Designing Scalable Routers with a New Switching Architecture

Zuhui Yue[1], Youjian Zhao[1], Jianping Wu[2], Xiaoping Zhang[3]
Department of Computer Science, Tsinghua University, Beijing, P.R.China, 100084
[1]{yuezuhui, zhaoyj}@csnet1.cs.tsinghua.edu.cn, [2]jianping@cernet.edu.cn,
[3] zhxp@tsinghua.edu.cn

Abstract

Today most routers consist of line cards and centralized switching fabrics. It is now more and more difficult to increase the speed of line cards. Each generation of routers consume more power, and it gets more difficult to package a router in one rack of equipment. The research on routers shifts from single-rack systems to multi-rack ones. Most switches in routers today reside out of line cards. As the number of line cards increases, they will meet scalability problems. The Cellular Router (CR) architecture introduced in this paper is a new switching architecture. This architecture can be used in scalable routers and shows excellent properties. We give a detailed discussion of the CR architecture and show the implementation of the basic CR architecture. There exist some problems in the basic CR architecture. We solve them by introducing Mirror Points (MPs). We also introduce two minimal routing algorithms.

1. Introduction

Routers play an important role in the Internet. The continual growth of user traffic requires a corresponding increase in router capacity. Nowadays most routers consist of line cards and centralized switching fabrics. A lot of work, such as addressing, classification, SAR (Segmentation and Reassembly), etc., resides in the line cards. The switching fabrics receive fixed-size packets (often referred as cells) from input ports and forward them to output ports. To meet the growing user traffic, switching fabrics are needed to support both faster ports and more ports. There exist many problems [1] that limit the increase of port-rate.

For low numbers of ports, crossbar is selected as the switch topology, owing to the simplicity and non-blocking properties. However, its cost grows as the square of the number of ports, and cannot be economically scaled to a large number of ports. Multistage switching fabric architectures are needed to handle modest or large numbers of ports. The Benes network [2] features a low cost $N \cdot 2\log N$ and is free of internal blocking. The Benes network is rearrangeably non-blocking, and setting up new connections may destroy the existing connections. The Clos network [3] is useful in practice and stimulating in research [4]. There are still some unsolved theoretical problems with Clos network. The Load-Balanced switch architecture proposed in [5] is considered promising for this approach eliminates the scheduler and can be realized with optics [6]. However the router described in [6] cannot be built with technology available today. There exists a centralized switch in the router implemented with the switches listed above. This architecture greatly limits the scalability and the centralized switch becomes the *Single Point of Failure* (SPF).

Many papers have appeared in the literature offering good taxonomies of switches. In this paper, we classified them into two categories: switches residing out of the line cards (SO) and switches residing on the line cards (SI). In a SO switch all the line cards are connected to the separated switches while in a SI switch line cards connect each other with links and switches reside on the line cards. All the switches listed above fall into the SO category. The 3-D Torus network is used in the Avici TSR [7] in which switches reside on the line cards. This architecture [8] shows good scalability.

The rest of this paper is organized as follows. Section 2 presents the basic *Cellular Router* (CR) architecture. In section 3, we give a brief introduction of the line card design. The basic CR architecture shows poor fault tolerance and some improvements will be described in Section 4. Section 5 introduces two minimal (choosing a shortest path for each packet) routing algorithms. Section 6 presents some other routing algorithms. Section 7 provides a summary of our work and talks about the future work.

This work was supported in part by grants NSFC-60473082 and 863-2003AA103210.

Proceedings of the Joint International Conference on Autonomic and Autonomous Systems
and International Conference on Networking and Services (ICAS/ICNS 2005)
0-7695-2450-8/05 $20.00 © 2005 IEEE

2. The basic CR architecture

The basic CR architecture consists of line cards or racks which are deployed on the vertices of regular hexagons.

