Hindawi Publishing Corporation
Journal of Electrical and Computer Engineering
Volume 2012, Article ID 728191, 14 pages
doi:10.1155/2012/728191

Research Article

Status Data and Communication Aspects in Dynamically Clustered Network-on-Chip Monitoring

Ville Rantala,[1,2] Pasi Liljeberg,[2] and Juha Plosila[2]

[1] *Turku Centre for Computer Science (TUCS), Joukahaisenkatu 3-5 B, 20520 Turku, Finland*
[2] *Department of Information Technology, University of Turku, 20014 Turku, Finland*

Correspondence should be addressed to Ville Rantala, vttran@utu.fi

Received 1 July 2011; Revised 28 October 2011; Accepted 1 November 2011

Academic Editor: Sao-Jie Chen

Monitoring and diagnostic systems are required in modern Network-on-Chip implementations to assure high performance and reliability. A dynamically clustered NoC monitoring structure for traffic and fault monitoring is presented. It is a distributed monitoring approach which does not require any centralized control. Essential issues concerning status data diffusion, processing, and format are simulated and analyzed. The monitor communication and placement are also discussed. The results show that the presented monitoring structure can be used to improve the performance of an NoC. Even a small adjustment of parameters, for example, considering monitoring data format or monitoring placing, can have significant influence to the overall performance of the NoC. The analysis shows that the monitoring system should be carefully designed in terms of data diffusion and routing and monitoring algorithms to obtain the potential performance improvement.

1. Introduction

Network-on-Chip (NoC) [1] based systems can be complex structures of tens or hundreds of processors, IP cores (Intellectual Property), and memory modules. These systems require versatile monitoring systems to handle the functionality and maintain their performance. While interconnect starts to increasingly dominate the overall performance, the monitoring systems become even more significant part of modern NoC systems. An advantage of the NoC paradigm is its scalability [2]. A fully scalable NoC architecture should have a scalable monitoring system which can be easily tailored to different NoC implementations and whose performance does not degrade when the size of the system increases.

NoC monitoring systems are typically designed for two purposes: system diagnostics and traffic management. The former aims to improve the reliability and performance of the computational parts while the latter concentrates to the same issues in the communication resources. The traffic management should take into account the status of the network resources including their load as well as their possible faultiness.

A technology-independent framework of the dynamically clustered monitoring structure for NoC is presented and its features are discussed in this paper. A SystemC-based NoC simulation model is also presented. The dynamically clustered monitoring structure is fully scalable monitoring system which is primarily aimed for traffic management purposes. This paper is organized as follows. NoC traffic management and different monitoring structures are discussed in Section 2. Section 3 gives a brief description about related works and writers' contributions. The SystemC-based NoC simulation model is presented in Section 4. The dynamically clustered monitoring structure is presented in Section 5, and its features are discussed and analyzed in Sections 6, 7, 8, and 9. Potential modifications to the monitoring structure are presented in Section 10. General discussion, future works, and conclusions are gathered to Section 11.

2. Monitoring in NoC

Traffic management is implemented into NoC to maintain network performance and functionality in the case of faults

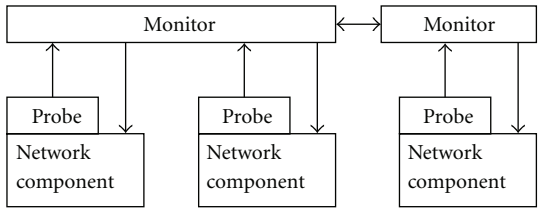


Figure 1: Network components.

and under high traffic load. Typically there is a monitoring system to collect traffic information from the network and an adaptive routing algorithm which adapts its operation when the conditions in the network change.

Two types of information are required in traffic management: traffic status in the network and locations of faults in the network. Traffic status can be observed from different network components: router activity, router FIFO occupancy, or link utilization, for instance. Fault information can cover the faultiness of different network components: routers or links, for instance. A network component is considered as faulty when it does not work as it should by its specification. The network components have to have mechanisms to detect these faults [3]. There are several methods to detect faults. For instance, faulty links can be detected using methods which are based on usage of spare resources or error control coding [4, 5].

2.1. NoC Monitoring Structures. The components of a monitoring system are monitors and probes. The probes are attached to a network components (e.g., routers, links, or network interfaces) to observe the functionality of a network component (see Figure 1). The observed data is delivered from probes to a monitor which can collect statistics or process the data to a format which can be utilized in different reconfiguration tasks. The processed monitoring data is finally delivered to the components which use it to reconfigure their operation. A monitoring structure can have dedicated resources for communication between probes and monitors, or it can share the resources of the data network. In our research we focus on shared-resource structures which require less additional resources than dedicated structures. We have also paid attention to dedicated resources for serial monitoring communication between monitors. In shared-resource structures nonintrusive operation of the monitoring system is a significant issue while in the serial monitoring communication the delays are crucial in terms of usefulness of the monitoring data.

Monitoring structure defines the number and type of monitors and probes, their placing, connections, and tasks. A centralized monitoring structure has one central monitor and several probes that observe the data and deliver it to the monitor. In centralized structure the central monitor has complete overall knowledge of the network but it causes significant amount of monitoring-related traffic in the network. A clustered monitoring structure has a few cluster monitors and several probes. The network is divided into subnetworks, clusters, each of them having a cluster monitor and several probes. The complete network knowledge can be reached using intercluster communication but most of the tasks can be executed inside a cluster. However, a clustered structure still causes a considerable amount of monitoring traffic [6].

In an NoC the data is typically transferred as packets which have a destination address. Routers forward these packets based on this address and the applied routing algorithm [7]. The NoC monitoring systems which use shared communication resources transfer the network status data using monitoring packets. When centralized or clustered monitoring structures are used, these packets have to be routed from probes to a monitor and from the monitor to the routers. Centralized control has its strengths and it is required for several tasks. However to optimize performance some of the traffic management tasks could be executed with simpler distributed, or dynamically clustered, monitoring structure to decrease the load of the centralized control system [6].

3. Related Work

NoC monitoring systems have been presented in several papers. A dedicated control network is used in a centralized operating system controlled NoC [8]. Dedicated resources are also used in [9] where dedicated embedded processors are used to monitor FIFO occupancies and transfer latencies. A monitoring system for Æthereal [10] is presented in [11]. This system monitors transactions in a network and can be configured to shared or dedicated communication resources. A congestion control system, which is based on monitoring the link utilization, is presented in [12]. All these systems include a centralized monitoring unit which collects the observed data and controls the system operation.

Clustered monitoring structures have been discussed in several papers. A monitoring system to collect error, run-time, and functional information from routers and network interfaces (NI) is presented in [13]. A traffic shaping mechanism with router monitoring is presented in [14]. In these two implementations there can be multiple central monitoring units, cluster monitors, so that the system is clustered.

These NoC monitoring structures are nonscalable or partly scalable. Our goal is to develop clustered monitoring towards yet finer granularity and better scalability. A scalable monitoring with regional congestion awareness is represented in [15]. It is aimed to balance the workload in the network based on the amount of congestion.

3.1. Contribution. Our research focuses on a scalable NoC monitoring structures where the knowledge about network conditions is spread widely enough over the network. There are two main factors taken into account while designing our NoC architecture. First, the structure should be not only aware of traffic but also aware of network faults so that network-level fault tolerance can be actively maintained during routing. Second, the structure should also be fully scalable to any size of mesh. All the probes and monitors

are identical, and they work autonomously without any centralized control. The presented ideas can be adapted to different kind of NoC topologies but, due to its popularity, we have decided to concentrate on the mesh topology.

