J Supercomput (2012) 59:569–588
DOI 10.1007/s11227-010-0454-7

Comparative analysis of Traffic Patterns on *k-ary n-tree* using adaptive algorithms based on Burton Normal Form

Nitin · Durg Singh Chauhan

Published online: 10 June 2010

Abstract *k-ary n-trees* are a particular type of Fat-Trees that belong to parametric family of topologies. In spite of their wide usage as an Interconnection Network topology, it has been quite unclear about the performance of Adaptive Routing Algorithms on them. In this paper, we consider a *4-ary 3-tree* and analyze two Adaptive Routing Algorithms namely the Non-Minimal Adaptive Routing Algorithm and Minimal Adaptive Routing Algorithm. Specifically, the application of these algorithms on *4-ary 3-tree* using various Traffic Patterns has been simulated. The six Traffic Patterns called BitTranspose, BitReversal, BitComplement, Uniform Distribution, k-shift and Ring are used as running examples throughout the paper. The simulation results show that the Network Latency for *k-ary n-tree* is much higher in case of the Non-Minimal Algorithm as compared to the Minimal Algorithm. However, in case of Ring Traffic, the results show a deviant behavior when compared to other patterns.

Keywords Index Terms · Interconnection networks · Fat-trees · *k-ary n-trees* · Traffic Patterns · Non-Minimal Adaptive Routing · Minimal Adaptive Routing · BigNetSim · Burton Normal Form · Congestion-free patterns and flow-control

Nitin (✉)
Department of CSE and IT, Jaypee University of Information Technology, Waknaghat, Solan 173234, Himachal Pradesh, India
e-mail: nitin@jes.juit.ac.in

Nitin
e-mail: delnitin@ufl.edu

D.S. Chauhan
Uttarakhand Technical University, Dehradun 248001, Uttarakhand, India
e-mail: pdschauhan@gmail.com

1 Introduction and motivation

Interconnection Networks (INs) enable fast data communication between the components of a digital system. INs are based on three aspects: Network Topology, the Routing Algorithm, and the Flow Control mechanism employed [1–10]. One of three prime aspects of INs is topology. The topology is packaged such that it is cost effective along with its ability to achieve good performance (in terms of throughput, latency, and scalability, etc.) [11]. Parametric family of regular topologies consists of *k-ary n-cube*, *k-ary n-butterflies,* and *k-ary n-trees* that can be built by varying the two parameters *k* and *n. k-ary n-trees* are a particular type of Fat-Trees built using processing nodes and constant Arity Switches interconnected in a butterfly like topology [12].

In this paper, we focus on Non-Minimal, Minimal Adaptive Algorithms, and their application on *k-ary n-tree* using various Traffic Patterns. Section 2 introduces a Fat-Tree, *k-ary n-tree*, and different types of Traffic Patterns. In Sect. 3, we explain routing algorithms used in the paper. In Sect. 4, we discuss Testbed and Simulation results of various Traffic Patterns on *4-ary 3-tree* using BigNetSim followed by conclusion and references.

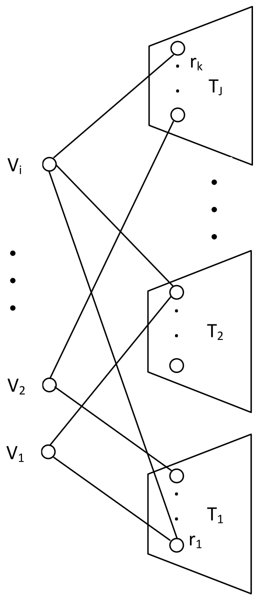
2 Preliminaries and background

An IN has a regular topology in which switches are identical and organized as a set of stages where each stage is only connected to the previous and to the next stage using a regular connection pattern. They are widely used for broadband switching technology and for multiprocessor systems. Besides this, they offer an enthusiastic way of implementing switches used in data communication networks. With the performance requirement of the switches exceeding several terabits/sec and teraflops/sec, it becomes imperative to make them dynamic and fault-tolerant. A number of techniques have been used to increase the reliability and fault-tolerance of the INs. A survey of the fault-tolerance attributes of these networks can be found in [1–7].

The typical modern day application of the INs includes fault-tolerant packet switches, designing multicast, broadcast router fabrics while system on-chip and networks on-chip are hottest nowa-days. Normally the following aspects always considered while deigning the fault-tolerant INs: the topology chosen, the routing algorithm used, and the flow control mechanism adhered. The topology helps in selecting the characteristics of the present chip technology in order to get higher bandwidth, throughput, processing power, processor utilization, and probability of acceptance from the IN based applications, at an optimum hardware cost. Soon, as the topology is frozen, the analytical bounds, which help for measuring reliability and availability, can be examined. The topology helps in determining the throughput and latency of the INs whereas the routing algorithm and flow control encourages in achieving the performance bounds.

In particular, here we focus on parametric family of regular topologies, which consists of *k-ary n-trees*, a type of Fat-Trees (that are most common type of INs in commercial machines) [13–15]. A Fat-Tree topology is an IN based on binary tree, which gets thicker near the root. A set of processors is located at the leaves or at the

Fig. 1 A Fat-Tree



first stage switches of the Fat-Tree and each edge of the underlying tree corresponds to a bidirectional channel (between a parent and a child). The number of wires in a channel connecting switches measures the capacity of a Fat-Tree; it is these wires, which create options for packets to flow during Congestion using Adaptive Routing Algorithm, i.e., a Fat-Tree is parameterized not only in the number of processors, but also in the communication bandwidth it can support.

Before we understand *k-ary n-tree* in detail, it is very important for us to know more about Fat-Trees.

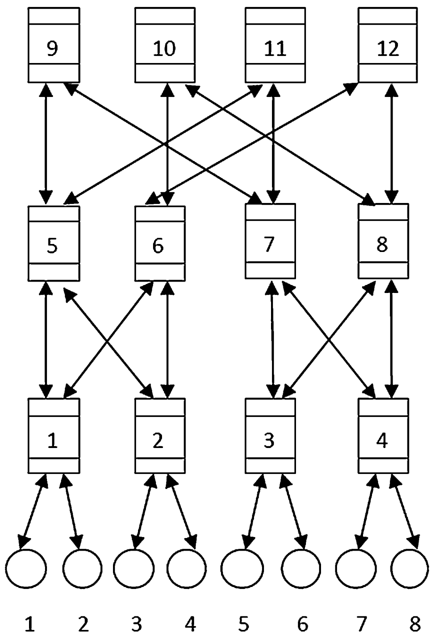
2.1 A Fat-Tree

A Fat-Tree is a collection of vertices connected by edges and is defined recursively as follows [15].

1. A single vertex by itself is a Fat-Tree. This vertex is also the root of the Fat-Tree.
2. If $v_1, v_2, \dots, v_i$ are vertices and $T_1, T_2, \dots, T_j$ are Fat-Trees, with $r_1, r_2, \dots, r_k$ as roots (j and k need not to be equal), a new Fat-Tree is built by connecting with edges, in any manner, the vertices $v_1, v_2, \dots, v_i$ to the roots $r_1, r_2, \dots, r_k$. The roots of the new Fat-Tree are $v_1, v_2, \dots, v_i$. See Fig. 1.

