# Cycle Breaking in Wormhole Routed Computer Communication Networks

Mehmet Mustafa
*Verizon Laboratories*
*40 Sylvan Rd, Waltham, MA 02451*
*E-mail: mehmet.mustafa@Verizon.com*

Mark Karpovsky, Lev Levitin
*Boston University, College of Engineering*
*8 St. Mary's Street, Boston, MA 02215*

## Abstract

Because of its simplicity, low channel setup times, and high performance in delivering messages, wormhole routing has been adopted in second generation multicomputing environments [1-3]. Furthermore, irregular topologies formed by ad-hoc interconnection of low cost workstations provide cost effective alternative to massively parallel computing platforms. Switches used in these networks of workstations, NOWs, implement wormhole routing [1, 12]. However, due to a number of channels being held up while requesting others, wormhole routing is susceptible to deadlocks. In this paper we investigated deadlock-free routing algorithms in wormhole routed irregular network topologies. We used a modified version of the Turn Prohibition algorithm [4] to break cycles and prevent channel deadlocks. We then used shortest path algorithm to determine the routing tables for each computational node in the topology, avoiding prohibited turns along the paths from source to destination. We used EMA to import topologies into Opnet and simulated for message delivery using both our approaches and for the competing Up/Down [1] approach. We then repeated this sequence for hundreds of different topologies and determine the average latencies for all algorithms. Our results show that the modified turn prohibition based routing has outperformed both, the original Turn Prohibition and the Up/Down algorithms.

## Introduction

Because of its simplicity, low channel setup times, high performance in delivering messages, wormhole routing has been adopted in second generation multicomputing environments [3]. Because of their incremental scalability, workstation clusters, also known as Network of Workstations, or NOWs, have enjoyed considerable popularity [3, 5, 6]. However providing, deadlock-free routing in irregular networks has proven to be a difficult problem. Many routing algorithms for regular topologies, such as meshes, hypercubes, tori etc, have been developed providing deadlock free, low latency message delivery [6-11]. Providing deadlock freedom in irregular topologies with low latencies for delivering messages, in a cost effective manner is not trivial [5]. Authors in [5] assumed virtual cut-through switching in the routers which had no routing tables. All worms or messages use wormhole routing until it is blocked at a router. Upon blockage, the message is absorbed in its entirety at the router and forwarded later when the requested outbound channel gets freed up. In their approach, authors used a spanning tree for delivering messages, hereby avoiding deadlocks. However, any spanning tree based approach is not very efficient in terms of link utilization. In a network of $N$ nodes only $N-1$ links are used with root node links being most heavily loaded. The Up/Down routing approach, first reported in [12], is also a spanning tree based approach. However, in Up/Down routing, nodes are labeled in a partial order with the root having the smallest label and the leaves having the largest labels. Authors prescribe direction to both tree links and non-tree links, latter referred to as cross-links. Routing is then permitted to follow a path formed by zero or more up-links followed by zero or more downlinks. One disadvantage of this approach is that the construction of the best spanning tree is a hard problem. As in any spanning-tree based approach, links of the root node usually are most heavily used. In addition, given a topology and a spanning tree, selection of the root node would result in unpredictable routing performance [14, 19].

Turn prohibition was first reported in [10, 13], where turn model was thoroughly investigated for multi-dimensional meshes, and some turns were prohibited to prevent all possible deadlocks. Authors considered only 90 degree turns which are sufficient for meshes to prevent deadlock formation. In [4, 14-18] authors generalized the notion of turn, and developed an algorithm to construct minimal sets of prohibited turns, to break all cycles and prevent deadlock formation. Authors also established that the fraction of prohibited turns could be used as one of the criteria of efficiency of a routing strategy. In [19] authors extended the use of turn prohibition as described in [4] to general topologies and applied the Network Calculus techniques, which, until then, used to be strictly for feed-forward routing networks.

In this paper, we modified the selection rule in the original Turn Prohibition algorithm, to investigate if the fraction of prohibited turns could be further improved. Motivation for this investigation is shown in **Figure 1**, adopted from [20] in which, we show the percentage gain in maximum sustained throughput versus the percentage improvement, i.e. reduction, in the fraction of prohibited turns. This plot, which is approximately linear, predicts an improvement of 7.7% in the saturation point per 1% reduction in the fraction of prohibited turns. In order to provide an unbiased comparison of competing routing algorithms, we developed a wormhole node model in Opnet, where flit level simulation has been used to determine the message latencies, and the saturation points for each topology and each algorithm. In our implementation of the model, we assume only one flit storage at the router per input port. We use routing tables constructed for each routing algorithm to determine which output port to use to forward the flits. Since our interest is to compare only the impact of fraction of prohibited turns on message latency, we assumed that routers are ideal, making atomic routing decisions with no table lookup latency.

In the rest of the paper, we briefly present the mathematical theory of turn prohibitions and the modified turn prohibition with examples, properties and complexities, and high level simulation results in the next section. In the subsequent section we discuss Opnet models for the wormhole node, random topology generation, use of EMA to import the topologies into Opnet environment, and performance enhancements for simulation. We present our simulation results in the subsequent section followed by conclusion and recommendations for further investigations.

