The Effect of the Router Arbitration Policy on the Scalability of ServerNet™ Topologies

V.Shurbanov[1], D.R.Avresky[1], and R.Horst[2]

[1] Network Computing Research Laboratory
Dept. of ECE, Boston University
15 St. Mary's St., Boston, MA 02215
Phone:(617) 353-9850
Fax: (617) 353-6440
E-mail: {avresky,vash}@bu.edu

[2] Tandem Computers Incorporated
19333 Vallco Parkway
Cupertino, CA 95014
Phone: (408) 285-1816
Fax: (408) 285-1819
E-mail: horst_bob@tandem.com

Abstract

In this paper we extend a previously introduced method for optimizing the arbitration policy employed by ServerNet routers and we evaluate the method's effect on scalability. The ServerNet^{TM} System Area Network (SAN), developed by Tandem Computers Inc., is a wormhole-routed, packet-switched, point-to-point network, with special attention paid to reducing latency and to assuring reliability. The ServerNet SAN uses multiple high-speed, low-cost routers to rapidly switch data directly between data sources and destinations.

The ServerNet router arbitration algorithm has the ability to assign different priorities to different links thus controlling the distribution of available bandwidth among channels. We propose an analytical method that allows us to determine the correct priorities for channels so as to achieve fair arbitration thus increasing throughput and scalability, minimizing latency, and eliminating tree saturation.

In our study we use data generated by a simulation tool to validate the proposed analytical model for determining the weights of router ports and to evaluate the resulting performance improvements.

1 Introduction

Multistage networks are finding increasing use in building clusters of workstations and PCs. The networks that support such connectivity must provide high bandwidth, low latency, scalability, low cost, high level of usability, and reliability [4]. A key parameter that influences the network performance characteristics such as maximum latency, throughput, scalability, and tree saturation, is the arbitration policy implemented in routers.

The ServerNet System Area Network (SAN) provides high-speed communications from processor to processor, processor to I/O device, or I/O device to other I/O devices. The development of the ServerNet SAN [3] has prompted the exploration of new topologies in search for better ways of constructing networks to optimize performance, while avoiding the possibility of deadlock. Several ServerNet SAN topologies based on have been studied through simulation in an attempt to estimate their performance characteristics including maximum latency, scalability, and tree saturation effects.

The weights (i.e., increment values) assigned to router ports have a profound influence on the performance of the network and in particular on the fairness of arbitration (i.e. QoS) because these weights determine the fraction of the channel bandwidth that will be provided to the device(s) connected to a port. We have developed a method that allows us to determine the amount of traffic arriving at router port X and competing for router port Y under a particular traffic pattern, and on the basis of this to calculate the port weights so as to provide a fair arbitration scheme i.e., each device should receive bandwidth proportional to its requirements as compared to the requirements of its competitors.

In this paper we extend the initial 16-CPU model and apply it to the 128-CPU Thin and Fat Fractahedron ServetNet topologies. We also study and evaluate the effect of the improved arbitration on the scalability of these topologies.

2 ServerNet basics

The ServerNet SAN is a wormhole-routed, packet-switched, point-to-point network, with special attention paid to reducing latency and to assuring reliability [3]. It uses multiple high-speed, low-cost routers to rapidly switch data directly between data sources and destinations.

The ServerNet SAN consists of the following components: a router that provides a six-port crossbar switch on a single chip, a processor interface that provides dual ports for fault-tolerant connection to two ServerNet fabrics, and a peripheral device interface that provides an open external interface to the ServerNet network and can support connections to standard buses, such as VME, PCI, and SCSI.

Using the ServerNet SAN, input/output and processing capacities can be scaled independently. This is possible because any ServerNet router may support connections to any type of ServerNet device, including processor and peripheral interfaces [3].

