Investigation of Self-Organized Criticality in Packet-Based Communications Networks

Nathaniel D. Sizemore, *Westminster College*
Vickie E. Lynch, *Oak Ridge National Laboratory*
Benjamin A. Carreras, *Oak Ridge National Laboratory*

Fusion Energy Division

Abstract

Investigation of Self-Organized Criticality in Packet-Based Communications Networks. NATHANIEL D. SIZEMORE (Westminster College, New Wilmington, PA 16172-0001), Vickie E. Lynch (Oak Ridge National Laboratory, Oak Ridge, TN 37831-8070).

A system is governed by self-organized criticality if it is a driven system that self-organizes to be close to some critical point. Communication systems have been shown to have a critical point where the system goes from continuous flow of information to a jammed state. Here we have examined the possibility of self-organization as a competition between information demand and congestion control. An object-oriented computer simulation was written in C++ to examine the self-organization properties in packet-based communication networks. Various congestion control were examined after confirming previously published results that did not include these methods. These included a simplified choke packet technique, congested signaling, backpressure, and dropping packets. Impacts of these schemes on self-organization were compared using a variety of diagnostics including throughput, average time traveled by packets, and probability distribution functions from time and distance traveled by packets. Studying the properties of self-organization can help better understand the macroscopic trends and properties such as throughput in large packet-based networks ranging from corporate LANs and WANs to the global Internet.

What is a Self-Organized Criticality, Anyway?

- An SOC is a driven system that moves itself to some critical point -- a “border” between behaviors
- Examples of SOC include mass transport in plasma, commercial power grids, and forest fires
- The movement of information in a communication network (like an office LAN or the Internet) might have similar behavior
  - Traffic in the network goes from a smooth flow to a “traffic jam” state as routers get more traffic then they can handle
  - Dynamics between information demand and congestion control might cause the system to self-organize

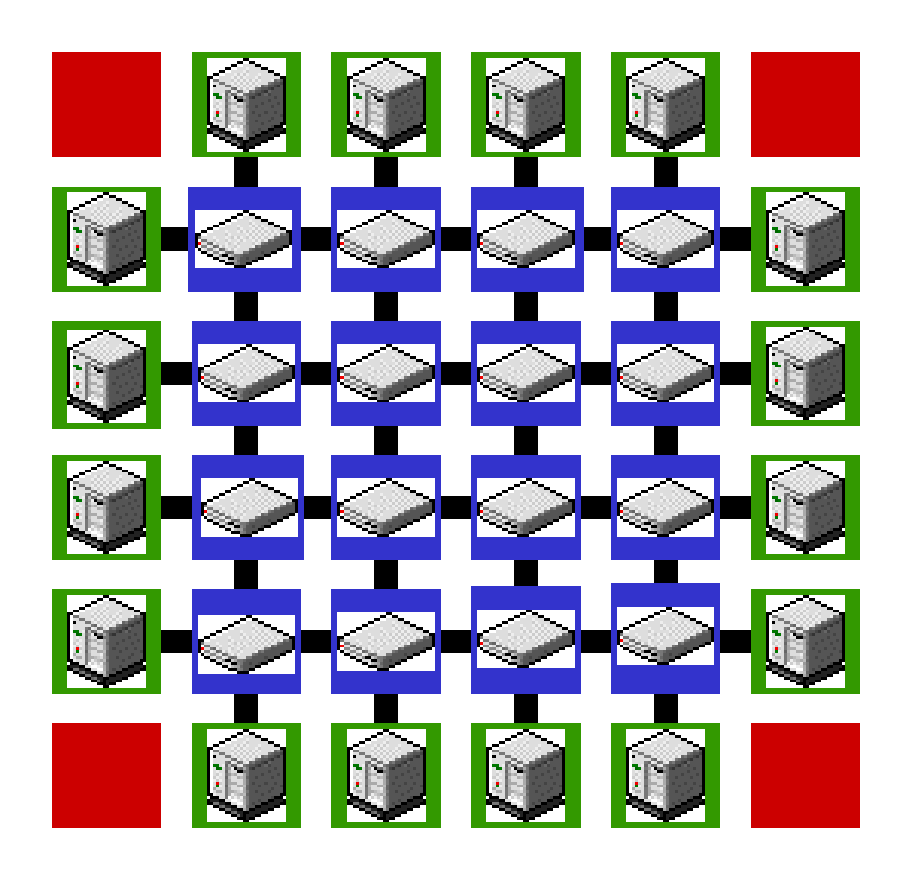
Development of computational model for network SOC

