MINSimulate – A MULTISTAGE INTERCONNECTION NETWORK SIMULATOR

DIETMAR TUTSCH and MARCUS BRENNER
Technische Universität Berlin
Real–Time Systems and Robotics
D–10587 Berlin, Germany
{*dietmart,mbrenner*}*@cs.tu-berlin.de*

Abstract: Multistage interconnection networks are frequently proposed as connections in multiprocessor systems or network switches. In this paper, a new tool for stochastic simulation of such networks is presented. Simple crossbars can be simulated as well as multistage interconnection networks that are arranged in multiple layers.
Keywords: network simulator, multicasting, multistage interconnection networks, crossbars, performance

1 Introduction

Multistage interconnection networks (MINs) with the banyan property are proposed to connect a large number of processors to establish a multiprocessor system [1]. They are also used as interconnection networks in Gigabit Ethernet [2] and ATM switches [3]. Such systems require high performance of the network. MINs were first introduced for circuit switching networks. To increase the performance of a MIN, buffered MINs were established as packet switching networks. For instance, Dias and Jump [4] inserted a buffer at each input of the switching elements (SE). Patel [5] defined delta networks. Delta networks are a subset of banyan networks (MINs with just one path between a given input and output). It is additionally required that packets can use the same routing tag to reach a certain network output independently of the input at which they enter the network.

Many variations of delta networks were introduced. Most of them result in MINs that lose the unique path property (and therefore the delta property) in order to reduce blocking. Clos [6] presented a MIN consisting of three stages and non-quadratic SEs. Turnaround MINs [7] are established by bidirectional links between the SEs. Network inputs and SE inputs operate also as outputs. Dilated banyan networks [8] arise by multiplying the links between the SEs: the link bandwidth is enhanced. Replicated banyan networks [8] originate from multiplying the whole banyan network. Multilayer multistage interconnection networks (MLMINs) are introduced to apply especially to multicast traffic. Those networks are established similarly to replicated MINs but a new replication starts at every network stage [9]. The network results in a growing number of layers from sources to destinations. Many other kinds of MINs are known. A detailed description can be found for instance in [3].

In this paper, a simulation tool for performance evaluation of MINs is presented. The tool is designed to investigate MINs with the delta property or with multiple layers. Various design parameters can be examined concerning network performance in terms of throughput and delay. Network traffic and resource scheduling is modelled by stochastic simulation.

The paper is organized as follows. The architecture of MINs is described in Section 2. Section 3 applies to the simulator features and shows how the simulator operates. Some examples of results are given in Section 4. Section 5 summarizes and gives conclusions.

2 Architecture of MIN

Various architectures of multistage interconnection networks exist. This section presents those architectures that can be modeled by the new simulator (called *MINSimulate*).

2.1 MIN with Banyan Property

Multistage interconnection networks with the banyan property are networks where a unique path from an input to an output exists. Such MINs of size $N \times N$ consist of $c \times c$ switching elements with $n = \log_c N$ stages. An 8×8 MIN consisting of 2×2 SEs is represented by Figure 1.

To achieve synchronously operating switches, the network is internally clocked. In each stage k ($0 \leq k \leq n-1$) of non-shared buffer networks, there is a FIFO buffer of size $m_{max}(k)$ in front of each switch input. The packets are routed by store and forward routing or cut-through switching from a stage to its succeeding one by backpressure mechanism.

Networks consisting of shared buffers are established by replacing the c FIFO input buffers of size $m_{max}(k)$ of a $c \times c$ switch with one common buffer of size $c \cdot m_{max}(k)$ [10]. This shared buffer is organized as follows: Each switch input reserves sufficient buffer space to store at least one packet in order to avoid the isolation of inputs (see below). The remaining buffer space of $c \cdot m_{max}(k) - c$ packets is available to all inputs. Each input forms a FIFO input queue of packets. If an input receives a new packet from the previous stage that has to be stored, the input allocates buffer space of the commonly used buffer part if available. If there is no further buffer available the packet is blocked at the previous stage.

An input with a queue of more than one packet deallocates buffer space if it sends a packet to the next network stage. This space is returned to the pool of the commonly available buffer space.

Guaranteeing at least one buffer space to each input avoids that an input without any buffer cannot participate in the switch routing process because it is not able to receive a packet that has to be forwarded. For instance, let us assume that one of the inputs (hot spot input) receives much more packets than the other ones. This input would allocate up to all of the buffers. Packets of the previous stage that are directed to the other inputs would be blocked at the previous stage even if their final destination is different from the first packet queued at the hot spot input. Only the hot spot input would contribute to the switch traffic and all other inputs would remain idle.