2.1 ServerNet Router Arbitration

In an actual ServerNet [3] router, each output port includes an autonomous arbiter that is responsible for packet ordering. ServerNet routers use the ALU arbitration algorithm which is a variant of Bresenham's Midpoint Line Algorithm. Each port is provided with a configurable increment parameter. The value of this increment parameter along with the increment parameters of the other ports competing for the same output port determines the portion of the output port's total bandwidth that each port is allowed to use. When an output port enters an arbitration cycle, it selects the requesting input port with the highest accumulator value. Each time an input port with a packet loses arbitration for an output port, the increment value is added to the port's accumulator. When a port wins arbitration for a particular output port, it subtracts the sum of the other contender's increment values from its accumulator.

The motivation for modifying the router arbitration biasing method is to enforce fair arbitration and thus improve the network performance characteristics (throughput and latency) in general and the scalability in particular.

2.2 Scalabilty of ServerNet

The up-sizing of ServerNet is achieved through a new fractahedral topology (Figure 1) that preserves symmetry, eliminates loops, and utilizes existing routers. The network is like a fractal, in that it has the same structure when viewed at different levels. We are currently calling this new class of topologies *Fractahedrons* - short for fractal-tetrahedrons. The simplest Fractahedron - a two-level thin fractahedron - can be constructed by connecting eight of the Tetrahedron topologies with the help of a tetrahedron. Similarly, this procedure can be repeated to create a three-level fractahedron by replicating the two-level Fractahedron eight times and connecting the eight fractahedrons by means of a tetrahedron. Theoretically, fractahedrons of arbitrary complexity could be created in the same manner [4].

We considered the two simplest classes of fractahedron topologies - the two-level thin and fat fractahedrons - with subnetworks of the 16-CPU Tetrahedron topology as the fractahedral branches (see Figure 1). Since the both the thin and fat two-level fractahedrons are composed of eight 16-CPU Tetrahedron topologies, they contain 128 CPUs and 832 I/O devices, we will refer to them as the *128-CPU-Thin and 128-CPU-Fat* topology, respectively. The two topologies differ from one another in the number of links from one level to another. The topologies are shown in Figures 1.

2.3 Scalability metrics

Although many scalability metrics have been proposed and applied to analyzing parallel computer systems [2, 7, 11, 12] there still does not exist a rigorous definition of scalability that has been widely accepted. In [5, 12] the authors loosely define scalability as the property of a computer system to exhibit performance linearly proportional to the number of processors employed. We based our analysis and conclusions on this definition. In [12] the authors further investigate the problem of what performance metric should be used for assessing scalability and point out that the chosen scalability measure should provide information about how the system size influences its performance. This important guideline was followed closely in our studies.

For a given topology, traffic pattern and request generation rate λ, the asymptotic increase of throughput *IT(λ,n)* is the best increase of throughput attainable, varying only in the number *(n)* of processing elements. Let *T(λ,1)* be the throughput of one processing element (16-CPU Tetrahedron topology), $T(\lambda, n)$ and $T_I(\lambda, n)$ be the real and ideal (i.e., maximum), respectively, throughput of a system with *n* processing elements, and *c(λ,n)* be the total loss of throughput due to congestion effects. The asymptotic increase of throughput is formally defined as:

$$IT(\lambda, n) = \frac{T_I(\lambda, n) - c(\lambda, n)}{T(\lambda, 1)}. \quad (1)$$

The scalability definition given in [8] was based on the relative performance of a real machine compared with that of an idealized theoretical version of that machine. In our case, the idealized model of the network is one in which contention and congestion effects do not occur. Therefore, in an idealized model, the throughput would be exactly proportional to the number of processing elements. The increase in throughput for the ideal machine can be calculated as:

$$IT_I(\lambda, n) = \frac{T_I(\lambda, n)}{T(\lambda, 1)} = n. \quad (2)$$

The scalability $\Phi(\lambda, n)$ of a topology under a given traffic pattern is defined as the ratio of the asymptotic increase

of throughput *IT(λ,n)* of the real network and the asymptotic increase of throughput $IT_I(\lambda, n)$ of the ideal network, as follows:

$$\Phi(\lambda, n) = \frac{IT(\lambda, n)}{IT_I(\lambda, n)} = \frac{T(\lambda, n)}{T_I(\lambda, n)}. \quad (3)$$

Hence, the larger the scalability, the higher the throughput that can be achieved under the traffic pattern; ideally, $\Phi(\lambda, n) = 1$. Note that this scalability measure takes into account only the communication capabilities of the network; it does not address the issue of the computer system's computational power or any other aspects of scalability.