The arity or ports of the internal switches of the Fat-Tree increases as we go closer to the root, hence the practical implementation of the same becomes almost unfeasible. Due to this, few alternative solutions are proposed, which intend to keep fixed switch degree, while focusing on *k-ary n-trees;* bandwidth is increased by replicating switches while going toward the root. Let us turn our attention to a particular class of fat-trees, the *k-ary n-trees*. *k-ary n-trees* borrow from a popular class of multistage interconnection networks, the k-ary n-butterflies [14, 15] (or *k-ary n-flies* for short), the topology of the internal switches.

Fig. 2 A *2-ary 3-tree*



2.2 A *k-ary n-tree*

The knowledge of Fat-Trees will help in exploring *k-ary n-trees*; In general, *k-ary n-trees* are a particular type of Fat-Trees that belong to parametric family of regular topologies, hence the structure of the same depends on the values of k and n, where k is the number of links of a switch that connects to the previous or next stage i.e. degree of switch is $2k$ while n is the number of stages [13–15]. After having knowledge of the values of k and n we see that a *k-ary n-tree* is constructed using $N = k^n$ processing nodes and $nk^{n-1}k * k$ communication switches [16].

1. A processing node is represented as a tuple $\{0, 1, \ldots, k\}^n$.
2. A switch is defined as an ordered pair $\{s, o\}$, where s is the stage at where the switch is located, $s \in \{0, 1, \ldots, n-1\}$ and o is a $n-1$ tuple $\{0, 1, \ldots, k\}^{n-1}$, which identifies the switch inside its stage.
3. Two switches, $\{s, o_{n-2}, \ldots, o_1, o_0\}$ and $\{s, o'_{n-2}, \ldots, o'_1, o'_0\}$ are connected if $s' = s + 1$ and $o_i = o'_i$ for all $i \neq s$. There is a link between the switch $\{0, o_{n-2}, \ldots, o_1, o_2\}$ and the processing node $p_{n-1}, \ldots, p_1, p_0$ if and only if $o_i = p_i$ for all $i \in \{n-2, \ldots, 1, 0\}$. We will use this link for numbering to the switches of a *k-ary n-tree*: descending links are labeled from 0 to $k-1$, and ascending links are labeled from k to $2k-1$. Figure 2 represent a *2-ary 3-tree*, which is a type of *k-ary n-tree*.

2.3 Definition of Traffic Patterns

In this section, we are providing the definition of following six types of existing Traffic Patterns.

Definition 2.3.1 BitTranspose Traffic: Address of the destination node is a transpose of that of the source node, i.e., $d_i = s_{(i+\frac{b}{2})\,\mathrm{mod}(N)}$.

Definition 2.3.2 BitReversal Traffic: Address of the destination node is a reversal of the bit address of the Source node, i.e., $d_i = s_{b-i-1}$.

Definition 2.3.3 BitComplement Traffic: Address of the destination node is a bitwise complement of the address of the source node.

Definition 2.3.4 Uniform Distributed Traffic: Here, the number of packets arriving at every node is statistically equal, where the performance gain of the slot reuse scheme strongly depends on the traffic destination distribution. The probability $p_{i,j}$ of sending packet from node i to node j is $\frac{1}{M} - 1$ where M is the number of nodes in the bus, i is the source and j is the destination node.

Definition 2.3.5 k-shift Traffic: In this strategy, each node p sends messages to k nodes $\{(p - \lfloor(\frac{k-1}{2})\rfloor, \ldots, p-2, p-1, p+1, p+2, \ldots, p + \lfloor(\frac{k-1}{2})\rfloor), p+k\}$. Each message that node p gets from node $p - k$, can be copied into its network interface card (NIC) before it is sent to the k neighbors. This is repeated for $\frac{P}{k}$ iterations to complete the collective operation.

Definition 2.3.6 Ring Traffic: In this strategy, messages are sent along a ring formed by all the nodes in the system. In all stages of the ring strategy, on receiving a message node p forwards that message to its neighbor $((p + 1) \bmod p)$ in the ring.

3 Adaptive Routing Algorithms

We have considered two Adaptive Routing Algorithms, namely the Minimal Routing Algorithm and Non-Minimal Routing Algorithm, which are further tested on six Traffic Patterns, results of which are discussed throughout the paper.

3.1 Minimal Adaptive Routing Algorithm

In a network due to existence of multiple shortest paths between source and destination called profitable links, the routing algorithm chooses amongst them based on local or temporal conditions to forward packets. The number of Minimal paths from source to destination is directly proportional to distance between them. Hence, the algorithm proves to be ineffective if the destination is close to source [12].

Similarly, in *k-ary n-tree* the profitable links exist and helps in the implementation of Minimal Adaptive Routing Algorithm in the same form as it does in other topologies. Figure 2 shows four Minimal paths from node 1 to node 4 via switches are as follows:

1-5-9-7-4
1-5-11-7-4
1-6-10-8-4
1-6-12-8-4

While two paths from node 1 to 2 via switches are as follows:

1-6-2

1-5-2

Hence, proving the direct proportionality between number of Minimal paths and distance between source and destination.

3.2 Non-Minimal Adaptive Routing Algorithm

To avoid congestion and improve fault-tolerance, it is very important that packets route themselves to Non-Minimal/longer paths when all Minimal paths are congested, this will help improving performance by reducing the time required between source and destination than otherwise. Our Simulation studies show that Non-Minimal Adaptive Algorithms are more effective than other Adaptive Algorithms in providing Deadlock freedom and prevention of Live-lock [17, 18].

In *k-ary n-tree,* this works very efficiently as well. Figure 2 shows Non-Minimal paths from node 1 to node 4 via switches (apart from ones shown in Minimal Adaptive Routing Algorithm) is as follows:

1-6-2-5-9-7-4

1-6-2-5-11-7-4

This comes to action when all Minimal paths are congested and packets route adaptively after sensing the congestion. Hence, even the paths are longer, it gives better performance.

4 Evaluation of Traffic Patterns using BigNetSim

4.1 Experimental setup and Testbed

In this section, we are providing the simulation results. For our simulation, we have used IBM System x86, running with Novell's SUSE Linux Enterprise Server 11 and with BigSim Network Simulator. The BigSim [19, 20] Network Simulator is also known as BigSimulator and lives in the subversion (SVN) repository of Parallel Programming Laboratory, University of Illinois at Urbana Champaign, USA. This comes to action when all Minimal paths are congested and packets route adaptively after sensing the same.

Definition 4.1.1 BigNetSim: The BigSim simulator along with the network simulator is together also known as BigNetSim [21]. Both the simulators run on top of the POSE framework [20], which is a Parallel Discrete Event Simulation framework built on top of CHARM++ [22].

Definition 4.1.2 CHARM++: It is a parallel object-oriented programming language based on C++ and developed in the Parallel Programming Laboratory at the University of Illinois at Urbana-Champaign. It is designed with the goal of enhancing programmer productivity by providing a high-level abstraction of a parallel program and delivering good performance on a wide variety of underlying hardware platforms.

