remains fractal.

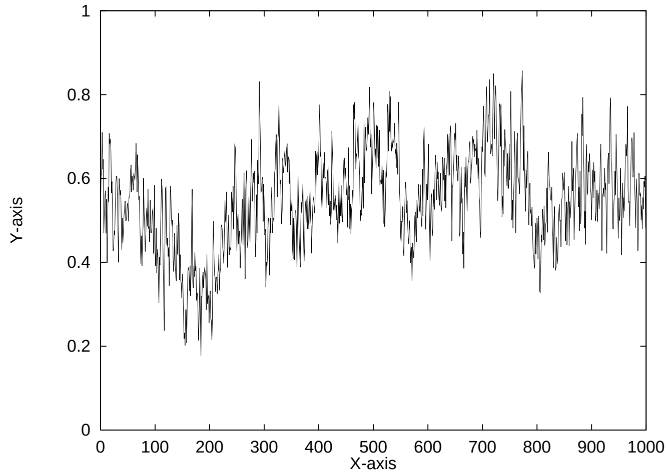

Figure 4: Aggregated Fractal Series with a Block Size of 10

Using the above methods, it is easy to show that a Poisson series is not self-similar. To visulize this, a 10,000-point Poisson series is generated (see Figure 5) for comparison. As shown in Figure 6, the aggregated series (with the same aggregation level of 10) is smoothened significantly which leads to the failure of the Poisson process to model actual network conditions. As shown in [1], aggregating real network traffic intensifies the burstiness of the traffic instead of smoothing it and the burstiness remains over many times scales.

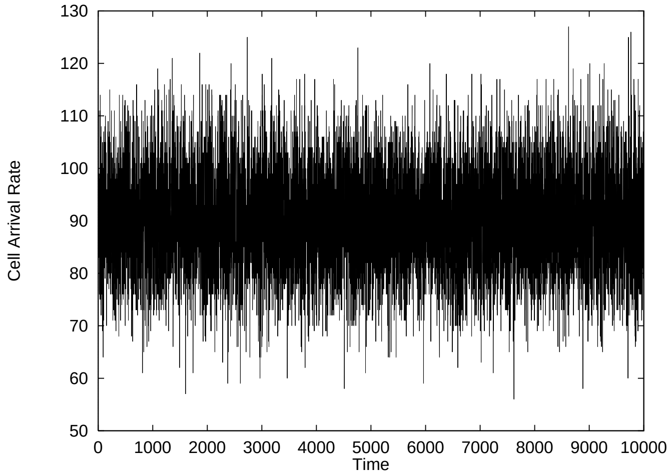

Figure 5: A Generated Poisson Series

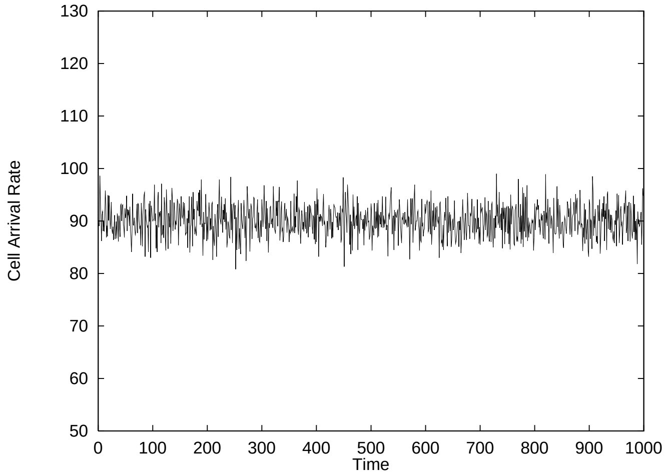

Figure 6: The Aggregated Series with a Block Size of 10

3 Omega Switch

The basic element in an omega switch is a 2×2 crossbar switch with 2 inputs and 2 outputs. These elements can be cascaded in series and in parallel to create larger switches. For N inputs and N outputs, where N is a power of 2, an omega switch has log_2N stages, with $N/2$ crossbar switches in each stage. Each stage has N inputs and N outputs. The interconnection between stages is defined by the perfect shuffle function: take the binary representation of an output port in the higher stage, do a logical rotate left to get the input port of the lower stage to which it connects. One advantage of the omega switch is the "self-routing" property. At each crossbar switch, the routing decision can be made by looking at only one bit in the cell destination port id and routing the cell to the upper port if the bit is 0 and to the lower port if the bit is 1.