3 Analysis of the Message Flow in the 128-CPU ServerNet SAN Topologies

The method we employed relies on aggregating the parameters used to model the devices and the traffic pattern in the network. Aggregate models are applicable to networks that exhibit regularities and fractahedron topologies exhibit the same structure at different levels of complexity.

Our model is based on the *aggregate message injection rate* (AMIR) of a port, which is, essentially, the number of messages arriving during a time interval (we used 1 *ms*) at a particular router port. The traffic arriving at a particular router port is generated by a set of devices; we will refer to these devices as *source* devices. We calculate the AMIR of a port as the sum of the aggregate request generation rate and the aggregate response generation rate of the source devices. The aggregate request generation rate is calculated as the *weighted* sum of the request generation rates (λ) of the source devices; the aggregate response generation rate is the *weighted* sum of the request generation rates (λ) of all devices that target the source devices because the generation of a response is triggered by the reception of a request. The sums described above are justified by the fact that the request generation rate, λ, is modeled as an independent Poisson stream [9, 6].

The weights employed in these calculations are in fact probabilities that are inherent to the topology and traffic model. To specify traffic patterns, the traffic model employs four parameters - L_c, L_i, L_x, and L_f - in the range $[0, 1]$. L_c is the percentage of CPU-to-CPU traffic; L_i is the percentage of I/O-to-CPU traffic that targets local CPUs; L_x is the percentage of CPU-to-I/O traffic that targets local I/Os; L_f is the percentage of the *remote* traffic confined within the fractahedral branch where it originated. I/O to I/O traffic is not modeled; therefore, all traffic generated by I/O devices targets CPU nodes. On the basis of these parameters, the traffic can be partitioned into eight distinct types as shown in Table 1-a where *branch* refers to fractahedral branches. Three basic traffic patterns were used in this work: *Common Subtree, Uniform, and Transpose*; their parameter settings are presented in Table 1-b. Table 1-c shows the actual percentages of remote and local traffic for the different traffic models.

Traffic Type	Probability
CPU-CPU within a branch	$L_c * L_f$
CPU-CPU among branches	$L_c * (1 - L_f)$
CPU to local I/O devices	$(1 - L_c) * L_x$
CPU to remote I/O devices in same branch	$(1 - L_c) * (1 - L_x) * L_f$
CPU to remote I/O devices in different branch	$(1 - L_c) * (1 - L_x) * (1 - L_f)$
IO to local CPUs	L_i
IO to remote CPUs in same branch	$(1 - L_i) * L_f$
IO to remote CPUs in different branch	$(1 - L_i) * (1 - L_f)$

(a)

	Common Subtree	Uniform	Transpose
L_c	0.9	0.13	0.13
L_x	1.0	0.25	0.00
L_i	1.0	0.25	0.00
L_f	1.0	0.25	0.00

(b)

	C.Sub.	Unif.	Trans.
within fract.branch *(%)*	91.6	34.7	13
among fract.branches *(%)*	8.4	65.3	87

(c)

Table 1. Fractahedron Traffic Types, Parameter Settings, and Traffic Distribution (with respect to fractahedral branches)

4 Analytical Model for the 128-CPU ServerNet SAN Topologies

The model estimates the total number of messages that enter a particular router port regardless of their destination. The analytical model relies on the following facts and assumptions:

- it ignores the fact that the incoming messages at one port usually target different output ports and, therefore, experience different amounts of contention (however the simulation model and the results presented in Figures 5-7 correctly take this contention into account);
- messages are routed through a *unique* minimal path (for details see Section 2), which is therefore known in advance; and vice versa, the sources of traffic through any particular link are predetermined;
- request and response generation rates are used inter-

changeably since the generation of responses by end nodes is triggered by the arrival of requests, i.e. their numbers are equivalent.

