Proceedings of the 1991 Winter Simulation Conference
Barry L. Nelson, W. David Kelton, Gordon M. Clark (eds.)

THE SIMULATION OF THE NON-LINEAR DYNAMIC BEHAVIOR OF DISTRIBUTED ROUTING NETWORKS USING DECSIM

Nicholas Ilyadis

Digital Equipment Corporation
158 Taylor Street
Littleton, MA 01460

Peter Drexel

Department of Computer Science
Plymouth State College
Plymouth, NH 03264

Andrzej Rucinski

Intelligent Structures Group
Electrical and Computer Engineering Department
The University Of New Hampshire
Durham, New Hampshire 03824

ABSTRACT

Network routers, which utilize distributed routing algorithms, are used to implement the highest levels of the modem computer communication hierarchy. As routing networks grow larger, their behavior under different load conditions becomes complex. Present analytical methods address these networks in their linear or load balanced regions of operation. In this paper, DECSIM - a logic hardware simulator, was used to model a router. Various load conditions were simulated on a mesh of 25 and 400 routers to characterize the network behavior both in the linear and non-linear regions. Different scheduling methods were used to characterize the systems' bandwidth. Simulation results are presented and compared to the non-linear behavior of natural systems.

1 INTRODUCTION

Interprocess communication over a network is one of the concerns in distributed processing. The communication routing issues have a direct impact on the performance of these distributed systems. Routers implement part of the communication channel between disjoint systems. The design of the routers, the routing algorithms and flow control mechanisms have a great impact on the throughput and efficiency of a network. This paper presents an investigation of a distributed routing network through the use of hardware simulation. Models of routers were used to simulate the distributed algorithm in a network governed by these routers. In past work, a network router model was designed in the DECSIM behavioral hardware modeling language. This model was based on the distributed algorithm used by Digital Equipment Corporation routers. A generic endnode model was also designed and implemented in DECSIM. This is important because DECSIM allows the writing of models that are concurrent in nature (as real hardware is).

The fundamentals of this research stress the complex behavior in mesh networks with distributed routing and they are described in Rucinski et al. (1990, 1991). The central question, posed in Rucinski et al. (1991), is whether networks utilizing local distributed algorithms could operate in the non-linear regime. In order to study the complex behavior of systems, a model must be constructed that can range in operation from the simple to the progressively more complex. A mesh lends itself to this approach because of its ability to map more complex operations on itself. The mesh can be simulated in either a synchronous static network with either full or partial connectivity, or a asynchronous dynamic network that experiences faults. The asynchronous case can be thought of as possessing wavefront processing characteristics.

Local routing that utilizes only information from adjacent nodes draws upon the analogies of fluid flow and diffusion (Rucinski et al. 1991). Traditional routers (Ahuja 1982, Cerf 1981, Gerla 1981, etc.) maintain a more global distributed algorithm. Analogies to natural systems may lend some insight into the operation of these systems. For instance, Martland (1989) showed that boolean networks correlate to periodic attractor theory presented by Feigenbaum (1980). Feigenbaum showed the onset of turbulence was characterized by period doubling. The above analogy to physical systems may show that this mesh system has the characteristics of a differential system.

The rest of the paper is organized as follows. Section 2 reviews the theoretical background presented by Rucinski et al. (1991) and puts the simulation work performed into perspective. This presents Mitchell Feigenbaum's work on the universal behavior of non-linear systems. This work shows the approach of turbulent behavior through period doubling. Martland showed this correlation with randomly connected boolean networks. Section 3 describes the relation of the simulation model to the theoretical background. In Section 4, DECSIM is briefly reviewed. A description of the router model is given in Section 5. Section 6 describes the mesh development and the simulation environment. Section 7 describes the basic modes of simulation. Conclusions and directions for future investigations are discussed in the final section.

2 THE MESH ROUTER WITH DISTRIBUTED ROUTING

For a detailed discussion of the mesh router and routing algorithm see Rucinski et al. (1991). It is presented here in brevity for completeness. The mesh network developed in Rucinski et al. (1991) has a computational node at each site. Each node is a self contained processor with memory. This type of mesh has many computational uses in computer science, here however, we will concentrate on its routing aspects. The methods and tools necessary to measure the performance of this system can be visualized by use of potential theory. A field of potential energy is defined as

$$E(X,\{i,j\},t) = \sum M_k \cdot D_k^p \qquad (1)$$

where:

D_k = destination address vector of packet

M_k = the cardinality of a packet set with the same destination address D_k

p = the stress coefficient which determines the impact of destination addresses on the energy field.

k = the index for packets concentrated at node (i,j) with the same destination D_k at time t.

The nodes in this mesh can send four concurrent packets, thus more than one node is involved in process $\{X(t)\}$. As messages get scheduled and moved the value of $E(X\{i,j\})$ changes. The difference

$$\Delta(X,\{i,j\},t+1) = E(X,\{i,j\},t+1) - E(X,\{i,j\},t) \qquad (2)$$

is a field of kinematic energy (power) dissipated by the mesh during one clock cycle when no new messages have been created. When messages reach their destination they are consumed or dissipated. Thus, the further from the destination a message is, the higher its energy. Related to this is the traffic of the system $F(X,\{i,j\},t)$ which measures the communication activity of the network. This is the number of messages that are trying to move in the network. Under light loads, the network can best be characterized by deterministic behavior. As the load increases, however, the network shows random behavior. This is caused by message interaction, which increases delay uncertainties and introduces non-determinism (Rucinski et al. 1991). The routing algorithm was developed using information flow parameters that physical systems possess, such as velocity and gradients. The local nature of this algorithm is that it routes towards the energy field gradient of the maximum descent. This should minimize the impact of the packet on the local energy field and this in turn should increase the probability of a global minimal energy equilibrium (Rucinski et al 1991).