In order to study the self-organizing characteristics of packet-based networks, a computational simulation was developed. The model contained many individual elements (such as routers, hosts, and packets) that were interacting, so an object-oriented paradigm was chosen. C++ was used as the development language both for its OO features as well as its portability. Additionally, the Standard Template Library (STL) was used extensively in the simulation both for flexibility and to deal with memory management (which had been a problem with earlier preliminary work). The network is stored in a graph data structure for maximum flexibility. Use of graphs also simplified distance calculations -- these could now be done with a shortest path algorithm, instead of empirical calculations which are limited to regular, well-defined networks.

The model is implemented as a Monte-Carlo cellular automata simulation due to the way the system evolves through time. Initial conditions are given by the initial value for λ for the hosts, router buffer size, time delay (if needed), and $\Delta\lambda$ (used for backpressure method).

Setup for the Computational Model

- Routers and hosts connected in a lattice
- Hosts are along the edges, and connect to the routers within the interior of the lattice
- Corner nodes are not used, as they cannot connect to routers

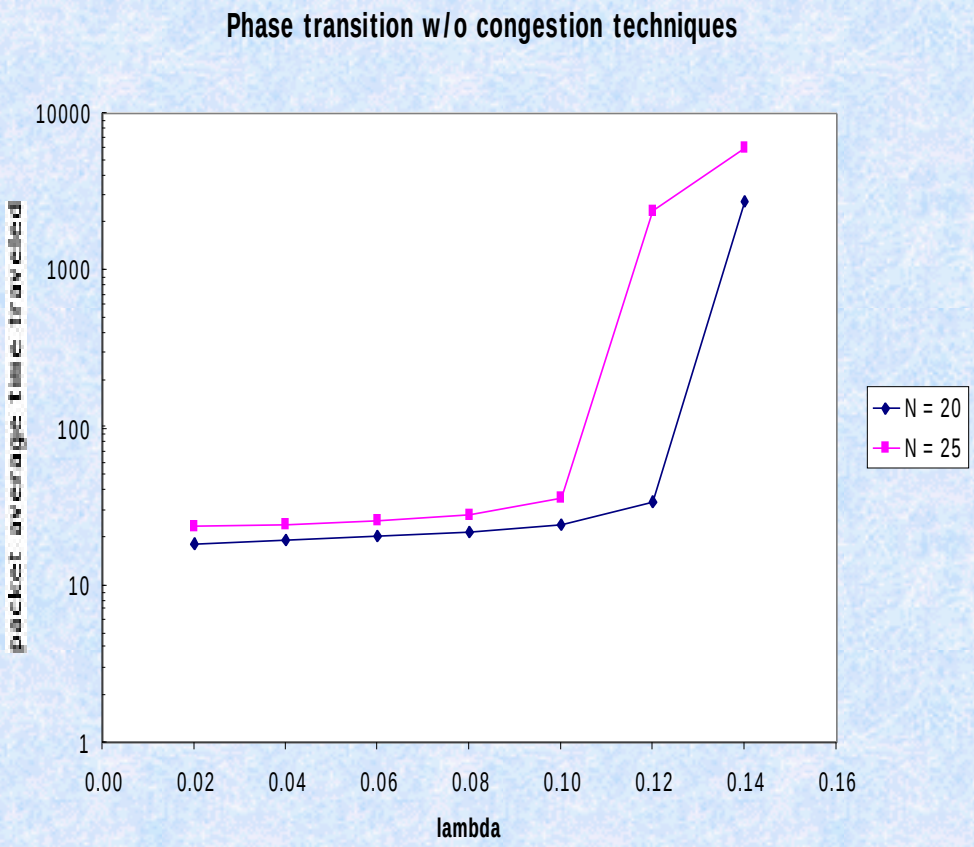


Rules of the Game

- Step 1: Hosts create packets
  - Hosts create packets with a probability λ and insert them into the system. Generated packets are given a random destination chosen from the hosts in the lattice.
- Step 2: Routers move packets
  - Routers forward the top packet in the queue to the neighboring node closest to the destination of the packet. If the closest distance is shared by more than one node, selection is made randomly between them. If a neighbor's queue is full, the router chooses the next closest neighboring node.
- Step 3: Hosts receive packets
  - Hosts that receive packets addressed to them delete the packets – this removes the packets from the system.

Need for Congestion Control

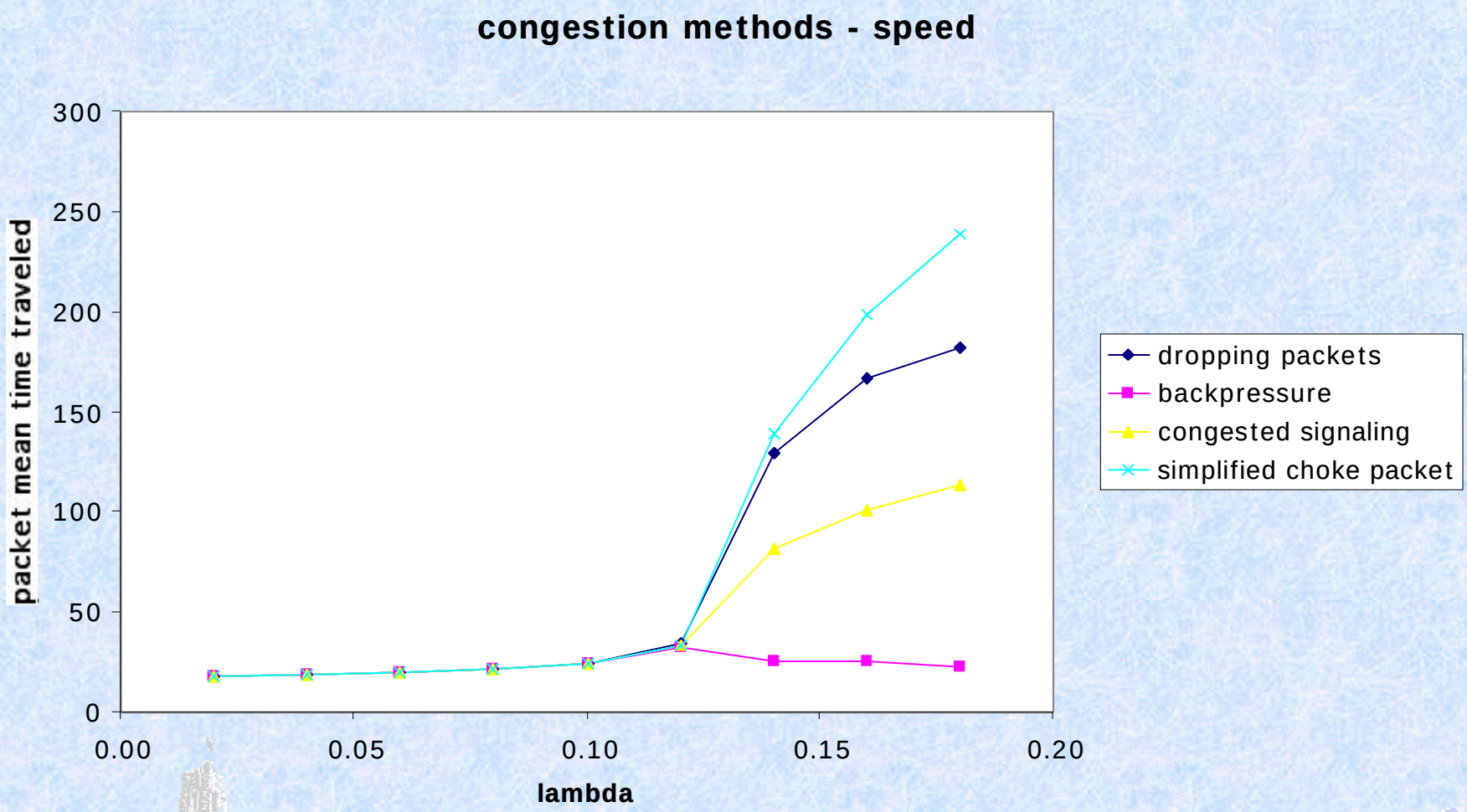
- Without any congestion control, the system becomes saturated above the critical point; packet travel time jumps by orders of magnitude.
- This situation is also unrealistic -- routers in the real world must have some method of congestion control, since they cannot have infinitely large buffers



