Chaotic Routing: *An Overview*

Wajahat Qadeer and Rehan Hameed

I. Introduction

Of the many router designs, most can be classified as either *oblivious* or *adaptive*, depending on whether the path selection is statically determined based on the network topology, or whether dynamic information about congestion, priorities, or faults is considered. Adaptive algorithms can be broken down into two broad categories as well: *minimal* or *non-minimal*. Minimal adaptive algorithms allow only topologically minimal-length paths, while non-minimal algorithms allow a larger set of paths, possibly including paths of infinite length. A wide variety of mechanisms exist within each of the categories to provide reliability and performance.

Regardless of the method used, the goals of routers are similar. All routers attempt to deliver the greatest number of messages in the least amount of time. Performance parameters for a routing algorithm encompass latency and throughput. Throughput is defined as the rate at which a network delivers data whereas the time for a message to travel from its source to its destination is known as *latency*. In general for higher performance, latency should be small, and throughput large.

Congestion degrades both latency and throughput. Since adaptive routers incorporate congestion information in their routing decisions, they can typically achieve better performance compared to oblivious routing algorithms. Additionally, adaptive schemes inherently have support for building fault tolerant schemes. Minimal adaptive algorithms are simpler to implement but are not as effective as non-minimal adaptive schemes. Non-minimal schemes provide the most flexibility in dealing with congestion but routing decisions are more complex making them harder to implement in hardware. Additional complexity is added because these schemes also need to deal with deadlock, live-lock and out of order packer delivery. In contrast, in oblivious routing algorithm these problems either do no exist or are easy to handle.

Despite the improved performance of non-minimal adaptive schemes, they were initially not employed in practical hardware designs due to their complexity. Thus, vast majority of routers continued to use oblivious schemes. *Chaotic routing* was presented as an alternate non-minimal adaptive scheme, which could be implemented efficiently in hardware unlike its predecessors. With its simpler hardware, it was possible to have a router which performed well under congestion and also enabled good fault tolerance techniques.

In this paper we present an overview of the *Chaotic routing* algorithm, including its operation, performance analysis and fault tolerance techniques. The papers that we have selected echo this. The paper by *Bolding, Fulgham* and *Synder* describes the *Chaotic routing algorithm* and compares its performance with other schemes. The paper by *Chinn* offers a more detailed analysis of the chaotic router performance under worst case traffic permutations and the paper by *Bolding* and *Yost* discusses implementation of fault tolerance schemes on a chaotic router.

II. The case for chaotic adaptive routing [I]

In this paper, authors present the case for chaotic routing algorithm, suggesting it to be a feasible and efficient adaptive routing scheme which improves upon the shortcomings of other adaptive scheme that existed before it. A practical implementation of this scheme has been described and its performance and implementation complexity has been compared to other existing routing schemes.

In 1994, when this work was first presented, most of the existing router implementations were based on oblivious schemes. The few implementations of adaptive routing that existed were used in single instance machines such as HEP, CM-2, CM-5 etc making it difficult to separate fundamental properties of the routers from artifacts of the specific instances. Authors view this as a stumbling block in the adaptation of those schemes by practical designs. They aim that their implementation of chaotic routing and its analysis would instill enough confidence in chaotic routing for it to be employed in future designs.

A. Existing Routing schemes

An overview of existing scheme highlights the problems associated with each of them. Issues considered include performance under congestion, deadlock, livelock and complexity. Following section summarize the findings of the authors in this regard.

1) Oblivious routing

Oblivious routing schemes do not perform well under highly congested loads. For certain traffic patterns, such as nearest neighbor, which cause little congestion, these schemes perform very well. However for non-uniformm traffic and in the presence of hot spots, these schemes perform poorly. Randomized oblivious algorithms such as Valiant, improve in this area but double the average path lengths of messages. Deadlock is not a problem when implementing these schemes on open-ended k-ary n-cubes. However on networks with wrap around links, the deadlock prevention mechanisms degrade network performance.

2) Minimal Adaptive routing

Deadlock is a more serious problem for minimal adaptive networks. Avoiding deadlock requires either placing restrictions on routing options or using extra resources such as buffer space. Again network performance is disturbed. Minimal adaptive algorithms often employ many virtual channels, resulting in quite complex designs. Also minimal adaptive scheme are constrained when talking to destination

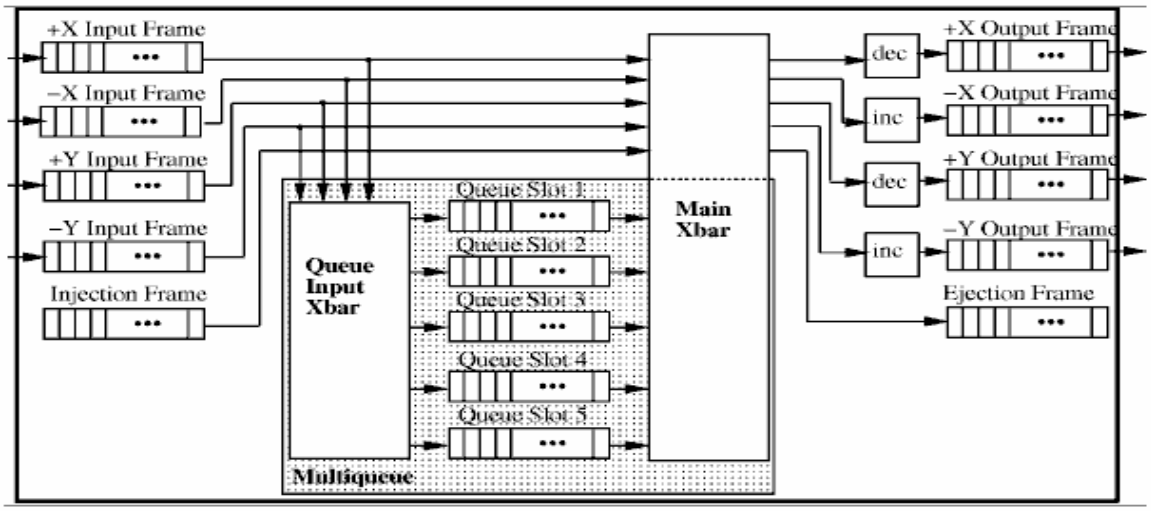


Figure 1: Chaotic router architecture

near them since there are not many paths available to maneuver around congestion.