Multicasting is performed by copying the packets within the $c \times c$ switches. In ATM context, this scheme is called cell replication while routing (CRWR). Figure 1 shows such a scenario for an 8×8 MIN consisting of 2×2 SEs. A packet is received by Input 3 and destined to Output 5 and Output 7. The packet enters the network and is not copied until it reaches the middle stage. Then, two copies of the packet proceed on their way through the remaining stages.

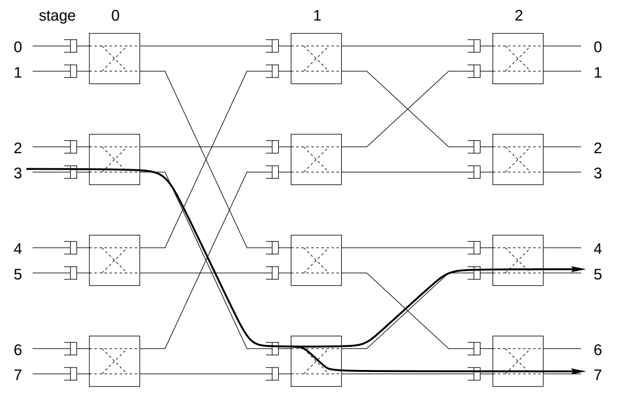


Figure 1: Multicast while routing

Packet replication before routing in the above example would copy the packet and send it twice into the network. Therefore, packet replication while routing reduces the amount of packets in the first stages.

Comparing the packet density in the stages in case of replication while routing shows that the greater the stage number, the higher is the amount of packets. In other words: there are much more packets in the last stages due to replication than in the first stages. The only exception is if the traffic pattern results in such a destination distribution that packet replication has to take place at the first stage. Then, the amount of packets is equal in all stages. But such a distribution is very unlikely, in general.

To set up multistage interconnection networks that are appropriate for multicasting, the previously mentioned different traffic densities of the stages must be considered. MLMINs, which are described later in this section, belong to this kind of networks. Their roots are in replicated MINs.

2.2 Replicated MIN

Replicated MINs enlarge regular multistage interconnection networks by replicating them L times. The resulting MINs are arranged in L layers. Corresponding input ports are connected as well as corresponding output ports. Figure 2 shows the architecture of an 8×8 replicated MIN consisting of two layers in a three-dimensional view. Such a concept was introduced by Kruskal and Snir [8]. Packets are received by the inputs of the network and distributed to the layers. Layers may be chosen at random, by round

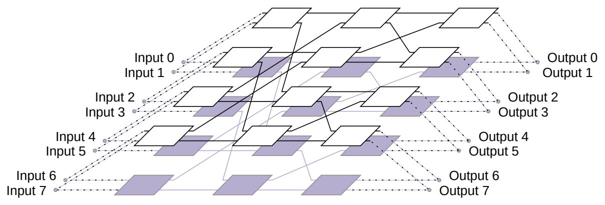


Figure 2: Replicated multistage interconnection network ($L = 2$, 3D view)

robin, dependent on layer loads, or any other scheduling algorithm. The distribution is performed by a 1:L demultiplexer.

At each network output, an L:1 multiplexer collects the packets from the corresponding layer outputs and forwards them to the network output. Two different output schemes are distinguished: single acceptance (SA) and multiple acceptance (MA). Single acceptance means that just one packet is accepted by the network output per clock cycle. If there are packets in more than one corresponding layer output, one of them is chosen. All others are blocked at the last stage of their layer. The multiplexer decides according to its scheduling algorithm which packet to choose.

Multiple acceptance means that more than one packet may be accepted by the network output per clock cycle. Either all packets are accepted or just an upper limit R. If an upper limit is given, R packets are chosen to be forwarded to the network output and all others are blocked at the last stage of their layer. As a result, single acceptance is a special case of multiple acceptance with $R = 1$.

In contrast to regular multistage interconnection networks, replicated MINs may cause out of order packet sequences. Sending packets belonging to the same connection to the same layer avoids destruction of packet order.

2.3 Multilayer MIN

Multilayer multistage interconnection networks (MLMINs) consider the multicast traffic characteristics. As mentioned above, the amount of packets increases from stage to stage due to packet replication. Thus, more switching power is needed in the last stages compared to the first stages of a network.