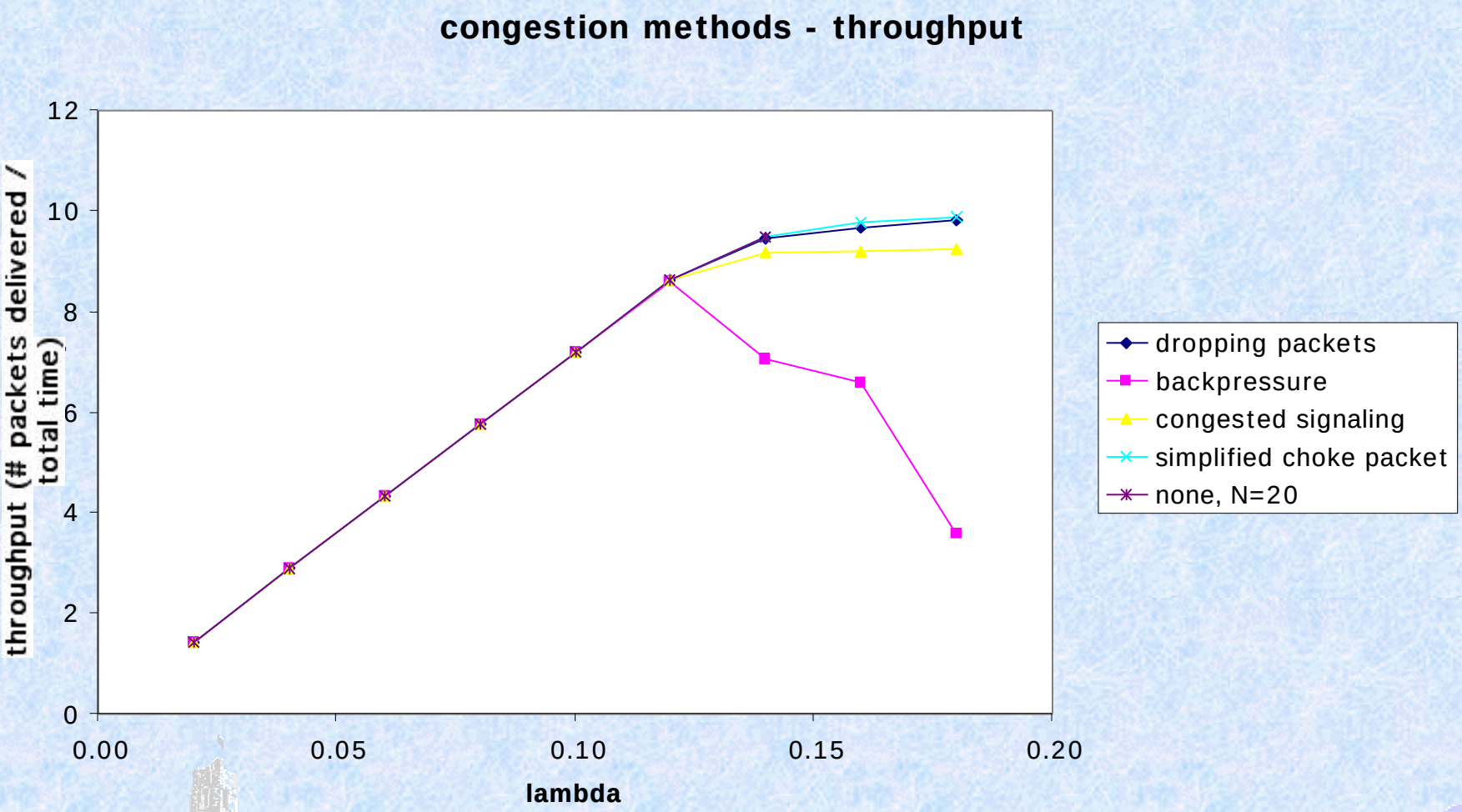
Congestion Control Techniques

- ***Dropping packets***
  - The simplest technique for dealing with congestion is to delete those packets that arrive at a node with a full buffer.
- ***Congested signaling***
  - Introduces a time delay to deal with congestion. If a buffer overrun occurs, the host which generated the packet that caused the overrun is put into a suspended state where it will not generate packets for a set number of time steps.
- ***Simplified choke packet***
  - Also introduces a time delay, but instead of suspending the generating host the node that delivered the packet is suspended.
- ***Backpressure***
  - Changes the value of λ at the generating host by an amount $\Delta\lambda$.

