Routing of guaranteed throughput traffic in a network-on-chip

Nikolay Kavaldjiev, Gerard J. M. Smit, Pascal T. Wolkotte, Pierre G. Jansen

Department of EEMCS, University of Twente, the Netherlands
{n.k.kavaldjiev, g.j.m.smit, p.t.wolkotte, p.g.jansen}@utwente.nl

ABSTRACT

This paper examines the possibilities of providing throughput guarantees in a network-on-chip by appropriate traffic routing. A source routing function is used to find routes with specified throughput for the data streams in a streaming multiprocessor system-on-chip. The influence of the routing algorithm, network topology and communication locality on the routing performance are studied.

The results show that our method for providing throughput guarantees to streaming traffic is feasible. The communication locality has the strongest influence on the routing performance while the routing algorithm has weakest influence. Therefore, the mapping algorithm is of greater importance for the system performance than the routing algorithm and it is profitable to use a more complex mapping algorithm that preserves the communication locality together with a simple routing algorithm.

I. INTRODUCTION

Source routing [1] is a widely used method for deterministic and oblivious routing in multiprocessor interconnection networks. Its popularity is due to its speed, simplicity and scalability. With source routing all routing decisions for a packet are made by a central routing entity and stored in the source node. Each node contains a table with routes, at least one route per destination. The routes are computed and loaded in the nodes tables during application initialization.

Source-table routing is used in the IBM 9076 SP1 multicomputer. The routing algorithm used for computing the routes that fill the tables in the nodes is discussed in [2]. Its main objective is a uniform distribution of the routes in the network such that congestions are avoided and the network channels are equally loaded.

In this paper we also study a routing algorithm that will be used for computing the routes to be loaded in the source-routing tables. It is based on the same algorithm used in the SP1 but in our approach together with routes computation the routing is used to provide throughput guarantees for data streams.

The routing algorithm we consider will be used at run-time and therefore has to be simple and fast. It will work in a dynamic, adaptable multiprocessor system-on-chip (MPSoC) as a part of its OS. The hardware architecture consists of a number of processing elements (PEs) communicating through a network-on-chip (NoC). At run-time the OS, running on one of the PEs, starts and stops applications in response to its interaction with the user and the environment. Several applications can run in the system at a time. An application can run on more than one PE. When the OS starts an application it allocates PEs where the application processes will run and routes the communication channels between the processes. The calculated routes are then loaded in the route tables of the corresponding PEs. Routing is done for the communication channels of the application only and consequently the source tables contain routes to a restricted number of PEs.

The MPSoC we consider will be used for mobile multimedia devices where most of the applications are streaming and energy is an issue. Streaming applications usually have simple linear application graphs – a data stream flows through a pipe of PEs. From our experience we can assume then the data streams account for more than 90% of the system traffic. In this work we consider only the streaming data traffic. The traffic patterns used in the SP1 are more typical for general parallel algorithms.

Most of the streaming applications running on the system are real-time applications and require guaranteed network services. In our system a routing function is used to find routes with a certain throughput guarantees for the data streams of the applications. Knowing the routes of all data streams that are already in the system, the routing algorithm can predict what the performance of a newly routed channel will be. Thus, the routing algorithm can search for a route that provides requested guarantees. This is the way in which throughput guarantees are provided to data streams in our network.

This paper focuses on the routing of guaranteed throughput (GT) traffic only. Our objectives are i) to show that the method for providing throughput guarantees is feasible ii) to find the limits for the provided guarantees. Three main factors influencing the routing performance are taken into account in this work: routing algorithm, network topology and traffic locality. Experiments are made with two routing algorithms, two network topologies, and three traffic locality models.

The paper is organized as follows. Section II presents the

approaches for providing service guarantees used in other on-chip networks. Section III introduces our network and explains how it provides throughput guarantees. In Section IV the traffic models used in the simulations are presented. Section V presents the routing function used during the simulation. Section VI presents and discusses the simulation results.