2.1 A single rack with a layer of line cards

We first consider the case in which there is only one layer of line cards in a single rack. We place 7 line cards in one layer. As shown in figure 1(a), the line cards are numbered 0 to 6. These numbers denote the order in which the line cards are added. When a line card is plugged into the rack, the related link(s) with the existing node(s) will also be added. We define such a hexagon as the *Basic Element* (BE). The numbers shown in figure 1(a) are defined as *BE Numbers* (BENs).

In practice, the line cards are always placed parallel in the rack. We can place one BE as shown in figure 1(b)~(j), in which arrowed dashed (solid) lines represent the data channels implemented on the backplane with equal length.

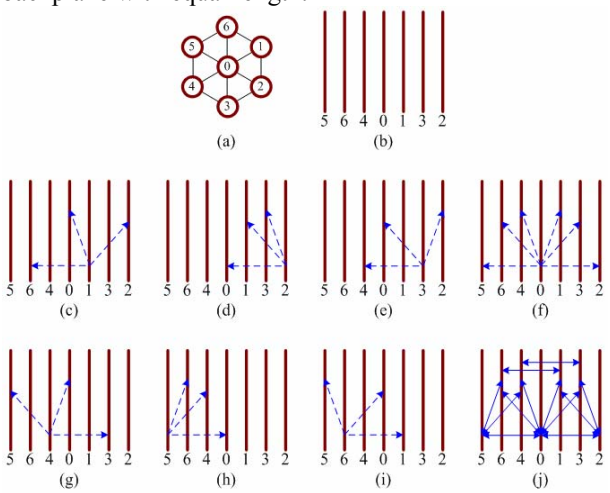

Figure 1: Basic Element

2.2 A single rack with multilayer line cards

We can place one BE in each layer and connect the corresponding line cards as shown in figure 2.

The maximum degree of each line card is 8. We reserve two of them for adjacent layers. In each rack, a line card can be identified with *(l, st)*. Here *l* represents the layer where the line card is located and *st* represents the *BEN* of the line card. In figure 2, to increase the connectivity of the nodes in the top layer and the bottom layer, we can connect the nodes in the top layer with the corresponding nodes in the bottom layer.

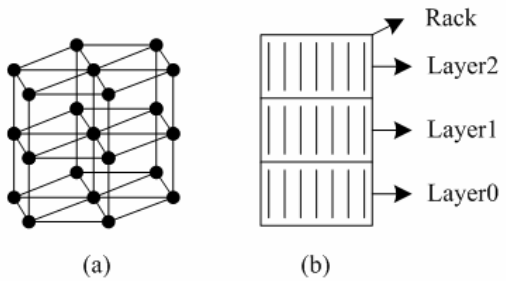

Figure 2: BEs placed in multiple layers

2.3 Multi-rack system

Each generation of routers consume more power than the last, and it is now difficult to package a router in only one rack of equipment. There has therefore been a move towards multi-rack systems. We can see that CR shows good scalability in a multi-rack system.

First, we show how to place the racks. In figure 3, each node represents a rack described in section 2.1 or 2.2.