Comparison of Techniques - speed



Comparison of Techniques - throughput

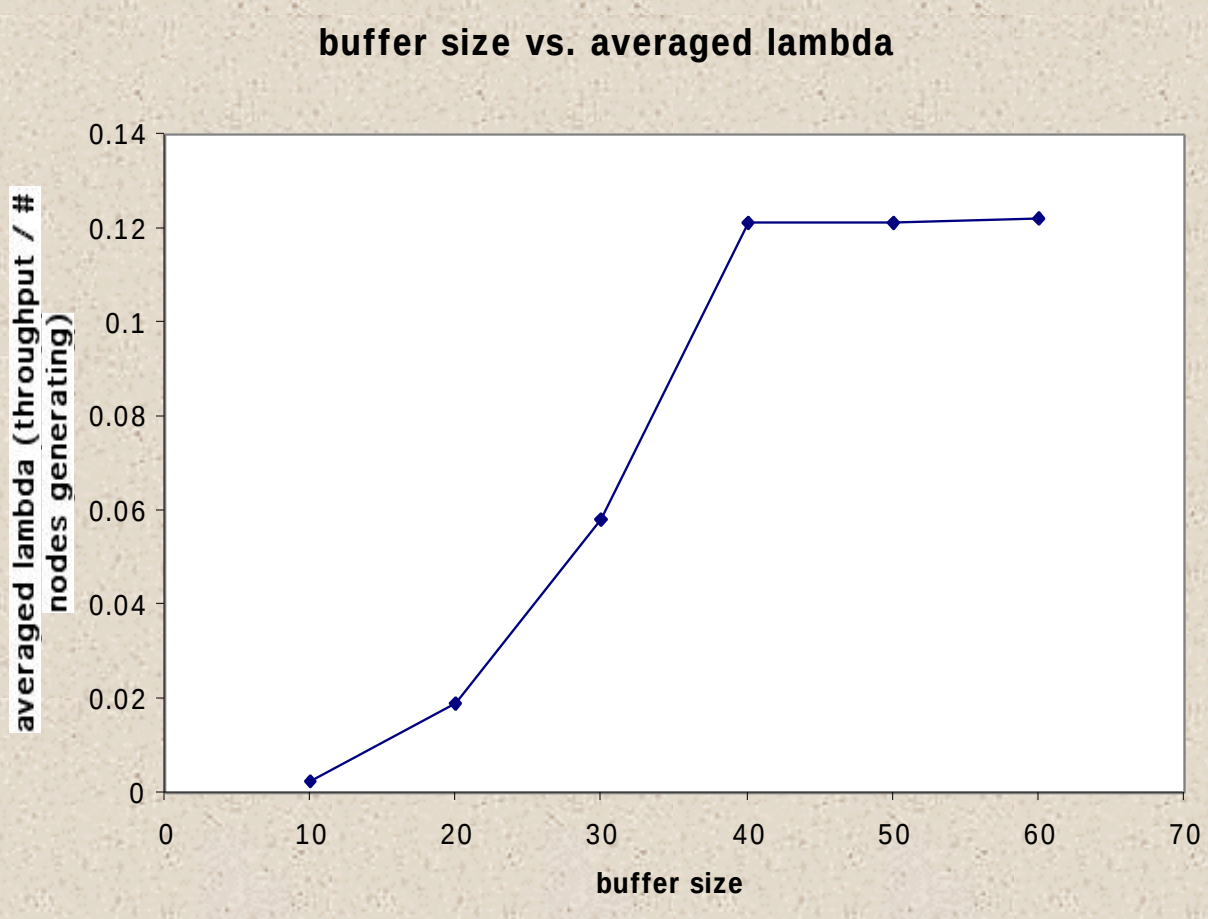


A New Method -- Finding the Critical Point Dynamically

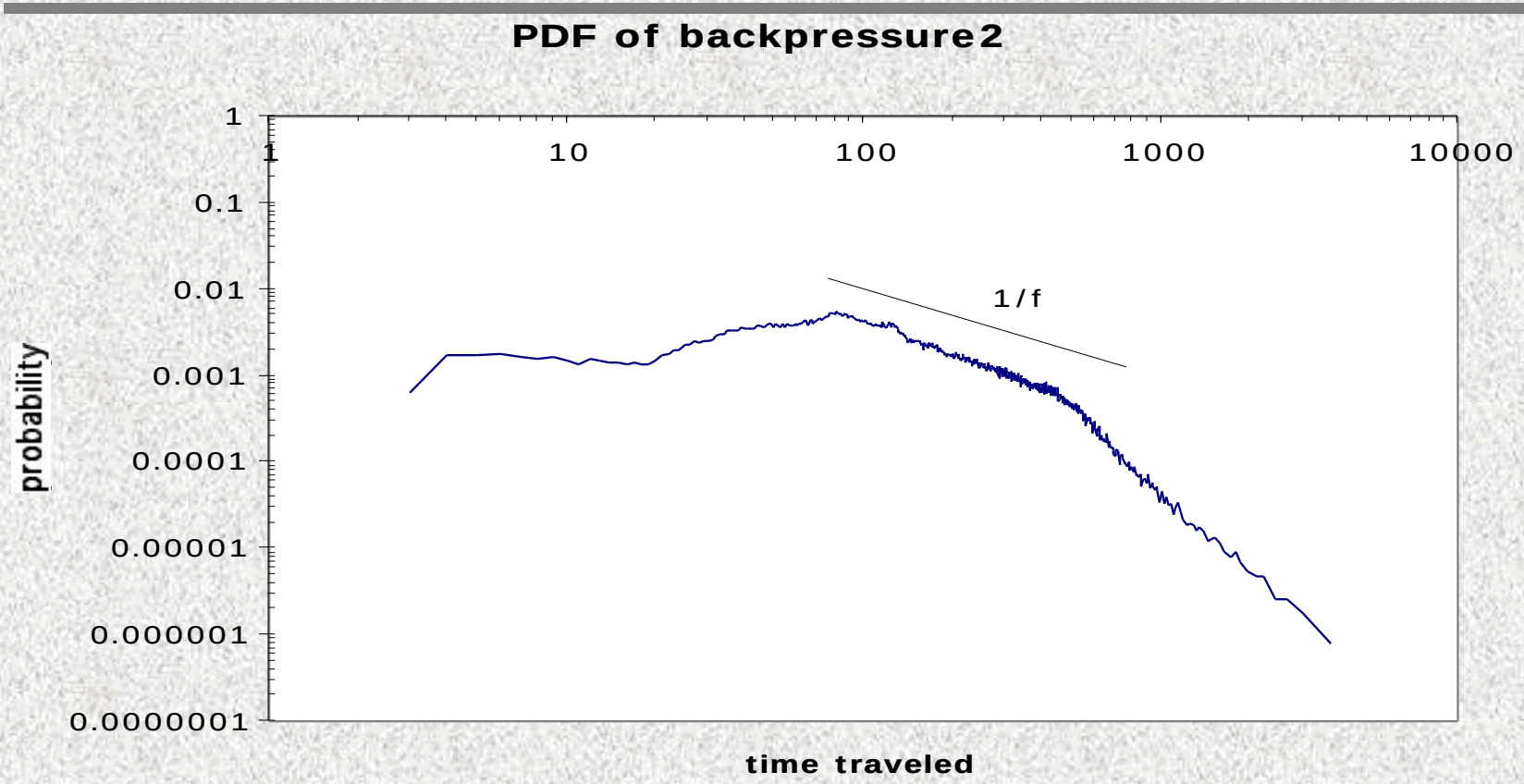
- The differences in the backpressure method prompted some thought -- why was it different, and is it possible to have its results look like the other methods?
- A closer look at the backpressure method showed that it was gaining its speed by virtually eliminating traffic on the network -- pushing λ down too far
- A modified version was created in which the hosts tried to increase λ as the routers tried to decrease it
- This method allowed the competition between information demand and congestion control to have a far greater and more immediate effect on the system. *As a result, this method finds and stays at the critical point -- no initial value for λ is needed.* This allows other conditions (such as buffer size or topology) within the system to be examined.