II. Related work

In the Ethereal network-on-chip [7] guaranteed services are based on the time-division multiple-access (TDMA) scheme. The best effort service is based on wormhole routing that utilizes the time slot not used by the guaranteed services.

In the Nostrum network-on-chip [9] guaranteed services are based on temporally disjoint networks (a technique related to TDMA) and looped container virtual circuits (a time slots reservation scheme). Since the approaches used in Nostrum and Ethereal are based on time slots, they require a global notion of time in the network and in the system.

Another approach for providing guaranteed services in network-on-chip is introducing traffic priorities. This approach is used in [8], where in a virtual channel network the virtual channels have static priorities. The virtual channels on a network channel have different priorities and provide different service guarantees. But these service guarantees are traffic dependent. The throughput of the lower priority channels depends on the bandwidth utilized by the higher priority channels. The relation between throughput and latency of a data stream is not direct but depends on the size of the packets passing the higher priority virtual channels, so it is hard to give guarantees.

III. Network and throughput guarantes

The network interconnecting the PEs in the system is a packet-switching virtual channel network [3]. In a virtual channel network on top of every network channel there are several virtual channels (VCs). In our network the number of VCs per network channel is 4. The network uses table based source routing – every PE has a table with routes to PEs to which it sends data. The routes are computed by a routing function (part of the OS), centrally for the system, at run-time. The new routes are computed and routing tables are updated every time a new application is started, however, this is done only for the data streams of the application being started.

The routing is done at the level of virtual channels – the routing function decides on which VCs a data stream is routed. The VCs are seen as a network resource which is distributed between the data streams by the routing function. The VCs which are used for GT traffic are not shared and used only by one data stream, while the VCs used for best effort (BE) traffic can be shared between many communication channels.

Since GT data streams are routed on separate VCs there are no resource dependencies between them. Therefore, a deadlock involving GT data streams cannot occur and the later discussed routing function does not deal with deadlock. Deadlock must be taken into account when routing BE traffic since BE communication channels may share a VC, but this paper discusses routing of GT traffic only.

The network can give throughput guarantees to a limited number of data streams. The throughput guarantees are based on throughput-fairness - bandwidth of a network channel is equally shared between its VCs (only those which are used). Knowing the number of occupied VCs on a network channel a lower bound on their throughput can be given. If a physical channel of bandwidth b, has v occupied VCs, then each of these VCs is guaranteed throughput greater than or equal to:

$$\Theta_{\min} = \frac{b}{v}$$

Since a network route is constructed as a concatenation of VCs, the throughput of the route is determined by the VC with the lowest throughput guarantees. Our network has 4 VCs per physical channel. Therefore, the throughput that is guaranteed to a route can be b, $b/2$, $b/3$ or $b/4$.

The number of occupied VCs on a network channel is known and under the control of the routing function which has a global overview of the traffic in the network. The routing function can provide guarantees to data streams by finding routes that provide the requested throughput.

This way for providing guarantees is suitable for streaming architectures where the application graphs are simple and where more than 90% of the GT traffic is due to streams flowing a same route for the duration of the application.

The same method can be used for providing guarantees in a circuit-switching network like that presented in [4]. Each network channel in this network is split in four sub-channels, called lanes. Switching can be done at different granularity level, from single lane to whole channel. Thus, paths with different throughput guarantees can be constructed. If the same routing function is used, then instead of reserving VCs it will reserve lanes.

The network topologies considered in our work are mesh and torus - the most popular topologies for NoC. We assume that the PEs are placed on a regular grid in a plane. Experiments are made with two network topologies of fixed size: 10-ary 2-mesh and 10-ary 2-cube (torus) [1]. The system consists of 100 PEs placed in an array of size 10-by-10. Energy results are obtained also for a folded torus - an alternative topological placement for the torus.