The equations in the model express the AMIR per input port regardless of the fact that the incoming stream of messages is, in actuality, a conglomerate of several streams targeting different output ports and, hence, each stream is subject to different conditions.

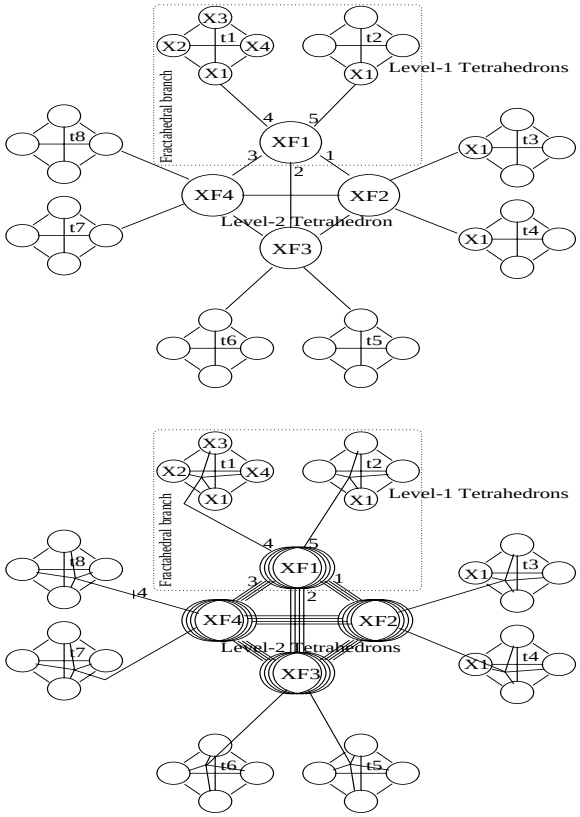

Figure 1. 128-CPU Thin (above) and Fat (below) Fractahedron Topology

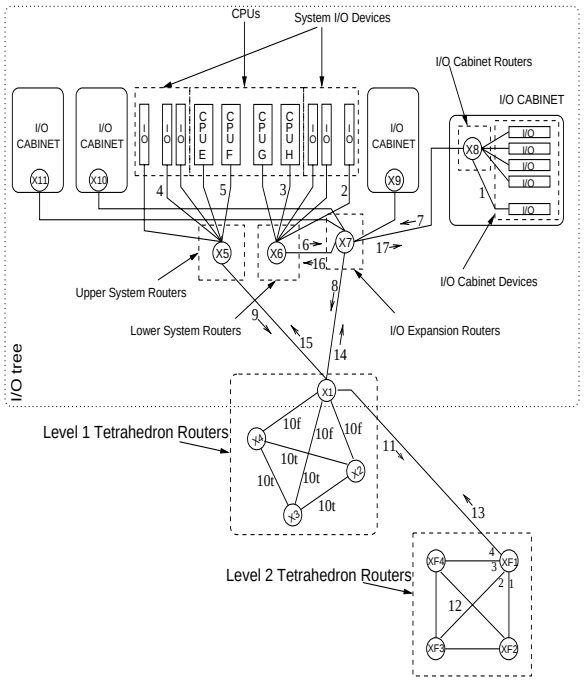

Figure 2. Device Categories in I/O Tree of 128-CPU Thin or Fat Fractahedron Topology

4.1 Determining the AMIR

The following equations represent a sample for calculating the AMIR of the XF1 router ports in the Thin Fractahedron ServerNet Topology in Figures 1 and 2. We use the following notation: • $AMIR_{R.P}$ - denotes the $AMIR$ of port P at router R e.g., $AMIR_{XF1.1}$ is the $AMIR$ of port 1 at router $XF1$;
• RQ - denotes number of requests generated;
• RS - denotes number of responses generated;
• $S_{G1}toD_{G2}\ F$ - denotes the number of messages of flow F (RQ or RS) which are: 1. generated by source devices of type S located in a topology substructure $G1$; 2. targeting destination devices of type D located in a topology substructure $G2$;
e.g., $CPU_{t3,4}toCPU_{\neg t1,2}\ RQ$ denotes the number of $requests$ generated by CPU devices located in the 16-CPU Level-1 tetrahedron substructures $t3$ and $t4$, and targeting CPU devices *not* located in the 16-CPU Level-1 tetrahedron substructures $t1$ and $t2$.