To supply the network with the required switching power, MLMIN structure increases the number of layers in each stage. The factor with which the number of layers is increased is called growth factor G_F ($G_F \in \mathbb{N}\backslash\{0\}$). Figure 3 shows an 8×8 MLMIN (3 stages) with growth factor $G_F = 2$ in lateral view. That means the number of

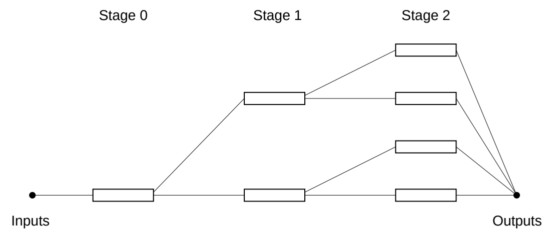


Figure 3: Multilayer multistage interconnection network ($G_F = 2$)

layers is doubled each stage and each switching element has twice as much outputs as inputs. Consider for instance that 2×2 SEs are used. Such an architecture ensures that even in case of two broadcast packets at the inputs all packets can be sent to the outputs (if there is buffer space available at the succeeding stage). On the other hand, unnecessary layer replications in the first stages are avoided.

Choosing $G_F = c$ ensures that no internal blocking occurs in an SE, even if all SE inputs broadcast their packets to all SE outputs. Nevertheless, blocking may still occur at the network output depending on R.

A drawback of MLMIN architecture arises from the exponentially growing number of layers for each further stage. The more network inputs are established, the more stages and the more layers result. To limit the number of layers and therefore the amount of hardware, two options are available: starting the replication in a more rear stage and/or stopping further layer replication if a given number of layers is reached.

The first option is demonstrated in Figure 4 in lateral view. The example presents an 8×8 MLMIN in which replication does not start before Stage 2 (last stage) with $G_F = 2$. A 3D view is given in Figure 5. The stage num-

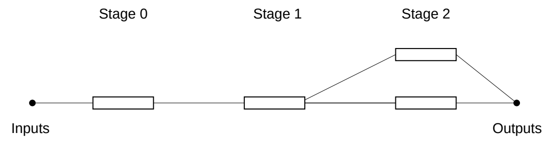


Figure 4: MLMIN in which replication starts at Stage 2 (lateral view)

ber in which replication starts is defined by G_S ($G_S \in \mathbb{N}$). Figures 4 and 5 introduce a MLMIN with $G_S = 2$. Of course, moving the start of layer replications some stages

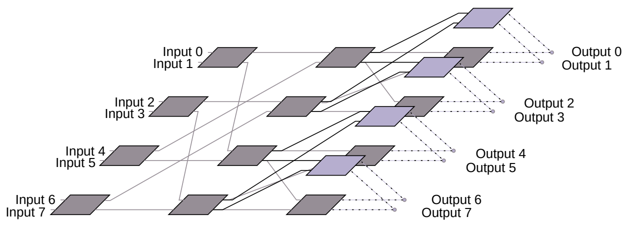


Figure 5: MLMIN in which replication starts at Stage 2 (3D view)

to the rear not just reduces the number of layers. It also reduces the network performance due to less SEs and therefore less paths through the network.

Stopping further layer replication if a given number G_L of layers is reached also reduces the network complexity ($G_L \in \mathbb{N}\backslash\{0\}$). It prevents exponential growth beyond resonable limits in case of large networks. Figure 6 shows such an MLMIN with limited number of layers in lateral view. 3D view is presented in Figure 7. The number of

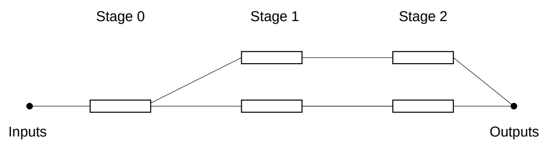


Figure 6: MLMIN with limited number of layers (lateral view)

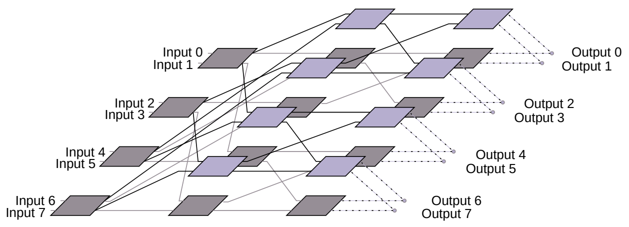


Figure 7: MLMIN with limited number of layers (3D view)

layers of this 8×8 MLMIN is limited to an upper number of $G_L = 2$. Layers are replicated with a growth factor of $G_F = 2$. As in the previous option, the reduced amount of SEs decreases network performance as well.