IV. TRAFFIC MODELS

Most of the streaming applications have a linear application graph (a pipeline), for example base-band processing, audio/video (de)coding, and image processing. We consider a liner graph communication pattern as representative for the GT traffic in a real stream-processing system. If the last node in a linear graph is connected to the first node a ring graph is constructed. A large ring topology graph can be seen as a concatenation of smaller linear application graphs. Thus, in the simulations a ring communication pattern is used.

An application graph with a ring topology containing 100 nodes is mapped on the architecture. Mapping is about deciding where on the multiprocessor architecture the application processes will run; for each node in the application graph a PE is chosen. In our experiment we arranged that the number of processes in the graph equals the number of PE in the architecture and each process runs on a separate PE. We refer to the result of this procedure, the fixed assignment of processes to PEs, as a *mapping*.

The mapping characteristic that is the most important for the traffic routing is the *communication locality* – the network distance between communicating processes. To model the effect of communication locality we use three different mapping strategies, each producing mappings with different locality characteristics.

The first mapping strategy models traffic without communication locality. It chooses PEs randomly. Therefore, no communication locality should be expected and the chance that there is a communication channel between a pair of PEs is equal for all the pairs in the system. Nevertheless the communication distance distribution after such a mapping is not uniform but depends on the network topology (the distance distribution over all pairs of nodes). For a 10-ary 2-mesh the distribution has the shape shown on the left graph in *Figure 1* (attained by simulation). The mean of the distance distribution equals the mean distance for uniform traffic in the same topology – *nk/3* for *k*-ary *n*-mesh and *nk/4* for *k*-ary *n*-cube, both for even *k* [1].

The other two mapping strategies are locality-preserving and model traffic with communication locality. They both use the same algorithm for choosing PEs. Every next process in the application graph is mapped on a PE chosen uniformly among those PEs which are at distance $\leq d$ from the previous PE (where the previous process is mapped). Here the distance d is a parameter of the algorithm. If there is no free PE within that distance, then a PE is chosen uniformly among all free PEs. The distances are calculated assuming a mesh topology.

The two locality-preserving mapping strategies differ only in the parameter d. For the first strategy d takes value 4 and for the second it takes value 1. These numbers are chosen to represent approximate average and best case of communication locality. The resulting distance distributions for the mesh topology are given in the second and the third graph in *Figure 1*. We see that the probability for a communication distance being $\leq d$ is higher than being $>d$. The distance distribution follows the shape of mapping with no locality, but the distances $\leq d$ and these $>d$ are weighted differently. As shown in the figure, for d=4 about 97% of the data streams have distance $\leq d$.

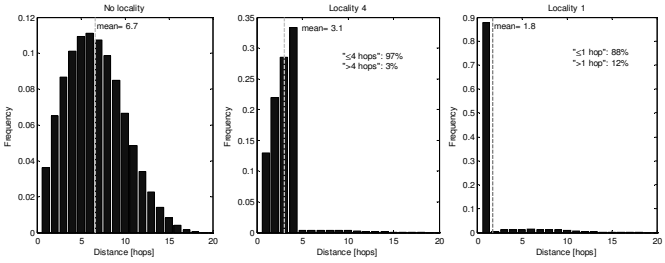

Figure 1 Distance distribution in a mesh network assuming ring communication pattern

Further in the text we refer to the traffic models corresponding to the three mapping strategies as *No locality*, *Locality 4* and *Locality 1* or for brevity only as *Lx*, *L4* and *L1*.

After all the nodes in the application graph are mapped on PEs a routing function is run to find routes for the data stream in the graph.

V. ROUTING FUNCTION

The task of the routing function is to find a route in the network for a data stream. The function takes as input a description of a data stream (source node, destination node and requested throughput) and returns as a result a description of a network route (an ordered list $\{pv_1, pv_2, \ldots, pv_n\}$ where pv_i defines the next network channel and its VC to be taken). If the function cannot find a route that satisfies the throughput request, an empty list is returned. Source and destination nodes can be any two nodes in the network. Since the network can provide throughput guarantees only for b, b/2, b/3 and b/4, these are the throughput guarantees that can be requested from the routing function (to identify them values 1, 2, 3 and 4 are used).