3) Non-minimal adaptive routing

These schemes could be further subdivided into Deflection, Queuing and Wormhole schemes.

Deflection schemes are complex and their complexity grows with the node degree, limiting their performance in high degree networks. Also their mechanisms limit the flow control options to Store-and forward, which results in unacceptably long latency for most packets. The operation of this algorithm inherently makes sure that no deadlock arises. Wormhole routers also have very complex deadlock prevention mechanisms require restrictions on routing, multiple classes of virtual channels or both. Queuing routers have to deal with complex queue operations. Deadlock in queuing routers is avoided usually by packet-exchange algorithm.

Livelock is a serious problem for non-minimal routing algorithms. Either priority methods or randomization are used to avoid it. Priority methods use the ages of various networks to decide which ones could be sent on non-minimal paths. The age is determined by a time stamp or the number of times packet has been routed to a non-minimal path. These methods are complex to implement and also consume extra bandwidth as additional information must be transmitted as part of packet header.

Randomization is another livelock avoidance scheme. This is the one used in chaotic routers and would be explained in next section as part of discussion about chaotic routing.

B. Chaotic Routing

Chaotic routing belongs to the queuing class of non-minimal adaptive routers. Chaotic routing attempts to minimize the impact of the queue management overhead by eliminating it from the critical path of the routing decision. In Chaos routers (see Fig. 1), single packet buffers are placed on both the input and output sides of each external channel. In "normal" operation, packets enter into an input frame of the node, wait for an output frame of a profitable outgoing channel to become available, and move to the output frame in a virtual cutthrough fashion. Thus, the "core" of a Chaos router looks like a minimal adaptive router without the need for multiple classes of queues for deadlock prevention. Packets in Chaos routers are moved from input buffers into the central queue, called the multiqueue, on two occasions: packet exchanges for deadlock prevention and "stalling." The packet-exchange deadlock prevention protocol mandates that, if routers on either side of a shared link have packets to send to each other, both packets must be sent, rather than only one side sending a packet. In order to guarantee this, the Chaos router implements the following protocol: If a packet is sent to the output frame for channel *i* and the input frame for channel *i* has an incoming packet, the packet in input frame *i* is moved to the central queue, as illustrated in Fig. 2.

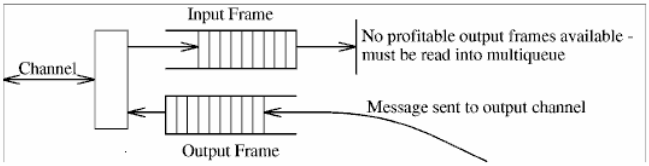


Figure 2: Input, output frame queues.

If there is no space in the queue, a packet is derouted from it to make room for the newly arriving packet. Packets are also moved into the queue when they become "stalled" at an input frame. The Chaos router determines that a packet is stalled when both its head and tail are buffered in the same input frame, i.e., it can no longer cut through. Whenever an output frame becomes available (empty), the router determines if there are any packets in the router that may be routed out the newly available output frame. The search for packets proceeds as follows:

1) If the queue is full, then a packet in the multiqueue is picked (randomly) to be routed to the output frame.
2) If the queue is not full, but a packet in the queue can be profitably routed out this channel, that packet is sent from the

queue to the output frame. If there is more than one such packet, priority is given to the packet that has been in the multiqueue for the longest time.
3) If no packets in the queue are to be sent to this output channel, but a packet in an input frame can be profitably routed out this channel, it is sent across the main crossbar to the output frame. If there is more than one such packet, a single packet is selected at random.

This *output-driven* search is initiated whenever a packet leaves the router through an output channel and frees up an output buffer. Because packets arriving in an input frame may find that a profitable output frame is immediately available, a newly-arriving packet will force all available output frames that are profitable to it to initiate an output-driven search as above. Since packets only enter the multi-queue when they have been delayed or when there is congestion in the router, most packets bypass the queue altogether, reducing the effect of the complexity of queue management. Also, separating the main routing mechanisms from the queue allows the lightly-loaded case, where latency is very critical, to proceed without delay while causing the heavily-loaded case, where intelligent buffer management is important, to suffer some extra delay in order to utilize the resources more efficiently.

C. VLSI Implementation of Chaotic routing

A VLSI implementation of this router was demonstrated to operate at 30 MHz in 1.2u technology. A 180 MHz version was expected to be released soon. A pipelined design was used which resulted in a minimum of 4 cycles of latency through the router. Further details of the internal design of this router have been published in [4], [5].

D. Comparisons Of oblivious, *-channels and Chaotic routing

Authors present a performance comparison of various routing schemes. In addition to oblivious and chaotic schemes, *-channels scheme has been compared, which is a minimal adaptive routing scheme. The comparison results have Following aspects of performance have been considered:

1) Uncongested Network Performance

With no congestion, even the chaotic scheme would always select the minimal paths. The latency for a packet for all three schemes has been modeled as follows:

c. $[(D + 1)d + L – 1]$

where c is the cycle time, D is distance in hops between source and destination nodes, L is the length of message in flits and d is the router delay in cycles. Out of these parameters only cycle time and router delay depend on the particular router used.

The speed of chip to chip interface imposes a fundamental lower bound on the network cycle time. Based on the existing chip designs and their own chaotic router design, authors conclude that all three schemes could be implemented such that they could achieve this lower bound for a given technology.

The router delay *d* essentially reflects the complexity of the routing decision. An oblivious router can typically manage *d* = 3. Chaotic router implementation has *d* = 4. No *-channels implementation was available but it has been estimated to have d = 4 as well.

Thus oblivious routers have lower router latency; however the serialization latency is similar. For reasonably long messages, the difference in overall latency would be insignificant.

2) Congested Network performance

Due to greater freedom in navigating packets around congestion, chaotic routing should achieve a better throughput then other two schemes. However, the analysis of congested performance is difficult for all three routers studied. Detailed simulation results that compare the routers on various simulated workloads are presented in later sections.