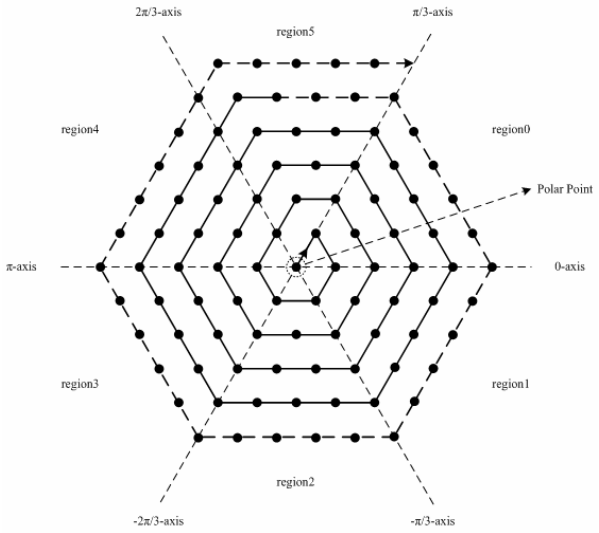

Figure 3: Multi-rack system

In figure 3, we define the central point as *Polar Point* (PP). There are six axes which start from the PP. They are defined as 0-axis, -π/3-axis, -2π/3-axis, π-axis, 2π/3-axis and π/3-axis respectively. These six axes divide the whole plane into six equal-area regions, and we define these regions as region 0~5. We first place one rack on the PP. Then we place the second rack on the π/3-axis. After we have placed six racks around the PP clockwise, we place other racks around these six racks. In figure 3, the arrowed solid line starting from the PP shows the order in which the racks are deployed in such a multi-rack system. We define the PP as *Circle*

0 and the six racks around it as *Circle 1*, etc. We call this architecture as *Cobweb* architecture. The sequence number the rack appears in Cobweb is defined as the *Cobweb Number* (CN). Combined with the line card identifier *(l, st)* in a single rack, a line card can be identified as *(CN, l, st)*. In 2-D case, we can identify a line card with *(CN, st)*. We say that line card *A* is larger than line card *B* if CN_A is larger than CN_B. When CN_A is equal to CN_B, we say that line card *A* is larger than line card *B* if st_A is larger than st_B. Line card *A* is equal to line card *B* only if they are the same line card. Table 1 shows the number of nodes in a regular Cobweb system.

Table 1 Number of nodes in a regular Cobweb system

Circle Number	Number of nodes in this circle	Number of nodes in each region	Range of nodes in this circle	Total nodes
0	1	-	0	1
1	6	1	1~6	7
2	12	2	7~18	19
…	…	…	…	…
n	6n	n	3n(n-1)+1 ~3n(n+1)	3n(n+1)+1

If we place BEs as shown in figure 3, we can find that some line cards are shared by two or three racks. Figure 4, in which each node represents one line card, shows a multi-rack system with three racks. *Line card* (LC) 0 and *LC2* are shared by two racks and *LC1* is shared by three racks.

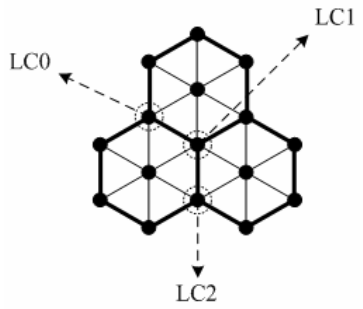

Figure 4: Shared line cards

We can find that there may be four types of racks as shown in figure 5 in a multi-rack system. They consist of 7, 5, 4 and 3 line cards respectively.

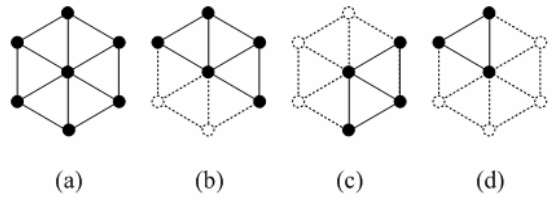

Figure 5: Four types of racks in a multi-rack system

3. Design of the line card

Unlike the architectures of most other routers, where dedicated switching fabrics are need, CR architecture distributes the switching task into each line card. This indicates that each line card should handle not only its own traffic but also the traffic from the neighbor line cards. As shown in figure 6, the input traffic could arrive from the eight ingress channels and the ingress link of its own. This input traffic is then distributed to the different output queues. The inner switching fabric can be normal crossbar. We can introduce separated queues, named *Virtual Channel Queue* (VCQ), for each output. There are many algorithms [1] [9] [10] which can solve the HOL problem. Unlike traditional switching fabrics, the inner switching fabric need not be changed as the number of line cards increases.