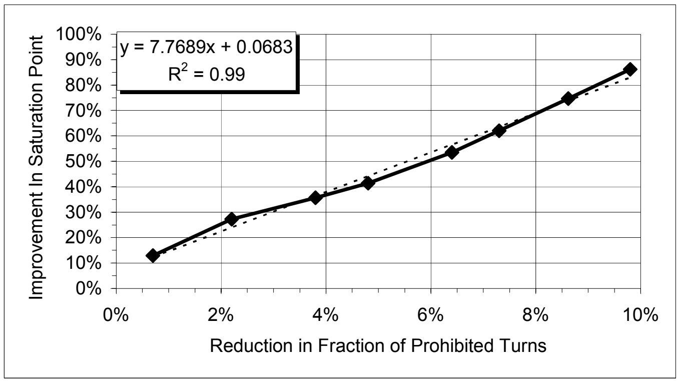


**Figure 1 Improvement in the Saturation Point with Reduction of Fraction of Prohibited Turns as Adopted from [20]**

## Turn Prohibition: A Mathematical Model

In this section we provide a brief and simplified overview of Turn Prohibition [4]. We assume that the network topology is represented by an undirected, connected graph $G=(V,E)$ of $N=|V|$ nodes and $M=|E|$ edges. A turn is defined as a three-tuple $(a,b,c)$, where $a,b,c \in V$ are nodes in the network in which $(a,b),(b,c) \in E$ are edges in $G$ incident on node $b$. We assume that turns are symmetric in that if a turn $(a,b,c)$ is prohibited so is the turn $(c,b,a)$. In enumerating turns, we count turns $(a,b,c)$ and $(c,b,a)$ as one and the same. If a node $a_i$ has degree $d_i$ then the total number of turns in the graph is $T(G)=\sum_i \binom{d_i}{2}$, where the summation is taken over all nodes. Not surprisingly, all deadlock free routing algorithms have prohibited turns. For example in spanning tree based routing algorithm, only turns along the tree are permitted and the rest of the turns are prohibited. In the Up/Down routing algorithm, after the spanning tree is constructed and nodes are labeled in partial order, in which the root has the smallest label, turn $(a,b,c)$ is prohibited if label of node $b$ is greater than the labels of both nodes $a$ and $c$. In the simplified version of the Turn Prohibition approach which is fully described in [4], first a minimum degree node is selected which has neighbors with the largest degree sum. According to the original Turn Prohibition algorithm, authors minimize the fraction $\frac{d_a(d_a-1)/2}{\sum_{i \in nbors} d_i}$, where the summation is over all neighboring nodes of the node $a$ that is being considered. According to the theory, when such a minimum degree node $a$ is found that satisfies the selection criterion, then all turns $(b,a,c)$ at the selected node are prohibited, and all turns $(a,b,c)$ starting with the selected node $a$ are permitted. Therefore, by minimizing the fraction, algorithm attempts to minimize the number of turns prohibited while simultaneously maximizing the number of permitted turns at every step of the algorithm. Application of the original Turn Prohibition algorithm is shown in **Figure 2**, where the prohibited turns are shown as arcs between the involved edges at the selected nodes. Node labels indicate the selection order. According to the original Turn Prohibition algorithm, node $0$ is selected first and all three turns at this node are prohibited and ten turns starting with this node are permitted. For example, turns $(1,0,3),(1,0,4)$, and $(3,0,4)$ are prohibited and turns $(0,1,3),(0,1,2),...$ are permitted. At the end of the algorithm, this approach prohibits nine turns as shown. With 37 total turns in the graph, fraction of prohibited turns becomes $z=0.243$.

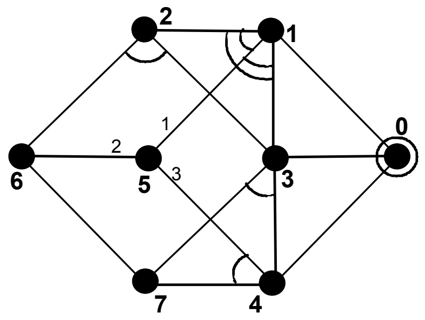


**Figure 2 Turn Prohibition Applied to a Simple Graph**

Modified Turn Prohibition changed the selection rule in the following way. Instead of selecting a node that minimizes the fraction, the modified selection process seeks a node with minimum degree, which has at least one neighbor with the smallest degree. For example in **Figure 2,** nodes *0, 2, 5, 6,* and *7* are all degree-3 nodes but only nodes *2, 5, 6,* and *7* have neighbors with degree-3 nodes. On the other hand, the minimum degree neighbor of node *0* is 4. The reason for selecting the node with minimum degree neighbors is a greedy reasoning in the following sense. *We select a node $a$ with minimum degree at the current iteration, such that, when the selected node is deleted, the remaining subgraph $G-a$ will be the best one for the next iteration of the algorithm.* We see, for example that, the original Turn Prohibition selection criterion during the first iteration selected node *1* which, when deleted would leave behind a subgraph with only degree-3 and degree-4 nodes. This means that the best that can be done during the next iteration is to select a degree-3 node with three prohibited turns. With the modified selection rule, any one of the four nodes *2, 5, 6,* or *7*, the subgraph would have at least one degree-2 node and therefore the next iteration of the algorithm would have to prohibit only one turn. This selection rule having one-step look-ahead feature promises to provide a smaller set of prohibited turns without adding any additional complexity to the original algorithm. Obvious extensions to the modified selection would be to have *k-step* look-ahead with the additional complexity at each additional step of the look ahead. For example, we can consider *2-step* look ahead where when we discover that there are multiple nodes satisfying the *1-step* look-ahead rule. Following the modified turn prohibition approach, one selection order would be *6, 2, 5, 7, 1, 0* with only eight prohibited turns, and a fraction of prohibited turns of $z = 0.216$, which is 12.5% smaller than $z = 0.243$. It should be noted that the modified turn prohibition algorithm is fundamentally same as the original algorithm. Both seek to identify a *minimum degree* node. If there are multiple minimum degree nodes then heuristics are used to identify which node among the set one should select. Original turn prohibition algorithm selects the node which has the largest degree neighbors, and the modified turn prohibition algorithm selects the node with minimum degree neighbors. Because of this, the properties and complexities of both algorithms are identical as discussed next.