3) Fault tolerance

Again with greater freedom in routing packets around faults, chaotic routing is expected to provide better fault tolerance. A more detailed study of this has been presented in another paper reviewed later.

4) Out of order Packet (OOO) delivery

OOO packet delivery is a problem with packet based routing schemes including chaos. It is suggested that a hardware scheme to assemble these OOO packers into a message could be used which has been detailed in [6]

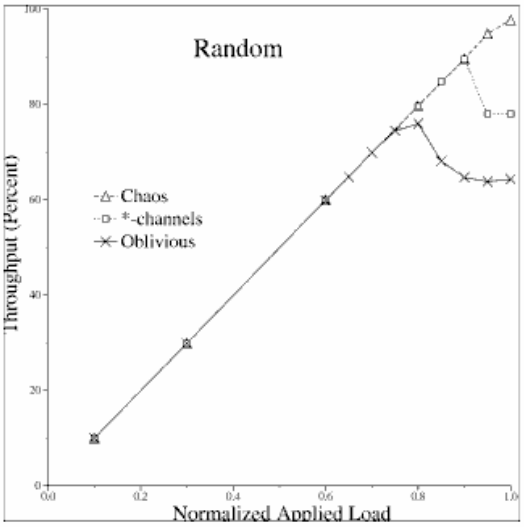
E. Simulation results

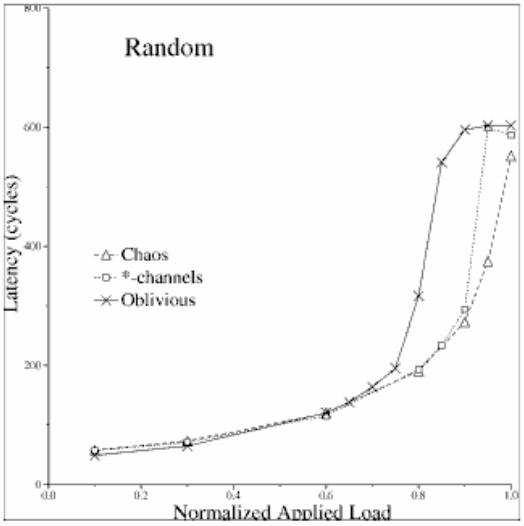
In this section, we present the results of simulating chaotic, oblivious, and minimal fully adaptive routing on torus networks. While Mesh networks presented some trouble to both adaptive schemes, authors believe that the additional links required to make a mesh into a torus add little cost so they concentrate on torus networks. Details of the simulation environment have been described in [7]

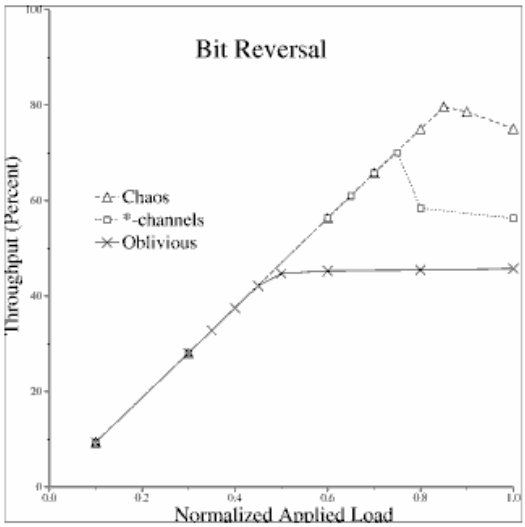
Traffic patterns simulated include Random, 4X hotspots, Complement, Transpose, Bit Reversal. The traffic patterns illustrate different features. As mentioned earlier, the random traffic is simply a standard benchmark used in network routing studies. The 4X hot spot traffic models cases where references to program data, such as synchronization locks, bias packet destinations toward a few nodes. The complement is a particularly difficult permutation, since it requires all packets to cross the network bisection in all topologies. Given *x* and *y* axes through the center of a torus network, the complement destination is the composition of the *x* and *y* axes reflection of the source. Transpose and bit reversal are important because they occur in practical computations and can cause worst case behavior in hypercubic oblivious routers [8].

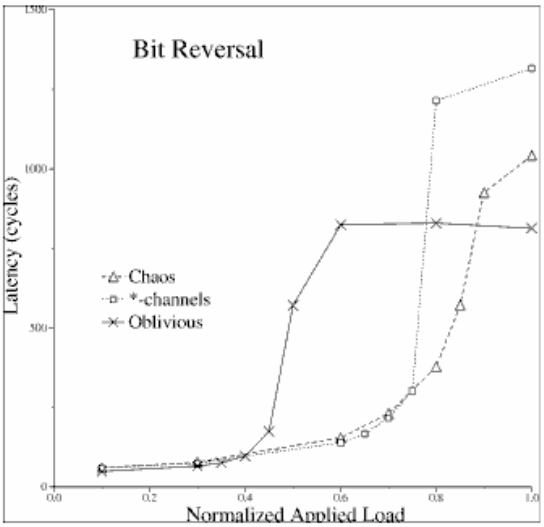
1) Saturation point

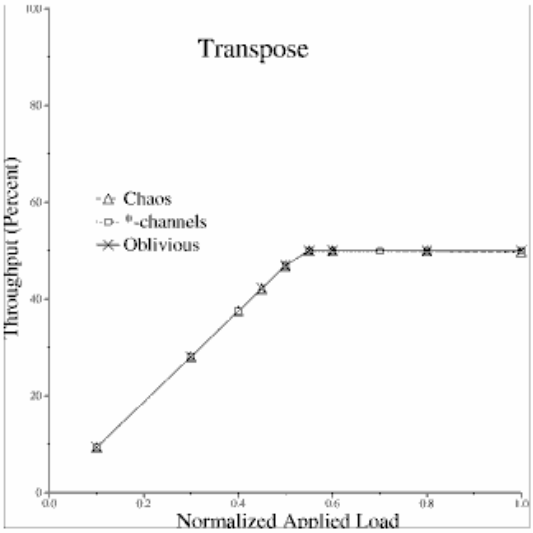
The saturation point reported is the first normalized applied load, using intervals of 0.05, that saturates the network. Table 1 reports saturation points for various traffic patterns. As can be seen, chaotic router mostly gives highest saturation point with minimal adaptive scheme coming close on most occasions. For complement pattern, the oblivious scheme performs better since it can route the complement traffic almost conflict free, while the adaptive schemes end up creating a hot spot for this pattern.