The energy balance equation shows that the changing potential energy field

$$E(t) = \{E_k\}, \text{ where} \qquad (3)$$

$$E_k = E_k(X_k,\{i,j\}_k,t) \qquad (4)$$

is the potential energy associated with communication task X_k with corresponding population sites $\{i,j\}_k$. In a deterministic case we have

$$E(t+1) = A(E(t),\ C(E(t),E(t-1),\ldots,E(0),t)\ \ ,t) \cdot E(t) + B(E(t),\ D(E(t),E(t-1),\ldots,E(0),t)\ \ ,t) \cdot E^0(E(t),t) \qquad (5)$$

where $A(E(t),\ C(E(t),E(t-1),\ldots,E(0),\ t),\ t) \cdot E(t)$ is the system matrix determined by the mesh connectivity with dynamic behavior control via *distributed routing algorithm* C, and

$B(E(t),\ D(E(t),E(t-1),\ldots,E(0),t\ \ )\ ,t) \cdot E^0(E(t),t)$ is the system matrix determined by the mesh connectivity with dynamic behavior control via *distributed scheduling algorithm* D. This has boundary conditions of $E(X_k,i,j,t) \leq E_{max}$ due to limited buffer capacity and $F(t) \leq 4n - 4\sqrt{n}$ due to limited bandwidth in a square mesh.

All the parameters in this equation are non-linear.

Matrix A controls the relaxation process while matrix B determines the activation of new messages with energy E^0. This equation also is presented in the probabilistic case as

$$e(t+1) = a(e(t),\ c(e(t),e(t-1),\ldots,e(0),t)\ ,t) \cdot e(t) + b(e(t),\ d(e(t),e(t-1),\ldots,e(0),t)\ ,t) \cdot e^0(e(t),t) \qquad (6)$$

where e(t) is a normalized population of nodes active at time t and a and b are stochastic matrices controlled by Markov processes c and d of order t (Gerla 1981).

3 RELATION OF SIMULATION MODEL TO THEORETICAL MODEL

This model development has maintained the above measures of energy and traffic. For the energy calculations the formula used on a per packet basis is:

$$E_{packet} = (DX + DY)^P, \qquad (7)$$

where:

$$DX = |\ My\_Xadd - Dest\_Xadd\ | \qquad (8)$$

and

$$DY = |\ My\_Yadd - Dest\_Yadd\ | \qquad (9)$$

and p = 2 (from Equation (1)).

My_Xadd and My_Yadd represent the cartesian coordinates of the node of interest. Dest_Xadd and Dest_Yadd are the cartesian coordinates of the destination node. The total energy of the mesh is the sum of all the unconsumed packets either in queues or buffers. The total traffic (FT) is the count of all attempted sends on all the ports in the mesh. Related to this are the acknowledgment (ACKs) and non-acknowledgment (NACKs) counts. The total traffic is the sum of all the ACKs and NACKs for a particular cycle. Under dissipative conditions the ACK count is very close to the FT count, or the NACK$\rightarrow$0. The node's potential, which is used in the routing algorithm, is a measure of its message load. This is basically the number of messages that a node owns, including non-acknowledged messages in its output buffers. The simulations generated two measures of the network state: mesh energy and mesh traffic. In the simulation section graphs were presented that depict the various operating conditions graphically as total mesh energy versus cycles as well as total mesh traffic versus cycles.

4 DECSIM AND ITS USE

DECSIM provides languages that allow a designer to model, simulate, test and debug multi-level logic networks, ranging from simple gates to whole CPU's. The DECSIM languages include a structural interconnect language, a behavioral modeling language and a interactive command language. The structural interconnect language allows the connection of "structural" components such as MOS gates, logic gates and flip-flops into networks that can be simulated. Larger models that have been previously compiled can also be interconnected, in a hierarchial fashion, to implement more complex systems. Signals are the connections between structural and/or behavioral components. Signals can take on the values of logic one, logic zero, high impedance, and undefined. States are variables that can be assigned an integer value within the bit range defined. The behavioral language allows the modeling of a block's operation in a software language rather than the actual structural interconnect. This allows creation of quick turnaround models that have faster execution speeds than their equivalent structural counterparts. The DECSIM behavioral language, which is based on BLISS, is structured like PASCAL and allows concurrent operation. The behavioral blocks communicate through ports. Ports are the points where external signals come into a software model. The internal program variables are states. The interactive command language allows the user to load a structural network that may contain structural and/or behavioral blocks and simulate it. During simulation the value of any signal or state may be examined, modified or saved to a log file. Command macros can be defined to encapsulate complex instruction sequences. The interactive command language can also be written in program flow style, similar to VAX DCL command files. The interactive environment runs in batch with indirect command files driving the simulation.

5 ROUTER MODEL DISCUSSION

The router model is written in DECSIM behavioral language. A block diagram is shown in Figure 1. The router has four I/O ports for data transmissions: 0 (right/east), 1 (up/north), 2 (left/west) and 3 (down/south). Each output port consists of a 48 bit Send Data (SD) output bus, a Request To Send Output (RTSO) signal and a ACKnowledge In (ACKI) input signal. The input port consists of a 48 bit Receive Data (RD) input bus, a Request To Send In (RTSI) input signal and an ACKnowledge Output (ACKO) output signal. The RTSO signal indicates that valid data is in the SD<48:0> lines. This is attached to the RTSI signal on the receiving node. The ACKO signal is attached to the ACKI signal on the sending node and indicates that the data was accepted. Each port also has a corresponding voltage sense input bus to receive the output potential of its neighbor in the indicated direction, ie. Voltage_0 (V_0) for direction 0. There are a set of global signals that are port independent. A XADDress and YADDress allow the X-Y address of the node to be programmed from the simulation. A Voltage Out (VO) is used to indicate the node's message load to its neighbors. The final input is the CLOCK. This is used to pace the nodes and keep them in synchronism. The program flow for the router is based around the routine ENGINE which is entered whenever the clock input changes. Sends occur during the high period of the clock, receives during the low period. The flowchart for the ENGINE routine is shown in Figure 2.