Both presented options can be combined to reduce network complexity further. Such a network is determined by parameters G_S (start of replication), G_F (growth factor), and G_L (layer limit). For instance, Figure 7 shows an MLMIN with $G_S = 1$, $G_F = 2$, and $G_L = 2$.

Regular MINs and replicated MINs can be considered as special cases of MLMINs. Regular MINs are equivalent to MLMINs with $G_F = 1$. In this case, G_S and G_L have no effect. Replicated MINs are equivalent to MLMINs with $G_S = 0$, $G_F = L$, and $G_L = L$.

3 Simulator *MINSimulate*

The new simulator presented in this paper is called *MINSimulate*. It is designed to model MINs with the banyan property, replicated MINs, and MLMINs, as well as simple crossbar switches.

3.1 Features

Stochastic simulation is performed by C++ code. According to the network parameters given by the user, the network is first established. It is represented as a directed graph starting at the sources (network inputs) and ending at the destinations (network outputs). The simulator is packet based. Packets are generated at the sources. Each packet is provided with a tag determining its destination. Due to multicasting this tag is modeled by a vector of N binary elements, each representing a network output. The elements of the desired outputs are set to "true". If the packet arrives at a $c \times c$ switch, the tag is divided into c subtags of equal size. Each subtag belongs to one switch output, the first (lower indices) subtag to the first output, etc. If a subtag contains at least one "true" value a copy of the packet is sent to the corresponding output containing the subtag as the new tag.

To keep the amount of allocated memory as small as possible, just a representation of the packets, referred to as containers, is routed along the network paths. These containers are replaced by the actual packets at the network outputs allowing evaluations. Figure 8 gives a short sketch of the simulation model. So called `ContainerMultiputs` (CM) receive the containers and store them in the queues. At the first network stage, `FirstContainerMultiputs` (FCM) additionally perform the replacement of the packets by containers. So called `ContainerOutputs` (CO) send the containers to the next network stage. At the last stage, `LastContainerOutputs` (LCO) additionally replace the containers by the corresponding packets. Each operation of a switch is controlled by its `Crossbar Manager`. The clocks perform the sequencing of the parallel actions due to computer simulation.

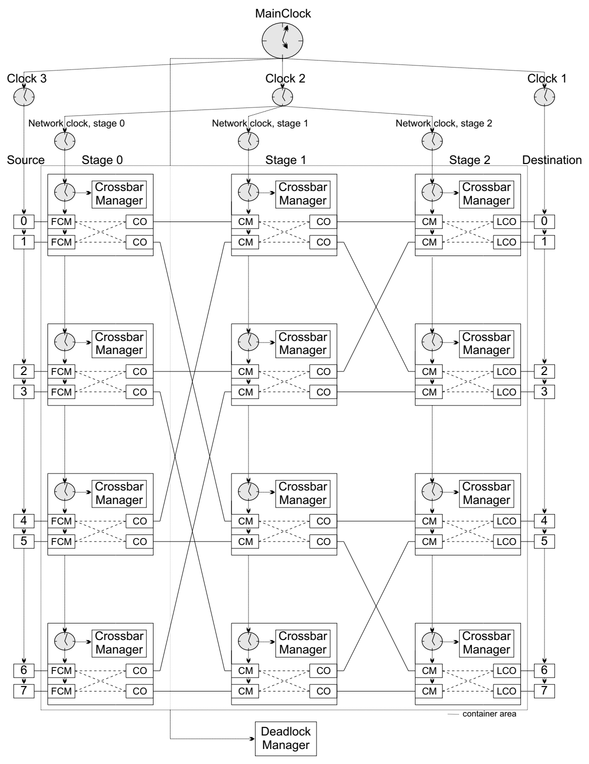
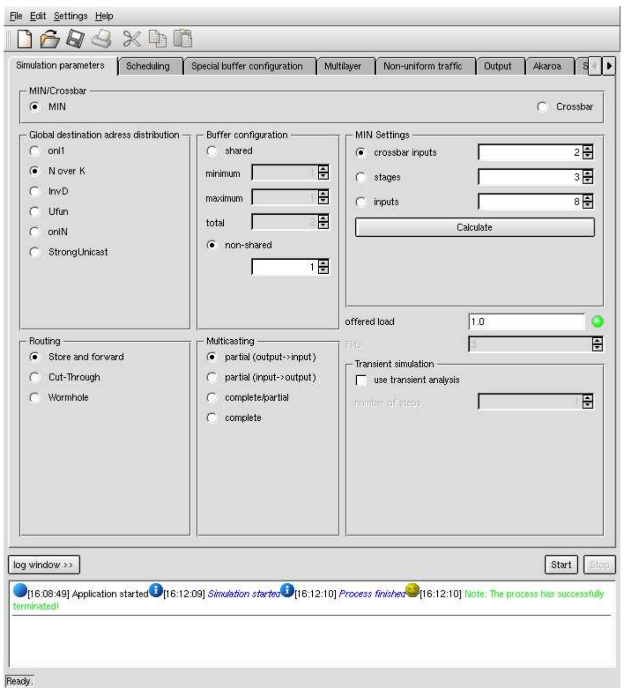