**Properties of Turn Prohibition**

Before we dwell into the experimental work, we review the properties of Turn Prohibition. Properties and proofs of the properties of the original Turn Prohibition algorithm have been discussed in great detail in [4, 14, 17-20] and for space considerations will be listed here without any proofs. Reader is encouraged to refer to cited papers for the proofs. Both the original and the modified Turn Prohibition algorithms create a set $Z(G)$ of prohibited turns. Following properties can be proven for the set $Z(G)$:

(a) Any cycle in $G$ has at least one turn included in $Z(G)$.

(b) The set $Z(G)$ is irreducible; deletion of any turn from $Z(G)$ to obtain a new $Z'(G)$ will create at least one cycle in which would have no turns included in the $Z'(G)$.

(c) Connectivity of the is maintained after prohibiting the turns in $Z(G)$.

(d) $|Z(G)| \le |T(G)|/3$ where $T(G)$ is the set of all turns in graph $G$.

(e) Computational and memory complexities of the Turn Prohibition algorithm is $O(N^2\Delta)$ and $O(N)$ where $N$ and $\Delta$ are number of nodes and the maximum node degree in the underlying graph.

First property (a) implies that all cycles in the given graph are broken. Second property (b) asserts that the set of prohibited turns is irreducible but it does not claim that the set $Z(G)$ is of minimum cardinality. Property (d) provides a non-trivial upper bound for the fraction of prohibited turns to be 1/3. It should be noted that this upper bound on the fraction of prohibited turns applies only to the family of complete graphs $G = K_n$.

**Routing Table Construction**

Since Opnet environment is used for message delivery in the randomly generated topologies, it is necessary to discuss the process of generation of the routing tables. Briefly stated, the construction of the routing tables involves identification of the shortest path between all node pairs that include no turns from the set $Z(G)$. As discussed in [20] in detail, each router is aware of its permitted and prohibited turns. This information is represented by a square matrix $P$ of $d+1$ where $d$ is the degree of the node. In our models port 0 is assumed to be the local consumption/injection channel to the local processor and that all turns from the local port 0 to any one of the remaining $d$ output ports. Matrix $P$ is such that at node $a$, $P_a(i,j)$ is 1 if the turn $(i,a,j)$ is permitted and 0 otherwise, where $i,j$ are the input and output ports respectively. It should be noted that $P$ is symmetric and that $P_a(0,j) = P_a(i,0) = 1$. Algorithm constructs two tables for each node; the routing table $R_a(i,n)$ and the distance table $D_a(i,n)$, where, $i = 0,...,d_a$ are the input ports and $n = 0,...,N-1$ are the destination node numbers. We note that since routing table generation depends on the prohibition matrix $P$, it can also be used for Up/Down approach or any other algorithm for which the prohibitions can be generated. In fact we used the same routing table generation algorithm in our work comparing the performances of Turn Prohibition, modified turn prohibition and Up/Down algorithms. Instead of describing the algorithm in detail here, we chose to sketch its operation and demonstrate it with an example. For more details readers should refer to [20]. Routing table contains the output port numbers for permitted turns and -1 for routes that are not available. For example in **Figure 2**, the routing table entries for node 1 will mostly be -1, since this node is not a forwarding node due to the fact that all turns at this node are prohibited. Only the local channel messages could be routed at this node. Distance matrix $D_a(i,n)$ at node $a$ contains the lengths of the shortest paths in hops for messages arriving at input port $i$ that are destined for node $n$.

For an intermediate node $a$, the algorithm estimates the length of the shortest permitted path between adjacent nodes of $a$ and the destination node. Router at node $a$ then would route the incoming message from an input port $i$ to the output port $j$ provided that $P_a(i,j) = 1$. Here we assume that output port $j$ of node $a$, is the port connecting to adjacent node with the shortest path length to destination node. As an example, during the construction of the routing table at node 1, $R_5(1,4) = 3$ showing that for a message arriving from node 1 on the input port $i = 1$ to destination node $n = 4$, should be routed on output port $j = 3$.

**Table 1 Routing Table $R_5(i,n)$ at Node 5 of Figure 2**

| i \ n | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 0 | 3 | 1 | 2 | 3 | 3 | X | 2 | 2 |
| 1 | 3 | X | 2 | 3 | 3 | 0 | 2 | 2 |
| 2 | 3 | 1 | 3 | 3 | 3 | 0 | X | X |
| 3 | 1 | 1 | 2 | 2 | X | 0 | 2 | 2 |

In **Table 1** we show the routing table for node 5 of topology in **Figure 2** in which we also identified the port numbers for node 5. Port 0 for the local consumption/injection channel is not shown for clarity in the figure. It is interesting to note that a message arriving at node 5 from input port 2 destined for node 7 cannot be delivered and hence the $R_5(2,7) = X = -1$. This does not mean that messages are not deliverable. In fact, routing table at node 6 will be such that no message destined for node 7 will be forwarded to node 5.

The prohibition matrix for node 5 is all ones with the exception of entries such as $(i,i)$, where input and output port numbers are equal.

Distance matrix $D_5(i,n)$ for node 5 is shown below for reference. Path lengths for unreachable nodes are represented by X = -1 in the table.