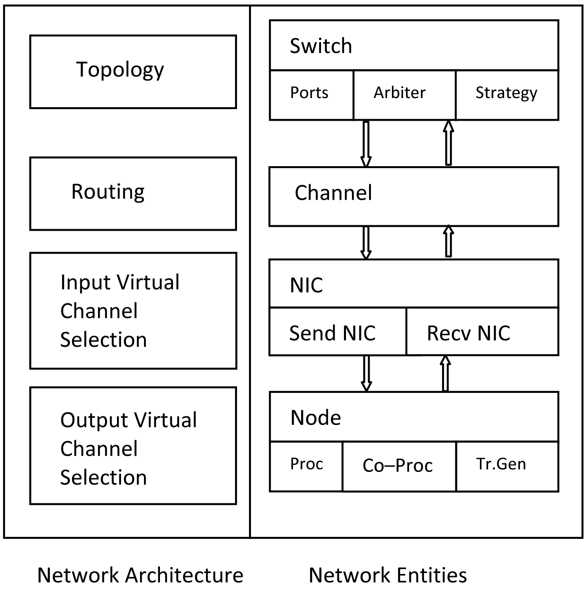


Fig. 3 Conceptual model of BigNetSim

Figure 3 shows the conceptual model of BigNetSim [19] used for our analysis, where using existing *k-ary n-tree* topology classes, both Routing Algorithms are implemented over it under different Traffic Patterns; some keywords mentioned below explains the model.

1. Switch: A switch decides the routing on a packet. Switches could be input buffered or output buffered. The former are implemented as individual posers per port of each switch while the latter are implemented as a poser per switch. In an Input Buffered (IB) switch, a packet in a switch is stored at the input port until its next route is decided and leaves the switch if it finds available space on the next switch in the route. While in an Output Buffered (OB) switch, a packet in a switch decides beforehand on the next route to take and is buffered at the output port until space is available on the next switch along the route.
2. Network Interface Card (NIC): Network cards packetize and unpacketize messages. A NIC is implemented as two posers. The sending and receiving entities in a NIC are implemented as separate posers. A NIC is attached to each node.
3. Channel: These are modeled as posers and connect a NIC to a switch or a switch to another switch.
4. Compute node: Each compute node connects to a network interface card. A compute node simulates execution of entry methods on it. It is also attached to a message traffic generator, which is used when only an interconnection network is being simulated. This traffic generator can generate any message pattern on each of the compute nodes. The traffic generator can send point-to-point messages, reductions, multicasts, broadcasts, and other collective traffic. It supports BitComplement, BitTranspose, BitReversal, Uniform Distribution, k-shift, and Ring Traffic. These are based on common communication patterns found in real applications.

Table 1 Key network parameters used in the experimental setup

Network parameters	Values for simulation scenario-1	Values for simulation scenario-2
1. Channel bandwidth	1.75 Bps	1.75 Bps
2. Packet size	256 Bytes	2048 Bytes
3. Channel delay	9 ms	9 ms
4. Switch delay	0 ms	0 ms
5. Switch virtual channel	2	2
6. Switch port	8	8
7. Switch buffer	1.75	1.75
8. Number of nodes	64	64
9. Message size	200	400

A uniform determines the frequency of message generation or Poisson distribution.

5. Topology, routing strategies, input, and output virtual channel selection strategies need to be decided for any interconnection network. Once we have all of these in place, we can simulate an interconnection network.

The simulator system includes these components:

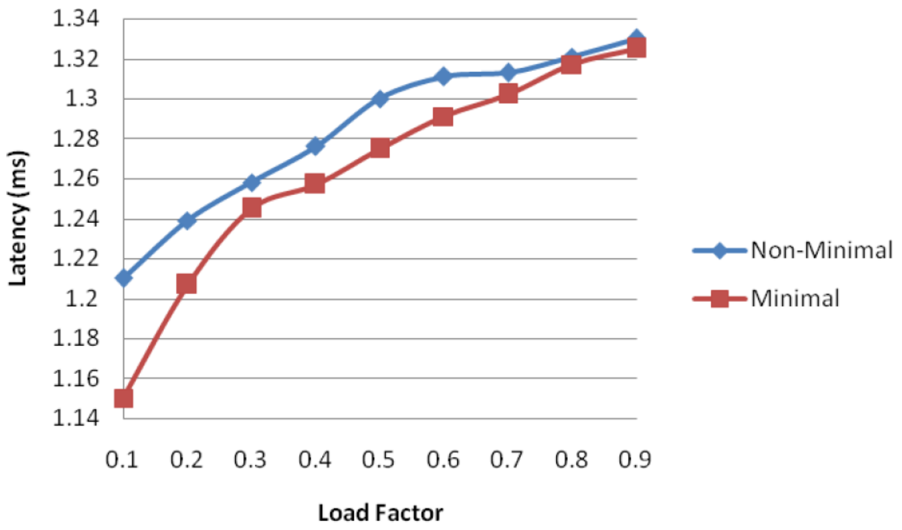
1. A parallel emulator that emulates a low-level machine API targeting architecture like Bluegene.
2. A message driven programming language (Charm++) running on the top of emulator.
3. The Adaptive MPI (an implementation of MPI on top of Charm++) environment.
4. A parallel post-mortem mode simulator for performance prediction, including network simulation.

The key networks parameters that passed to Charm++ tabulated in Table 1. All the simulations results have been tabulated in Tables 2, 3, respectively, and simultaneously put forth using Figs. 4(a–f), 5(a–f). Two types of simulation scenario i.e. scenario-1 and scenario-2 have been setup for the analysis. Scenario-2 differs from scenario-1 in terms of packet size and the message size.

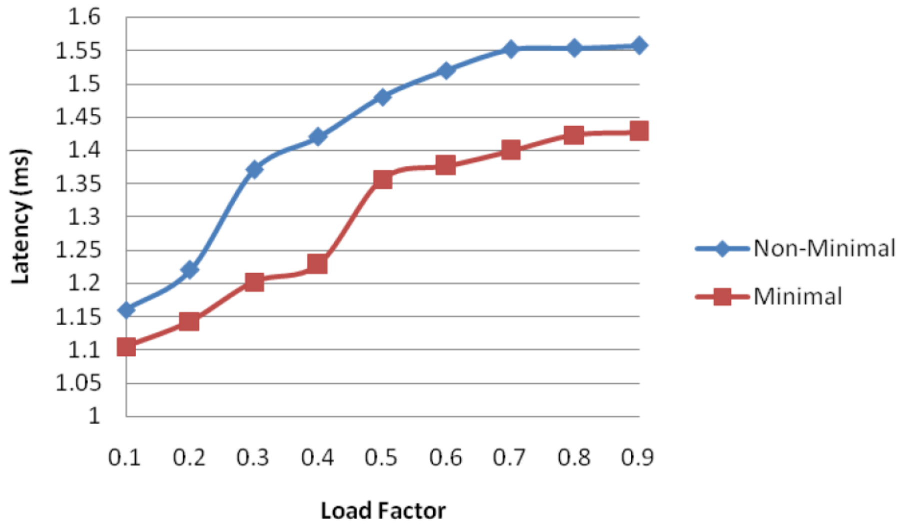
4.1.1 Key network parameters for simulation scenario-1 and scenario-2

4.2 Key parameters used for analysis of simulation results

The performance of an IN under dynamic load is usually assessed by two quantitative parameters, Throughput and the Latency. Two important characteristics are the saturation point and the sustained rate after saturation. Saturation is defined as the minimum offered load where the accepted bandwidth is lower than the global packet creation rate at the source nodes. The behavior above saturation is important because the network and/or the routing algorithm can become unstable, leading to sharp performance degradation. We usually expect the accepted bandwidth to remain stable after saturation, both in the presence of bursty applications that require peak performance for a short period of time and applications that operate after saturation in normal conditions, e.g. when executing a global permutation pattern.