Effect of Buffer Size on System

Examining the effect of buffer size on the efficiency and averaged λ of the system with the second backpressure method shows that another critical point exists in the model. Buffer sizes above this critical point have essentially no effect on the system. However, if the buffer size is below this critical value, the result is an extreme loss of performance.



Evidence for SOC



The most convincing evidence for SOC behavior in the system is the decade of 1/f behavior that can be seen in the above probability distribution function for the modified backpressure method. This is characteristic of all SOC systems.

Conclusions

- These congestion control methods produce a model of network communication that self-organizes
- The interaction between data demand and congestion control produces a system that deals with the traffic jam events gracefully (i.e. doesn't produce exponential growth of packet travel time)
- The code developed for this model is easily extendable, and lays the groundwork for further investigation in areas such as:
  - Changing the topology to be less homogeneous
  - Combining various congestion control techniques
  - Packet trains (groups of packets with a common destination generated by a host in sequence or during the same time step)

References

- Bak, P., Tang, C., & Weisenfeld, K. (1987). Self-Organized Criticality: An Explanation of 1/f Noise. *Physical Review Letters*, 59, 381-384.
- Jain, Ran, & Routhier, Shawn A. (1986). Packet Trains – Measurements and a New Model for Computer Network Traffic. *IEEE Journal on Selected Areas In Communications*, SAC-4, 986-995.
- Ohira,T., & Sawatari, R. (1998). Phase transition in a computer network traffic model. *Physical Review E*, 58, 193-5.
- Stallings, William. (2000). Data and Computer Communications (6th ed.). Upper Saddle River, NJ: Prentice Hall.
- Willinger, Walter, Taqqu, Murad S., Leland, Will E., & Wilson, Daniel R. (1995). Self-Similarity in High-Speed Packet Traffic: Analysis and Modeling of Ethernet Traffic Measurements. *Statistical Science*, 10, 67-85.
- Yuan, Jian, & Shan, Xiuming. (2000). Self-organized criticality in a computer network model. *Physical Review E*, 61, 1067-1071.

Acknowledgements

- This study was performed at the Fusion Energy Division of Oak Ridge National Laboratory 2001 June 04 through 2001 August 10.

- The presenter wishes to thank his ERULF program mentor, Vickie E. Lynch, as well as Benjamin A. Carreras for invaluable guidance and assistance in this study. Additional thanks to the Department of Energy, the National Science Foundation, and Oak Ridge National Laboratory for making this work possible through the creation and continued support of the ERULF program.