**Table 2 Distance matrix $D_5(i,n)$ for Node 5 in Figure 2**

| i \ n | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 0 | 2 | 1 | 2 | 2 | 1 | X | 1 | 2 |
| 1 | 2 | X | 2 | 2 | 1 | 0 | 1 | 2 |
| 2 | 2 | 1 | 3 | 2 | 1 | 0 | X | X |
| 3 | 2 | 1 | 2 | 3 | X | 0 | 1 | 2 |

**High Level Simulations**

High level simulation experiments provided more convincing evidence that modified approach would reduce the fraction of prohibited turns. In **Figure 3**, we see that the modified approach, in all but very few topologies would result in better results. **Figure 4** shows the average percent improvement for a number of topologies. Given these observations, according to **Figure 1** we expect approximately 30% improvement in the performance with the modified turn prohibition when compared with the original Turn Prohibition algorithm.

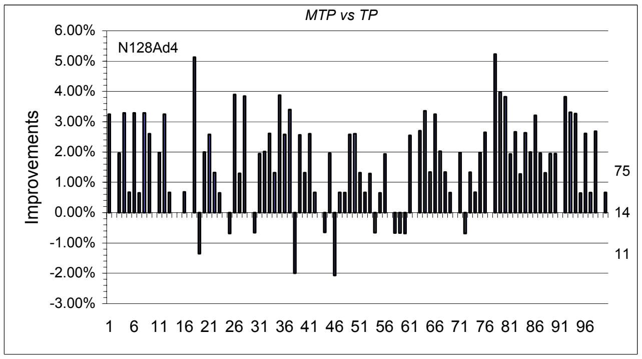


**Figure 3 High Level Experimental Results for 100 Irregular Graphs of 128 Nodes of Average Degree 4 Predicting an Expected Gain with the Modified Turn Prohibition Algorithm (MTP) over the Original Turn Prohibition Algorithm (TP).**

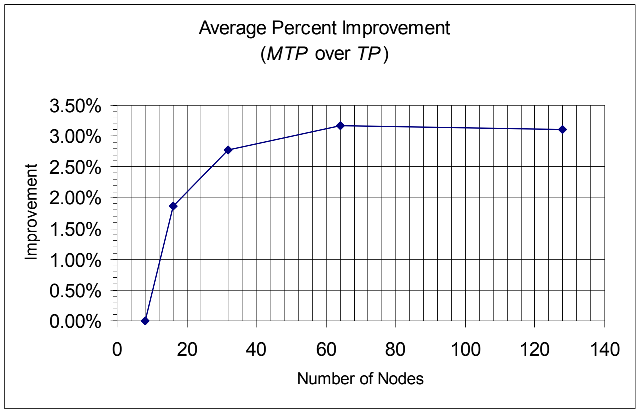
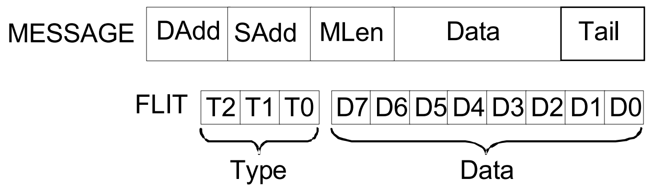


**Figure 4 Results Showing the Average Improvement of the Modified Approach**

**An OPNET Model for a Wormhole Node**

We used OPNET to develop a simulation model for the wormhole node. Structure of a wormhole message is depicted in **Figure 5**. A typical unicast message has three header flits, where the first flit is Destination Address Flit, the second flit is Source Address Flit and the third flit is the Message Length Flit. Header is followed by zero or more data/payload flits. Last flit in a wormhole message is a tail flit, which contains the last payload data for the message. We used an 11-bit long packet with two fields as our packet format simulating one flit. Three bits wide Type field identifies the type of the flit and the eight bits wide Data field is the flit payload.



| T2 | T1 | T0 | Flit Type |
|---|---|---|---|
| 0 | 0 | 0 | Data Flit |
| 0 | 0 | 1 | Dest Address Flit |
| 0 | 1 | 0 | Src Address Flit |
| 1 | 1 | 0 | Message Length Flit |
| 1 | 1 | 1 | Tail Flit |

**Figure 5 Wormhole Message Format, Flit Format and Flit Types**

**Wormhole Node Model**

A wormhole node consists of a router and a processor, the latter being modeled by two queues; PGQueue for generation of messages at the source node, and PSQueue, for consumption of messages at the destination node. Router module is responsible for forwarding all flits arriving at the input ports to the appropriate output ports. Wormhole handshaking [11] is implemented using the statwires within the node, and remote interrupts with adjacent nodes. Reception of a stream interrupt is interpreted as the arrival of a flit. If a flit arrives from the local injection channel from the PGQueue module, router uses the statwire to acknowledge the receipt and completion of handling of the flit. When PGQueue module receives a statwire event, it retrieves the next flit of the message from the queue and sends it to the router. When the tail flit arrives from an adjacent router, a remote interrupt is used to acknowledge that router is ready for the next flit.

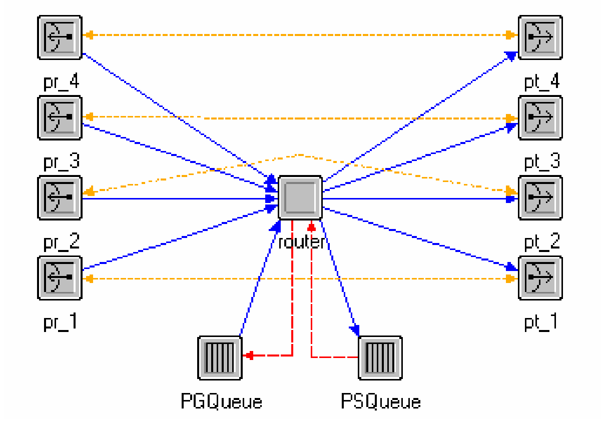


Figure 6: OPNET Node Model for a Four-port Wormhole Node

When the Destination Address Flit arrives at a router, it is stored temporarily in the flit buffer. Reference to the routing table identifies which output port to use. If the output port is busy, the header flit is blocked and no acknowledgement is sent to the sender. When the output port is finally freed up, it is associated with the waiting header flit. If multiple header flits are waiting for the same output port that has just been freed up, binding is done on a FIFO basis.