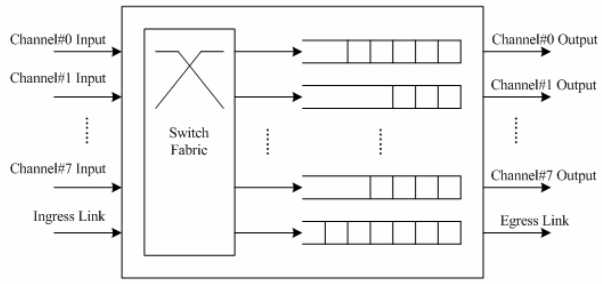

Figure 6: The line card structure in CR architecture

4. Improvement of the basic architecture

4.1 Fault tolerance considerations

Here we first discuss the BE again. As shown in figure 7(b), it can be found that if the central point (line card) is down the BE will change to a dual-ring. If one node in the left nodes is down, the dual-ring will turn to a line. If another node breaks down, the left nodes become unconnected as shown in figure 7(d). It is known that the maximum degree of each line card is 6 (excluding the reserved two links used for adjacent layers). However, in the BE the degree of each node, except for the central point, is only 3. When some nodes fail, the left nodes are affected greatly.

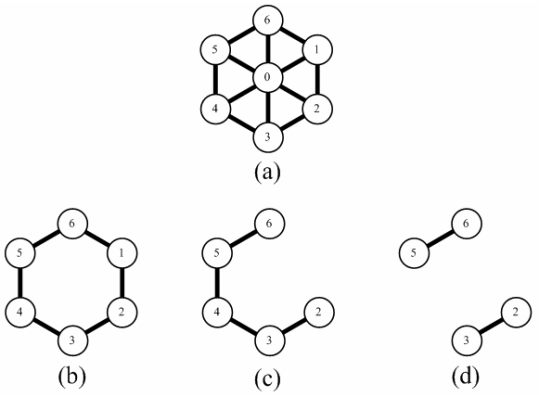

Figure 7: Node failure in the BE

4.2 Properties of the CR architecture

To ease the problem, here we only consider the *Cobweb* architecture with only one layer.

We define the set Γ, which consists of nodes with *linked degree* (LD) of less than 6. We also define the set Λ, which consists of nodes with LD of 6. For example, the node 1, 2, 3, 4, 5 and 6 in figure 7(a) belong to Γ and the node 0 belongs to Λ. We can easily find that there are only two types of edge nodes, nodes with LD of 3 and nodes with LD of 5 in a regular Cobweb system. Table 2 shows the number of nodes with LD of 3 or 5.

Table 2 Number of line cards with LD of 3 or 5

Circle Number	Number of line cards with LD of 3	Number of line cards with LD of 5
0	6	0
1	12	6
2	18	12
3	24	18
…	…	…
n	6(n+1)	6n

4.3 Mirror Points

First we introduce the concept of *Envelope*. Envelope is the regular hexagon that can encapsulate the CR architecture. If no nodes appear on the envelope, we define it as *External Envelope*. If some nodes appear on the envelope, we define it as *Internal Envelope*. As shown in figure 8(a), the dashed regular hexagon is the External Envelope of the BE. In figure 8(a), *node 4'* on the External Envelope is defined as the *Mirror Point* (MP) of *node 4* related to *node 0*. *Node 4'* can be connected to node 6, 1 and 2. That is, in practical system we will connect node 4 with node 6, 1 and 2. We can also connect node 1 to node 3, 4 and 5 with the help of MP, etc., as shown in figure 8(b). In the end we get a full-connected system as shown in figure 8(c). Figure 9 shows the case in which there are two circles of racks. In this way, we can improve the connectivity of edge nodes.