The following analytical result is obtained for comparison with simulation results.

Under the assumption of Bernoulli cell arrival and random destination assignment, the cell loss *probability, P, of a bufferless $N \times N$ omega switch is*

$$P = \frac{1}{4} \sum_{m=0}^{log_2 N-1} P_m^2 \tag{12}$$

where P_0 is the input load and $P_{m+1} = P_m(1 - P_m/4)$ $(m = 0, 1, ..., log_2 N - 1)$.

Proof: With an input load of P_0, there are NP_0 cells entering the switch. For each crossbar at Stage $log_2 N - 1$, the probability that both input ports have cells is P_0^2 and the probability of contention is 1/2. Therefore, the probability that a crossbar has a contention is $(1/2)P_0^2$. Since there are $N/2$ crossbars at each stage, the average number of contentions at Stage $log_2 N - 1$ is $(N/2)(1/2)P_0^2 = (N/4)P_0^2$. For the standard switch, re-routed cells can be discarded. The input load, P_1, for the next stage (i.e., Stage $log_2 N - 2$) is $(NP_0 - (N/4)P_0^2)/N$. That is, $P_1 = P_0(1 - P_0/4)$. For the rest of the stages, $P_{m+1} = P_m(1 - P_m/4)$ $(m = 0, 1, ..., log_2 N - 1)$. The total number of lost cells is $(N/4)(P_0^2 + P_1^2 + ...)$. Therefore, the cell loss probability is

$$P_s = \frac{(N/4)(P_0^2 + P_1^2 + ...)}{N} = \frac{1}{4} \sum_{m=0}^{log_2 N-1} P_m^2 \tag{13}$$

The above analytical cell loss probability shows good agreement with simulation results.

Simulations are performed on an 8×8 omega switch. Eight fractal series are generated and are used as an input to each of the eight input ports of a 8×8 omega switch. The cell sequence of an input port consists of 0's (indicating no cell arrivals) and 1's (indicating cell arrivals) which are generated using a decision line. At each time unit, if the corresponding fractal value is above the line, then a cell is generated (i.e. 1 appears in the sequence). Otherwise, no cell is generated (i.e., 0 appears in the sequence). The obtained cell sequence is actually made up of bursts (1's) and lulls (0's). The bursts are termed as "packet trains" in the literature. Each of the sequences is long range dependent and the total number of cells arrived at each time unit (i.e., the sum of the eight input sequences) is fractal with an estimated Hurst value of 0.711.

For a bufferless omega switch, a small performance difference (Bernoulli i arrivals vs. fractal arrival) is observed. This is probably due to the heavy cell loss in the switch. Both Eq. (13) and the simulation results show that 30% of the cells will be lost at a load of 0.5 and almost half of the cells will be lost at full load. As cells are lost, the traffic loses its fractal property resulting in small performance difference. When the cell destinations are randomly chosen, the cells observed from a specific output port are almost white noise. However, the sum of the output sequences remains fractal but with significantly reduced Hurst value (about 0.6). In other words, the fractal degree is significantly reduced if the cell destinations are randomly chosen. A simulation is performed for fixed output ports (i.e., cell output addresses are determined using a complete port permutation and are used for the entire simulation). It turns out that the traffic at each output port remains long range dependent and the sum of out traffic is still fractal with a Hurst value of 0.706, which is slightly less than the Hurst value of the input traffic. Since cell destinations in real traffic are not randomly chosen, self-similarity of network traffic persists over many spacial scales. This explains why aggregated network traffic remains bursty.

The performance of a buffered omega switch is evaluated under both Bernoulli and fractal traffic. For an infinity buffer size and 10,000 cycles, the peak values of buffer occupancy at different loads are shown in Figure 7. Note that results using fractal traffic are presented by dotted lines and solid lines are used to present results from non-fractal traffic in the figures. It can be seen that the Poisson model generally underestimates the required buffer size. A significant difference is observed for switch loads between 0.4 and 0.8. This can be explained as follows. At low traffic load (0.3 or less), the cell contention probability is low and bursitiness becomes less important. At high traffic load (0.9 or greater), the switch is flooded. In this case, the switch truncates peak bursts and the fractal traffic becomes non-fractal.