In order to provide throughput guarantees the routing function needs to know the current state of the network. The network state is stored in a data structure where every VC is represented by one bit and one integer indicating respectively whether the VC is occupied and what throughput is guaranteed. (This information is sufficient when only GT traffic is routed. When GT and BE traffic are routed together then one more bit is needed per VC to identify whether the VC is used for BE or GT. When a VC is used for BE the integer stores the number of communication channels routed

through it.)

After a data stream has been routed the routing function updates the network state. When the data stream is not needed anymore (e.g. the application is stopped) its route is cleared from the network state.

The routing function is based on an algorithm for shortest path search in a graph. The algorithm runs over a sub-graph of the network topological graph. The sub-graph is derived by deleting all network channels that cannot be used for routing the data stream either because they cannot guarantee the requested throughput or either because the guarantees of the data streams already routed there will be violated if another one joins them.

This work studies two routing functions based on two graph algorithms: *Breadth-first* search and *Dijkstra's* single-source shortest-path algorithm [5]. The algorithms are used for solving the single-pair shortest-path problem – finding the shortest path in a graph between given source and destination vertices.

Breadth-first search (BFS) is one of the simplest algorithms for searching in the graph. It works on a non-weighted graph and finds shortest paths in terms of number of edges. The runtime of the algorithm is O(V+E), where V and E are the number of vertices and edges in the graph. The same algorithm is used for routing in [2].

Dijkstra's algorithm (DA) works on a weighted, directed graph, where all edge weights are nonnegative. It finds shortest paths in terms of minimal weight sum. In our case the weighs assigned to the edges are the number of occupied VCs on a channel plus one. Thus, if no VC is occupied the weight of a channel is one and if all four VCs are occupied the weight is 5 (one is added to avoid zero weight and thus provide that the algorithm always prefers shorter paths). When searching for the shortest path Dijkstra's algorithm will prefer to use less occupied channels and so will spread the traffic more uniformly in the network. The runtime of the algorithm is $O(V^2)$.

Both algorithms need also a description of the network graph. To store the graph of the considered topologies we use an adjacency list. Detailed description of the BFS and Dijkstra's algorithm can be found in [5].

We have implemented two routing functions, one using the simpler Breath-first algorithm and another using the more complicated Dijkstra's algorithm. Our goal is to see whether by balancing the load over the network channels the more complex Dijkstra's algorithm will bring any practical advantages. The evaluation criteria are the amount of traffic that the routing function can fit in the network, the average path length and respectively the average communication energy cost. The routing algorithms are examined considering two other factors that influence the routing performance – traffic locality and network topology. In this paper we refer to the routing functions with the abbreviations of their algorithms – BFS and DA.

VI. SIMULATION RESULTS

The conducted simulation experiments are as follows. For each topology, traffic model and throughput requirement 1000 mappings of a ring application graph are generated; each mapping is random but according the same traffic model. All data streams in the graph are set to have equal throughput requirements (this is not a very realistic assumption but it will provide worst case performance results). For each mapping both routing algorithms are run separately to find routes that guarantee the requested throughput for all the data streams of the graph. Routing for a given mapping is considered successful if all data streams are routed. If the routing algorithm cannot find a path for even one data stream, the routing fails. The successfully routed mappings are counted. Results are collected only for the successfully routed mappings.

A. Throughput

The two routing algorithms are tested for the highest requested throughput for which they still manage to route all data streams. In other words, what maximal throughput they can achieve for a ring communication pattern.