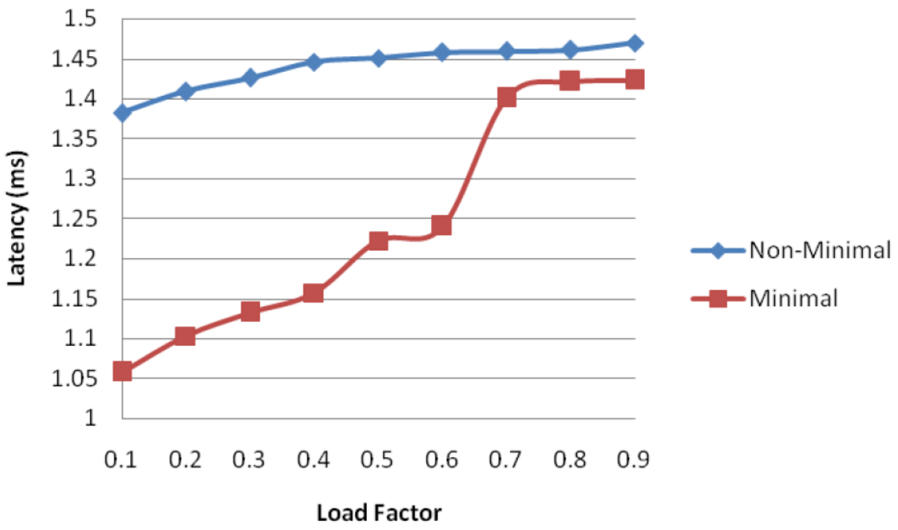
(a) *4-ary 3-tree*, BitComplement Traffic



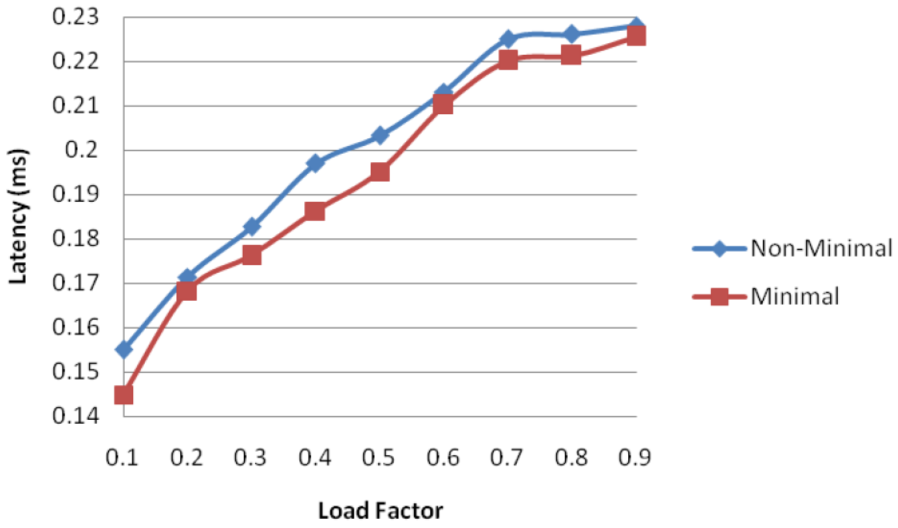
(b) *4-ary 3-tree*, BitTranspose Traffic

Fig. 4 (a–f) Graphical representation of simulation results (scenario-1) of lower Load Factor versus lower Latency (microseconds) for Non-Minimal and Minimal Adaptive Routing Algorithms for six Traffic Pattern on *4-ary 3-tree* using BigNetSim. The simulation results obtained from BigNetSim are tabulated in Table 2

Definition 4.2.1 Network Latency: The Network Latency is the time taken for messages to travel from source to destination, is an important measure of the communication performance. It does not include the source queuing delay. The end-to-end latency rises to infinity above saturation and is impossible to gain any information



(c) *4-ary 3-tree*, BitReversal Traffic

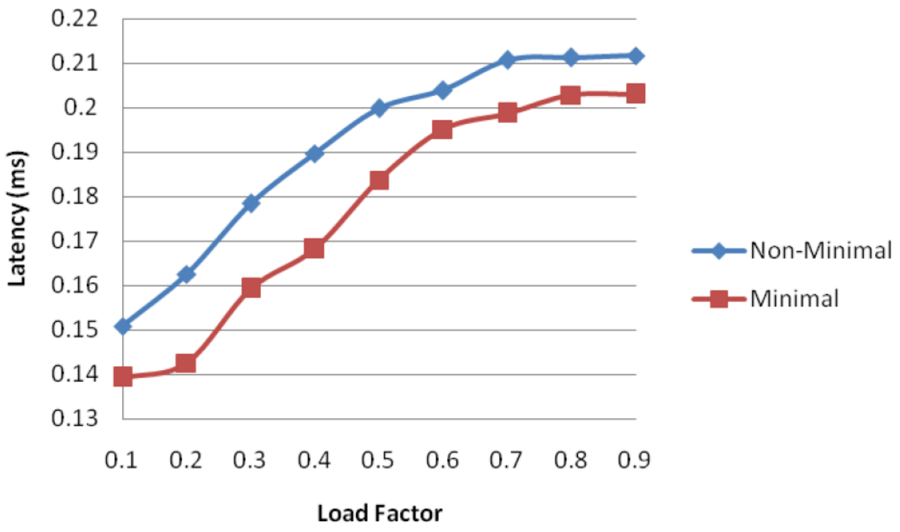


(d) *4-ary 3-tree*, Uniform Distribution Traffic

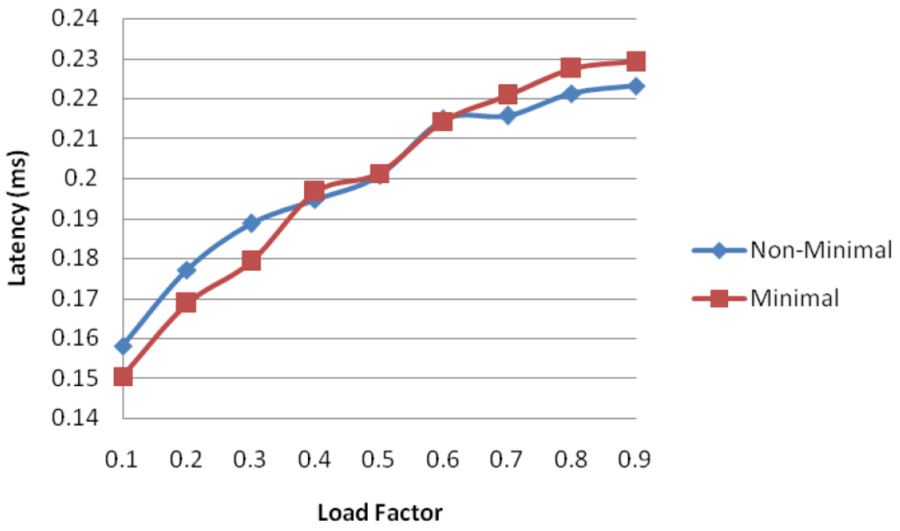
Fig. 4 (*Continued*)