In this environment, sending a 200 flit long message would involve sending 200 packets; a very inefficient use of resources. Because of this we incorporated an optimization mechanism, in which, source node processor generates only as many flits as necessary for the Source Address flit to arrive at its destination. Once the destination is reached, the destination node, knowing the sending node address, sends a remote interrupt to the source node. When this remote interrupt is received, source node generates a tail flit, stores it in the queue and schedules a self interrupt. Scheduling of the self interrupt is based on number of flits already generated, message length, and the flit transfer time, or the flit transmission time. When this tail self interrupt arrives, the tail flit is transmitted to the router from the PGQueue. This optimization significantly improved the simulation run time since for each message of 200 flits; only five to ten flits per message are generated and processed.

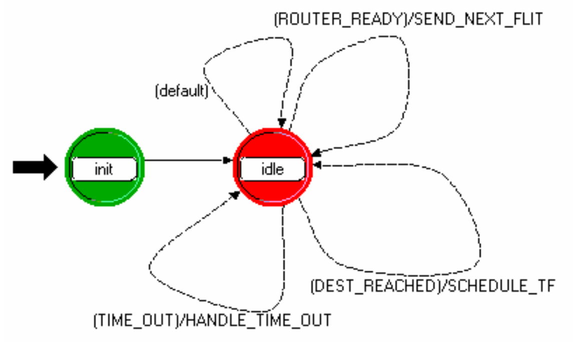

(ROUTER_READY)/SEND_NEXT_FLIT
(default)
init
idle
(DEST_REACHED)/SCHEDULE_TF
(TIME_OUT)/HANDLE_TIME_OUT

Figure 7 Process Domain Representation of the PGQueue Module of the Processor

**PGQueue Module**
In **Figure 7** we show the process model of the PGQueue module of the wormhole node. The forced init state performs the one - time setup necessary at the module, reads in the values for the run-time attributes, schedules a self interrupt for generating a message, and transitions into the only blocking state labeled *idle*. In the *idle* state, if ROUTER_READY event is triggered, it implies that the local router has just sent an acknowledgement via the statwire. In this case the PGQueue process sends either a flit already in the queue, or if necessary generates one and transmits it to the router. In our implementation, we use the stream interrupt as the READY signal, and the statwire as the ACK signal of the wormhole handshake. When the DEST_REACHED event fires, it implies that we just received a remote interrupt from the destination, and we schedule a tail flit to be transmitted when self interrupt expires. When this self interrupt expires, the TIME_OUT macro is triggered, and the HANDLE_TIME_OUT is executed. In this function, the tail flit is transmitted if the interrupt code indicates so. Otherwise, time-out implies that time for generating another message has arrived and a new worm is generated.

**PSQueue Module**
As shown in **Figure 8**, the process model of the PSQueue of the wormhole node processor is very simple. Again, all of the one-time, module level processing in done in the forcing state called *init*. In the blocking state labeled *idle,* flits that arrive from the local router are handled. Arrival of a flit is triggered by the ARRIVAL event, which causes the RECEIVE_FLIT exec to run. When run, it stores the flit in the receive queue using the wormhole handshaking with the local router. If the incoming flit is a Source Address Flit, then a remote interrupt is sent to the PGQueue of the source node. If the incoming flit is a tail flit, then end-to-end delay is computed and saved and message in the PSQueue is discarded. One other activity that takes place in RECEIVE_FLIT is monitoring the attainment of network stability. PSQueue process, computes the cumulative average with every message, and if the stability criterion is reached the simulation is terminated. Since we are interested in running many simulations on many different topologies, unattended by the user, we opted to determine the stability condition in this process. We compute the percent difference between the current and the previous values for the cumulative running average. If this difference is less than $\pm 0.1\%$ for 300 consecutive messages, we assume stability is attained and terminate the simulation. **Figure 9** shows the time evolution and attainment of network stability at low message injection rates, requiring about 1000 messages.