$$
\begin{aligned}
AMIR_{XF1.1} = \\
&= CPU_{t3,4}toCPU_{t1,2}\ RQ + CPU_{t3,4}toI/O_{t1,2}\ RQ + \\
&\quad + I/O_{t3,4}toCPU_{t1,2}\ RQ + CPU_{t3,4}toCPU_{t1,2}\ RS + \\
&\quad + CPU_{t3,4}toI/O_{t1,2}\ RS + I/O_{t3,4}toCPU_{t1,2}\ RS = \\
&= CPU_{t3,4}toCPU_{t1,2}\ RQ + CPU_{t3,4}toI/O_{t1,2}\ RQ + \\
&\quad + I/O_{t3,4}toCPU_{t1,2}\ RQ + CPU_{t1,2}toCPU_{t3,4}\ RQ + \\
&\quad + I/O_{t1,2}toCPU_{t3,4}\ RQ + CPU_{t1,2}toI/O_{t3,4}\ RQ = \\
&= \frac{1}{3} \times 2 \times 16 \times (1 - L_f) \times L_c \times \lambda_{CPU} + \\
&\quad + \frac{1}{3} \times 2 \times 16 \times (1 - L_f) \times (1 - L_x) \times (1 - L_c) \times \lambda_{CPU} + \\
&\quad + \frac{1}{3} \times 2 \times 104 \times (1 - L_f) \times (1 - L_i) \times \lambda_{I/O} + \\
&\quad + \frac{1}{3} \times 2 \times 16 \times (1 - L_f) \times L_c \times \lambda_{CPU} + \\
&\quad + \frac{1}{3} \times 2 \times 104 \times (1 - L_f) \times (1 - L_i) \times \lambda_{I/O} + \\
&\quad + \frac{1}{3} \times 2 \times 16 \times (1 - L_f) \times (1 - L_x) \times (1 - L_c) \times \lambda_{CPU}
\end{aligned}
$$

$$AMIR_{XF1.2} \quad = \quad AMIR_{XF1.1}$$

$$AMIR_{XF1.3} = AMIR_{XF1.1}$$

The reasoning behind the coefficients used in the above equation is as follows:

• L_c, L_x, L_i, L_f - see Table 1-a;

• $\frac{1}{3}$ - each pair of Level-1 Tetrahedron substructures (linked to a Level-2 Tetrahedron router) generates a flow of messages that traverse the Level-2 Tetrahedron (and router XF1 in particular); each message may target, with equal probability, one of the other $three$ such pairs of Level-1 Tetrahedron substructures, therefore $\frac{1}{3}$ of the messages will traverse this router;

• 2 - as pointed out above, the messages are generated by a pair of Level-1 Tetrahedron substructures;

• 16, 104 - each of the two Level-1 Tetrahedron substructures mentioned above contains 16 CPU and 104 I/O devices.

Similar calculations were performed for the Fat Fractahedron topology, as well as for each router and type (category) of link. The AMIR results for link categories (see Figure 2) under three different traffic patterns (Common Subtree, Uniform, Transpose) with $\lambda_{CPU} = \lambda_{I/O} = 60$ (which was chosen to be above the saturation point of the network for all three traffic models) are presented in the form of graphs in Figure 3. The figure shows the AMIR of each link category.