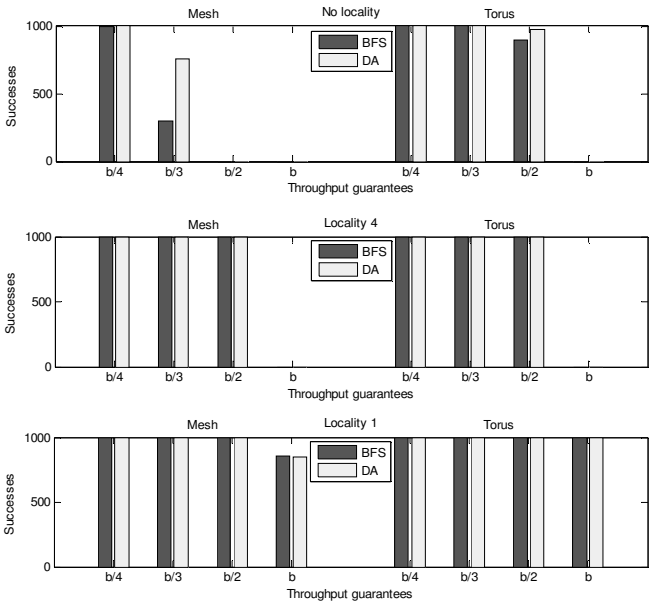

Figure 2 Number of successfully routed mappings

The results are presented in *Figure 2*, where the three graphs give the number of successfully routed mappings for the three traffic locality models. Each graph shows the results for mesh and torus topology.

The traffic locality has strongest influence on the routing performance. The more local the traffic is the higher

throughput that can be guaranteed. For example, in the mesh topology the local traffic (Locality 1) allows the routing algorithms to fit traffic with throughput requirements *b/2* in the network, while for traffic with no locality (No locality) the routing always succeeds only for lower throughput *b/4* requested. Of course these results are dependent on the network size, especially for traffic with no locality.

The dependence of the routing performance on the network topology is significant only for traffic with no locality. For such traffic the routing in a torus outperforms the routing in a mesh, which is due to the double bisection of the torus. When the traffic shows strong locality the higher bisection is not so advantageous and the routing performs almost the same in a torus and a mesh.

The two routing algorithms perform differently only when the network is near saturation (like for b/3 in the mesh with no locality). In this case DA outperforms BFS as it more often manages to fit the traffic in the network. But still, for a network near saturation it does not succeed for all the mappings, which make the use of the network under such conditions undesirable.

B. *Extra distance*

Both routing algorithms try to route the data streams using minimal distance paths. But this is not always possible since some network channels along the minimal distance paths might already be occupied. In such cases the routing takes a path which is not minimal. For a particular mapping *extra distance* is defined as the difference between the sum of the real distances and the sum of the minimal distances for all data streams. The real distances are the result of the routing, while the minimal distances are idealistic (assuming there is no other traffic in the network).

The extra distance is used to compare the performance of the routing algorithms. A better routing algorithm will manage to route the traffic using shorter paths and therefore with less extra distance. More extra distance leads to more network traffic and more energy spent on communication.

Figure 3 presents the results for the average extra distance for a mapping. The three graphs in the figure correspond to the three traffic locality models. Each graph shows the results for two network topologies, mesh and torus.

It can be seen that in most of the cases the extra distance is negligible compared to the total minimal distance, especially when the requested throughput is low and when the traffic shows locality. In most of the cases the extra distance is less than 10 hops in total for all 100 communication channels in the application graph, which is small. Compared to the results of *Figure 2*, we see that the extra distance becomes significant only when the routing algorithm works in a network near saturation. The results for the routing algorithms differ significantly only in such conditions as DA gives a smaller extra distance than BFS.