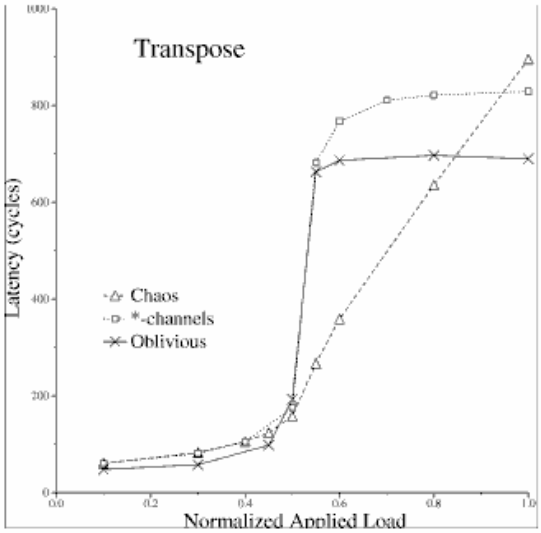

Random
Throughput (Percent)
Normalized Applied Load
Chaos
*-channels
Oblivious


Random
Latency (cycles)
Normalized Applied Load
Chaos
*-channels
Oblivious


Bit Reversal
Throughput (Percent)
Normalized Applied Load
Chaos
*-channels
Oblivious


Bit Reversal
Latency (cycles)
Normalized Applied Load
Chaos
*-channels
Oblivious


Transpose
Throughput (Percent)
Normalized Applied Load
Chaos
*-channels
Oblivious


Transpose
Latency (cycles)
Normalized Applied Load
Chaos
*-channels
Oblivious

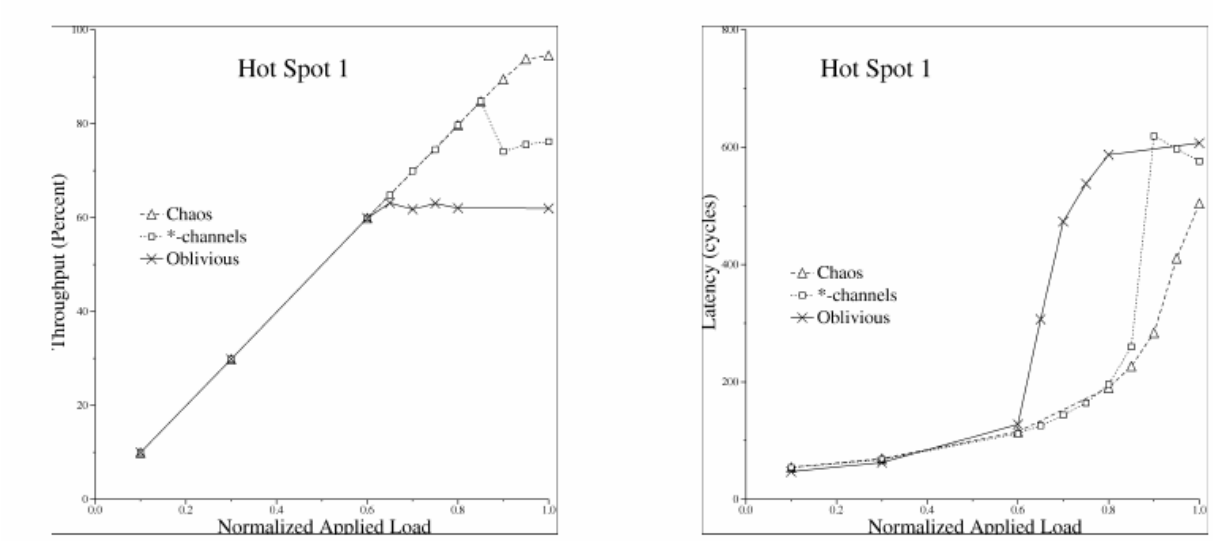


Figure 3: Latency and throughput versus applied load

2) Throughput and latency

The graphs in Figure 3, display the throughput and latency versus load for the traffic patterns studied. Throughput of a Chaos network is greater than or equal to that of the oblivious or *-channels network for the torus topology for pretty much all traffic patterns at all loads. Even above saturation point the throughput of chaotic router increases with load for most patters unlike other two schemes for which the throughput above saturation degrades for some patterns.

At very low loads, latency for the torus is slightly higher for the Chaos and *-channels routers than for the oblivious router, due to the lower per-hop latency of the oblivious router. At higher loads before Chaos saturation, the Chaos network generally has the lowest latency. After the Chaos router saturates, the relative latencies between the routers depend upon the traffic pattern under consideration.

3) Hot Spots

When hot spots are added to random traffic, the maximum throughput of the oblivious torus is reduced from 76 percent to 63 percent, as shown in Fig. 8. With the same hot spots, however, the Chaos torus achieves a maximum throughput of 95 percent, a slight reduction from its peak throughput of 98 percent achieved with random traffic. The *-channels router experiences a drop in maximum throughput from 89 percent to 85 percent.

F. Critique

The paper is well written and addresses all issues of concern. These include the following:

- Comparison with existing schemes in terms of hardware complexity, saturation points, throughput and latency
- Discussion about handling deadlock, livelock, out of order packet delivery
- Fault tolerance mechanism

It establishes successfully that chaotic routing is a practically realizable scheme which exceeds the performance of typical oblivious and minimal schemes with a design simple enough to be competitive with most streamlined oblivious routers.