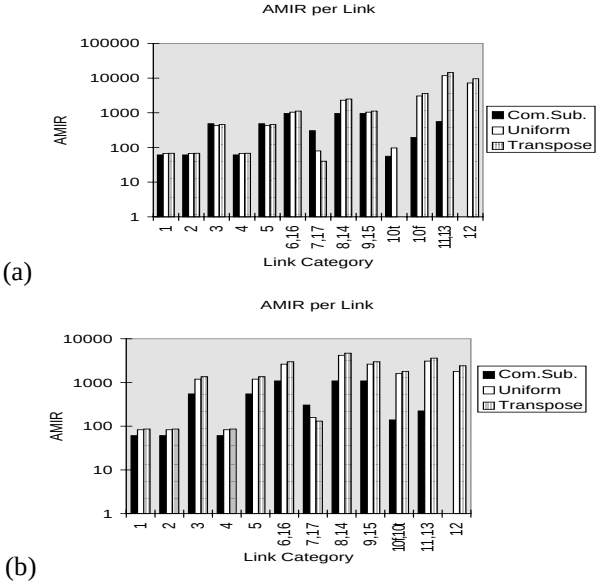

Figure 3. AMIR for Link Categories of (a) Thin and (b) Fat Fractahedron

Next we proceeded to apply the calculations to model individual routers. The results produced by the model are shown in the graphs of Figure 4. The AMIR values are *normalized* by using the smallest value obtained (from each traffic pattern and router) as a unit and scaling the other values proportionally. The graph for Router X1 in Figure 4 shows the normalized AMIR measures obtained for the three traffic patterns. The Round Robin arbitration policy would assign equivalent increment values (hence portions of bandwidth) to all links. Such a policy would provide fair arbitration only for the Common Subtree traffic pattern. For the Uniform and Transpose traffic patterns this policy will not provide increased bandwidth proportional to the activity of the category 11 links which will cause them to become congested and to degrade the performance of the entire network.

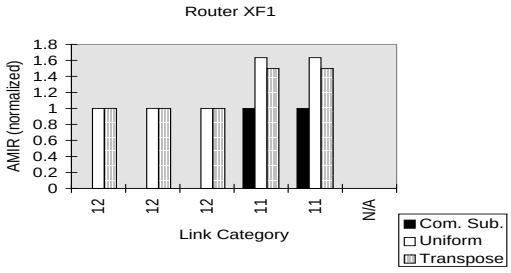

Figure 4. AMIR of 128-CPU Thin and Fat Fractahedron XF1 Router Ports

5 Performance and Scalability Analysis of the 128-CPU Topologies

Using the AMIR measure produced by the analytical model we proceeded to assign the port increment values (weights) proportionately to the calculated port AMIR. We obtained simulation data using the new sets of weights and compared it with the data obtained for Round Robin weights. The simulation results provided proof for the validity of our approach in determining the optimal values of the router port increment values on the basis of the port AMIR.

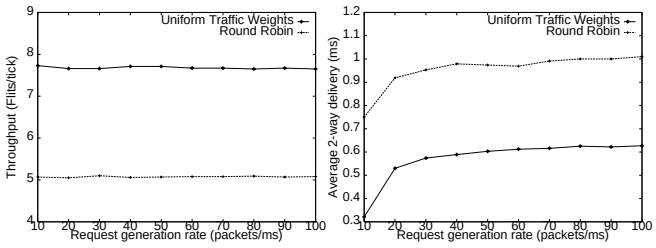

Figure 5. Simulation Results for 128-CPU Thin Fractahedron with Uniform Traffic

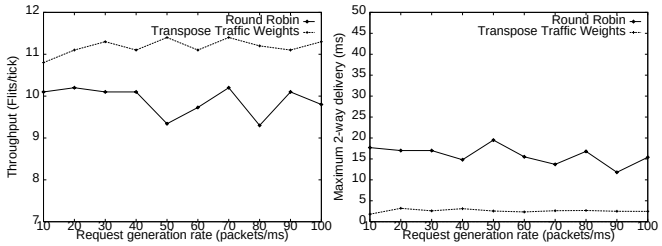

Figure 6. Simulation Results for 128-CPU Fat Fractahedron with Transpose Traffic

The simulation data obtained with the new weight assignments showed a marked improvement of the network's performance characteristics. For example, the 128-CPU Thin Fractahedron with Uniform traffic and weights proportional to the AMIR exhibits more than 50% higher throughput (Figure 5) and scalability (Figure 7-a) and more than 40% lower average delivery time in comparison with Round Robin arbitration (Figure 5).