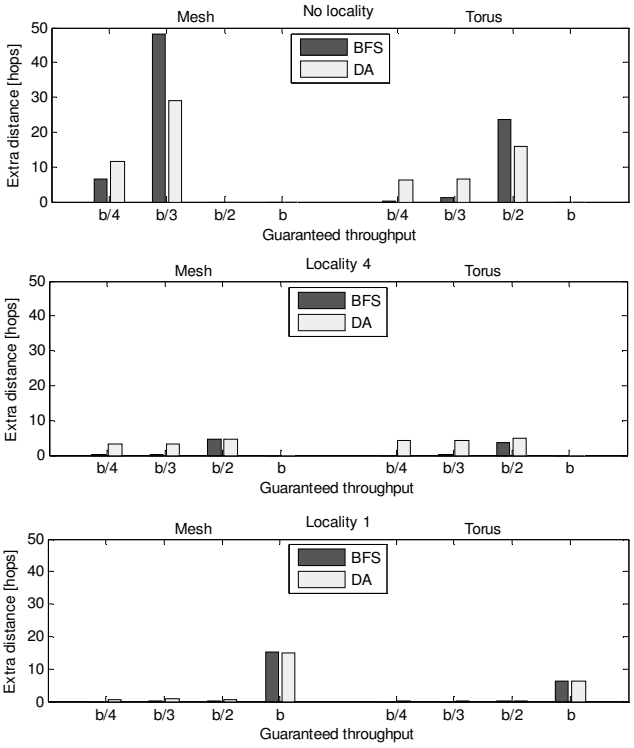

Figure 3 Average extra distance

C. *VC utilisation*

The main difference between the two routing algorithms is the way they spread traffic in the network. While BFS checks only whether a network channel can provide the requested throughput, DA also checks how heavy a channel is loaded and among all possibilities selects a channel with lower load. Thus, it can be expected that DA spreads the traffic more uniformly in the network keeping the channels equally occupied. It should be expected that spreading the traffic will cost extra distance compared to BFS, but as *Figure 3* shows this extra distance is usually negligible.

The effect of traffic spreading can be seen in *Figure 4*. The figure shows the average channel utilization in a torus network. The traffic has no locality and the throughput guarantee is *b/4*. The two graphs correspond to the two routing algorithms; they give the average number of channels in the network with utilized 0, 1, 2, 3 or 4 virtual channels. The number of channels is normalized.

Comparing the two graphs it can be seen that DA reduces the number of network channels with a high number of utilized VCs. More network channels are utilized but with fewer VCs occupied. This effect is better observed when the network is heavy loaded but still not in saturation. When the network load is reduced the channels become weakly utilized and both routing algorithms distribute the traffic in the same

way. When the network is loaded near saturation all channels are heavy loaded and optimizations are not possible.

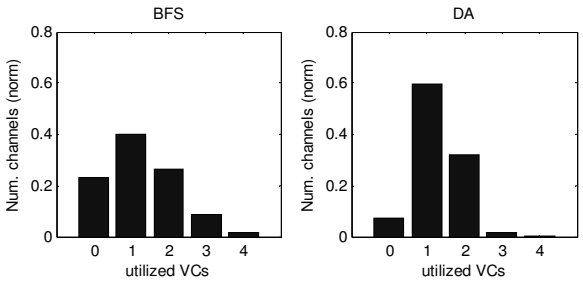

Figure 4 Utilization of the network channels in a torus network, traffic with no locality and throughput guarantees b/4

D. Energy cost

The average data stream distance obtained by the simulations can be used for calculating the average communication energy cost, assuming that all data streams have a same throughput. We use the energy models of a packet-switching network and a circuit-switching network presented in [6]. The models estimate the energy cost for data transportation in the network (for 0.13um technology). The energy cost in [pJ/bit] is:

$$E_{ps} = E_R\left(N_{hop}+1\right)+\left(0.39+0.12l_{wire}\right)N_{hop}$$

l_{wire} is the network channel length in *mm*. In the formula N_{hop} is the network distance in number of hops. E_R stands for energy for traversing a router; for packet-switching and circuit-switching network takes values $E_{R\_PS} = 0.98$, $E_{R\_CS} = 0.37$. The second term estimates the energy for traversing a network channel. The model captures only the dynamic energy cost to transport a bit. An extra amount of energy is consumed due to leakage (static energy) and an offset in the frequency dependent power consumption.