Our dynamically clustered Network-on-Chip is previously discussed and analyzed in [6, 16]. The study in [6] includes extensive introduction and related work sections and presents the proposed dynamically clustered Network-on-Chip architecture. The architecture is simulated and analyzed on theoretical concept level including analysis of monitoring traffic overhead, status data diffusion, and cost of monitoring system. Brief performance analysis on transaction level is also presented in that paper.

Our in-house NoC simulation model has been presented and different status update intervals in the dynamically clustered NoC has been discussed and analyzed in [16].

This paper includes broader and more in-detail presentation of the in-house NoC simulation model (see Section 4). The status data diffusion analysis, originally presented in [6], is extended from theoretical concept level to transaction level using the presented NoC simulation model. In addition, issues concerning network status data format, monitoring communication protocols, and monitor mapping are studied in this paper. The NoC architecture is studied as a technology-independent framework, and therefore the analysis is mostly based on comparison of different features, not on pure analysis of absolute performance values.

4. Simulation Environment

The proposed DCM structure and its features are simulated and analyzed using a *SystemC*- and *TLM 1.0*-based NoC simulation model. The cycle-accurate NoC simulation model is designed for the analysis of different mesh-shaped NoCs. The NoC simulation model includes implementations of a router, a link, a monitor, and a general core. In real implementations the cores include processors and memories but in our NoC simulation model they operate as the senders and receivers of data following some traffic pattern. The NoC simulation model also includes structures to tie the components together and to model data packets. These packets can be considered as flits. In this paper each packet consists of a flit.

The analysis, presented in this paper, is performed using NoC with 64 cores and routers arranged to rows and columns of eight. Two-level traffic pattern has been used. The NoC simulation model is widely customizable which enables the analysis of several different design aspects.

4.1. Simulation. A simulation mechanism of the NoC simulation model is able to execute transient analysis where the cores are sending packets following to a specific traffic pattern and key figures are documented during the simulation. These figures include the numbers of sent and received packets (including the data packets and the monitoring packets, see Section 6), the transfer delay, and the number of dropped packets. The simulation durations can be customized, and it is possible to run simulation in multiple NoCs simultaneously with identical simulation parameters. In this case, the results are represented as averages from the simultaneously running NoCs. The amount of traffic during the simulation can be adjusted with the traffic pattern. There is also an option to put link faults in the network. This feature enables the fault tolerance analysis of NoC. All the network faults are modeled using only the link faults; for example, a faulty router can be illustrated with a bunch of faulty links. Faults can be put in the system randomly or manually so that modeling of larger uniform fault areas is also possible.

4.2. Router. The router model is designed for mesh networks and so it has five ports, four for traffic to and from the neighboring routers and one for traffic to and from the local core. There is a small FIFO buffer in each input port and a centralized FIFO for packets which cannot be routed at the first attempt but can be rerouted later. If the routing fails constantly, the packet is dropped and the router should notify the sender when dropping a packet. The reporting feature is not implemented at this point. The packet dropping typically happens in severe situations where routing is inhibited permanently due to permanently faulty network resources making destination unreachable. Sizes of the FIFO buffers are customizable. The router model includes several different routing algorithms. The used algorithm (see Section 5.1) can be chosen with the simulation parameters.

4.3. Monitor. The monitors are used to observe the functionality and state of the system. The monitor component in our NoC simulation model includes both a probe to collect the monitoring data as well as the monitor to process the collected data. The monitor component can be also configured to act only as a probe or a monitor. This is useful, for instance, when analyzing centralized monitoring structures [6]. Monitoring algorithms are discussed in Section 6. The monitors communicate with each other using shared or dedicated resources. The NoC simulation model includes both implementations. When shared resources are used, the monitors send packets in the data NoC. However, when dedicated resources are used, the amount of resources is limited and therefore the serial communication protocol is utilized.

4.4. Link. The model of a link in the NoC simulation model is unidirectional so that they are used in bunches of two in between two routers, one in each direction. The link is a simple component which forwards the incoming packets. A link can be set on usable or unusable state to model faults in the network.

4.5. Traffic Patterns and Fault Injection. The NoC simulation model has three traffic patterns to be used in analysis. A traffic pattern has two parameters, one defines the amount of sent data during a unit time while another defines how the destination cores are determined. The simplest pattern is a fully random traffic pattern which randomizes the packet destinations among all the cores in the network. A weighted

random traffic pattern is adjusted so that one-third of traffic is between neighboring cores, another third between neighbor's neighbors, and the last part between all the other cores in the network. This pattern roughly imitates a traffic pattern in a real NoC implementation.

The third implemented traffic pattern is a two-level pattern which includes uniform random traffic and varying hot spots each of which sends a relatively large number of packets to a single receiver during a certain time interval. A relatively small number of cores operate as hot spots simultaneously and send packets to a statically chosen receiver cores. At the same time, other cores are sending relatively smaller amount of traffic to random destinations. This two-level traffic pattern imitates real applications where most of the traffic takes place between certain cores at a time. It is aimed for even more realistic performance simulations. The simulations, presented in this paper, are done using this two-level traffic pattern.

During simulation the network links can be set faulty. In Network-on-Chip simulations the fault information can be simplified by using only the information on faulty links and representing other faulty components by marking the links around these components to be faulty. In our simulation framework the number of faults is defined by the user and the simulator places the faults randomly in the network. The simulation is executed several times with different fault patterns and the results are averaged from the original simulation results. This procedure gives overall insight of the system's operation when parts of the network are faulty.

5. Dynamically Clustered Monitoring Structure

Dynamically clustered monitoring (*DCM*) can be considered distributed monitoring because it does not require any centralized control. There is a simple monitor and a probe attached to each router in the network. The used mesh topology is illustrated in Figure 2. Centralized control is not required but the monitors exchange information autonomously with each other.

Each router has a dynamic cluster around itself from where the router receives the data it needs for traffic management. There are no fixed cluster borders as there is in traditional clustered networks and a router can belong to several different dynamic clusters. A dynamic cluster is the area around a single router of which the router has knowledge and to where the router shares its own status. Router's own status is delivered to all the routers in this cluster area. The delivery of router status is called as status data diffusion. The dynamic clusters of different routers overlap with each other. The simplest dynamic cluster includes 4 closest neighbors of a router but it can be expanded to neighbors' neighbors and so on. A system which uses DCM for traffic management could have, for instance, operating system level control for tasks that need complete knowledge of the system. When traffic management is implemented with a DCM structure, the load of the network can be optimized.

There are several issues which affect the functionality of a monitoring system. The used simulation environment

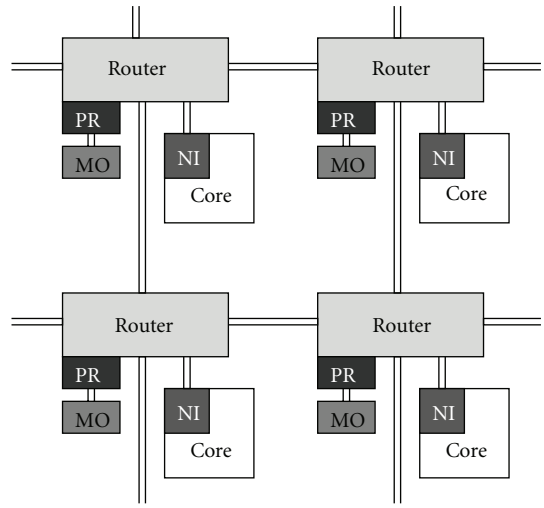


Figure 2: Network topology showing the connections between routers, networks interfaces (NI), monitors (MO), probes (PR), and cores in a part of mesh shaped Network-on-Chip.

is presented in Section 4. The monitoring communication is discussed in Section 6 while Section 7 concentrates on the diffusion and calculation of network status data. The different formats of network status data are presented and analyzed in Section 8.