Figure 8: Sketch of the simulation model

Confidence level and relative error of simulation results is observed by the toolkit *Akaroa*. The simulation is stopped when those termination criteria are met. *Akaroa* is developed at the University of Canterbury, New Zealand [11].

3.2 Graphical User Interface

The network to be evaluated is determined by the user via a graphical user interface (GUI). Figure 9 shows the main tab to settle simulation parameters. A short sketch



MIN/Crossbar: [✓ MIN] Crossbar

Global destination adress distribution: onl1 [✓ N over K] InvD Ufun onlN StrongUnicast

Buffer configuration: shared (minimum 1, maximum 1, total 2) [✓ non-shared] 1

MIN Settings: [✓ crossbar inputs] 2, stages 3, inputs 8

offered load 1.0

Flits 3

Routing: [✓ Store and forward] Cut-Through Wormhole

Multicasting: [✓ partial (output->input)] partial (input->output) complete/partial complete

Transient simulation: ☐ use transient analysis, number of steps 1

[16:08:49] Application started [16:12:09] *Simulation started* [16:12:10] *Process finished* [16:12:10] Note: The process has successfully terminated!

Figure 9: Main tab of *MINSimulate*

of the parameters and their available settings is given below. They are described in more detail in [12].

First, it can be chosen whether to simulate single crossbars or MINs. If crossbars are chosen the number of inputs can be defined. If MINs are chosen two of the three parameters according to equation $n = \log_c N$ must be set by the user. The third one is determined by the GUI. Input buffers can be chosen as shared ones (with a minimum size and maximum size for each crossbar input, as well as the overall buffer size of a crossbar) or as non-shared ones (with the size per crossbar input). Tab `Special Buffer Configuration` allows individual buffer settings for each stage.

The global address destination distribution of packets entering the network can also be varied. The most important patterns are `onl1` (only unicast; all targets are with equal probability a packet's destination), `N over K` (multicast; all target combinations are with equal probability a packet's destination), `Ufun` (multicast with many unicasts and many broadcasts), and `onlN` (only broadcast). If single sources are desired to produce deviating address distributions, tab `Non-uniform Traffic` helps.

When choosing the routing algorithm the following packet switching schemes are available: store and forward routing, cut-through switching, or wormhole routing. Multicast in case of wormhole routing usually suffers from deadlocks. The wormhole routing algorithm of *MINSimulate* avoids deadlocks by grouping appropriate parts of the network [13]. Wormhole routing requires dividing the packets into flits. The number of flits per packet is also a parameter for the simulation.

The kind of multicast can be set to complete multicast, partial multicast, or a two phase version of both. A further parameter represents the offered load to each network input. The last parameter in the main tab determines whether to observe measures transiently instead of observing the steady state. In case of transient simulation, the number of clock cycles to simulate can be fixed.

Tab `Multilayer` allows to configure MLMINs as presented in Section 2.3. Choosing `constant number of layers` refers to replicated MINs.

Instead of simulating a particular network configuration, a parameter can be varied to deal with parameter dependent results. In tab `Simulation Series`, the parameter to vary is chosen. A start value, end value, and step size determines the variation. If desired, step size can be changed once in the parameter interval.