For a fixed buffer size of 3 at each output port at each stage, the cell loss probability for both fractal and non-fractal traffic is shown in Fig. 8. As expected, fractal traffic results in a higher cell

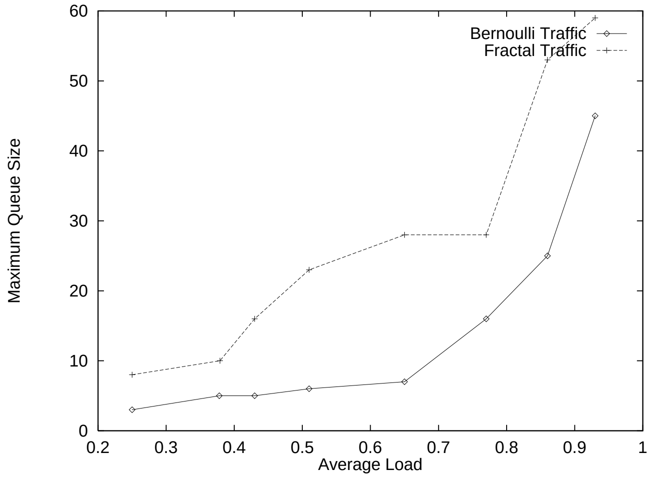

Figure 7: Comparison of Peak Queue Occupancy

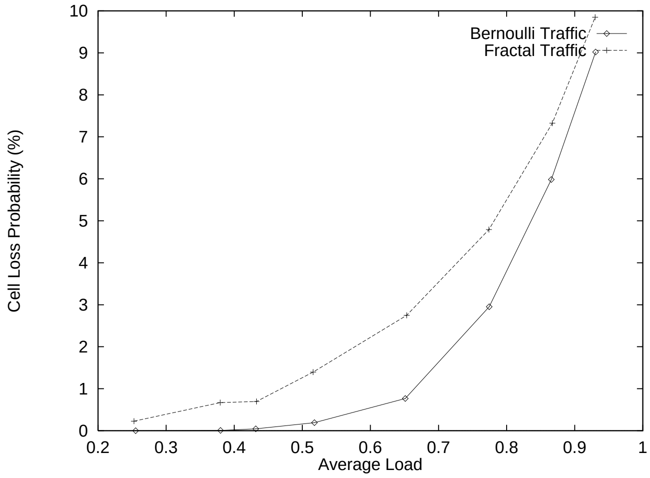

Figure 8: Comparison of Cell Loss Probability

loss probability. It is interesting to note that the cell loss probability (shown in Figure 8) has a shape similar to the peak occupancy (shown in Figure 7) in the sense that intermediate input loads result in a greater performance difference than extreme input loads.

4 Hypercube Switch

The hypercube has long been of interest as an interconnection network for parallel computing systems. It is a blocking packet switching network where the number of message links between a given router and the rest of the network varies as the logarithm (to the base 2) of the number of nodes in the network. Communication network designers are drawn to its geometric regularity and simple, yet powerful routing algorithms. A D-dimensional hypercube has 2^D nodes and $D2^{D-1}$ edges (or communication links). Any two nodes are connected by a communication link if and only if the binary identifiers of the two nodes differ in only one bit. A re-routing algorithm can be used since multiple paths exist between any two nodes. When a cell is considered, the node examines which outgoing links are on shortest paths to the cell's destination and sees if any of those are free. If at least one of those links is free, the cell is assigned to that link. Otherwise, the cell is assigned to any free link [7]. Detailed information on hypercube switches can be found in [5], [7], [6], [8] and [9].

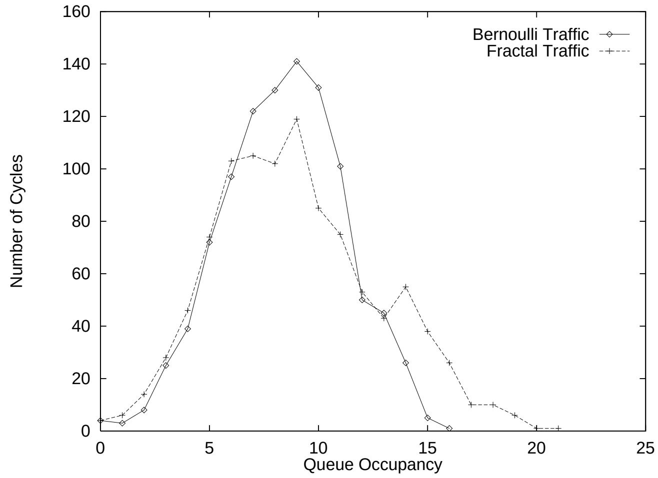

Figure 9: Comparison of Queue Occupancy (Load = 0.686)