However there are some flaws in the paper which have been discussed below:

1) Paper presents chaotic routing as an alternative to existing non-minimal routing schemes, but no comparisons have been made with any of these exiting schemes in terms of either complexity or performance or both. It is, therefore, not possible to judge whether chaotic routing should be a preferable alternative to other non-minimal schemes.
2) For oblivious schemes, only a deterministic approach has been considered. Authors acknowledge that random oblivious schemes do well in avoiding congestion and achieving good throughput but then drop it out of consideration by just mentioning that it doubles the average latency in un-congested loads. It seems to us that a performance comparison should have been made with these schemes with congested loads.
3) It has been mentioned that live-lock is provided in the probabilistic sense, where each packet should eventually get to its destination. However no analysis has been made of the latencies which could be incurred by packets before they eventually get to the destination.

III. THE PERFORMANCE OF ADAPTIVE ROUTERS ON WORST CASE PERMUTATIONS [II]

Since C*haotic routing* is a randomized, non-minimal adaptive routing algorithm, it can potentially avoid network bottlenecks by routing packets around “hot spots”. However, in contrast to minimal adaptive algorithms, *Chaos routing* does not always choose the minimal path which makes analysis of worst case traffic permutations for such algorithms difficult. Chin, Leighton and Tompa (CLT) [9] have provided a lower bound for permutation routing problems on the nxn

mesh for a large class of minimal adaptive algorithms. It has been shown that for any such routing algorithm, there exists a permutation that requires $\Omega(n^2/k^2)$ steps to route all the packets in the permutation, where k is the number of packets a node can contain. In this paper the author has presented experimental results showing performance improvement of non-minimal *Chaos router* over the worst-case permutation traffic for the minimal version of *Chaos adaptive router*. The results presented in the paper show that the Chaos router for the worst case permutation of the *Chaos router* takes time which is polynomial in n to the degree $3/2$.

All the state-of-the-art routers are *oblivious* in nature which makes them simple and deadlock free. However, their performance suffers from congestion and/or faults because such algorithms cannot use the bandwidth effectively due to the absence of flexibility in routing. Conversely, adaptive routing offers much more flexibility as it can potentially use the available bandwidth to relieve congestion or to route around faults. In contrast to minimal adaptive routing, non-minimal adaptive can route packets towards any path and is not limited to the minimal paths between the source and the destination only. However, this flexibility has higher complexity and much more complex logic is needed to avoid *livelock*.

One of the benchmarks for a router's performance is how it performs in the presence of permutations in which each processor sends at most one packet and each processor receives at most one message. Permutations that arise in practice such as transpose cause poor performance on some networks. On the nxn mesh there is theoretically enough bandwidth to deliver all packets in any permutation in time proportional to n. Thus a good routing algorithm should be able to route packets in $O(n)$ time.

CLT have shown that for minimal adaptive algorithms employed in the nxn mesh, there exists a permutation such that it takes $\Omega(n^2/k^2)$ steps to route all the packets in the permutation, where k is the number of packets a node can contain. This applies to the class of routing algorithms that rely on local traffic and employ only profitable paths to destination. Dimension order routing with first in first out (FIFO) queues falls in this category.

Since *Chaos routing* is a randomized, non-minimal adaptive routing algorithm, the question that arises is how well does it perform on a worst case permutation. For this purpose a worst case permutation is constructed on an nxn mesh with a minimal version of the *Chaos router* such that it gives performance comparable to the $\Omega(n^2/k^2)$ bound presented by CLT. The difference on the running times of the non-minimal and the minimal versions of the *Chaos router* are a measure of the how effective the de-routing mechanism is in *Chaos*. By running non-minimal *Chaos* over a range of mesh sizes it is observed that *Chaos* performance on a CLT worst case permutation for an nxn mesh is closely approximated by a polynomial of degree $3/2$.

A. *Experiment 1*

In the first experiment, CLT permutation was built for minimal chaos router. This permutation is constructed by injecting one packet into each of the nodes in a *cn x cn* corner of the mesh. These packets have destinations in one of several rows or columns just besides the corner (see figure 4). A hot spot develops at the intersections of the destinations.

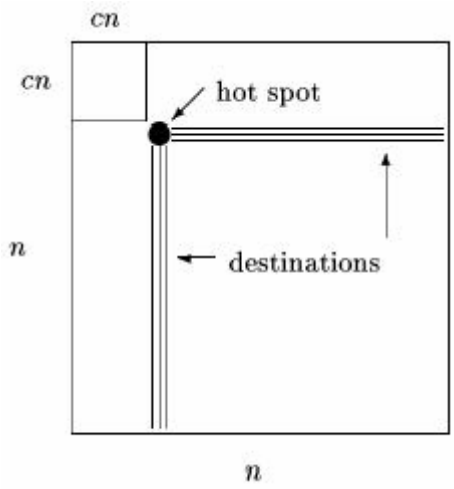


Figure 4. Worst case permutation traffic pattern

The packet transfer time consists of the time required to escape the congestion and then the additional time required afterwards to reach the destination. In CLT results, the *congestion component* grows qudratically with n on the n x n mesh, and the second component, which we call *distance component,* grows linearly with n. To isolate congestion component, the CLT permutation was altered such that the last packets to be delivered were near the edge of the mesh. This ensured that a linear dependency is achieved for the distance component in the chaos minimal routing. It was then possible to observe how congestion component grows. The same experiment was then run on the Chaos router.

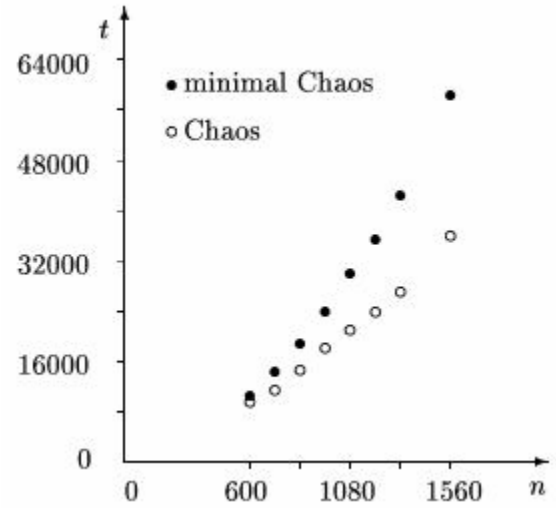


Figure 5: Results of experiment 1

The results of the experiments are shown in figure 5. As predicted by the CLT result, the congestion component of

minimal Chaos grows quadratically with n and dominates the distance component. The results for Chaos are however, inconclusive: it is difficult to estimate what is the asymptotic behavior of the curve for Chaos.