5.1. Routing Algorithms in Dynamically Clustered NoCs. The NoC simulation model utilizes an adaptive routing algorithm [7]. The algorithm determines the routing direction among the eight candidates which include four main directions (north, south, east, and west) and the intermediate directions (e.g., northwest and southeast). These directions are illustrated in Figure 3. The algorithm chooses an output port to be used among the actual routing direction and its nearest neighbor directions. The decision is based on the traffic status values and the link statuses in potential directions. A packet which cannot be delivered is put back in the router's memory and rerouted. A packet lifetime is also utilized to prevent undeliverable packets from blocking the network. To prevent congestion the packets are not sent in directions where receivers' buffers are fully occupied.

We also propose an experimental routing algorithm where the destination's distances to the core in different routing directions are better taken into account. In this algorithm the destinations are classified to 24 different routing directions which differ in varying distances in different routing dimensions (X and Y dimension). These routing directions are illustrated in Figure 4. The idea behind this algorithm is that a packet should be routed always in a dimension where the distance to the destination is longer. This way the possibility to change the dimension remains making it possible to evade problematic areas without extending the routing path. The distance resolution in this experimental routing algorithm has three levels. The algorithm distinguishes the routing directions based on the following criteria: the destination core is (1) in the current

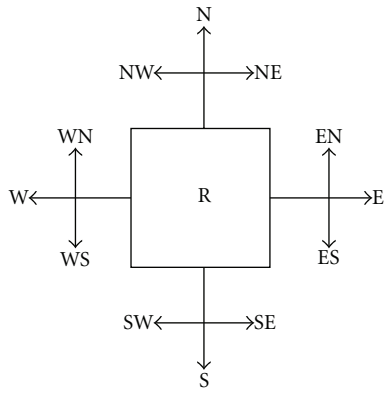


Figure 3: Routing directions. N: North, S: South, E: East, W: West, and R: Router.

Distance between current router and destination in *X*-direction

Distance between current router and destination in *Y*-direction	<−1	−1	0	1	>1
<−1	1	2	3	4	5
−1	6	7	8	9	10
0	11	12	Current router	13	14
1	15	16	17	18	19
>1	20	21	22	23	24

Figure 4: Routing directions in our experimental routing algorithm.

row/column, (2) in the next row/column in some direction, or (3) further away. Hence, there are altogether 24 different routing directions. These routing directions enable extensive classification of different routing cases and that way each case can be handled optimally.

In each of these 24 routing directions we have ranked the possible output ports based on the destination and network conditions. Every time a packet is routed the algorithm identifies the routing direction and uses available traffic status and fault information to select the appropriate output port.

6. Monitoring Algorithms and Communication

In the basic DCM structure the monitoring data is transferred in packets using the actual data network. These packets are called monitoring packets. The monitoring packets have a higher priority in the routers so that they can be transferred even when there is congestion in the network. The monitoring packets are sent from a monitor to a monitor, but because the monitors are not directly connected to each other, the packets are transferred via routers and links.

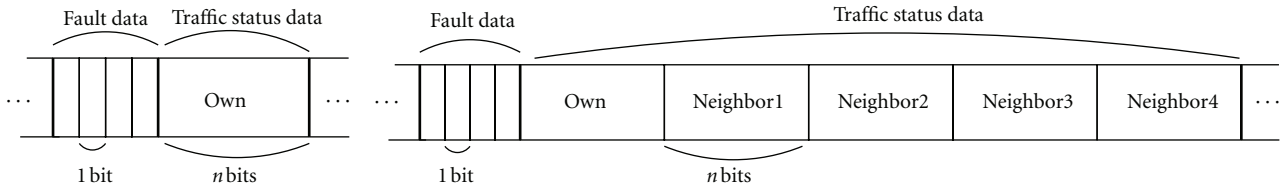
The router statuses in the DCM structure are represented with two binary numbers, one for traffic status and another for fault information. The status of a router is based on the occupancy of the FIFO buffer where packets are waiting to be routed forward. The faultiness of a single component can be represented using a single bit while number of bits in the traffic status values is related to the size of the FIFO buffer, required accuracy as well as the used additional status data processing (see Section 7.1). The resolution of the traffic status data is defined with status data granularity. The granularity defines the number of different values which can be used to illustrate the level of traffic load. For instance, when the status granularity of a router is 4 there is 4 different levels of traffic (1: no or just a little traffic, 4: highly loaded and cannot receive new packets, 2-3: scaled linearly between the edge values). The finer the granularity the better the accuracy of the status values is.

In the DCM structure the monitors exchange their own and their neighbors' statuses with each other. Typically monitoring packets include fault statuses of nearby links and one or more traffic statuses of routers depending on the size of the monitoring cluster. The structure of a monitoring packet payload in systems with monitoring cluster sizes 5 and 13 is presented in Figure 5. The contents of a monitoring packet payload are discussed in Section 7.

In centralized and clustered monitoring structures the monitoring packets are transferred in a network in the same way as the data packets (see Section 2.1). The dynamically clustered approach simplifies the monitoring communication because the routing of the monitoring packets is not needed but substituted with a packet-type recognition. Every monitor sends its status data and the neighbor status data it is forwarding to all its neighbors. The receiver recognizes these packets as monitoring packets and does not send them forward. The transfer distance of a monitoring packet is always one hop, from a router to its neighbor router. This simplicity of monitoring packet transferring combined with the simple routing procedure of monitoring packets makes it possible to keep the latency overhead on tolerable level for most applications. The presented DCM structure is targeted for applications without strict real-time constraints because the in-time delivery of packets cannot always be guaranteed. This is a trade-off of the improved fault tolerance.

A monitor stores the status data from received monitoring packets to its memory and provides this information forward to its own neighbors. This way the routers are able to receive information not only from their neighbors but also from the neighbors of their neighbors. In dynamically clustered monitoring structure the network status data spreads over the network without centralized control and without routing related processing.

The update interval of a monitoring data denotes the conditions when a monitor sends an up-to-date monitoring packet to its neighbors. In [16] we analyzed different update intervals including static and dynamic intervals as well as a hybrid interval combining both of the previous ones. The static interval sends a new packet after a certain time interval



(a) $C_{\text{Size}} = 5$ (b) $C_{\text{Size}} = 13$

Figure 5: Structure of monitoring packet payload with cluster sizes 5 and 13. n is the number of bits required to represent a traffic status value with a certain granularity.

while the dynamic interval has a status value threshold and sends a new monitoring packet when the difference between the previously sent data and the current one reaches the threshold. The third proposed interval is a hybrid of static and dynamic interval in a way that it is basically a dynamic interval but it also sends a monitoring packet after a certain time interval if the dynamic threshold has not been met during that time.

Our research has shown that, while being the most complex at the implementation level, the hybrid status update interval leads to the highest network performance in terms of throughput [16]. The difference to the static interval is significant while the dynamic interval is almost as good as the hybrid interval. We also found that the complexity difference between the implementations is negligible on large systems so we decided to use the hybrid interval in our works. The designer can have impact on the monitoring data overhead by adjusting these interval parameters. In our previous works we chose the parameters in a way that the monitoring traffic overheads were equal to enable the fair comparison of different interval methodologies. The used time interval and threshold values have straight influence on the feasibility of the status values. If these parameters are too high the status values could became badly outdated before next update. This is more crucial in networks where the traffic pattern is changing quickly. The designer should find a compromise between enough up-to-date status values and monitoring traffic overhead.

7. Status Data Diffusion

The network status data diffusion defines how far the status of a network component (router) spreads in the network. A wider spread area makes it possible to react to problems early and avoid routing packets to the worst hot spots or faulty areas.