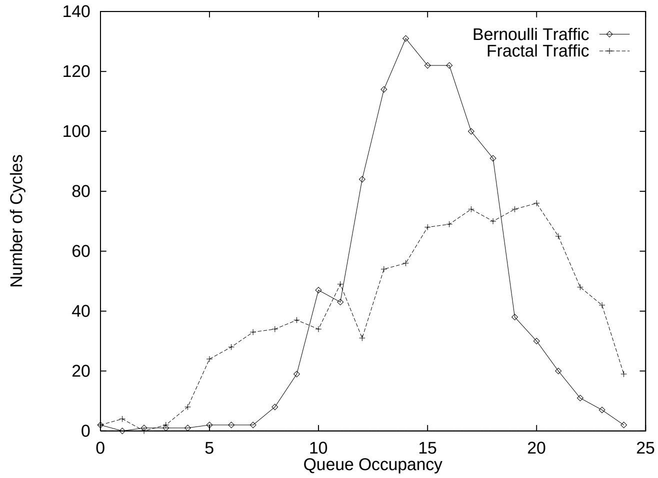

Figure 10: Comparison of Queue Occupancy (Load = 0.784)

The fractal series used in Section 3 are used as the traffic inputs to a 3-dimensional hypercube. The buffer size for each router is 3 and, therefore, the maximum buffer occupancy is 3 X 8 = 24. Although two delivery channels can be used, this study uses one delivery channel for all the simulations. That is, each router can deliver at most one cell during each cycle. The buffer occupancy statistics for three different loads are shown in Figures 9-11 for 1000 cycle simulations. Again, results using fractal traffic are presented by dotted lines and solid lines are used to present results from non-fractal traffic in the figures. It can be seen that fractal traffic results in a higher queue occupancy. The number of cell loss, average buffer occupancy, and average latency at different input loads are presented in Tables 1 and 2. The input load is calculated using the total number cells generated during each simulation. At the load of 0.686, there is no cell loss for both Bernoulli and fractal traffic. The average buffer occupancy for fractal traffic is 5.485 and the average buffer occupancy under Bernoulli traffic is 4.319. At the load of 0.784, 3 cells are lost under fractal traffic and there is no cell loss under Bernoulli traffic. The average buffer occupancy under fractal traffic is 9.043 and the average buffer occupancy under Poisson traffic is 8.416. At the load of 0.864, 102 cells are lost under fractal traffic and 35 cells are lost under

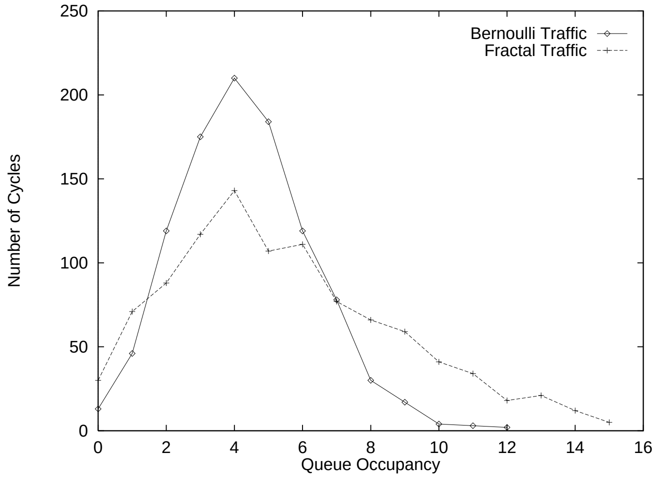

Figure 11: Comparison of Queue Occupancy (Load = 0.864)

Poisson traffic. The average buffer occupancies under fractal and Bernoulli traffic are 15.314 and 14.86, respectively. It can also be seen from Figures 8-10 that the buffer occupancy distribution under fractal traffic is not uni-modal. Although the difference of the average buffer occupancies under fractal and non-fractal traffic is small, the difference of the peak buffer occupancies could be very large. This is also true for cell latency.