in this case. For this reason, the Network Latency is often preferred to analyze the network performance.

Definition 4.2.2 Load Factor: Load Factor is the ratio of the mean arrival rate of packets and the arrival rate that saturates a link.



(e) *4-ary 3-tree*, k-shift Traffic



(f) *4-ary 3-tree*, Ring Traffic

Fig. 4 (*Continued*)

The experimental results of each Traffic Pattern presented according to the Burton Normal Form (BNF) [2].

Definition 4.2.3 Burton Normal Form: The BNF uses single-graph plot of both latency and throughput. The prime reason for using BNF is that BNF shows both throughput and latency, before and after saturation. The X-axis corresponds to achieve throughput, and the Y-axis corresponds to latency (in microseconds). In ad-

Table 2 Simulation results (scenario-1) of lower Load Factor versus lower Latency (microseconds) for Non-Minimal and Minimal Adaptive Routing Algorithms for six Traffic Pattern on *4-ary 3-tree* using BigNetSim

Load Factor	Traffic Patterns											
	BitComplement		BitReversal		BitTranspose		Uniform Distribution		k-shift		Ring	
	Routing Algorithms		Routing Algorithms		Routing Algorithms		Routing Algorithms		Routing Algorithms		Routing Algorithms	
	Non-Minimal	Minimal	Non-Minimal	Minimal	Non-Minimal	Minimal	Non-Minimal	Minimal	Non-Minimal	Minimal	Non-Minimal	Minimal
0.1	1.21	1.15	1.382	1.058	1.16	1.105	0.155	0.1448	0.1508	0.1392	0.1581	0.1505
0.2	1.239	1.207	1.409	1.103	1.22	1.143	0.1714	0.1683	0.1626	0.1426	0.1771	0.1687
0.3	1.258	1.245	1.426	1.132	1.37	1.202	0.1827	0.1762	0.1784	0.1593	0.1887	0.1791
0.4	1.276	1.257	1.446	1.157	1.42	1.228	0.1969	0.1863	0.1897	0.1682	0.1947	0.1966
0.5	1.3	1.275	1.451	1.222	1.48	1.356	0.2032	0.1951	0.1998	0.1837	0.2007	0.2013
0.6	1.311	1.291	1.458	1.24	1.52	1.376	0.2131	0.2101	0.2039	0.1951	0.2147	0.2143
0.7	1.313	1.302	1.459	1.401	1.551	1.399	0.225	0.2201	0.2107	0.1987	0.2156	0.2207
0.8	1.321	1.317	1.461	1.421	1.553	1.423	0.2261	0.2213	0.2113	0.2028	0.2212	0.2274
0.9	1.33	1.325	1.47	1.423	1.557	1.427	0.2279	0.2256	0.2117	0.2029	0.2231	0.2291

dition, it is worth noting that *k-ary n-trees* are not bisection-bandwidth limited as the *k-ary n-cubes*, whose BNF is normalized on the bisection bandwidth and the upper bound on the throughput for the uniform traffic.

We have used *4-ary 3-tree* for our experiment; hence, the tree has 64 (i.e. $k^n = 64$ as $k = 4$ and $n = 3$) processing nodes and 48 (i.e. $nk^{n-1} = 48$ as $k = 4$ and $n = 3$) switches.

4.2.1 Comparative analysis of simulation results obtained for scenario-1

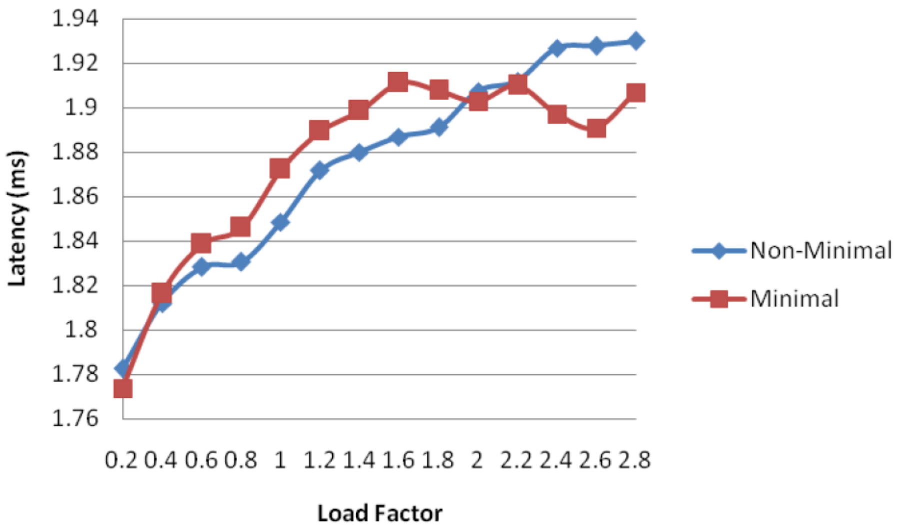
Refer Table 1 for key parameters used in the experiment, Table 2 show the results of the simulation scenario-1 while Table 3 shows the results of the simulation scenario-2.

4.2.1.1 BitComplement Traffic The BNF of the Complement Traffic shows a surprising behavior, at least at first glance. As can be seen in Fig. 4(a), the saturation point is at about 67% of the capacity for all flow control strategies. The use of more than a virtual channel (VC) is counterproductive in terms of network latency [10] so we have fixed the VC and thereafter analyzed the network. The Complement Traffic belongs to a wide class of permutations that map a *k-ary n-tree* into itself. These permutations do not generate any congestion on the descending phase and called congestion-free. Here, we see that for the Minimal Routing Algorithm graph; the message latency is approximately same for a Non-Minimal. However, in case of low network load, the network latency for Non-Minimal Routing Algorithm was 30% more. Thus, for lesser load the Non-Minimal Algorithm gives better performance.