There are two factors which affect the status data diffusion: the size of a dynamic cluster and status data processing. The cluster size defines how far the status data diffuses from a router and the amount of neighbor status data a router has. The cluster size has an effect on the quantity of neighbor status data to be transferred in a monitoring packet. If the size of a dynamic cluster is 5, it contains the neighbors of a router and the router itself. In this case it is enough to send only the router's own status to the neighbors. If the size of a cluster is 13, it also includes the neighbors of the neighbors and then four neighbor statuses should be included in every monitoring packet. In larger dynamic clusters the amount of monitoring data which has to be included in a monitoring packet increases to a level which is not practical. Therefore, we have limited our analysis to dynamic cluster sizes of 5 and 13.

7.1. Additional Status Data Processing. In DCM structure the network traffic status values are based on the load of the routers. When additional status data processing is used, each router status value is based on the state of the router itself and the state of its neighbors. When the neighbor routers' status is defined using also its own neighbors, the status data diffuses over the network.

A processed status of a router (S), which represents the traffic load in a router and its surroundings, is calculated using (1), where $S_x, x \in \{\text{L}, \text{N}, \text{S}, \text{E}, \text{W}\}$ is the status of a neighboring router in a specified direction (Local router, or a neighbor in North, South, East or, West direction):

$$S = \alpha S_{\text{L}} + \frac{1-\alpha}{4}(S_{\text{N}} + S_{\text{S}} + S_{\text{E}} + S_{\text{W}}), \tag{1}$$

where α is a factor which defines which part of the router status is related to its own utilization level. When status data of twelve neighbors is used, the statuses of different routing directions are calculated using, for example, (2), where $D_x, x \in \{\text{N}, \text{S}, \text{E}, \text{W}, \text{NW}, \text{WN}, \text{NE}, \text{EN}, \text{SW}, \text{WS}, \text{SE}, \text{ES}\}$ represents the status of a routing direction and N_i, where $1 \leq i \leq C_{\text{Size}} - 1$ is the status of a neighboring router in a specific index. Other ten statuses are calculated using corresponding equations. The routing directions are represented in Figure 3 and the used indexes of the neighbor routers in Figure 6:

$$\begin{aligned} D_{\text{N}} &= \beta N_2 + (1-\beta) N_7, \\ D_{\text{ES}} &= \gamma N_3 + \frac{1-\gamma}{2}(N_9 + N_{10}), \end{aligned} \tag{2}$$

where β and γ are factors which define at which intensity the status of a neighboring router affects the status of a direction. Coefficients α, β, and γ ($0 \leq \alpha, \beta, \gamma \leq 1$) are in here defined experimentally, but can be individually adjusted to avoid or favor routing packets to certain routing directions. $\beta, \gamma = 1$ when the size of a dynamic cluster is four. Therefore, (2) is simplified in that case and the complexity of the system is reduced.

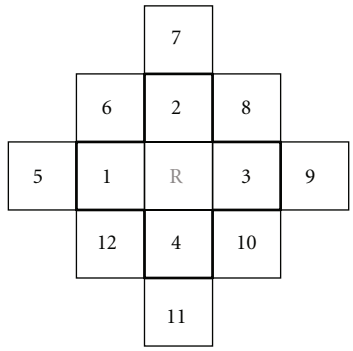


Figure 6: Indexes of the neighbor routers.

7.2. Diffusion Analysis. The analysis of network status diffusion is presented in Figures 7 and 8. The same simulation was executed with two different dynamic cluster sizes without faults and with 10% of network links being faulty. The analysis was not done with larger cluster sizes because the amount of transferred monitoring data then increases to intolerable level. The additional status data processing was used in Figures 7(a) and 8(a). The processing coefficients were experimentally chosen to obtain maximal throughput so that $\alpha = 0.5$, $\beta = 0.6$, and $\gamma = 0.6$. When the additional processing was not utilized (Figures 7(b) and 8(b)) the raw monitored status data was used and the statuses of neighbor routers did not have influence on the router status values.

The differences in network performance appear when the throughput has been fully or nearly saturated. The figures show that in a faultless network the performance differences are notable. The proportional differences were measured at the point where 60% (0.6 on the X-axle) of the maximum capacity of packets were sent during a routing cycle. The 60%point was chosen because it is clearly after the saturation point but still far from the maximum load.

All the following performance increment percentages are in proportion to the performance of an NoC with similar fault pattern, deterministic routing algorithm, and without network monitoring. In a faultless network (see Figure 7) the performance increase is 19% when status data processing is used, regardless of the monitoring cluster size. Surprisingly, when status data processing is turned off, the throughput increases 23% and 21% in systems with monitoring cluster sizes 5 and 13, respectively. Simulations were also executed in faulty networks where 10% of the links are set to unusable state (see Figure 8). These links were randomly chosen and simulations were run with several different random fault patterns. When status data processing is used, the performance increases are 78% and 74% in networks with cluster sizes 5 and 13, respectively. Without status data processing the corresponding values are 78% and 72%.

The analysis shows that DCM with small cluster size improves network performance significantly. An especially notable feature is its ability to maintain the network throughput in a faulty network. Without network monitoring the throughput decreases 41% when 10% of links become faulty. However, if the presented monitoring is used the decrement is only 11%. Furthermore, the throughput in a faulty network with monitoring is 6% higher than it of a faultless network without monitoring.

A noteworthy observation is that a larger cluster size does not have positive impact on the performance but actually reduces it. This phenomenon can have multiple reasons. One reason for the inefficiency can be that too much data processing leads to inaccurate status data and dissolves the differences between the statuses. Another reason could be the latency in the status data propagation which makes it outdated before it is utilized.

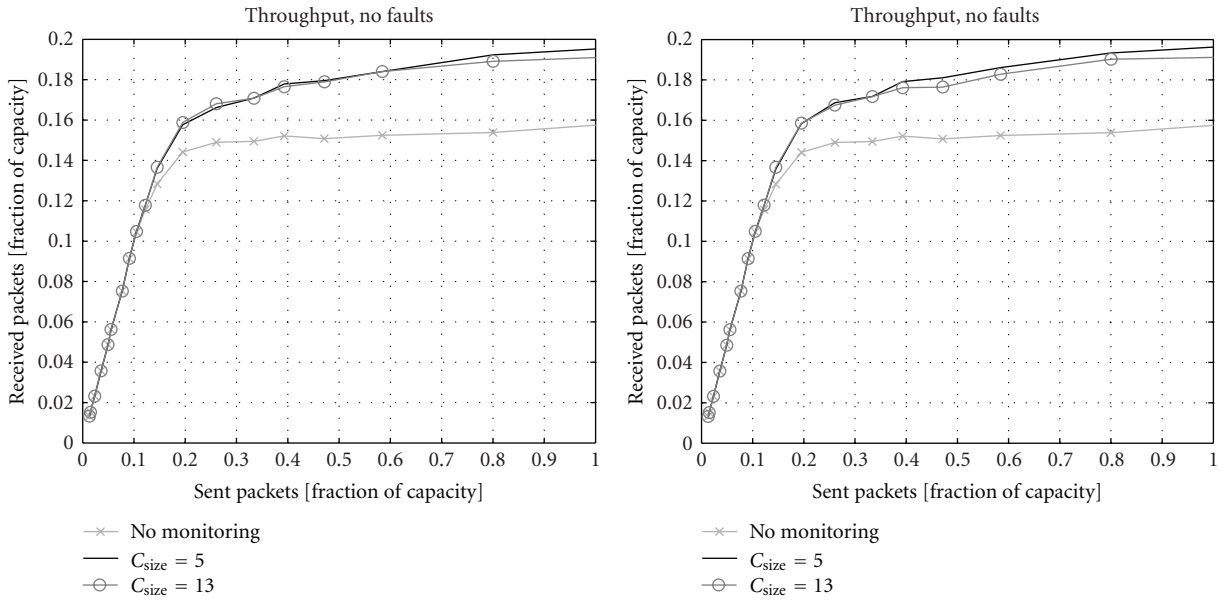
The influence of the additional status data processing is small or even nonexistent. In a faulty network there is a very small increase in the throughput. However, in faultless network the impact is even negative. For example, Figure 9 shows the difference in throughput in faulty network with $C_{\text{Size}} = 5$. As can be seen in this specific comparison the difference is negligible.