Performance measures are chosen via tab `Output`. Most important ones are throughput, delay times, and queue lengths. A histogram of delay times within an interval is also available. Deadlines can be added to packets and packets that exceed their deadline are then removed. In such scenario packet loss results as a measure.

Akaroa parameters to determine confidence level and relative error of results are set in tab `Akaroa`.

4 Example

To present an example of the results obtainable using *MINSimulate* the following evaluation will be performed: Given the task to design a multistage interconnection network of size $N = 64$, how will the network's performance be affected by the choice of the switching element size c?

Simulations are run for MINs composed of 2×2 and 4×4 SEs. This example's simulations were performed using an accuracy of 0.02 and a confidence level of 98%.

The size of the buffer in front of each SE (non-shared buffering) is set to $m_{\max}(k) = 2$ for all stages k. Routing is performed according to a store and forward scheme.

The MINs to be evaluated are being offered multicast traffic governed by multicast traffic pattern *N over K*, that is, all possible combinations of target ports have an equal probability of being a packet's destination. Packets are offered to the network at a constant (time-independent) rate.

Figure 10 shows the output throughput for both network configurations. There is little to no difference in throughput performance between MINs based on 2×2 and 4×4 SEs. When the offered load rises above approx. 0.02 (average number of offered packets per input and clock cycle) the network begins to saturate: Due to the multiplication of packets inside the SEs, queues build up and by the aforementioned backpressure mechanism the actual input throughput is limited to about 0.02 as well. I. e., in saturation there are less packets accepted by the MIN than are being generated and ready to be sent.

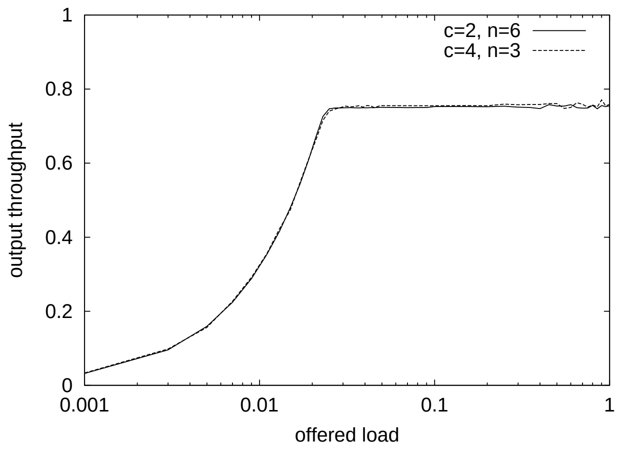


Figure 10: Throughput at the network's outputs

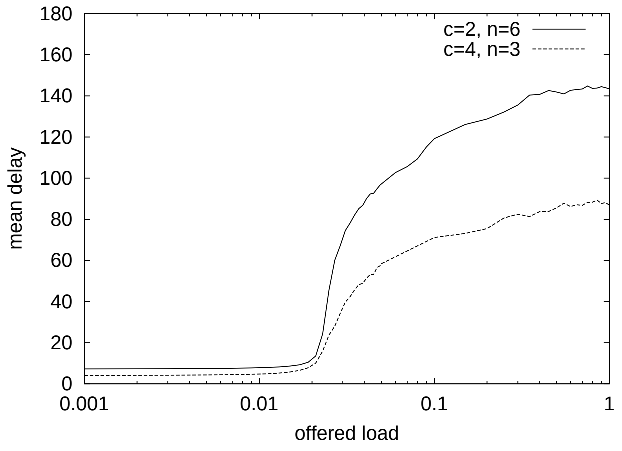


Figure 11: Mean packet delay times

When looking at the average time it takes a packet to reach its destination, differences between the two network configurations become apparent (Figure 11). In case of store and forward routing the minimum amount of time required to traverse the MIN is equal to the number of stages in the network (3 and 6, respectively). Of course, this requires that the packet is not being delayed even a single time. With rising offered load the number of conflicts between packets increases and buffers in front of the switching elements fill up. Thus, for an offered load greater than approx. 0.02 the mean delay time rises considerably. In case of an unlimited buffer space $m_{\max}(k)$ the delay times would grow beyond limit, similar to an unstable queuing system.

The most apparent difference between the two MIN configurations is the lower mean delay of the 4×4-based one in saturation (offered load > 0.02): 80–90 opposed to 120–140 clock cycles for the 2×2-configuration. Because of the lower number of stages there occur less conflicts between packets, resulting in an overall lower delay. In addition, the slope of the curve in the area of 0.02 to 0.07 is less steep for the MIN consisting of only 3 stages.