B. Experiment 2

Observing the packet flow of Experiment 1 showed that the behavior of packets some distance beyond the hot spot region becomes quite regular and doesn't effect the over all analysis. These facts allowed simulating just a small portion of the mesh, and still observe the same behavior. Experiment two is identical to experiment 1 except that it simulates only a part of the mesh, and thus much larger mesh networks are simulated.

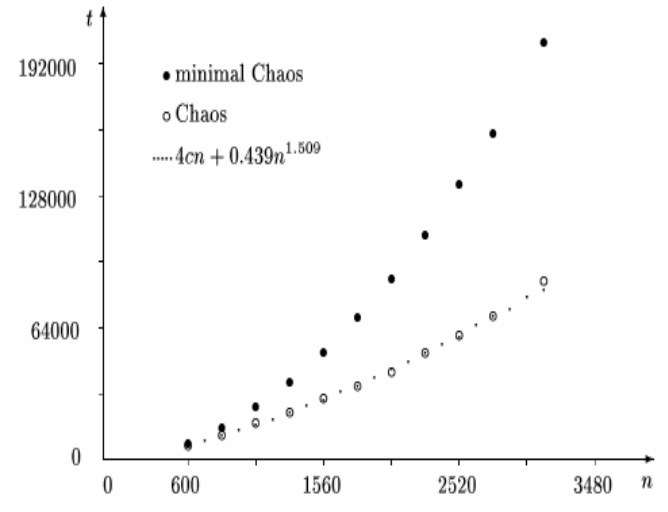


Figure 6: Asymptotic behavior of minimal and non-minimal *Chaos*

As shown in Figure 6, the curve for *Chaos* closely fits the curve defined by $an + bn^{3/2}$ where a and b are constants. If the congestion component grows proportionally to n^x, the ratio of congestion components for two different problem sizes is related to the ratio of the problem sizes as $\left(\frac{m}{n}\right)^x$ where m and n are the problem size of mxm and nxn meshes respectively. The value of x as computed by authors is 1.5 for almost all pairs except for the case when the problem size differs by 480 in which case it lies between 1.4 and 1.7.

C. Conclusions

Experiments presented in the paper show that *Chaos router* gives polynomial performance in n of degree $3/2$ on worst case permutations constructed for a deterministic minimal adaptive version of it. However, some permutations can still cause quadratic behavior for *Chaos* but the author believes it is highly unlikely given the way *Chaos* routes packets. *Greedy hot potato* or *deflection* routing, where a node must send on the next step any packets it receives in the current step, might be a non-minimal adaptive solution to route arbitrary permutations in time linear in n on the nxn mesh.

D. Critique of the paper

The strong point of the paper is that it analyzes the behavior of *Chaos routing*, a non-minimal adaptive algorithm based upon worst-case permutations for minimal adaptive algorithms. Since it is generally hard to find worst-case traffic permutations for non-minimal adaptive algorithms, the approach seems promising as it provides a fair comparison between the performance of minimal adaptive and *Chaos routing* algorithms. However, the weak points of the paper are as follows:

- The results presented in the paper are inconclusive as they are based upon just one worst-case traffic permutation, which shows *Chaos* performs better than its minimal version. Other permutations could exist on which *Chaos* might perform worse than existing minimal adaptive algorithms. This point has also been reflected in the paper but the author's argument as to why these situations will not arise is not substantive.
- The performance results presented in the paper are based upon mesh networks. Since other topologies have not been considered, it is possible for *Chaos* to perform worse than minimal adaptive for a worst-case permutation on some topology. Consequently, other topologies are needed to establish the case that *Chaos* is better than minimal adaptive on worst-case permutations.
- The paper does not present a comparison between *Chaos* and other non-minimal adaptive algorithms. This comparison is needed to analyze how well *Chaos* compares with other non-minimal adaptive algorithms and why the behavior differs.

IV. Design of a Router for Fault-Tolerant Networks [III]

A. Introduction

The paper *Design of a router for fault-tolerant networks* describes various fault-tolerance methods that the authors have proposed for the enhancement of *Chaos routing*, a non-minimal adaptive routing scheme. Although, non-minimal adaptive routing algorithms provide a basic level of fault-tolerance as compared to existing routing schemes, their response time is rather slow. However, due to inherent fault-tolerance capabilities of such schemes, only a limited amount of logic is needed to support fault detection, identification and reconfiguration of the networks in the presence of faults.

Since the size of interconnection networks is increasing substantially, the need for fault-tolerant networks is also growing. Numerous networks have built-in redundancy in terms of the number of paths from the source to the destination; however, exploiting this redundancy requires development of complex routing algorithms. Since routers need to make quick decisions such algorithms are not feasible. Secondly, absence of global information about the presence of faults further constrains the development of fully fault-tolerant networks. In this paper, the authors have leveraged the inherent fault-tolerant capabilities of *Chaos routers* to present

minimum enhancements to boost the fault-tolerance capabilities of *Chaos routers*.

B. Fault-tolerant routing

The goals of fault-tolerant networks as defined in the paper are to:

1- Provide reliable message delivery between all pairs of nodes that have some non-faulty path between them
2- Detect when a fault has occurred and notify the operating system
3- Provide mechanisms to reconfigure the network around faulty areas to ensure quick reliable message delivery to all non-faulty areas of the network

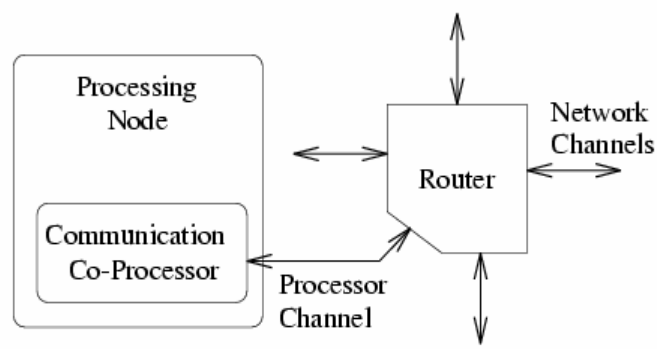


Figure 7: Three basic classes of failures, link, node and router

In this regard, the paper considers three basic classes of faults: link failures, node failures and router failures as shown in Figure 7. The fault detection and recovery schemes discussed in the paper cover both transient (temporary) as well as persistent (repeatable) faults. However, faults that do not cause lost or corrupted packets are not detected. If a node or a link is found to be faulty, it is removed from service but associated routers continue to operate.