The energy cost is estimated for three topologies – mesh, torus, and folded torus – all of size 10-by-10 nodes. For the mesh we assume a channel length of 1.5 mm corresponding to PE size of 2.25 mm^2. In the torus the wrap around channels are ten times longer, 15 mm. In the folded torus the network channels in the middle have to cross two PEs and thus are 3 mm long.

Figure 5 presents the results for the communication energy cost of a system with a packet-switching network. For each topology three traffic models are considered: No locality (Lx), Locality 4 (L4) and Locality 1 (L1) (see *Section IV*).

The graph shows that the communication locality has the strongest influence on the communication energy cost. By introducing communication locality the average communication energy cost can be reduced by 50% to 70% for the different topologies.

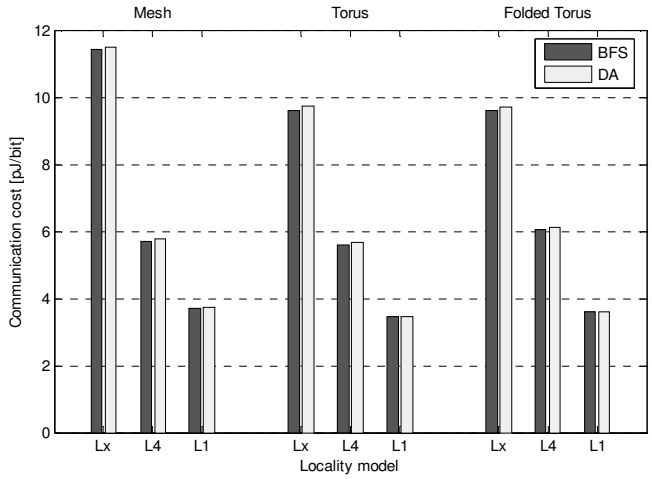

Figure 5 Average energy cost for communication of a system with packet-switching network

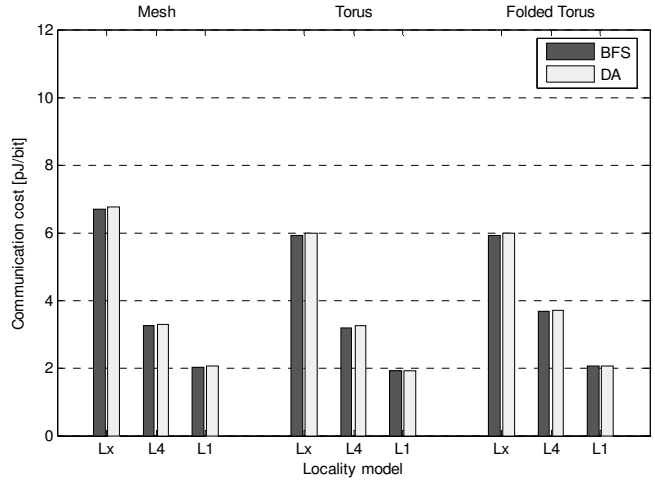

Figure 6 Average energy cost for communication of a system with circuit-switching network

Dependency on the network topology is significant only for traffic with no communication locality. For such traffic a torus can reduce the communication energy compared to mesh topology due to its smaller network diameter. The distant data streams, which in a mesh topology have to cross the middle of the mesh, in a torus network take the wrap-around channels as a shortcut and save energy by avoiding routers passing. Since the communication locality reduces the communication distances and the consequently number of data streams that take the wrap-around channels, the torus does not bring significant energy reduction for local traffic.

Changing the network topology from torus to folded torus does not influence significantly the communication energy cost. Because the logical hop distribution for both topologies is identical but the length of the network channels in folded torus is doubled it should be expected the energy cost for

folded torus to be higher. But because the energy cost per wire length is relatively small compared to a router energy cost the increase is insignificant.

The influence of the routing algorithms on the energy cost is also small. The cost is slightly higher when using DA, especially when the traffic does not show locality and this is mainly due to the small extra distance this routing algorithm adds.