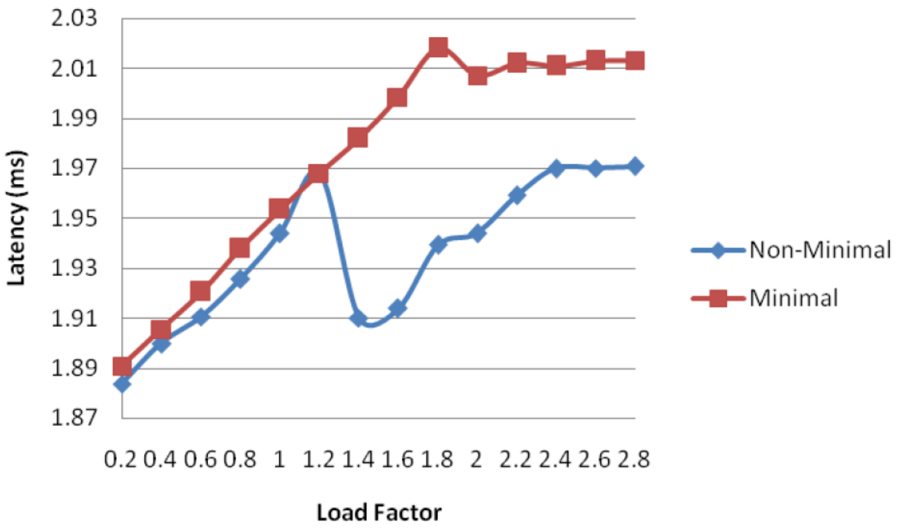
4.2.1.2 BitTranspose Traffic BitTranspose and BitReversal Traffic are considered as a standard for testing algorithms on INs. Both have similar distribution in terms of the distance from source to destination. Figure 4(b) that deals with BitTranspose Traffic, the network latency shown by Non-Minimal Algorithm is again 23% more than that of the Minimal Algorithm. There is a steep rise in network latency between the Load Factor 0.4 and 0.5. However, In case of higher Load Factor values, it is saturated.

4.2.1.3 BitReversal Traffic In Fig. 4(c), we can see a drastic difference between Minimal and Non-Minimal Routing strategies as when the Load Factor is considered as 0.1. The Non-Minimal curve is saturated much earlier than the Minimal curve that shows a steep rise from 0.6 to 0.7. Finally, there is rising in 10% more network latency, again, in case of Non-Minimal algorithm.

4.2.1.4 Uniform Distribution Traffic In Fig. 4(d), for Uniform Distribution, the Non-Minimal Algorithm again performs slightly better than Minimal Algorithm throughout the range of load applied. There is saturation in the network when the value of applied load is 0.7.



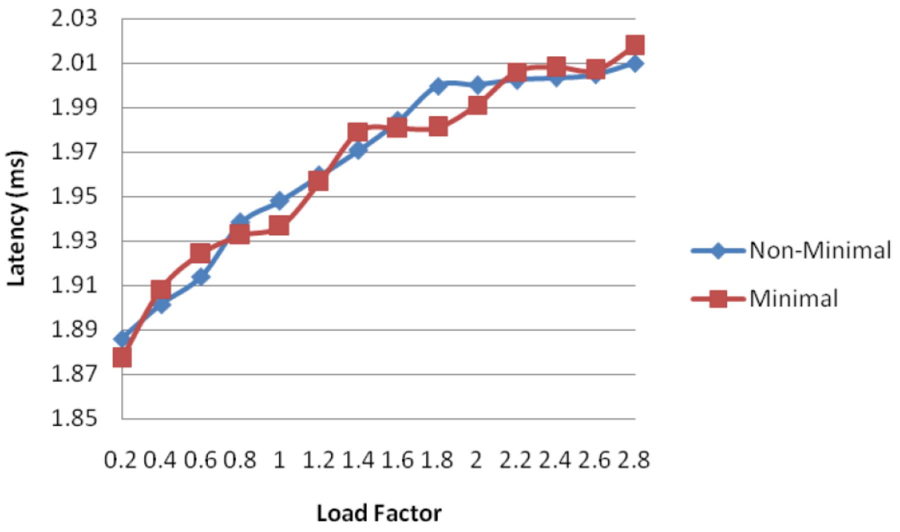
(a) *4-ary 3-tree*, BitComplement Traffic



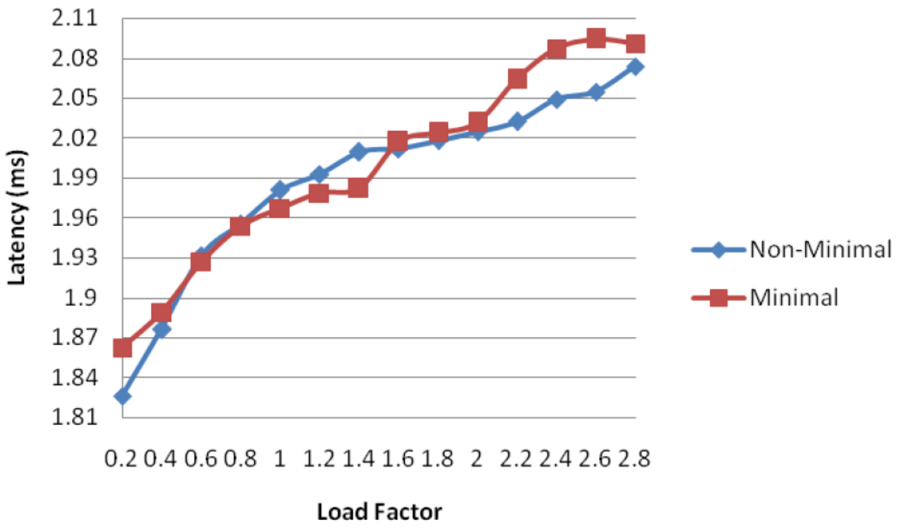
(b) *4-ary 3-tree*, BitTranspose Traffic

Fig. 5 Graphical representation of simulation results (scenario-2) of higher Load Factor versus higher Latency (microseconds) for Non-Minimal and Minimal Adaptive Routing Algorithms for six Traffic Pattern on *4-ary 3-tree* using BigNetSim. The simulation results obtained from BigNetSim are tabulated in Table 3

4.2.1.5 k-shift Traffic In case of k-shift Traffic, shown by Fig. 4(e), Non-Minimal curve shows 10% more latency as compared to the Minimal curve. Again, Minimal curve shows less Latency, and hence better performance throughout the range of load applied, finally saturating at 0.8 Load Factor.