To conclude this example, if achieving low packet delay times was an issue in designing a MIN, one would opt for 4×4 switching elements (if the additional cost would be acceptable), although throughput would not benefit.

5 Conclusion

This paper presents the tool *MINSimulate* for stochastic simulation of multistage interconnection networks (MINs). Due to the tool's GUI support the wide variety of simulation parameters is easily accessible to the user.

MINSimulate is able to simulate simple (i. e. fully meshed) crossbars, multistage interconnection networks (MINs) with the delta property, and MINs that are arranged in multiple layers. Within each network architecture, simulations can be performed using a wide variety of input parameters such as offered load, multicast traffic pattern, routing scheme or (internal) buffer configuration. The simulations yield performance measures such as throughput, mean delay, delay time distributions or mean buffer queue lengths in individual network stages. During simulation, all data is evaluated according to pre-set levels of confidence and accuracy using the statistical library *Akaroa*[11].

Transient network behavior can also be evaluated with *MINSimulate*, this is useful for studies of the fine grain time-dependent performance, especially when observing traffic that changes with time.

The presented tool allows for evaluating various network configurations under different traffic conditions. Therefore, one can easily establish knowledge whether a particular network design suits a task at hand.

Currently, *MINSimulate* is extended in the way that also non-delta networks can be simulated. As a first step, Clos networks and turnaround MINs are incorporated.

6 Acknowledgements

We would like to thank Daniel Benecke, Matthias Hendler, Rainer Holl-Biniasz, and Arvid Walter for implementing and maintaining *MINSimulate*. Without their work and ideas, this project would never have been as successful.

References

[1] Gheith A. Abandah and Edward S. Davidson. Modeling the communication performance of the IBM SP2. In *Proceedings of the 10th International Parallel Processing Symposium (IPPS'96); Hawaii*. IEEE Computer Society Press, 1996.

[2] Toshio Soumiya, Koji Nakamichi, Satoshi Kakuma, Takashi Hatano, and Akira Hakata. The large capacity ATM backbone switch "FETEX-150 ESP". *Computer Networks*, 31(6):603–615, 1999.

[3] Ra'ed Y. Awdeh and H. T. Mouftah. Survey of ATM switch architectures. *Computer Networks and ISDN Systems*, 27:1567–1613, 1995.

[4] Daniel M. Dias and J. Robert Jump. Analysis and simulation of buffered delta networks. *IEEE Transactions on Computers*, C–30(4):273–282, April 1981.

[5] Janak H. Patel. Performance of processor–memory interconnections for multiprocessors. *IEEE Transactions on Computers*, C–30(10):771–780, October 1981.

[6] C. Clos. A study of nonblocking switching network. *Bell System Technology Journal*, 32:406–424, March 1953.

[7] Hong Xu, Yadong Gui, and Lionel M. Ni. Optimal software multicast in wormhole-routed multistage networks. *IEEE Transactions on Parallel and Distributed Systems*, 8(6):597–606, June 1997.

[8] Clyde P. Kruskal and Marc Snir. The performance of multistage interconnection networks for multiprocessors. *IEEE Transactions on Computers*, C–32(12):1091–1098, 1983.

[9] Dietmar Tutsch and Günter Hommel. Multilayer multistage interconnection networks. In *Proceedings of 2003 Design, Analysis, and Simulation of Distributed Systems (DASD 2003); Orlando*, pages 155–162. SCS, April 2003.

[10] Dietmar Tutsch, Matthias Hendler, and Günter Hommel. Multicast performance of multistage interconnection networks with shared buffering. In *Proceedings of the IEEE International Conference on Networking (ICN 2001); Colmar*, pages 478–487. IEEE, July 2001.

[11] Krzysztof Pawlikowski, Victor W. C. Yau, and Don McNickle. Distributed stochastic discrete-event simulation in parallel time streams. In *Proceedings of the 1994 Winter Simulation Conference; Lake Buena Vista*, pages 723–730, December 1994.

[12] http://uscream.cs.tu-berlin.de/minsimulate/.

[13] V. Varavithya and P. Mohapatra. Asynchronous tree-based multicasting in wormhole-switched multistage interconnection networks. *IEEE Transactions on Parallel and Distributed Systems*, pages 1159–1178, November 1999.