C. Fault-tolerance capabilities of various routing schemes

Although, oblivious routers are simple and fast as the paths from a message source to its destination are statically determined (according to the paper), they are not robust in the presence of faults due to the absence of a mechanism for avoiding faulty components. In contrast, minimal adaptive routing algorithms do provide limited flexibility in routing messages; their fault-tolerant extensions come at the cost of added complexity. Conversely, non-minimal adaptive routers with the exception of deflection routers have inherent fault-tolerance capabilities, they respond slowly to faults. On the other hand, deflection routers require all links to be functional and cannot tolerate faults without special hardware considerations.

D. Fault-tolerance and basic Chaos router

1) Basic support

Chaos routers have basic fault-tolerance built into their design. If a link or a router fails to respond to its inputs, the output frames associated with each link do not empty causing the router to route the rest of the packets away from the faulty components. Packets that have many *profitable* (minimum) paths to the destination are routed out though other available channels. Conversely, the ones that have no alternative minimal paths to destination are routed out through non-minimal channels after a certain delay. This allows the communication to continue despite the presence of faults.

2) Shortcomings

Packets waiting in the output frame become permanently stuck waiting for faulty channels as they cannot be de-routed. Additionally, packets that do get de-routed incur large delays as they wait in the multi-queue of the router to fill up before being randomly chosen and de-routed. Moreover, under light loads, the multi-queue might never fill up enough to allow de-routing. Other errors which the basic *Chaos router* cannot handle are corrupted or lost packets.

E. Enhancements in Chaos router

To maintain flexibility and to avoid complexity in the system, policy decisions are deferred to the software. *Chaos router* utilizes a global, synchronous procedure for fault diagnosis. The router performs fault detection based upon programmable watchdog timers and passes the information to its associated processing node. The software running on the node decides whether to initiate a diagnostic procedure or not. The software performs fault detection, identification and recovery at the system level. Fault diagnostic procedures are performed under the control of local processing node and generate a map of usable and unusable local links. This map is combined globally to generate a map of the entire network.

1) Communication

For fault management a single dedicated bi-directional line is provided on a per channel basis. This line is used for communication and synchronization and passes a *red alert* signal indicating a fault might have occurred somewhere in the network. This signal can be initiated by any processing node in the network and is communicated to each of the neighbors of the asserting node, which in turn broadcast it to their neighboring nodes. Each node marks the channel from which the *red alert* signal arrives, which allows the originating node to control various phases of the fault diagnosis procedure while maintaining synchronization within the network. The processor channel in each router is also provided with the same line but with some additional functionality. This line is called *Error-detected* line. The router strobes this line to indicate the presence of a possible fault and similarly the processor can signal on this line to indicate to the router to assert the *red-alert* signal.

2) Fault detection

1- Watchdog timers associated with each output channel indicate when a packet in the output frame has

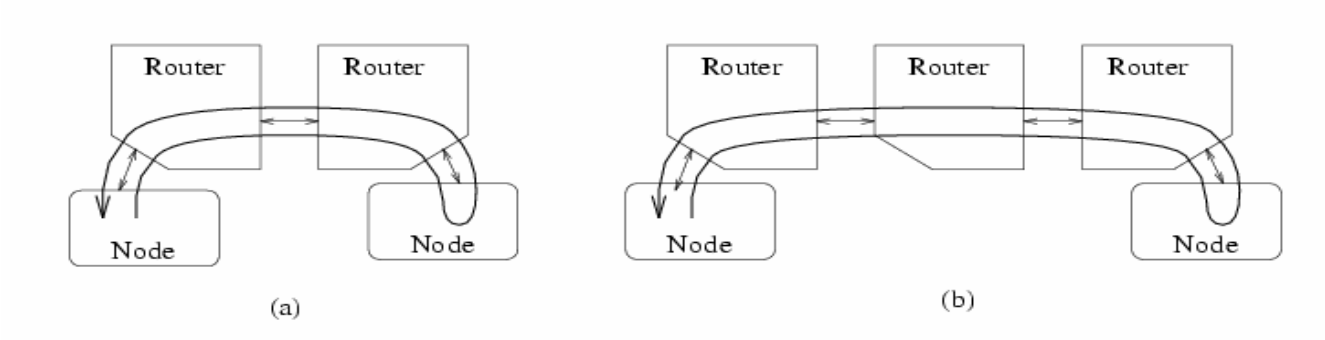


Figure 8: a- one hop diagnostics and b-two-hop diagnostics

exceeded the maximum allocated time. This indicates that the channel is un-responsive and might be faulty and a strobe is sent to the processor on the *Error-detected* line.

2- In *Chaos channel protocol* the end-of-message (EOM) is indicated by the tail of the packet. If a packet fails in mid-transmission, the length of the packet exceeds the maximum permissible packet length. In this case the transmission is terminated and the processor is strobed.

3- A checksum is included in the header of every packet that is based upon the static bits of the packet. Checksum is verified by both the injection and delivery nodes. Since some of the bits of the header change from node to node, parity is also included in the header which is updated on a per hop basis. If a parity error is detected, the packet is delivered to the processing node and a fault is indicated. Additionally, the parity error changes the profitable paths for the erroneous packet to indicate just the packet delivery node.