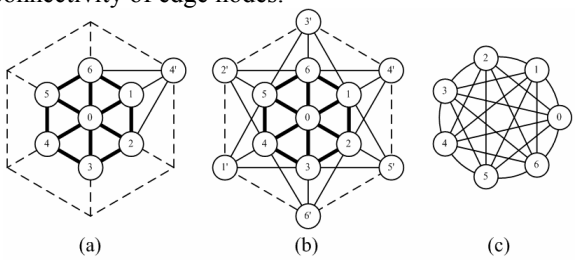

Figure 8: Modification of the BE

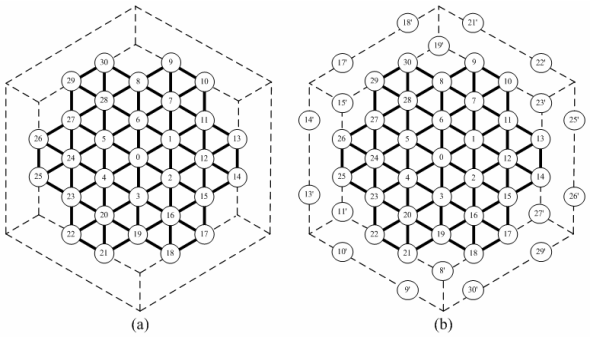

Figure 9: Mirror Points in a system with two circles of racks

5. Minimal routing algorithms in the CR architecture

In the CR architecture, when cells are forwarded from source node(s) to destination node(s), they may pass some intermediate nodes. It is very important to find a routing algorithm. Before discuss the algorithm, we introduce some basic concepts.

5.1 Preliminaries

The following discussions describe routing algorithms that are *minimal*. That is, they select the shortest path among all optional paths. We restrict our discussion to 2-D CR architecture and ignore the impact of mirror points. Each link is unidirectional, so there are two links between any adjacent nodes. We define the length of each link as 1.

We further assume that the architecture uses *store-and-forward* flow control with each node having buffers of infinite length. Whenever contention between packets for the same outgoing link in a node occurs, the oldest packet is chosen. This can isolate the effect of the routing algorithm from flow control issues.

Suppose the source node is *s*, and the destination node is *d*. When cells are forwarded from *s* to *d*, they pass a series of intermediate nodes:

$J = (j_1, j_2, \cdots, j_n)$. We define D_{sd}, the distance between *s* and *d*, as the length of the shortest path between them. For shortest paths, we get $D_{sd} > D_{j_1 d} > D_{j_2 d} > \cdots > D_{j_n d} = 1$.

We denote the six neighbors of *s* as n_0, n_1, n_2, n_3, n_4 and n_5. If we connect *s* and *d*, there exist two cases. In the first case, *d* appears on the extended line connecting *s* and one of its neighbors n_i. In the second case, *d* appears between two extended lines. When we choose the next-hop node on the shortest path, n_2 is selected in figure 10(a) while in figure 10(b) we can choose one between n_2 and n_3. We call n_2 as *Right Neighbor* (RN) of $\overrightarrow{sd}$ and n_3 as *Left Neighbor* (LN) of $\overrightarrow{sd}$.

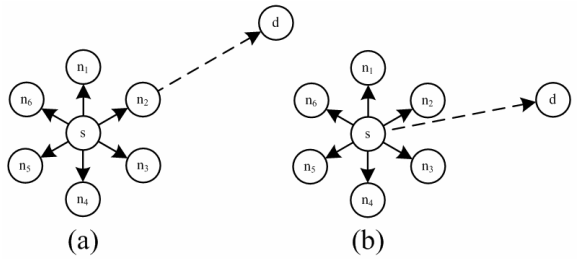

Figure 10: Relationship between s and d

5.2 Selection of next-hop node

To find the next-hop node on the shortest path, we should first identify each node in the architecture. There are two methods to achieve this: method based on the PP and method based on the source node.

5.2.1. Method based on the PP. According to the characteristic of the CR architecture, it is proper to identify each node under polar coordinate. We choose the *0-axis* as polar axis *OX*. So node *P* can be identified as *(ρ, θ)*. Here *ρ* represents the length of *OP*, and *θ* is the value of ∠ POX. For example, in figure 11, $tg\theta = PN/ON = (PM\sin\pi/3)/(OM - PM\cos\pi/3) = (3\sin\pi/3)/(5 - 3\cos\pi/3)$. In this way, we can easily calculate the values of *ρ* and *θ* for each node.