6 MESH DEVELOPMENT AND INITIAL SIMULATIONS

The previously discussed router is the basic building block for a mesh network. The work in Rucinski et al. (1990, 1991) is based upon a 5 X 5 mesh of processors. The published results of this 25 node mesh were used as a baseline to verify the operation of this router. The VALID schematic body of ROUTR was instantiated in a mesh as shown in Figure 3. This mesh is called TEST25.

All the nodes have their address busses assigned unique names so that they can be driven to a unique value at the initialization of the simulation. The addressing scheme is in an X-Y coordinate system. The upper left hand corner has an X address of 1 and a Y address of 1. From this point on the notation will be parenthesized as (X,Y). The X addresses increment as one progresses left to right, the Y addresses increment from top to bottom. The node in the bottom right hand corner is designated (5,5). The boundary nodes have their unconnected ports assigned signal names but not connected to any other device. These unconnected busses are driven with values at the initialization of the simulation. In the case of the RTSI signals a constant logic 0 maintains inactivity from that direction. The voltage sense busses are driven to their highest value, 255 in this case.

This schematic was processed by the G2S (GED to SPIDER) tool to extract a netlist. This netlist, when processed through SPIDER, resulted in a DECSIM netlist that could be compiled into a simulation model. Once the model was available, simulations were performed that corresponded to the routing experiments detailed in Rucinski et al. (1990, 1991). This first simulation has node (1,1) sending a group of messages to node (5,5). In turn (5,5) is also sending a group of messages to (1,1).

This simulation required a command file that drove the simulation. DECSIM has a indirect command file feature that allows commands to be read in from an external file. These files can also contain command level state variables that can be used as global simulation variables.

Graphs of the mesh energies, traffic and energy attractors are shown in Figure 4. Figure 4a is a composite of several different energy measures. EMT is the total mesh energy, which includes the originating processors and their instantaneous message loads. EM is the total mesh energy of all nodes less the two originators. E1 and E2 represent the energy of the (1,1) and the (5,5) nodes respectively. The composite graph of the traffic measures is shown in 4b. This shows total traffic and its ACK and NACK components. Graph 4c shows the attractor for the EMT over 200 cycles. The discontinuities represent the introduction of message loads into the corner nodes. The mesh is very dissipative with the majority of the operation occurring along the central $E(t+1) = E(t)$ line. The EM attractor, in 4d, represents the mesh energy of the 23 other nodes. These plots correlate very well with work in Rucinski et al. (1990, 1991). This exercise was meant to checkpoint the design so that it could be shown that a hardware simulation gave equivalent results to the software model. Some differences do exist due to the scheduling differences between the two simulations but these do not affect the overall results.

The unique work done on this project relates to how this mesh and larger meshes operate under load conditions that represent messages originating within the mesh rather than the two corner nodes. To maintain the link with the past work, the 25 node mesh was exercised in parallel with a 400 node mesh. The work of generating a 400 node mesh was initially challenging. The hierarchial structure of VALID and DECSIM allowed a 400 node mesh to be generated without having to directly connect up 400 nodes. The 25 node mesh that had been generated up to this point was used as a hierarchial entity for the larger simulation. A model called FIVEBYFIVE was created from this 25 node mesh. The connection points for the model were the boundaries of the 25 node mesh.

The computer resources that were used on this project included a DEC VAXstation and a pair of VAX 8800's clustered together. The 8800 is a dual processor machine. The netlist processing (G2S/SPIDER) for the 400 node mesh took approximately 1 hour on a single 8800 CPU with no other significant user load. The actual DECSIM compilation took 10 minutes. The resultant executable model is approximately 2.5 Mbytes.

7 THE DRIVE SERIES OF MESH SIMULATIONS

The task of loading the mesh with messages was challenging in that a scheduling scheme had to be designed that showed the dynamic behavior of the mesh under different load conditions. After some trial and error four different scheduling schemes were devised. These are designated DRIVE1 through DRIVE4. They range from most deterministic to least deterministic. DRIVE1 schedules 1/K nodes per cycle to send a single message to their complement node. The next cycle schedules the next 1/K nodes. This continues until all nodes have been scheduled. Once all nodes have been scheduled the process begins once again. The complement node is defined as the node mirrored across the center of the mesh. In the 25 node case the complement of (1,1) is (5,5). The complement can be mathematically derived as:

$$\text{destination } X = (\text{max\_x} + 1) - \text{My\_xadd} \quad (10)$$

$$\text{destination } Y = (\text{max\_y} + 1) - \text{My\_yadd} \quad (11)$$

Where max_x and max_y are the maximum X and Y coordinates of the mesh. In the case of an odd array size, the center node sends to the (1,1) node. When K is not modulo of the mesh total there are some cycles when one less node gets scheduled to make up the total.

Figure 5 shows the 25 node mesh under DRIVE1 conditions. The figure includes temporal, attractor and FFT plots of total mesh energy and total traffic. The FFT plot clearly shows that the temporal plot of mesh energy is periodic. Mesh operation is linear.

The DRIVE2 simulation maintains the same 1/K node scheduling scheme but rather than sending to the complement node it sends to a randomly determined node. This randomly determined node is selected at the onset of the simulation and maintained throughout the simulation. Figure 6 shows the mesh under DRIVE2 conditions. The ACK power spectrum indicates an increase in periodicity in this signal. The mesh operation is linear under these conditions.

DRIVE3 changes the scheduling scheme from the deterministic 1/K to a random set of nodes. The number of nodes that are scheduled each cycle is constant but the actual nodes are random. The destination is once again the complement node. Thus DRIVE3 has a random scheduling scheme that maintains a constant load of scheduled nodes. The temporal plots, attractors and FFT spectra of Figure 7 indicate the onset of chaos. Mesh operation has ceased to be linear here.