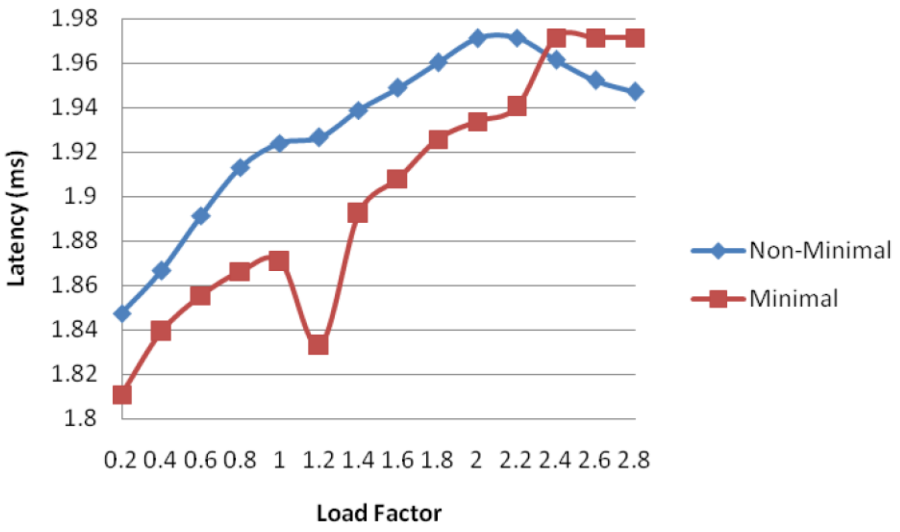
(c) *4-ary 3-tree*, BitReversal Traffic



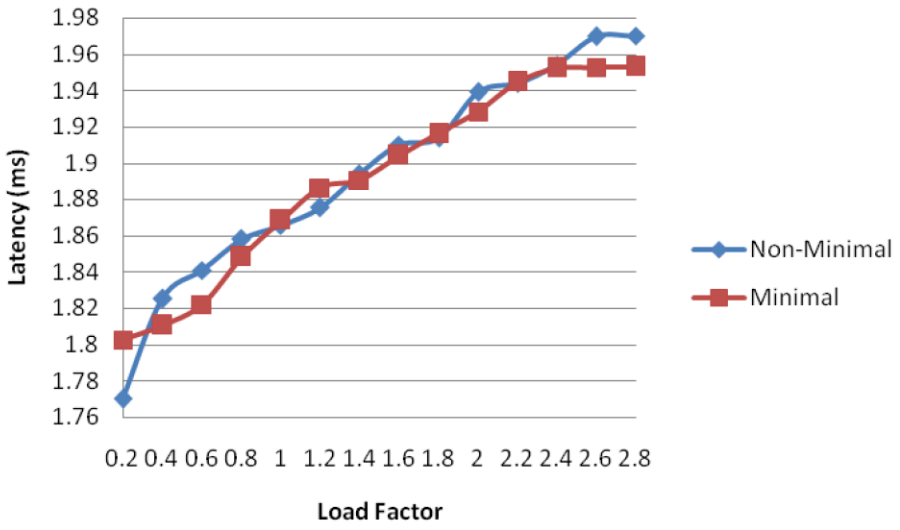
(d) *4-ary 3-tree*, Uniform Distribution Traffic

Fig. 5 (*Continued*)

4.2.1.6 Ring Traffic The Ring Traffic shows a deviant behavior when compared to other Traffic Patterns. As shown by Fig. 4(f), for higher load values, i.e., from 0.7 to 1, the Non-Minimal curve shows better performance than Minimal curve that is contrary to what observed in other patterns. For lower loads, the nature of the curve is same as in other Traffic Patterns.



(e) *4-ary 3-tree*, k-shift Traffic



(f) *4-ary 3-tree*, Ring Traffic

Fig. 5 (*Continued*)

4.2.2 Comparative analysis of simulation results obtained for scenario-2

In scenario-2, attempts have been made to create congestion like conditions by increasing the packet and message size from 256 to 2048 bytes and 200 to 400, respectively. Refer to Table 1 for key parameters used in the experiment.

4.2.2.1 BitComplement Traffic The BNF of the Complement Traffic shows a surprising behavior, at least at first glance. As can be seen in Fig. 5(a), for initial loads

Table 3 Simulation results (scenario-2) of higher Load Factor versus higher Latency (microseconds) for Non-Minimal and Minimal Adaptive Routing Algorithms for six Traffic Pattern on *4-ary 3-tree* using BigNetSim

Load Factor	Traffic Patterns											
	BitComplement		BitReversal		BitTranspose		Uniform Distribution		k-shift		Ring	
	Routing Algorithms		Routing Algorithms		Routing Algorithms		Routing Algorithms		Routing Algorithms		Routing Algorithms	
	Non-Minimal	Minimal	Non-Minimal	Minimal	Non-Minimal	Minimal	Non-Minimal	Minimal	Non-Minimal	Minimal	Non-Minimal	Minimal
0.2	1.7825	1.77358	1.886	1.8772	1.8837	1.8909	1.826	1.8622	1.8474	1.8108	1.77	1.8026
0.4	1.8119	1.8163	1.9013	1.9082	1.8998	1.9055	1.8764	1.8889	1.8665	1.839	1.8254	1.8107
0.6	1.8283	1.8384	1.9139	1.9241	1.91054	1.9206	1.9317	1.9272	1.8915	1.8554	1.841	1.8222
0.8	1.8304	1.846	1.9381	1.9324	1.9257	1.9379	1.9557	1.9537	1.9131	1.8663	1.858	1.8484
1	1.8486	1.8721	1.9479	1.9366	1.944	1.9535	1.981	1.967	1.9239	1.8707	1.8656	1.8684
1.2	1.8717	1.8893	1.9593	1.9565	1.968	1.9679	1.9928	1.9784	1.9264	1.8327	1.8755	1.8865
1.4	1.8799	1.8985	1.9709	1.9792	1.9101	1.982	2.0095	1.982108	1.9389	1.8924	1.8939	1.8905
1.6	1.8869	1.911	1.984	1.9806	1.91389	1.9984	2.012	2.0172	1.9489	1.9079	1.9101	1.9041
1.8	1.8912	1.9077	1.9997	1.9811	1.9392	2.018	2.018	2.0238	1.9602	1.9256	1.91389	1.9162
2	1.9074	1.9027	2.0001	1.9911	1.944	2.007	2.0247	2.0316	1.9711	1.9337	1.9392	1.9282
2.2	1.9119	1.9099	2.0024	2.006	1.9592	2.012	2.0322	2.064	1.9713	1.9403	1.944	1.9448
2.4	1.9267	1.8972	2.0035	2.0081	1.9699	2.011	2.0488	2.087	1.9615	1.9713	1.9542	1.9524
2.6	1.9278	1.8909	2.0047	2.0067	1.9701	2.013	2.0545	2.094	1.9522	1.9713	1.9699	1.9527
2.8	1.93	1.9069	2.0099	2.0177	1.9708	2.0133	2.0737	2.091	1.9471	1.9715	1.9702	1.9529

Minimal Algorithm shows 10% better performance than Non-Minimal. Thereafter, comparable and difference decreases with the increase in load.

4.2.2.2 BitTranspose Traffic BitTranspose and BitReversal are considered as a standard for testing algorithms on INs. Both have similar distribution over destinations in terms of the distance from source to destination. In Fig. 5(b) that deals with BitTranspose Traffic, we observe that Minimal is better than Non-Minimal again, similar to scenario-1, saturation occurs toward the end i.e. on higher loads, and the difference decreases.

4.2.2.3 BitReversal Traffic Unlike scenario-1, here in Fig. 5(c), we did not observe any drastic difference between the Minimal and Non-Minimal Routing Algorithm. The Minimal Algorithm shows 22% better performance than the Non-Minimal.