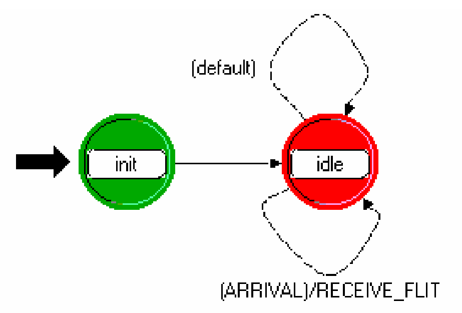


Figure 8: PSQueue Module of the Processor

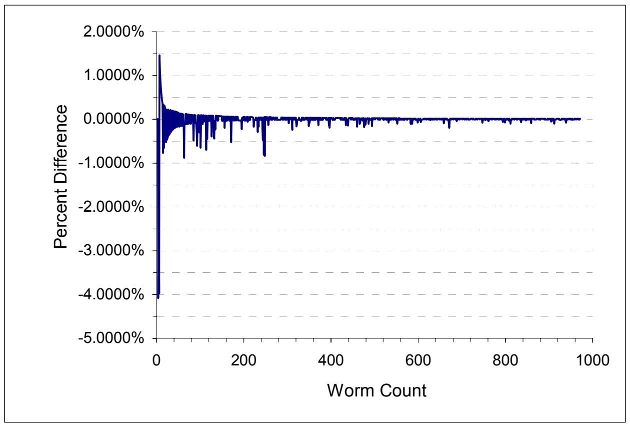


**Figure 9 Monitoring the Attainment of Network Stability in PSQueue Module**

## Router Module

Router module process model is depicted in **Figure 10**, where we have two blocking states and a forcing state. In *Init* state, router identifies its node number from its name, reads in the run-time attribute values, identifies the attached PGQueue and PSQueue objects, initializes all state variables, dynamically allocates memory for routing tables, and reads in the node specific routing table. It then transitions to the *Identify* blocking state. In this state, each node identifies its neighbors by sending just a Source Node Address Flit to each of its active ports. When the router receives all responses from its active ports, it schedules a self interrupt with no delay and transitions to the next blocking state called *Listening*.

In *Listening* state, FLIT_ARRIVAL is defined to be a stream interrupt and all flit types are processed by a function represented by the HANDLE_FLIT macro. This function calls other procedures that handle individual flit types. For example, if the flit type is a Destination Node Address flit, then routing table is referenced to identify the output port that flit needs to be transmitted out of. If the output port is busy then process of handling the waiting header flit is blocked. When a tail flit is transmitted out on any busy output port, then list of waiting headers is examined. If there is a header flit waiting for the output port that has just been freed up, then the input port that it came from is associated with the, now free, output port and binding takes place. When the HANDLE_FLIT macro, identifies that the destination node address is equal to its own node address, then the local output port to the PSQueue is bound to the port delivering the flit. From now on, all incoming flits and the tail flit from the associated input port are sent out to this output port. With each transmitted flit, router process will send a stat wire to the PGQueue object and will be awaiting a remote interrupt from the adjacent router. As discussed earlier, wormhole handshaking takes place at the router at three interfaces; first between the two adjacent routers, second between the PGQueue and the local router, and third between the PSQueue and the local router.

The transition event called ACK is defined to be a remote interrupt from an adjacent node. Router process identifies the adjacent node by the interrupt code and interprets it to mean that remote node is ready for the next flit. All remote interrupts are processed by the function defined in the HANDLE_ACK macro.

PSQUEUE_READY transition event is defined to trigger when a statwire interrupt is received from the PSQueue module, indicating that it is ready for the next flit. This interrupt is managed by the function represented by the HANDLE_SWIRE macro.

The TIME_OUT macro is for debugging purposes and is not used.

## Topology Generation

One important requirement in our research is the generation of large number of random topologies with given specifications. We developed a topology generator which is capable of generating topologies with given average and standard deviation for the node degree, number of nodes and the bisection width for the topology. When the standard deviation for the node degree is zero, then each node in the generated topology is of fixed degree.  We can also generate topologies that have no bisection width constraints imposed. The generator creates a text file for the underlying graph in the successor list form.

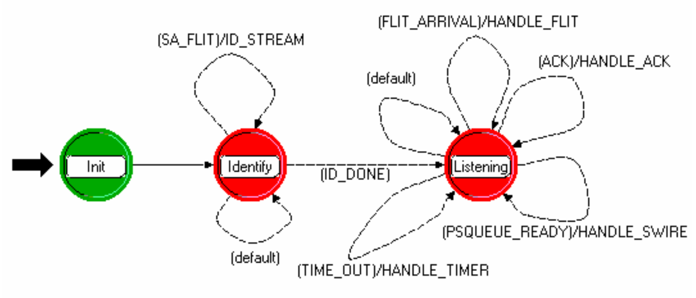


**Figure 10 Process Model for the Router Module in the Wormhole Node**

EMA tool provided the necessary bridge between the topologies generated outside into networks recognized by Opnet. An EMA program was created for this purpose that reads in the network topology and connectivity information from the text file generated by the topology generator. It uses the wormhole node and wormhole link model files to create the subnet network model. Nodes are positioned on a square grid at such small node to node distances that propagation time is insignificant compared to transmission time. A typical subnet network model created following this process is shown in **Figure 11**. Even though node_14 and node_15 are shown next to each other, in fact these two nodes are not neighbors in the topology. This figure is one of the topologies for a 64 node network with average node degree of four, node degree standard deviation of one, and a bisection width of two. Bisection width is the minimum number of links that when deleted separates the graph into two equal sized sub-graphs.

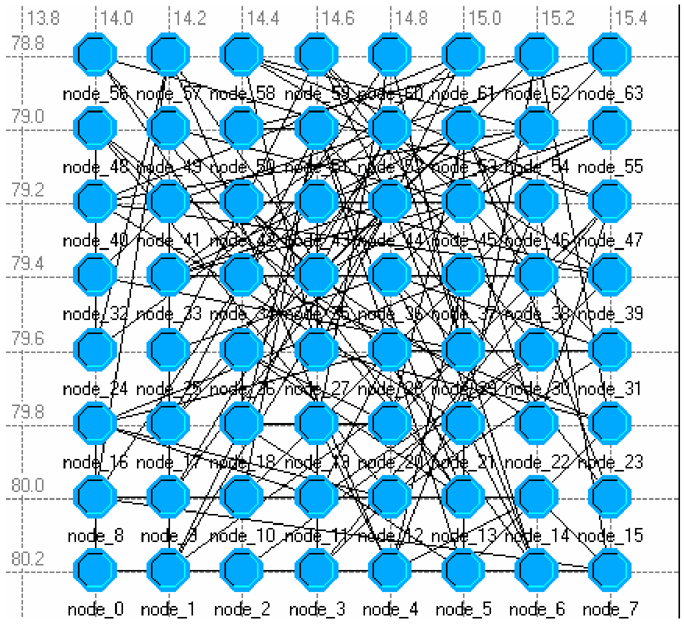


Figure 11: Subnet Network Model Imported into Opnet by Importing the Scenario Generated by EMA