DRIVE4 is a combination of DRIVE2 and DRIVE3. It schedules a constant number of randomly determined nodes that send to the same list used in DRIVE2. The temporal plots of Figure 7 show the mesh moving from linear operation to a frozen, deadlocked state. From the ACK attractor, it is possible to see that the system has passed through a chaotic state before deadlock.

Several other scheduling schemes were also simulated but lack of time did not allow for the analysis or full simulation of these. These included variations of DRIVE2 that determined a new random destination node every cycle rather than using the list. Others included scheduling a cyclic number of random nodes each cycle, or even a totally random number under some upper bound. The point here is that once the network was compiled it was very easy to modify the scheduling and destination schemes with simple macros. This really shows the flexibility of this approach. The problem came with the analysis and plotting of the tremendous amount of information that was generated by these simulations. VAX DCL level command files could be written that ran DECSIM for each particular type of simulation under different load conditions. This would generate a disk full of data that required reduction and analysis. Interactive runs could also be programmed in which network behavior could be monitored as the simulation ran.

The 25 node mesh could be simulated for 256 cycles for a particular load condition in about 10 minutes. The 400 node mesh unfortunately was slightly slower. The average run of a mesh that did not saturate was 8 hours. Simulations that caused mesh saturation ranged from 9 to 10 hours on a single CPU. Thus the 25 node mesh was useful in developing the scheduling schemes and then applying them to the larger mesh.

8 CONCLUSIONS

For this routing algorithm the deterministic and random scheduling schemes did not show appreciable differences in network message routing capacity prior to saturation. Fundamental congestion problems occurred in both cases under similar load conditions. Evidence was shown that the random behavior of the coin flip introduce enough variability in the system to allow some extra routing capacity once saturation was entered. This gives direction for further research in more robust routing algorithms.

The flexibility of a hardware simulator for this type of work was shown. The ability to program and monitor network activity in an interactive or batch environment makes this a flexible approach. This method can be used to model many different types of networks for subsequent simulation.

The simulations presented here confirm that this router mesh exhibits complex behavior that is analogous to that seen in natural systems. This similarity appears to become more acute as the size of the mesh increases. This work confirms the taxonomy of behavior put forth in the theoretical work in Rucinski et al. (1990, 1991). Further work on this and larger meshes should be performed to further quantify this relationship.

In previous work, selected routers were made unavailable. These routers create turbulence in the message flow and increase the noise level of the system. This larger mesh could easily be programmed to compare the results on a larger mesh. This could be taken a step further by interconnecting routers in non-regular meshes or hierarchies of meshes. The schematic entry portion of the simulation environment allows this up front capability.

Leaving the mesh intact one could modify the underlying model to find out whether modifications to the routing algorithms would change the network behavior. Once the network netlist is in place only the simulation compiler need be used to modify the model. Work could be done to characterize the system's response under different initial conditions. Does this system possess the sensitivity to initial conditions that natural systems do? The simple scheduling schemes presented here could be modified to include Markov processes or other probabilistic models. A full characterization of the mesh operation with different queue limits would give more insight into the mesh's elasticity, or its ability to absorb transients.

Visualization techniques that map either network activity or energy gradients could be used with the above investigations to give insights to the routing behavior that are not apparent by other means.

It is apparent from the simulations that while this system has a relatively simple routing algorithm, nevertheless it produces complex behavior.

REFERENCES

Ahuja, Vijay, Design and Analysis of Computer Communication Networks, McGraw-Hill, 1982.

Cerf, Vinton G., "Packet Communication Technology". Protocols and Techniques for Data Communication, pp 1-30, Ed. F.F. Kuo, Prentice Hall, 1981.

Feigenbaum, Mitchell J., "Universal Behavior in Non-linear Systems", *Los Alamos Science*, Summer 1980.

Gerla, Mario, "Routing and Flow Control", Protocols and Techniques for Data Communication, pp. 122-169, Ed. F.F. Kuo, Prentice Hall, 1981.

Lam, Simon S. and Hsieh, Ching-Tarng, "Modeling, Analysis and Optimal Routing of Flow-Controlled Communication Networks", Technical Report TR-87-24, University of Texas at Austin, June 1987.

Martland, D., "Dynamic Behavior of Boolean Networks", Neural Computing Architectures, Ed. I. Aleksander, The MIT Press, 1989.

O'Leary, Art, "Distributed Routing", Ph.D. Dissertation, Massachusetts Institute of Technology, Jan. 13, 1981.

Perlman, Radia, "Routing Algorithms: Why Should We Care?; Pt 1, Implications for DECNET" (Video), Digital Equipment Corporation, Video NCS-850410, April 10, 1985.

Rucinski, Andrzej & Drexel, Peter & Dziurla, Barbara, "An Integrated Model of Complex Behavior in Networks with Distributed Routing", Frontiers of Computing Systems Research Vol III , Ed. S.K. Tewksbury, Plenum Press, 1991.

Rucinski, Andrzej & Drexel, Peter & Dziurla, Barbara, "Chaotic Nature of Mesh Networks with Distributed Routing", International Symposium on Advances in Interconnection and Packaging, Boston, November, 1990.

Tanenbaum, Andrew S., Computer Networks, Prentice Hall, 1988.

Toueng, Sam, "A Minimum-Hop Path Failsafe and Loop-Free Distributed Algorithm", IBM Research Report RC8530, October 14, 1980.

AUTHOR BIOGRAPHIES

NICHOLAS ILYADIS is a principal engineer with Digital Equipment Corporation. He is involved in VLSI design for use in network and communications equipment. His interests are in design synthesis and mixed analog-digital, semicustom IC's.