To find the next-hop node on the shortest path, we should find the association between $\overrightarrow{sd}$ and $\overrightarrow{sn_0}$, … , $\overrightarrow{sn_5}$. In figure 12, we know the values of ρ_s, θ_s, ρ_d and θ_d. Then we can calculate the value of θ_d' according to these values. In the end we can choose the next-hop node in the way discussed in section 5.1.

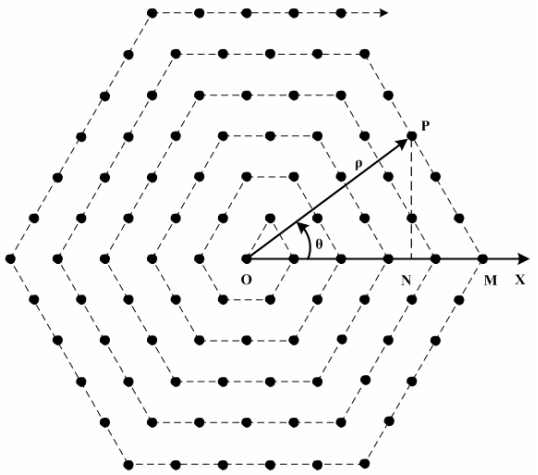

Figure 11: Polar coordinate

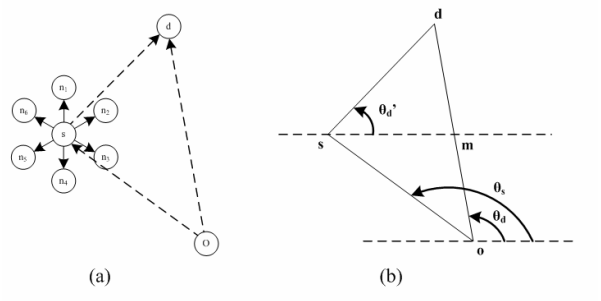

Figure 12: Calculation of θ_d'

5.2.2. Method based on the source node. It is easy to calculate the coordinate of each node in the method based on the PP. However, the relationship of *s* and *d* should be calculated again to determine the next-hop node.

If we set up the coordinate of each node on the source node, the speed of switching will be improved greatly. All the calculation can be carried out offline. That is, we can get these values before switching.

5.2.3. Selection of next-hop node. When there is only one next-hop node on the shortest path, we choose it without considering the queue state. When there are two next-hop nodes, we can choose one randomly or the one with lower queue load.

5.3 Performance evaluation

We choose the BE At the beginning of each time slot, a fixed sized packets arrives at each node. We define the length of each packet as 1. The destination of each arrival packet is randomly chosen. The queues on each node are organized as VOQs and the length of each queue is 100. We choose *Right Neighbor First* (RNF) algorithm. That is, when there are two optional

next-hop nodes we always select the right neighbor. Packets are injected into the BE for a specific time. After all the packets have left the BE, we record the number of packets that arrive at the BE or leave the BE. Then we can calculate the ratio between them as shown in figure 13. For the length of each queue is finite, some packets are dropped when the ingress queues are full.

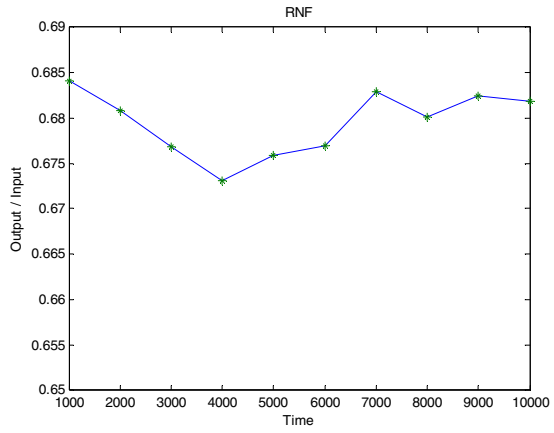

Figure 13: Throughput of RNF algorithm

6. Other routing algorithms

Some routing algorithms for Torus network are introduced [11] ~ [14]. These algorithms can also be applied to the CR architecture with some modification. These algorithms introduce new problems, such as deadlock and livelock [14], packet reordering, etc.