## Simulation and Experimental Results

We used Perl scripts to generate different graphs and their Opnet subnet models using EMA. We then used the original Turn Prohibition algorithm to generate set of prohibited turns and the routing tables. Using Perl script, we simulated the network using the *op_runsim_opt* with required command line parameters and generated both the vector and scalar output files. For all of our simulations, we used 200 flits long messages. Uniform traffic model was used for selection of destination nodes, in which each node has the same probability of being selected as a destination. We identified the saturation point as the message generation rate, at which the average latency is $2 \times 10^2$ times the low injection latency. We then used the *op_cvos* command to convert each scalar file to GDF text file. A utility program has been written to parse the GDF text files to identify the saturation points for each topology. We determined the distribution of the saturation points for each algorithm and plotted the results. In **Figure 12** we show the distribution of the saturation points for Turn Prohibition (TP), Up/Down (UD) and Modified Turn Prohibition (MTP), for one family of 100 randomly generated topologies, each with 64 nodes of average degree four and bisection width of eight. In the figure, we show the saturation points on the horizontal axis in units of $10^3$ [worms/(sec.node)]. The vertical axis shows the number of topologies in the listed bins. For this specific case the mean of the distributions are 53.9, 61.0, and 88.4 for Turn Prohibition (TP), Up/Down (UD), and Modified Turn Prohibition (MTP) respectively. In **Figure 13**, we show the saturation points versus the minimum bisection width. One interesting observation is that for bisection width less than ten, Up/Down algorithm performs better than the original Turn Prohibition algorithm. For the topologies that we investigated, performance of the Up/Down algorithm worsens after bisection width of ten. In all cases, the average performance of the modified turn prohibition algorithm is considerably and consistently better than the other two. Percentage improvement attained by the modified turn prohibition over the original Turn Prohibition approach is 26% at bisection width of two, which peaks at 40% at bisection width of six.

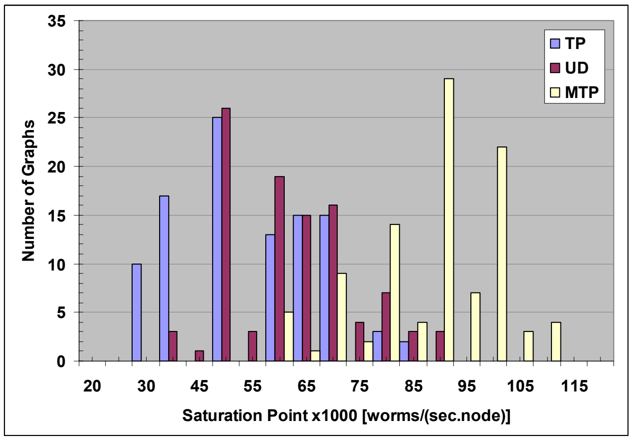


Figure 12 Distributions of the Saturation Points for the Three Algorithms for 64 Node Graphs for Bisection Width Eight

To explain the observed behavior of the Turn Prohibition and the Up/Down algorithms, we computed the average distances in these topologies, first without any prohibition (NP), then with Turn Prohibition (TP), then with Up/Down (UD), and finally with modified turn prohibition (MTP) algorithms. Our results for average distances shown in **Figure 14** have similar behavior and a crossover at about the same bisection width. For these topologies, the Turn Prohibition introduces larger dilations than the Up/Down algorithm for bisection widths smaller than twelve. However, for bisection widths lager than fourteen, dilation introduced by Up/Down algorithm is larger. We believe that, the observed behavior is due to the worse dilation performance of the original Turn Prohibition algorithm for topologies with small bisection widths. We note that the modified turn prohibition has the best dilation properties for the entire family of topologies that we investigated.

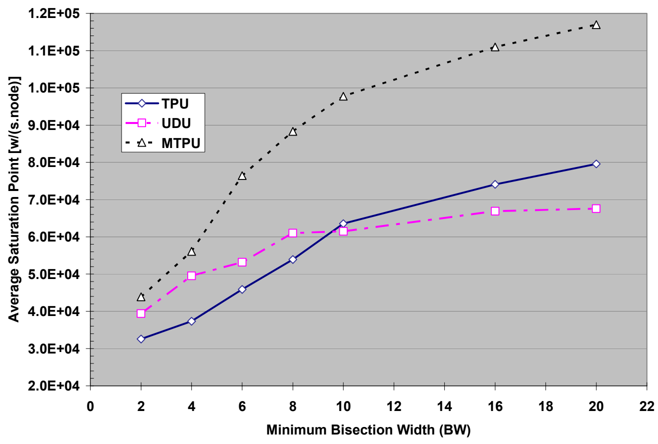


Figure 13 Simulation Results Comparing Turn Prohibition, Up/Down and Modified Turn Prohibition Saturation Points