PETER DREXEL is an instructor in the Department of Computer Science at Plymouth State College. His background includes over 15 years in the semi-conductor industry. His research interests are distributed systems, distributed routing techniques and chaos. He is a member of the Intelligent Structures Group at the University of New Hampshire.

ANDRZEJ RUCINSKI is an associate professor in the Department of Electrical and Computer Engineering at the University of New Hampshire. His research interests are distributed systems, fault tolerant computers and chaos. He is coordinator of the Intelligent Structures Group and Student Exchange Program (Technical University of Budapest).

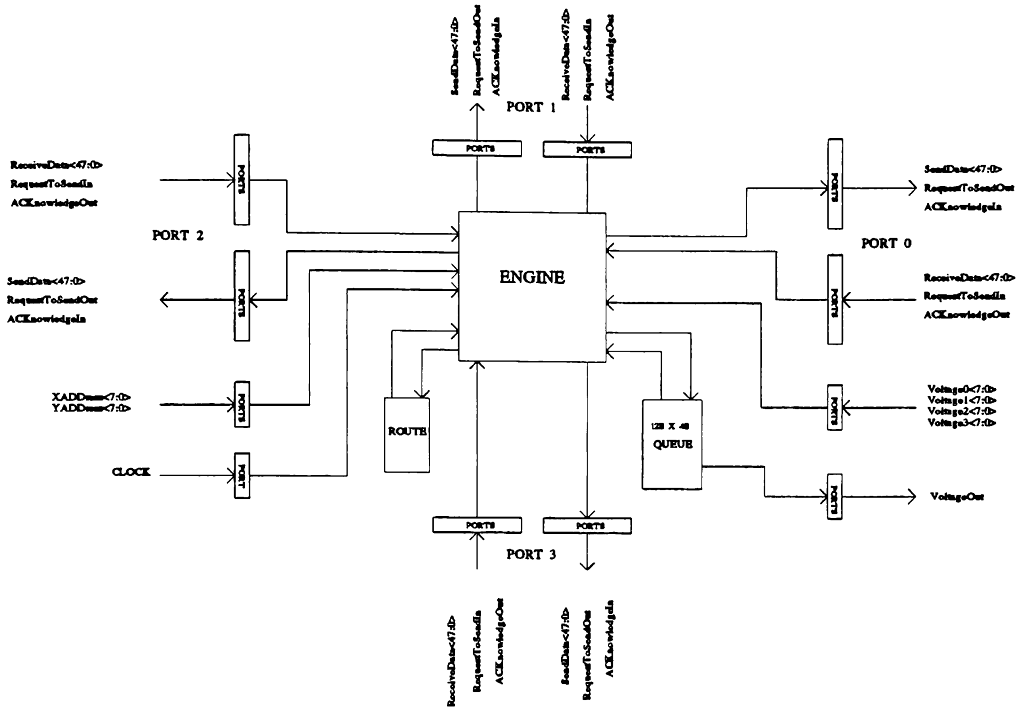

Figure 1: Mesh Router Block Diagram

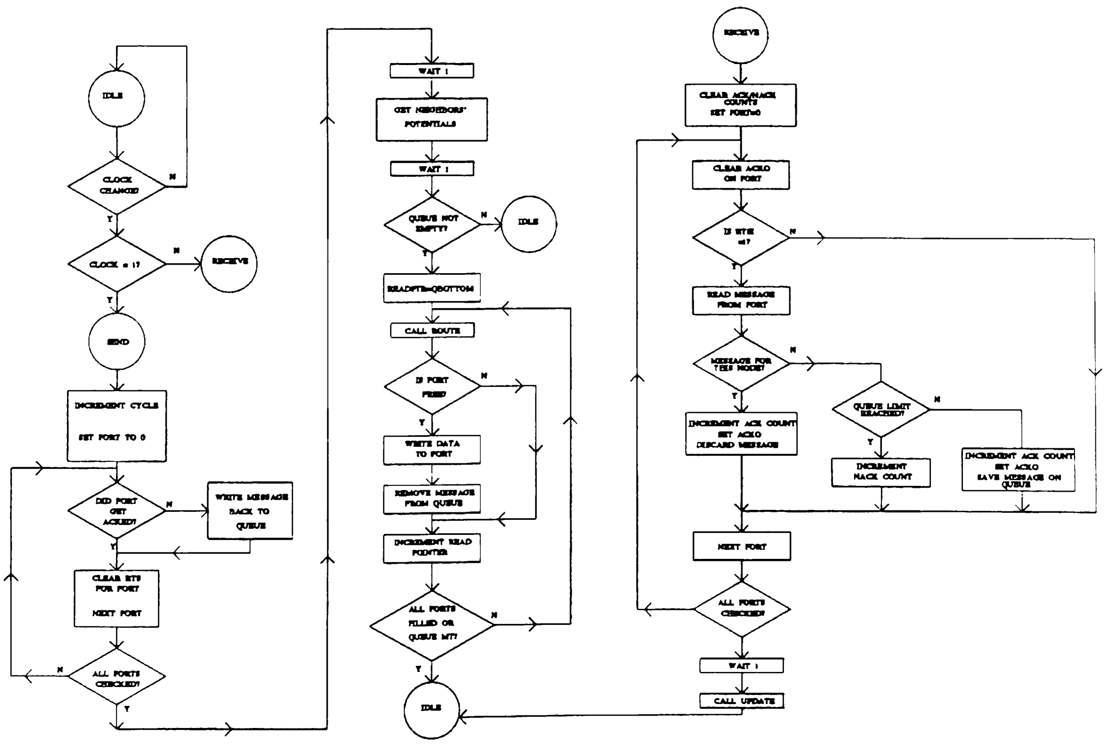

Figure 2: ROUTR Engine Routine Flow Diagram

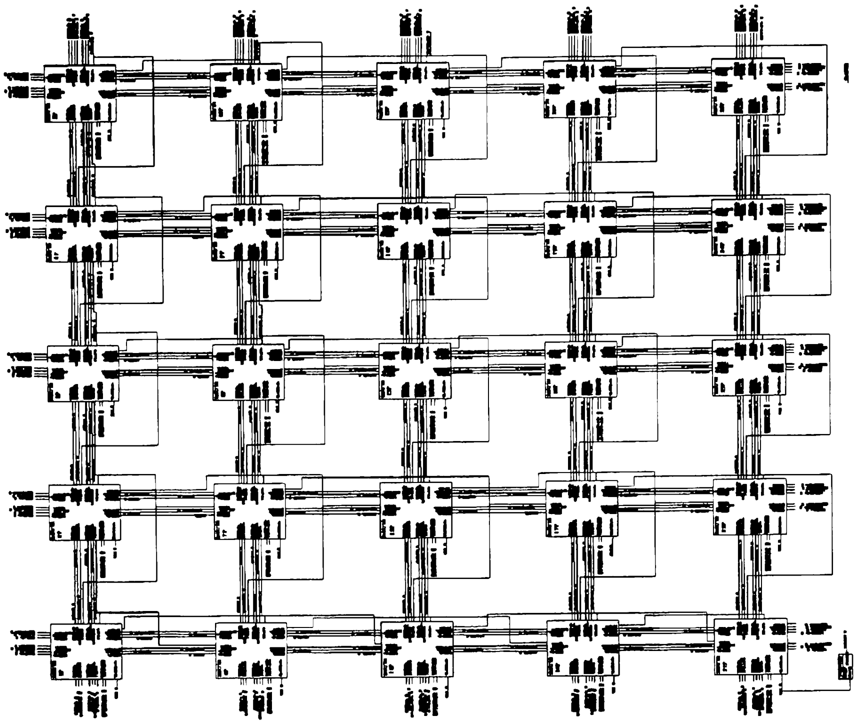

Figure 3: Schematic Of TEST25 (5 X 5 Router Mesh)