Table 1. Hypercube Switch Performance Under Poisson Traffic

Load	0.686	0.784	0.864
Cell Loss	0	0	35
Average Buffer Occupancy	4.319	8.416	14.860
Average Latency	2.438	3.163	4.485

Table 2. Hypercube Switch Performance Under Fractal Traffic

Load	0.686	0.784	0.864
Cell Loss	0	3	102
Average Buffer Occupancy	5.485	9.043	15.314
Average Latency	2.534	3.187	4.681

Table 3. Hypercube Switch Performance Under Fractal Traffic

Load	0.51	0.65	0.78	0.87
Cell Loss Probability (Omega)	7.14%	11.94%	16.00%	20.11%
Cell Loss Probability (Hypercube)	0.00%	0.00%	0.18%	1.19%

A 3-dimensional hypercube switch has 8 routers and each of the routers both injects outgoing cells and delivers incoming cells. Therefore, a 3-dimensional hypercube switch can be viewed as a 8-input-8-output switch. Its performance under non-fractal traffic is better in terms of low cell loss probability than that of a 8×8 omega switch. To compare the performance of the two switches with fractal traffic, a total number of 24 buffer spaces is used for both switches. The simulation results are presented in Table 3. It can be seen that, for the same buffer size, the hypercube switch performs better (lower cell loss probability) than the omega switch under fractal traffic. This is due to the multiple paths between any two routers in the hypercube switch and the re-routing strategy used in the routing algorithm.

5 Conclusions

Two types of switches are simulated using both fractal and non-fractal traffic models. Fractal traffic is generated using an algorithm from the literature; the degree of self-similarity of the generated traffic is tested using several commonly used methods. A Bernoulli model is used to generate non-fractal traffic. The simulation results can be summarized as follows.

1. Fractal traffic results in more cell loss than Bernoulli traffic. Since real network traffic traffic is often fractal, the conventional models underestimate required buffer sizes for real network traffic.

2. Due to the multiple paths among the routers in a hypercube switch and due to the re-routing policy, the hypercube switch performs much better in terms of low cell loss probability than the omega switch.

3. With fractal traffic input to a switch, the aggregated output traffic remains fractal, but with a reduced degree of self-similarity (due to cell loss and other smoothing effects in the switch fabric). The degree of self-similarity of input traffic can be significantly reduced if the output destination for each cell is selected randomly.

As a by-product, a analytical solution for the cell loss probability of a bufferless omega switch is obtained. The analytical result is in good agreement with the simulation results.

References

[1] Leland, W.E., Taqqu, M.S., Willinger, W. and Wilson, D.V. On the self-similar nature of Ethernet traffic (extended version). *IEEE/ACM Transections on Networking 2*, 1 (Feb. 1994), 1-15.

[2] Paxson, V. and Floyd, S. Wide-area traffic: the failure of Poisson modeling. In *Proceedings of SIGCOMM'94*, Oct. 1994, London, pp. 257-268.

[3] Garrett, M.W. and Willinger, W. Analysis, modeling and generation of self-similar VBR video traffic. In *Proceeding of ACM SIGCOMM'94*, Oct. 1994, London, pp. 269-280.

[4] Hosking, J.R.M. Modeling persistence in hydrological time series using fractal differencing. *Water Resources Research 20*, 12 (1984), 1898-1908.

[5] McCrosky, C. Message passing in synchronous hypercubes. *Computer Systems Science and Engineering 4*, 2 (1989), 89-96.

[6] Varvarigos, E.A. and Bertsekas, D.P. Performance of hypercube routing schemes with or without buffering. *IEEE/ACM Transactions on Networking 2*, 3 (1994), 299-311.

[7] Greenberg, A.G. and Hajek, B. Deflection routing in hypercube networks. *IEEE Transactions on Communications 40*, 6 (1992), 1070-1081.

[8] Abraham, S. and Padmananhan, K. Performance of the direct binary n-cube network for multiprocessors. *IEEE Transactions on Computers 28*, 7 (1989), 1000-1011.

[9] Stamoulis, G.D. and Tsitsiklis, J.N. The efficiency of greedy routing in hypercubes and butterflies. In *IEEE Transactions on Communications 42*, 11 (1994), 3051-3061.