The inefficiency of the status data processing may stem from the same factors as that of the large cluster size. The differences between status values dissolve and are not on display so that the routing algorithm could make right decisions.

8. Format of Network Status Data

The network status data is used to deliver the information of the state of the network and it can be used to different purposes. When the main application is traffic management the data typically includes information concerning network load and faults. Faults are simply denoted using binary values which indicate if a component of the network is usable or faulty. In more complex systems multilevel fault indicators could be considered. The network load is denoted using a scale where different values represent different amounts of load on a network component. In our simulations the network load representation is linear. Different scales can be considered in some specific applications.

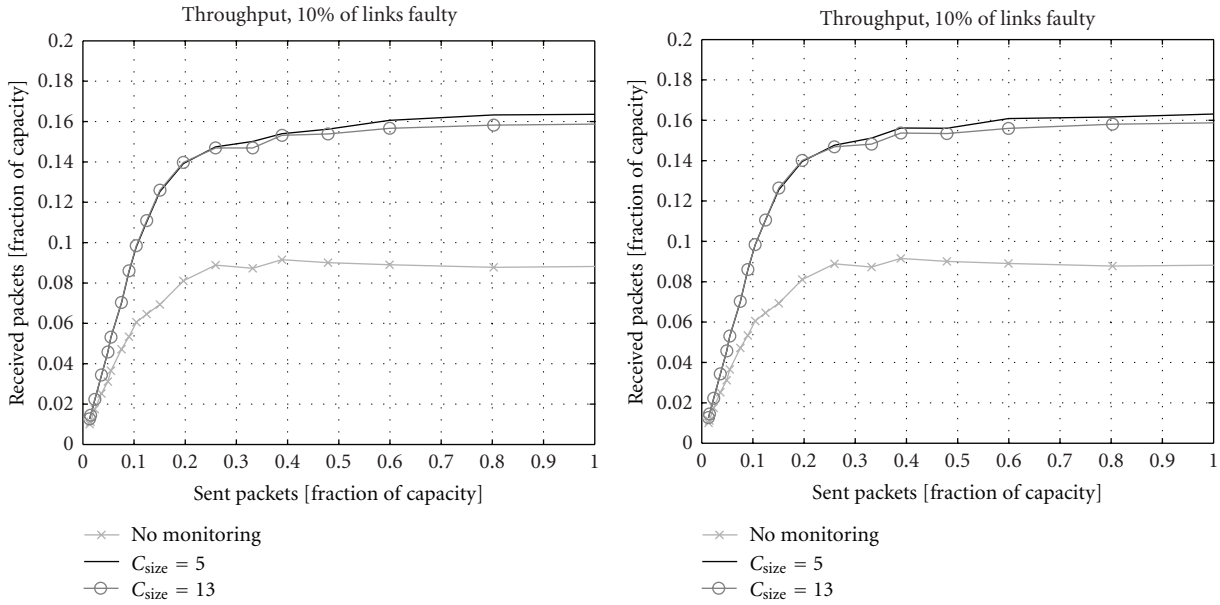
8.1. Granularity of the Router Status Values. The status data granularity defines the resolution of the status data values or how many different values there are on the scale which is used to represent the load on a network component. The smallest used value indicates that the load of a network component is very low and the highest value represents high load of the component. Rest of the values indicate component load linearly between the extreme values. An example of the status data granularity was given in Section 6. In physical implementations the status data values are represented as binary numbers which means that finer granularity requires more bits and that way increases the size of monitoring packet payload. The status data granularity impacts on the amount of the monitoring data to be transferred as well as to the required computational resources in the monitoring components. The granularity should be chosen so that the required data transfer and status data processing resources are adequate in the framework of the current NoC implementation.



(a) With additional status data processing

(b) Without additional status data processing

Figure 7: Throughput with different sized clusters (C_{size}) without network faults.



(a) With additional status data processing

(b) Without additional status data processing

Figure 8: Throughput with different sized clusters (C_{size}). 10% of links are faulty.

The 64-core NoC has been simulated with different granularity alternatives. 32 was defined to the maximum possible granularity because of the limited size of payload in the monitoring packets. A status value with 32-level granularity can be indicated with 5 bits. When monitoring cluster size is 13, a monitoring packet should include information on router's status and statuses of its four neighbors. With granularity of 32, this takes 25 bits which can be considered a realistic amount of data in a monitoring packet. The same data granularity is also used between probes and monitors.

The throughput of a 64-core NoC with diverse status granularity is presented in Figures 10 and 11. The simulations were carried out in a faultless network and in a network where 10% of links were faulty. The results show that in a fully functional network the performance is only slightly improved when granularity is larger than eight. It

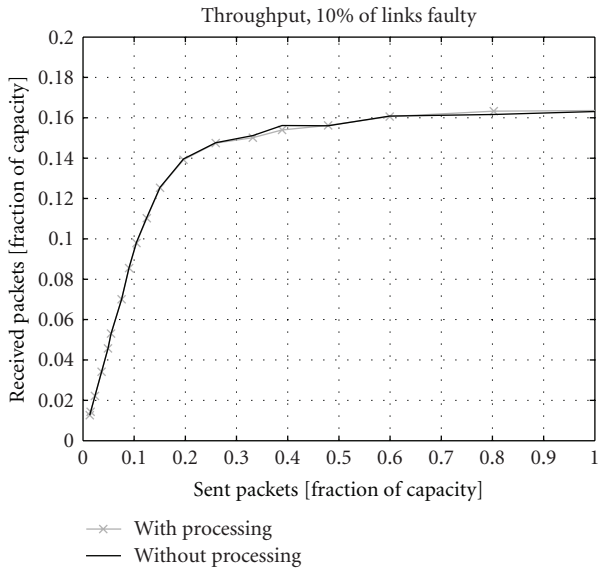


Figure 9: Throughput with and without processing. $C_{size} = 5$, 10% of links are faulty.

can be also noted that in faulty network the granularity should be at least 16. In both cases 4-level granularity leads to significantly lower performance which means 7% decrement in the faultless and 4% in the faulty network compared with the 16-level granularity. This supports the use of at least 16-level granularity. The performance of system without monitoring is illustrated as a reference.

8.2. Combining Router and Link Statuses. A method to simplify monitoring status data is to combine traffic and fault information. In the original status data format (see Figure 5) there is a binary number to represent the traffic load and a bit to indicate the resource faultiness. To decrease the monitoring complexity and the size of monitoring data payload, we analyzed two approaches where the monitoring data is combined to hybrid forms. These two hybrid data formats are presented below.

8.2.1. Hybrid Status Data Using Traffic Status Values. Traffic status values can be used to indicate faults by defining that the maximum status value does not only indicate high traffic load but also faulty resources. If there is a faulty component in some direction, the traffic status value of that direction is set to its maximum value. In this case the payload of a monitoring packet (see Figure 5) is reduced by 4 bits because the fault values are not included. When this format is utilized the routing algorithm has to be configured to totally avoid the routing directions with maximum traffic values.