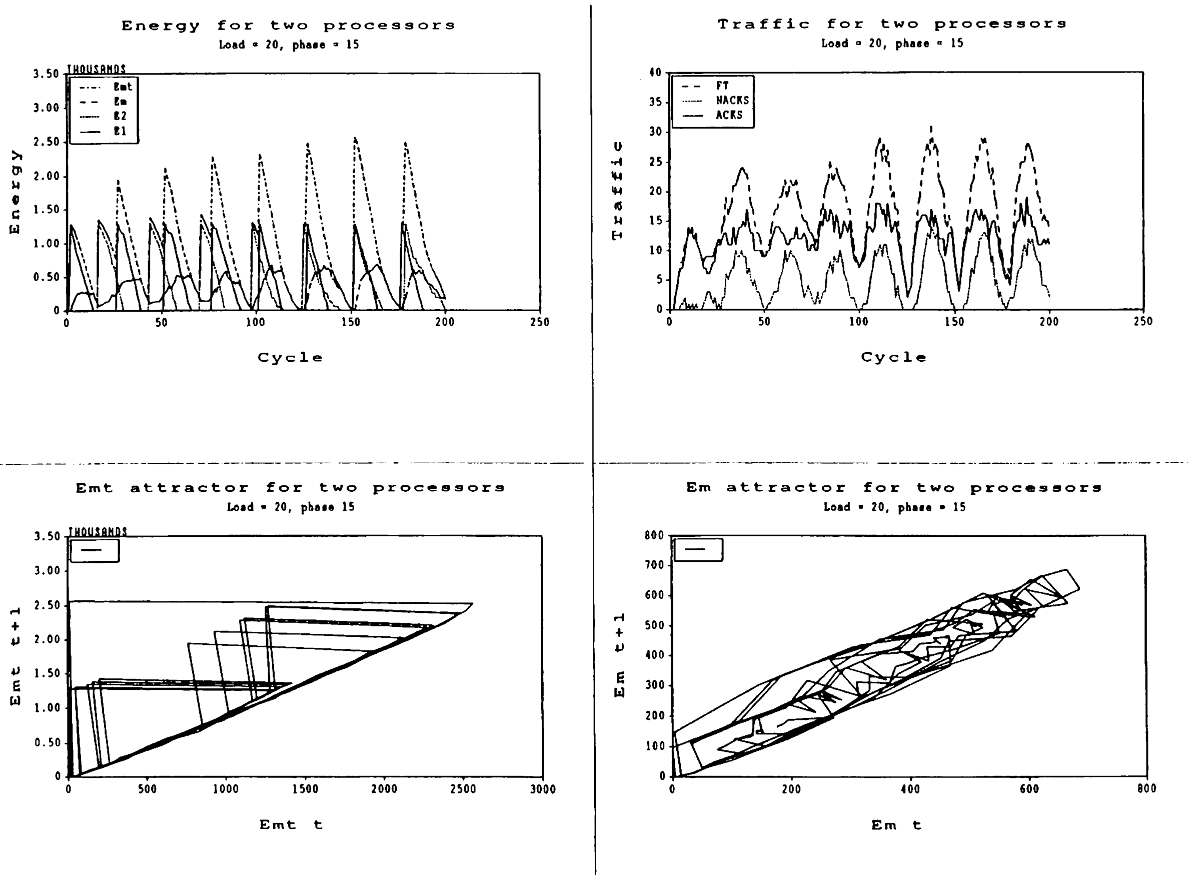

Figure 4: Energy And Traffic For Two Processors

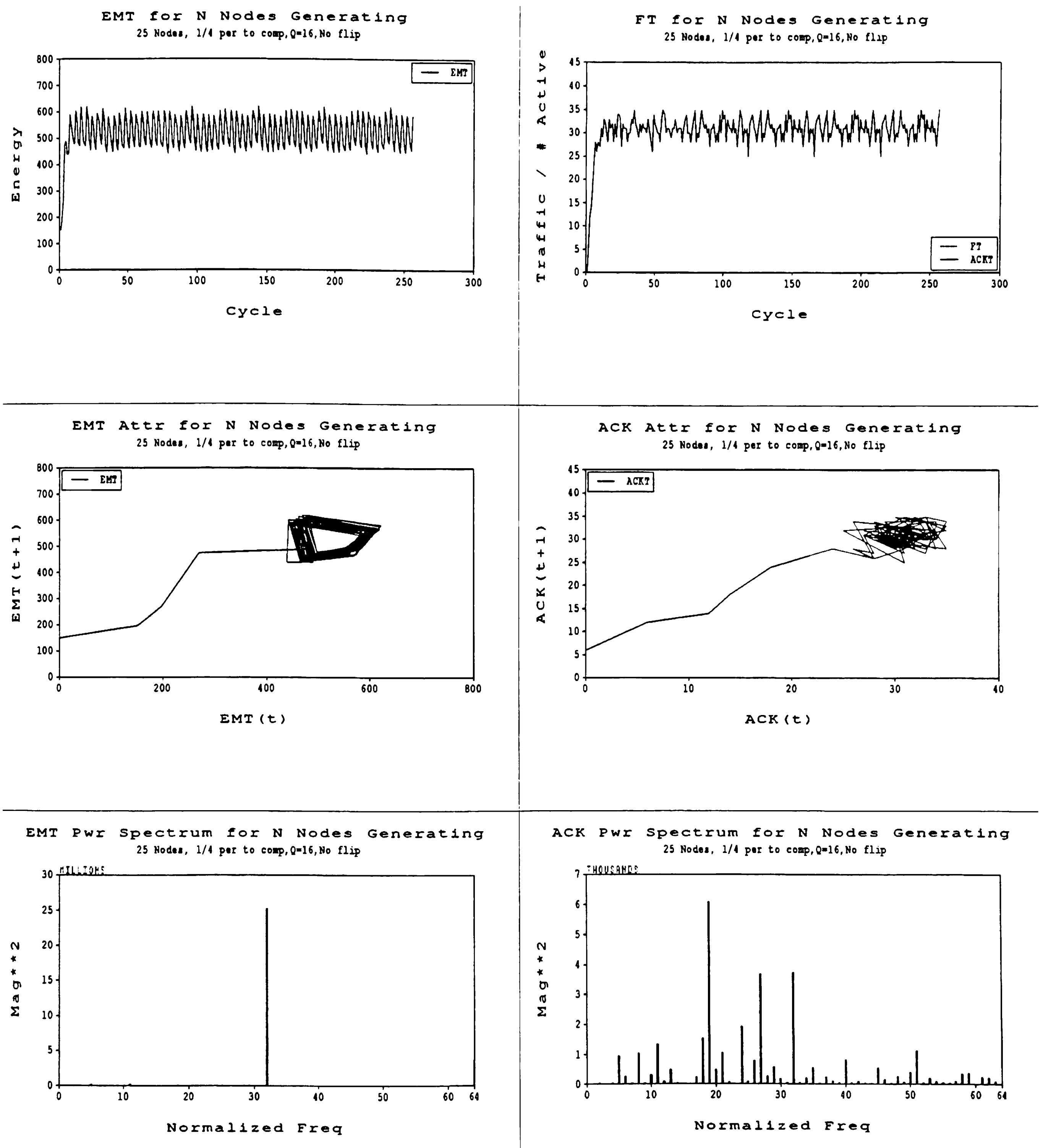