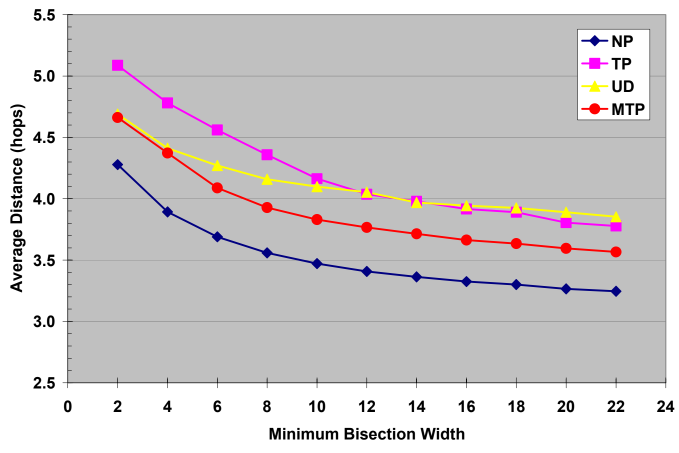


Figure 14 Computed Average Distances vs. Bisection Width

## Conclusions

In this paper we developed Opnet based tools and investigated three competing routing algorithms in irregular network topologies. Specifically, we validated that as the fraction of prohibited turns is reduced there is a significant improvement in average message delivery times. We have shown that by modifying the node selection rule in the original Turn Prohibition algorithm, we achieved significant improvement in increasing the maximum sustainable throughput. Modified turn prohibition provided no less than 26% and up to 40% improvement over the original Turn Prohibition algorithm, and up to 42% improvement over Up/Down algorithm. Inconsistent average distance dilation behavior of the original Turn Prohibition algorithm has not only been improved, providing consistency for all of the graphs that we investigated, but also providing consistently smaller average distance dilations. By breaking all cycles by means of prohibiting a small subset of turns in a given irregular network, we proactively prevented all deadlocks in wormhole routed networks.

Our next investigation would be to extend these ideas into multicasting in wormhole routed computer networks.

## References

[1] N. Boden and e. al. "Myrinet: A Gigabit per second Local Area Network," IEEE Micro pp. 29-35, 1995.

[2] R. Horst, W. "ServerNet(TM) Deadlock Avoidance and Fractahedral Topologies," Proc. of IEEE Int. Parallel Processing Symp. pp. 274-280, 1996.

[3] F. Silla and J. Duato "High-Performance Routing in Networks of Workstations with Irregular Topology," IEEE Trans. on Parallel and Distributed Systems vol. 11, no. 7, pp. 699-719, 2000.

[4] L. Zakrevski "PhD Thesis: Fault-Tolerant Wormhole Message Routing in Computer Communication Networks," College of Engineering pp. 21-27, 2000.

[5] H. Chi and C. Tang "A Deadlock-Free Routing Scheme for Interconnection Networks with Irregular Topologies," 1997 International Conference on Parallel and Distributed Systems pp. 88-95, 1997.

[6] W. Dally and C. Seitz, L. "Deadlock-Free Message Routing in Multiprocessor Interconnection Networks," IEEE Trans. on Comput. vol. 36, pp. 547-553, 1987.

[7] R. Boppana and S. Chalasani "A Comparison of Adaptive Wormhole routing Algorithms," Computer Architecture News vol. 21, no. 2, pp. 351-360, 1993.

[8] A.A. Chien and J. Kim "Planar Adaptive Routing: Low-Cost Adaptive Networks for Multiprocessors," Journal of ACM vol. 42, no. 1, pp. 91-123, 1995.

[9] J. Duato "A New Theory of Deadlock-Free Adaptive Routing in Wormhole Networks," IEEE Trans. on Parallel and Distributed Systems vol. 4, pp. 1320-1331, 1993.

[10] C. Glass and L. Ni "The Turn Model for Adaptive Routing," Proc. of the 19th Annual Int.. Symp. on Computer Architecture pp. 278-286, 1992.

[11] L. Ni, M. and P. McKinley, K. "A Survey of Wormhole Routing Techniques in Directed Networks," Computer vol. 26, pp. 62-76, 1993.

[12] M. Schroeder and t. al. "Autonet: A High-Speed self configuring Local Area Network Using Point-to-point Links," 1990.

[13] C. Glass and L. Ni "The Turn Model for Adaptive Routing," Journal of ACM vol. 5, pp. 874-902, 1994.

[14] L. Zakrevski, S. Jaiswal, L. Levitin and M. Karpovsky "A New Method for Deadlock Elimination in Computer Networks With Irregular Topologies," Pro. of the IASTED Conf. PDCS-99 vol. 1, pp. 396-402, 1999.

[15] L. Zakrevski and M. Karpovsky, G. "Fault-Tolerant Message Routing for Multiprocessors," Parallel and Distributed Processing pp. 714-731, 1998.

[16] L. Zakrevski and M. Karpovsky, G. "Fault-Tolerant Message Routing in Computer Networks," Proc. of Int. Conf. on PDPA-99 pp. 2279-2287, 1999.

[17] L. Zakrevski, S. Jaiswal and M. Karpovsky "Unicast Message Routing in Communication Networks With Irregular Topologies," Proc. of CAD-99 1999.

[18] L. Zakrevski, M. Mustafa and M. Karpovsky "Turn Prohibition Based Routing in Irregular Computer Networks," Proc. of the IASTED International Conference on Parallel and Distributed Computing and Systems pp. 175-179, 2000.

[19] D. Starobinski, M. Karpovsky and L. Zakrevski "Application of Network Calculus to General Topologies Using Turn Prohibition," IEEE/ACM Transactions on Networking vol. 11, no. 3, pp. 411-421, 2003.

[20] S. Jaiswal, L. Zakrevski, M. Mustafa and M. Karpovsky, G. "Unicast Wormhole Message Routing in Irregular Computer Networks," Twelfth IASTED International Conference on Parallel and Distributed Computing Systems - PDCS 2000, pp169-174, 2000.