8.2.2. Hybrid Status Data Using Fault Indicator Values. The monitoring data is simplified even further when all the status data is combined to the boolean fault indicator values. In this approach, a routing direction is marked as faulty when there is high traffic load. The status can be restored when the traffic load decreases. This way packets are not routed in highly loaded directions. A drawback in this approach is the loss of knowledge about differences between routing directions with low and medium traffic load. Because the traffic status values are not used, the reduction of the monitoring packet payload is n bits if $C_{Size} = 5$ and $5n$ bits if $C_{Size} = 13$.

The monitoring data combination approaches were simulated with the NoC model, and the results are presented in Figure 12. Obviously, in faultless network the traffic-status-based combination works similarly as separate data. When the traffic data is integrated into the fault statuses the decrease in throughput is 8%. However, the performance is still 12% better than without monitoring. In faulty network separated status data is notably the best solution. The traffic-data-based hybrid format causes 24% performance loss which is even larger with the fault-data-based format, 40%. Nevertheless, these hybrid formats increase the performance by 36% and 8%, correspondingly, compared to the system without traffic monitoring.

The presented analysis leads to a resolution that both monitoring data classes are necessary in a system where faults are a realistic threat. In less vital applications the hybrid formats could be a good compromise.

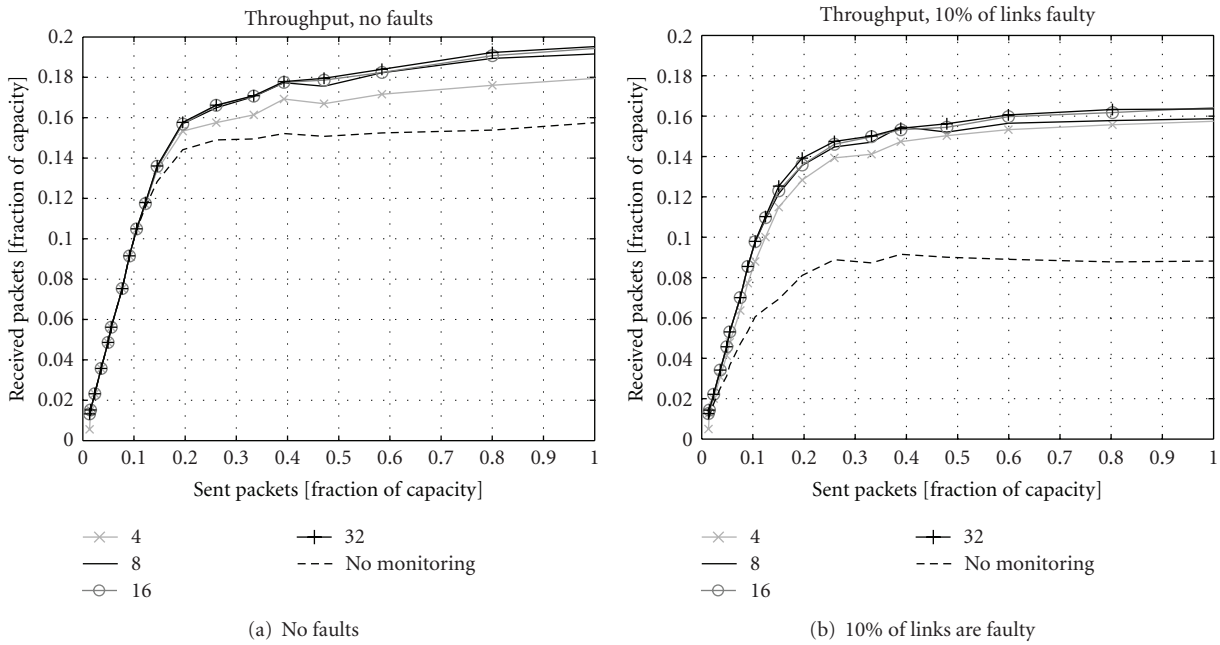
9. Serial Monitor Communication

In the DCM structure the monitoring data is transferred in the same network which is used by the original data packets. It is a straightforward solution which minimizes the requirement of additional resources. However, a shared-resource structure is always at least somewhat intrusive and it consumes the network resources which otherwise could be used by the actual data packets.

An alternative solution to the intermonitor communication is serial communication which is implemented with dedicated channels. It can be realized with relatively small amount of additional resources. A drawback in serial communication is the increased transfer delay. However, because the serial communication resources are dedicated to the monitoring communication there can be a nonstop status update without paying attention to update intervals [16].

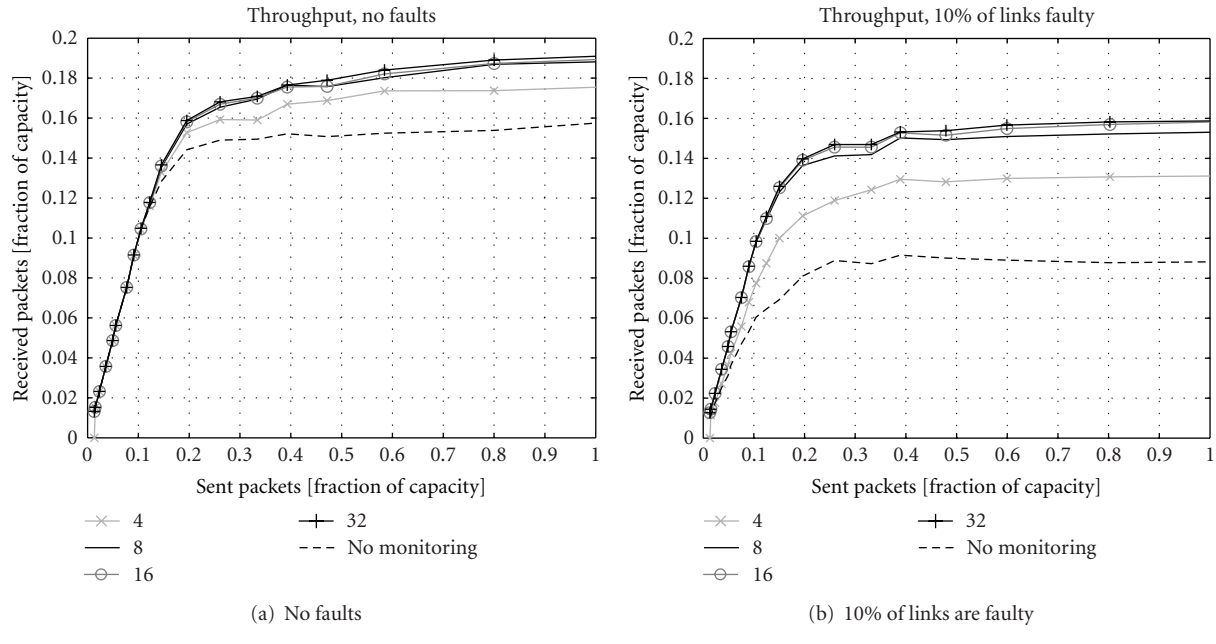
Serial monitor communication was simulated with the SystemC-based NoC simulation model. Throughput with different status data granularities and serial communication is presented in Figure 13. The serial transmitter operates at the same clock frequency as the maximum frequency of the monitoring packet transmitter. However, the latter rarely works on its maximum frequency because of the status update interval conditions.