Figure 5: DRIVE1, 25 Nodes, 1/4, No-Flip

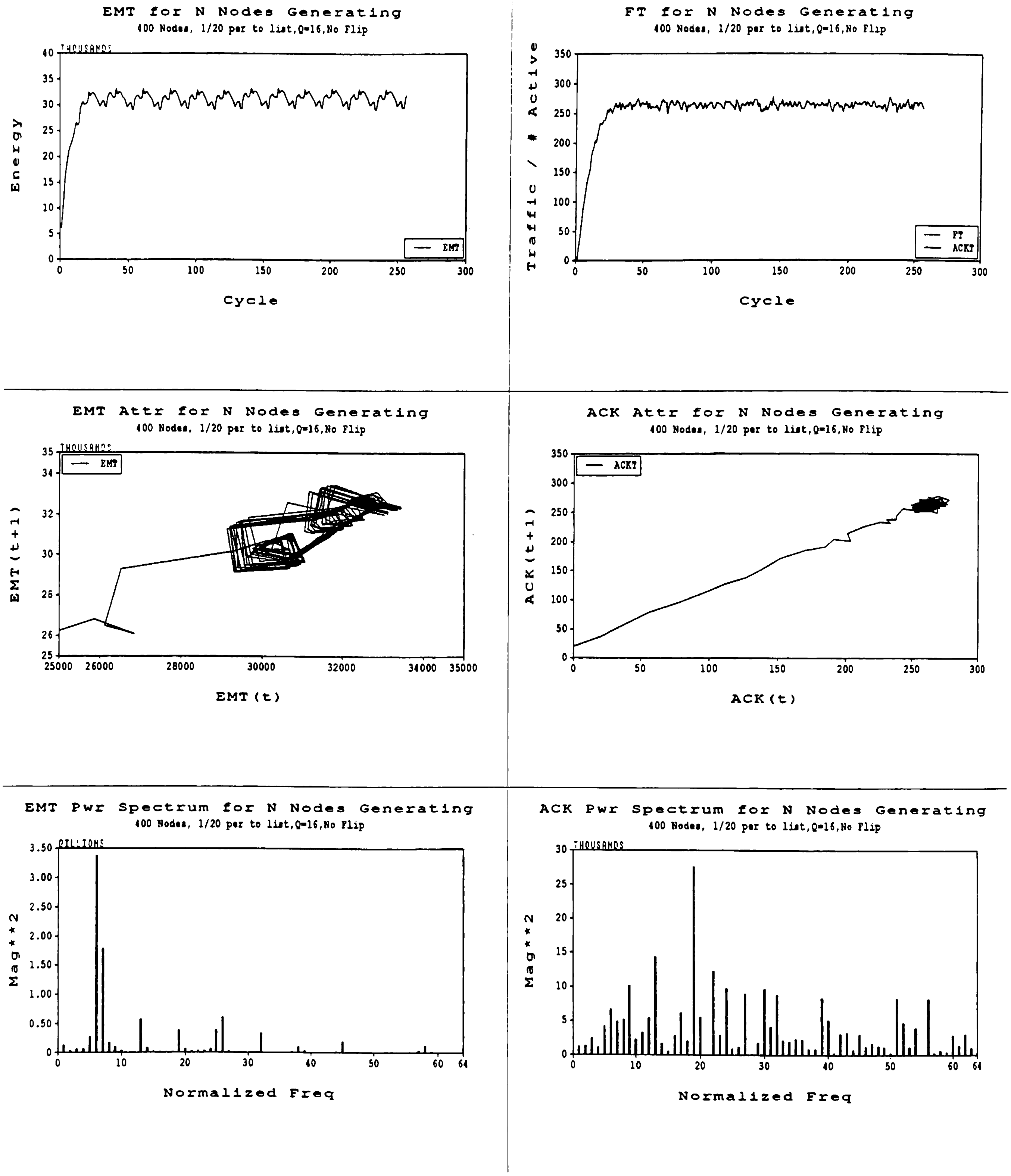