3) System drain

Once a *red-alert* signal is asserted, the router halts the injection of new packets into the network and a system drain is initiated in which the packets existing inside the network are delivered to their destination nodes. The system drain is terminated when either the watchdog timer associated with each router expires or the network is emptied. The drain time of a faulty network is unbounded and after the expiration of the drain timer, the *red-alert* signal by any node is used to indicate that it contains undelivered packets. After the expiration, all the undelivered packets are sent to the processing node associated with each router and a diagnostic procedure is initiated to assess faults. Since packets are not necessarily delivered in order in non-minimal adaptive protocols, the communication co-processor must reorder the arriving packets. If a packet is missing for a certain time or if the buffer used to reorder packets overflows an error is indicated and a system drain results.

4) Fault diagnosis

Each node injects a packet destined for its neighbors. The receiving node sends an acknowledgment back to the sender as shown in Figure 8. If the message is lost or corrupted, the link is flagged as bad. If the combination of tests performed by many nodes provides conflicting information about a link, operating system resolves conflicts. However, before initiating the tests, de-routing must be disabled in the routers.

If a processing node associated with a router is dead, all the one-hop tests to that router result in a failure. In order to get around this problem, another two-hop test is conducted in which the processing node of the intermediate router is not employed. As a result, the router is detected as working even though its associated processing node is dead.

5) Test coverage

Diagnostic tests covered in the paper can find most of the static failures in the network links. They can also detect when nodes fail enough to upset the testing protocol, however, router failures are not covered fully. Additionally, the tests are not exhaustive and a rudimentary test of the router's ability to correctly route packets is made. Transient faults are not detected but through the procedure outlined in the paper, transient errors are caught.

F. Reconfiguration

After the diagnosis, each node has knowledge of the faulty links and they are marked as dead and are masked out from the routing decisions. In *Chaos router* a mask of the functional channels is created at configuration time. This list is updated whenever a link goes dead and any packet whose profitable channel list is zero is immediately de-routed. The hardware for this is quite simple and does not impact the critical path of packets delivery.

Each node must communicate fault data to a central controller to preserve information about faults for the long term. There can be redundant controllers for the purpose of fault-tolerance. The information collected by these controllers can be used in the following ways:

- Help in reconfiguring the system after re-boot using global fault information
- Communication of new and existing faults to the system
- Configuration of the network to remove dead or unreachable nodes
- Removing messages destined for dead nodes

G. Summary of hardware costs

It is claimed that the additional hardware required to implement fault-tolerance is a small fraction of the router cost. One line per channel along with a bit of logic is needed for communication purposes. The output watch dog timers are generally less than 10-bits with the worst case timer for any purpose is 20-bits. During system drain a state machine is needed to control the router during system drain and diagnostic procedures. Additionally, each router must be able to report if it is buffering any packets and once again the logic required for that is small. In brief the hardware cost for making *Chaos* fault-tolerant is small as most of the fault-tolerance features are provided by the non-minimal adaptive algorithm.

H. Remaining problems

- *Orphaned messages*: If a node stops reading messages destined for it, messages back up in the network without ever being detected. Such a problem can only be detected using a system drain
- *Global problems:* Presented solutions work for isolated faults and convex-shaped fault areas; however, certain patterns of faults might still not be detectable and would require global techniques.
- *Error correction*
- *Router failures*

I. Critique of the paper

The design of fault-tolerant *Chaos router* presented in the paper is thorough and covers all the issues related to fault-detection, identification and reconfiguration of routers. Additionally, the authors leverage inherent fault-tolerance capabilities of non-minimal adaptive routers and keep the hardware cost needed to add additional fault-tolerance features to a minimum. The weak point of the paper is that it does not provide a thorough comparison between the performance of fault-tolerance features added to *Chaos* and other existing non-minimal adaptive fault-tolerance architectures.

V. Overall Critique and Conclusion

Based on the literature reviewed we conclude that chaotic routing is a practically realizable scheme which exceeds the performance of typical oblivious and minimal schemes with a design simple enough to be competitive with most streamlined oblivious routers. It also enables effective fault tolerance schemes to be implemented in a router, thus resulting in improved reliability. It is, however not clear if Chaotic routing forms a preferable alternative to other non-minimal adaptive routing schemes, since no comparisons with other schemes is available. In fact one of the papers points out that *hot potato* or *Deflection routing* could achieve better worst case performance than chaotic routing and also has a simple logic and algorithm. Therefore, more analysis is required to establish whether chaotic routing should be the routing of choice within the class of non-minimal routing algorithms.

References

[1] Kevin Bolding, Melanie Fulgham, Lawrence Snyder. *The Case for Chaotic Adaptive Routing*, Technical Report UW-CSE-94-02-04, University of Washington, Feb. 1994.

[2] Donald D. Chinn. "The Performance of Adaptive Routers on Worst Case Permutations", Proceedings of the 1994 Parallel Computer Routing and Communication Workshop, May 1994.

[3] Kevin Bolding, William Yost. "Design of a Router for Fault-Tolerant Networks", Proceedings of the 1994 Parallel Computer Routing and Communication Workshiop, May 1994.

[4] S. Konstantinidou and L. Snyder, "The Chaos Router: A Practical Application of Randomization in Network Routing," *Proc. Symp.* Parallel Algorithms and Architectures, pp. 21-30, 1990.

[5] K. Bolding, "Chaotic Routing: Design and Implementation of an Adaptive Multicomputer Network Router," PhD thesis, Univ. of Washington, Seattle, July 1993.

[6] N. McKenzie, K. Bolding, C. Ebeling, and L. Snyder, "CRANIUM: An Interface for Message Passing on Adaptive Packet Routing Networks," *Proc. Parallel Computer Routing and Comm. Workshop*, pp. 266-280, May 1994.

[7] K. Bolding, M. Fulgham, and L. Snyder, "The Case for Chaotic Adaptive Routing," Technical Report CSE-94-02-04, Univ. of Washington, Feb. 1994.

[8] F.T. Leighton, Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes. Morgan Kaufmann, 1992.

[9] D. D. Chinn, T. Leighton and M. Tompa. Minimal adaptive routing on the mesh with bounded queue size. In *Proceedings of the 1994 ACM symposium on Parallel Algorithms and Architectures*, Cape May, NJ, June 1994.

.