The 128-CPU Fat Fractahedron with Transpose traffic shows a 10-20% higher throughput (Figure 6) and scalability (Figure 7-b). The most noticeable effect of the new arbitration policy is the order of magnitude decrease of the maximum two-way delivery time (Figure 6). This is an important result since it will allow the timeout counters to be configured with much lower values and thus the error detection time will be reduced.

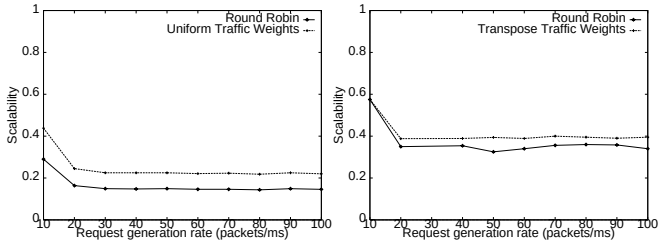

Figure 7. Scalability of the 128-CPU Fractahedron Topologies: (a) Thin with Uniform Traffic; (b) Fat with Transpose Traffic

6 Conclusions

An analytical method for determining the increment values (weights) of the ServerNet router ports was extended and applied to the 128-CPU Thin and Fat Fractahedron Topologies. Simulation data was obtained to evaluate the effect of the new weight assignments. The data was used also to assess the improved scalability of the two topologies.

The performance analysis of the ServerNet SAN with the *increment values of router ports assigned proportionally to their AMIR* showed that the weights (arbitration policy) have a strong influence on the performance of the network in general and its scalability in particular. This implies that a simple round robin scheme is a poor arbitration policy for these multistage networks. Our study demonstrates that it is important to have routing hardware which supports a flexible arbitration policy. By adjusting the policy for typical communication patterns according to the method described in this paper, overall network performance can be improved significantly.

References

[1] D. R. Avresky, V. Shurbanov, R. Horst, et al. Performance modeling of ServerNet™ topologies. *Accepted J. of Supercomputing*, 1997.

[2] A. Grama, et al. Isoefficiency: Measuring the scalability of parallel computer systems. *IEEE Parallel and Distributed Technology*, 1(3):12–21, August 1993.

[3] R. W. Horst. Tnet: A reliable system area network. *IEEE Micro*, pages 37–45, February 1995.

[4] R. Horst. ServerNet™ deadlock avoidance and fractahedral topologies. In *Proc. of IEEE Int. Par. Proc. Symp.*, pp.274–280, Honolulu, HI, USA, April 1996.

[5] Kai Hwang. *Advanced Comp. Arch.: Parallelism, Scalability, Programmability*. McGraw-Hill, 1993.

[6] R. Jain. *The Art of Computer Systems Performance Analysis*. John Wiley & Sons, Inc., 1991.

[7] V. Kumar and A. Gupta. Analyzing scalability of parallel algorithms and architectures. *J. Par. & Dist. Comp.*, 22(3):379–391, September 1994.

[8] D. Nussabum and A. Agrawal. Scalability of parallel machines. *Comm. of ACM*, 34(3):57–61, 1991.

[9] M. Schwartz. *Telecommunication Networks*. Addison-Wesley Publishing Co., 1987.

[10] V. Shurbanov, D. R. Avresky, R. Horst, et al. A scalability study of ServerNet™ topologies. *ISCA 10th Int. Conf. on Par. & Dist. Comp. Sys.*, October 1997.

[11] A. Sivasubramaniam, et al. A simulation-based scalability study of parallel systems. *J. Par. and Dist. Comp.*, 22(3):411–426, September 1994.

[12] X. Sun and D. Rover. Scalability of parallel algorithm-machine combinations. *IEEE Trans. on Parallel and Distributed Systems*, 5(6):599–613, June 1994.