Essentially serial communication is slower than the earlier discussed parallel, packet-based communication, and the theoretical delays of the serial communication are even more increased when there is large amount of data to be transferred, for example, in systems with relatively large monitoring clusters. However, in contrast the serial communication is operating in dedicated communication resources which can be used only to this purpose all the time. This way the status values can be updated actually more often than when the monitoring packets are transferred in



(a) No faults (b) 10% of links are faulty

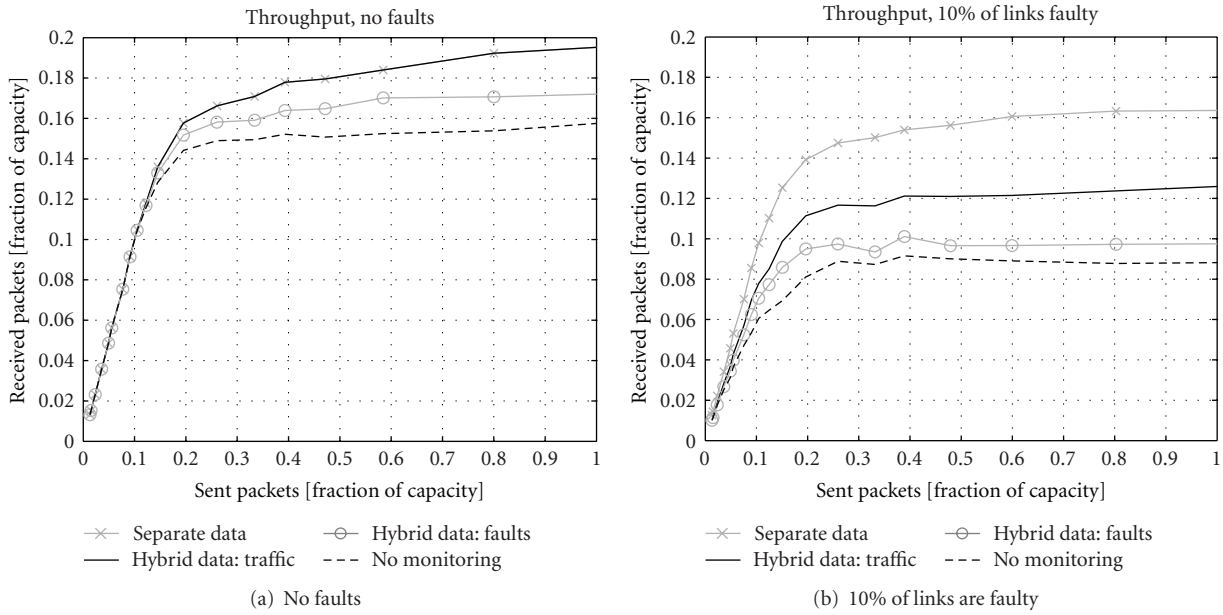
Figure 10: Throughput with different traffic status granularity alternatives. $C_{\text{Size}} = 5$ and data processing is enabled.



(a) No faults (b) 10% of links are faulty

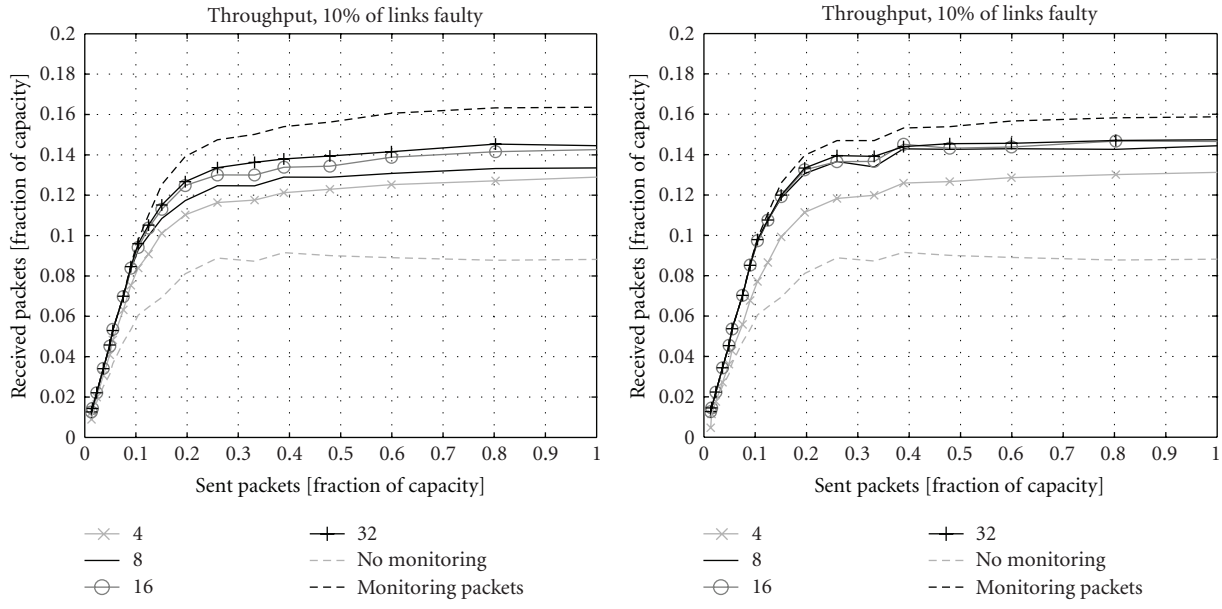
Figure 11: Throughput with different traffic status granularity alternatives. $C_{\text{Size}} = 13$ and data processing is enabled.

the shared resources. Somewhat surprisingly the system with $C_{\text{Size}} = 13$ works well also for coarser status granularities when serial communication is utilized. This is a result of shorter traffic status update interval even though in this case the amount of serially transferred data is quite large. In this case the granularity of 4 clearly stands out. Possibly the granularity of 4 is simply too rough to be used with the data amount of a system with large monitoring clusters. Figure 13(a) shows that when serial communication and small cluster size are used the performance differences are more notable also between granularities 8, 16, and 32. When serial communication is used in system with $C_{\text{Size}} = 5$ the performance with granularity of 32 is 11% less than with corresponding system using monitoring packets. Respectively, the performance of a system with serial communication and granularity of 4 is 39% better than the performance of system without monitoring. The corresponding percentages for system with $C_{\text{Size}} = 13$ are 6% and 42%.



(a) No faults (b) 10% of links are faulty

Figure 12: Throughput with separate traffic and fault data as well as with the hybrid formats. $C_{\text{Size}} = 5$ and data processing is enabled.



(a) $C_{\text{Size}} = 5$ (b) $C_{\text{Size}} = 13$

Figure 13: Throughput with different granularity alternatives using serial communication. 10% of links are faulty and data processing is enabled.

The serial communication could be a useful option when a designer wants to keep the communication resources of different applications separately. The serial approach guarantees that the monitoring communication does not disturb the actual data which is transferred in the network. It may be possible to increase the clock frequency of the serial transmitter from what was used in the presented analysis. In this case the performance differences should shrink.