7. Conclusions

The CR architecture is promising and suitable for scalable routers. In theory this architecture can be scaled from a system with only one line card to a system with an infinite number of line cards. With some improvements, this architecture works well when some nodes or links fail.

In future, we plan to go deep into the properties of the regular and irregular CR architectures. We plan to develop routing algorithms which will apply to normal operation and some other algorithms which can be used in case of faults. With this architecture, we also can make some research on multicast or QoS switching.

References

[1] Nick McKeown, "iSLIP: A Scheduling Algorithm for Input-Queued Switches", *IEEE Transactions on Networking*, Vol 7, No.2, April 1999.

[2] V. Benes, "Optimal Rearrangeable Multistage Connecting Networks", *Bell Systems Technical Journal*, vol. 43, no. 7, pp. 1641–1656, July 1964.

[3] Charles Clos, "A Study of Non-blocking Switching Networks", *Bell System Technical Journal*, 1953, vol. 32, no. 2, pp. 406-424.

[4] Andrzej Jajszczyk, "50 Years of Clos Networks: A Survey of Research Issues", available at: http://www.tlc-networks.polito.it/HPSR2003/talks/Clos_HPSR_AJ.pdf

[5] C.-S. Chang, D.-S. Lee and Y.-S. Jou, "Load balanced Birkhoff-von Neumann switches, Part I: one-stage buffering", *Computer Comm.*, Vol. 25, pp.611-622, 2002.

[6] Isaac Keslassy, Shang-Tse Chuang, Kyoungsik Yu, David Miller, Mark Horowitz, Olav Solgaard, Nick McKeown, "Scaling Internet Routers Using Optics", *ACM SIGCOMM* Aug. 2003, Karlsruhe, Germany.

[7] [Online] Available at: http://www.avici.com/products/tsr.shtml

[8] William J. Dally, "Scalable Switching Fabrics for Internet Routers", available at: http://www.avici.com/technology/whitepapers/TSRfabric-WhitePaper.pdf

[9] Nick McKeown, Adisak Mekkittikul, Venkat Anantharam and Jean Walrand, "Achieving 100% Throughput in an Input-Queued Switch," *IEEE Transactions on Communications*, Vol.47, No.8, August 1999.

[10] Adisak Mekkittikul, and Nick McKeown, "A Practical Scheduling Algorithm to Achieve 100% Throughput in Input-Queued Switches," *IEEE Infocom 98*, Vol 2, pp. 792-799, April 1998, San Francisco.

[11] Arjun Singh, William J. Dally, Amit K. Gupta and Brian Towles,"Adaptive Channel Queue Routing on k-ary n-cubes", *ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, Barcelona, Spain, June, 2004.

[12] Arjun Singh, William J. Dally, Brian Towles and Amit K. Gupta,"Globally Adaptive Load-Balanced Routing on Tori", *Computer Architecture Letters*, Volume 3, March, 2004.

[13] Arjun Singh, William J. Dally, Amit K. Gupta and Brian Towles,"GOAL: A Load-Balanced Adaptive Routing Algorithm for Torus Networks", *International Symposium on Computer Architecture (ISCA)*, San Diego, California, USA, June, 2003.

[14] Arjun Singh, William J. Dally, Brian Towles, and Amit K. Gupta,"Locality-Preserving Randomized Oblivious Routing on Torus Networks", *ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, Winnipeg, Manitoba, Canada, August, 2002.