Figure 6: DRIVE2, 400 Nodes, 1/20, No-Flip

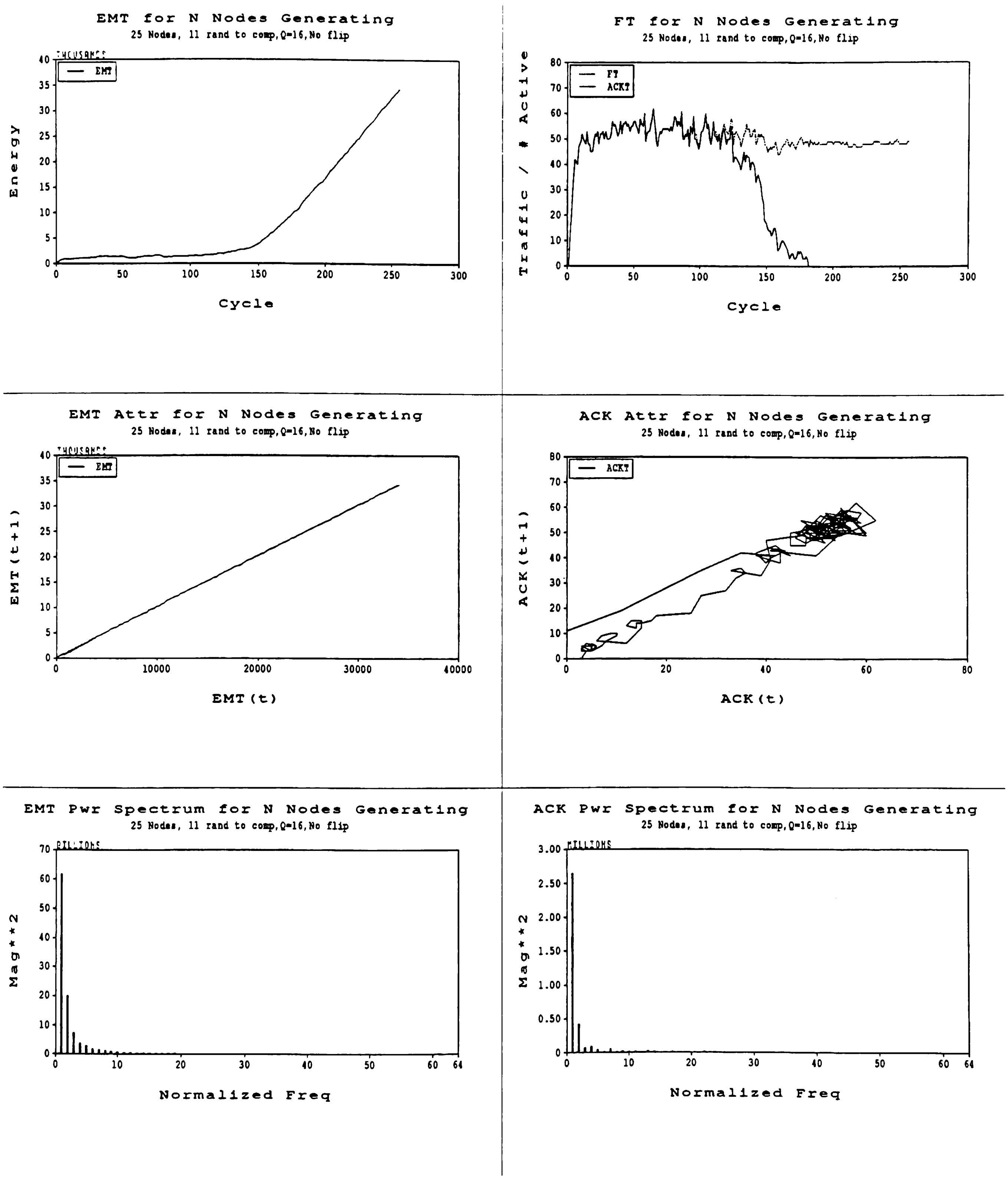

Figure 7: DRIVE3, 25 Nodes, 11, No-Flip