4.2.2.4 Uniform Distribution Traffic In Fig. 5(d), for Uniform Distribution, the Minimal Algorithm again performs slightly better than the Minimal Algorithm throughout the range of load applied. Minimal algorithm shows 22% better performance than the Non-Minimal for higher loads.

4.2.2.5 k-shift Traffic In case of k-shift Traffic, shown by Fig. 5(e), we observe that the performance of Minimal is better than Non-Minimal throughout. However, it is interesting to note that there exists a wide gap of 31% between Minimal and Non-Minimal in case of 1.2 Load Factor.

4.2.2.6 Ring Traffic The Ring Traffic shows expected behavior in scenario-2, i.e., when congestion is increased by increasing message and packet size. As shown by Fig. 5(f), for all load values, the Minimal curve shows better performance than Non-Minimal curve, the difference increases for post 0.8 value of Load Factor.

5 Conclusion

We have introduced Parametric Family of regular topologies that consists of *k-ary n-trees* and analyzed two Adaptive Algorithms, on discussed six Traffic Patterns.

As shown in Figs. 4(a)–4(e) and Table 2 under scenario 1, the Network Latency for Non-Minimal Routing Algorithm is higher as compared to Minimal Routing Algorithm. Moreover, from these results, we can conclude that the network performance for *k-ary n-tree* is much higher in case of the Minimal Algorithm as compared to the Non-Minimal Algorithm. However, for the Ring Traffic Pattern, on higher loads, i.e., from 0.7–1.0, there is an increase in latency value for the Minimal Adaptive Algorithm as compared to the Non-Minimal Adaptive that decreases the Minimal Algorithms performance over the Non-Minimal.

However, in Figs. 5(a)–5(e) and Table 3 under scenario-2 when congestion is increased by increasing packet and message size while keeping the number of virtual channels constant, we observe that for all Traffic Patterns, including Ring we get higher performance for Minimal than Non-Minimal thereby proving the Minimal adaptive algorithm to be better in every case, even after an increase in congestion. Congestion-free patterns are a powerful tool to reach optimal performance and guaranteed scalability.

6 Future work

We will analyze the various Traffic Patterns on Fat-Tree topology (having larger bandwidth on higher nodes) and to analyze whether it has better efficiency than *k-ary n-tree* topology.

Acknowledgements A very special thanks goes to the Jaypee Education System (JES) and Jaypee University of Information Technology (JUIT) for providing excellent Research Environment and Learning Ambience and Sun Solaris and High-end Parallel Computing Lab at JUIT.

This research work is dedicated to my Shri Shri 1008 Swami Shri Paramhans Dayal Sacchidanand Maharaja Ji and the loving memories of my departed maternal grandfather, grandmother, and father-in-law, who continue to guide me in spirit.

References

1. Nitin, Subramanian A (2008) Efficient algorithms to solve dynamic MINs stability problems using stable matching with complete TIES. J Discrete Algorithms 6(3):353–380
2. Nitin, Garhwal S, Srivastava N (2009) Designing a fault-tolerant fully-chained combining switches multi-stage interconnection network with disjoint paths. J Supercomput. doi:10.1007/s11227-009-0336-z
3. Nitin, Chauhan DS, Sehgal VK (2008) Two $O(n^2)$ time fault-tolerant parallel algorithm for inter NoC communication in NiP. Springer, Berlin, ISBN: 978-3-540-79186-7, pp 267–282. Invited
4. Nitin, Vaish R, Shrivastava U, Rana M (2009) Adaptive deterministic routing algorithm for k-ary n-cube torus network. In: 7th Annual workshop on Charm++ and its applications. Parallel Programming Lab, University of Illinois, Urbana Champaign
5. Nitin, Sehgal VK, Bansal PK (2007) On MTTF analysis of a fault-tolerant hybrid MINs. WSEAS Trans Comput Res 2(2):130–138
6. Nitin (2006) Component level reliability analysis of fault-tolerant hybrid MINs. WSEAS Trans Comput 5(9):1851–1859
7. Dally WJ, Towels B (2005) Principles and practices of interconnection networks. Morgan Kaufmann, San Mateo
8. Wani MA, Arabnia HR (2003) Parallel edge-region-based segmentation algorithm targeted at reconfigurable multi-ring network. J Supercomput 25(1):43–63
9. Arabnia HR (1990) A parallel algorithm for the arbitrary rotation of digitized images using process-and-data-decomposition approach. J Parallel Distrib Comput 10(2):188–193
10. Arabnia HR, Oliver MA (1989) A transputer network for fast operations on digitised images. Int J Eurograph Assoc 8(1):3–12
11. Singh A, Dally WJ, Gupta AK, Towles B (2004) Adaptive channel queue routing on k-ary n-cubes. In: Proceedings of the 16th annual ACM symposium on parallelism in algorithms and architectures, pp 11–19
12. Dally WJ (1990) Network and processor architecture for message-driven computers, VLSI and parallel computers. Morgan Kaufmann, San Mateo, pp 140–222
13. Gomez C, Gilabert F, Gomez ME, Lopez P, Duato J (2007) Deterministic versus adaptive routing in fat-trees. In: Proceedings of the international parallel and distributed processing symposium, p 297
14. Leighton FT (1992) Introduction to parallel algorithms and architectures: arrays, trees, hypercubes. Morgan Kaufmann, San Mateo
15. Petrini F, Vanneschi M (1997) k-ary n-trees: high performance networks for massively parallel architecture. In: Proceedings of the 11th international parallel processing symposium, pp 87–93
16. Gomez C, Gilabert F, Gomez ME, Lopez P, Duato J (2007) An efficient fault-tolerant routing methodology for fat tree interconnection networks. In: Proceedings of the 5th international symposium on parallel and distributed processing and applications, pp 509–522
17. Nguyen TD, Synder L (1994) Performance analysis of a minimal adaptive router. In: Lecture notes in computer science, vol 853. Springer, Berlin, pp 31–44
18. Bolding K, Fulgham M, Synder L (1997) The case for chaotic adaptive routing. IEEE Trans Comput 46(12):1281–1292

19. Zheng G, Kakulapati G, Kale LV (2004) BigSim: a parallel simulator for performance prediction of extremely large parallel machines. In: Proceedings of the 18th international parallel and distributed processing symposium, vol 1, p 78b
20. Wilmarth TL, Zheng G, Bohm EJ, Mehta Y, Choudhury N, Jagadishprasad P, Kale LV (2005) Performance prediction using simulation of large-scale interconnection networks in POSE. In: Proceedings of the 19th workshop on principles of advanced and distributed simulation, pp 109–118
21. Kale LV, Krishnan S (1993) CHARM++: a portable concurrent object oriented system based on C++. ACM SIGPLAN Notices 28(10):91–108
22. Choudhury N, Mehta T.L. Wilmarth Y, Bohm EJ, Kale LV (2005) Scaling an optimistic parallel simulation of large scale interconnection networks. In: Proceedings of the 37th conference on winter simulation, pp 591–600