Figure 6 presents the communication energy cost results for a system with circuit switching network. As can be seen the main difference between the results for the circuit-switching and the packet-switching network is that the energy cost in the circuit-switching network is proportionally reduced. This is due to the smaller energy cost for passing a router in a circuit-switching network – a clear advantage of the circuit switching network.

In this work experiments are made only with a system of size10-by-10 PEs. This size is large enough for demonstrating the effect of the traffic locality while still feasible in the present silicon technology. In a system of smaller size the network will have a smaller diameter; the network distances will be naturally shorter which leaves less room for locality improvements. In a larger size system preserving the communication locality will play a more important role since system operation will be impossible without it. But it can be also expected that with the increase of the system size the possibilities for exploiting the communication locality will get more and more limited. Preserving communication locality is done at the level of a single application, while the inter-application communications are not so local. With the increase of the system size the number of applications and therefore the amount of the non-local inter-application communications will increase. The network conditions in such a system can be improved by a network topology of higher dimension.

VII. CONCLUSION

In this paper we examined with traffic routing in a streaming multiprocessor architecture. The goal is to study the possibilities for providing throughput guarantees in a network-on-chip by appropriate traffic routing. The influence of three major factors is taken into account – routing algorithm, network topology and traffic locality.

The results show that for stream-dominated traffic and assuming traffic locality it is feasible for a routing algorithm to provide throughput guarantees. The traffic locality has strongest influence on the routing performance. It allows for higher communication throughput and reduction of the energy cost of 50% to 70%. When no traffic locality is presented a torus network topology improves the network conditions and the routing performance compared with the mesh topology. Increase in traffic locality reduces the advantages of the torus. While different in complexity, the two routing algorithms, based respectively on the simple Breath-first search algorithm and the more complex Dijkstra's algorithm, do not show significant difference in performance. The Dijkstra's algorithm outperforms the Breath-first algorithm only when the network is near saturation and because this is not a desirable operating condition for the system we cannot take advantage of that.

We can conclude that when routing is used to provide throughput guarantees to streaming traffic in a multiprocessor system-on-chip it is reasonable to use a simple routing algorithm but more complex and clever mapping functions that keep the traffic local.

REFERENCES

[1] William J. Dally and Brian Towles, "Principles and Practices of Interconnection Networks" Morgan Kaufmann, 2004

[2] Bülent Abali, Cevdet Aykanat, "Routing Algorithms for IBM SP1", *Proceedings of the First International Workshop on Parallel Computer Routing and Communication*, 1994, pp. 161–175.

[3] Nikolay Kavaldjiev, Gerard J. M. Smit, Pierre G. Jansen, "A Virtual Channel Router for On-chip Networks", *Proceedings of IEEE International SOC Conference,* September 2004, pp. 289-293.

[4] Pascal T. Wolkotte, Gerard J.M. Smit, Gerard K. Rauwerda, L.T. Smit, "An Energy-Efficient Reconfigurable Circuit Switched Network-on-Chip", *Proceedings of the 12th Reconfigurable Architecture Workshop RAW*, 2005, p. 155a

[5] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein, "Introduction to Algorithms", The MIT Press, 2nd edition, 2001

[6] Pascal T. Wolkotte, Gerard J.M. Smit, Nikolay Kavaldjiev, Jens E. Becker, Jurgen Becker., "Energy Model of Networks-on-Chip and a Bus", *International Symposium on System-on-Chip*, 2005

[7] K. Goossens, J. van Meerbergen, A. Peeters, and P. Wielage, "Networks on Silicon: Combining Best-Effort And Guaranteed Services", *DATE 2002*, pp. 423-428.

[8] Tomaz Felicijan, Steve Furber,"An Asynchronous On-Chip Network Router with Quality-of-Service (QoS) Support", *Proceedings of IEEE International SOC Conference*, Sep. 2004, pp.274-277.

[9] Mikael Millberg, Erland Nilsson, Rikard Thid, and Axel Jantsch, "Guaranteed bandwidth using looped containers in temporally disjoint networks within the Nostrum network on chip", *DATE 2004*, p. 20890