10. Using Fewer Monitors

The DCM system is based on a structure where there is an identical monitor attached to each router. These monitors include both monitoring and probing components. One potential way to reduce monitoring structure complexity is to decrease the number of monitors systematically by removing every nth monitor. In this approach there is a probe

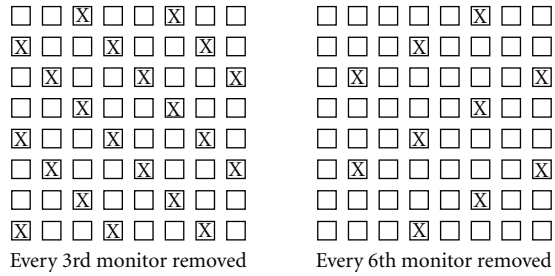
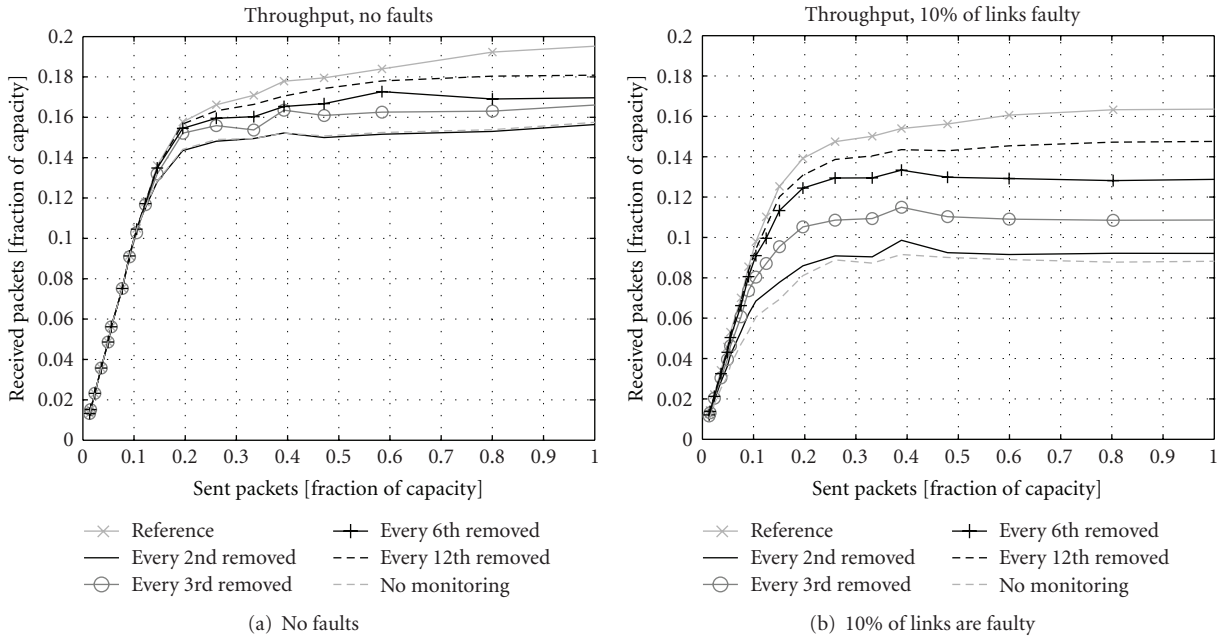


Figure 14: Two patterns of removed monitors. Removed monitors are marked with X.



(a) No faults

(b) 10% of links are faulty

Figure 15: Throughput with fewer monitors. $C_{\text{size}} = 5$ and data processing is enabled.

attached to every router but a monitor is attached only to a limited number of routers. This means that there is still complete knowledge of the network state in the monitors because there is probes attached to every router. Two monitor removal patterns are illustrated in Figure 14. The monitors receive probed data, process it, and deliver it to the local router. The routers that do not have their own monitor should utilize a deterministic routing algorithm because they do not have access to the probed status data from the neighboring routers. An adaptive algorithm is utilized in the other routers [7]. The simplified deterministic routers forward packets based on a deterministic routing algorithm. In problematic traffic or fault cases, a simplified router could route packets randomly just directing them to some of its neighbors. In any case the neighbors of a deterministic router are adaptive routers which can route the packet forward adaptively.

In addition to performance, the reduction of monitors affects the complexity of the NoC implementation. Routers which do not have their own monitoring component could have less complex routing logic which decreases the router area. This is due to the deterministic routing algorithm which substitutes the adaptive routing algorithm in the routers which do not have a monitoring component. However, the probing components cannot be simplified because they have still have to offer status data to other monitors.

This approach was analyzed using our SystemC-based NoC simulation model and the results are presented in Figure 15. The simulation cases were chosen so that the unmonitored routers are placed as evenly as possible in the network. This way we defined four simulation cases where every second, every third, every sixth, and every twelfth monitor was removed from the network.

The figure shows that the removal of a monitor, even if it is just every 12th, has notable influence on network throughput and the influence is even more remarkable when there are faults in the network. Removal of every second monitor causes 18% performance decrement in faultless network and 43% in the network with 10% of faulty links. In faultless network the performance is equal with the performance of a system without traffic monitoring. In faulty network there is 2% performance increase compared

to the unmonitored system. If just every 12th monitor is removed, the performance decreases by 3% and 10%, respectively.

The removal of monitors has positive impact to area and traffic overheads caused by the monitoring system. However, the total area of the monitoring system is almost negligible compared to the area of a 64-core NoC. This way the removal of monitors cannot be justified with the reduced complexity when the performance decrement is as large as presented here. In application-specific NoCs it could be reasonable to remove monitors from areas where traffic is predictable so that the resources can be sized properly during the design phase and adaptivity is not necessary. However, in our work the focus is on homogeneous general-purpose NoCs so the monitors are placed evenly over the network.

11. Discussion and Conclusions

The dynamically clustered monitoring structure for fault-tolerant Networks-on-Chip has been presented and analyzed in this paper. Dynamically clustered monitoring does not require any centralized control. There is a simple monitor and a probe attached to each router in the network. Centralized control is not required but the monitors exchange information with each other. Each router has a dynamic cluster around itself from where a router collects the data it needs for traffic management. The different features of the dynamically clustered monitoring structure were analyzed and their influence on the overall performance of the system were studied. Most of these presented features can be utilized in different NoC implementations with various requirements and limitations. However, due to nature of adaptive, shared-resource system, the presented DCM structure could not be the best solution to systems with strict real-time requirements.

In future works the analysis of individual cores and routers will be improved. In this phase, the NoC simulation model does not enable the analysis of specific senders and receivers but concentrates only on overall performance. This makes it possible to analyze how different monitoring methods and parameters affect performance from a component's point of view.

Performance of the DCM structure could be adjusted by using different sized monitoring clusters in different areas in the network. Areas with low traffic load may work at reasonable performance using very simple deterministic routing algorithms. At the same time in the same system there could be performance critical areas with high traffic loads and tight quality of service requirements. It could be necessary to use larger monitoring clusters and adaptive routing on these areas. Another useful feature could be on-fly reconfiguration of cluster size and the routing algorithm. Performance and energy consumption of the communication resources could be optimized by using more complex mechanisms in the critical areas of the network.

Our SystemC-based NoC simulation model has been proved to be an efficient tool to analyze and simulate different aspects in Networks-on-Chip. The model is reasonably easy to configure for the analysis of different features. The NoC simulation model was used to analyze monitoring algorithms, monitoring data diffusion areas, format of monitoring data and communication as well as number of monitors. The presented research shows that in most cases simple monitoring algorithms and small monitoring cluster areas perform at least as well as more complex implementations. In this paper the most complex structures did not cause significant improvements to performance. However, these structures and algorithms may be developed further in the future works. Another observation is that even small adjustments in the system parameters can have significant influence to the overall performance. Therefore the parameters should be chosen carefully while designing a complex DCM structure.

Acknowledgments

The authors would like to thank the Academy of Finland, the Nokia Foundation, and the Finnish Foundation for Technology Promotion for financial support.

References

[1] W. J. Dally and B. Towles, "Route packets, not wires: on-chip interconnection networks," in *Proceedings of the 38th Design Automation Conference*, pp. 684–689, June 2001.

[2] L. Benini and G. De Micheli, "Networks on Chip: a new paradigm for systems on chip design," in *Proceedings of the Design, Automation and Test in Europe (DATE '02)*, pp. 418–419, 2002.

[3] C. Grecu, A. Ivanov, R. Saleh, and P. P. Pande, "Testing network-on-chip communication fabrics," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 26, no. 12, pp. 2201–2213, 2007.

[4] D. Bertozzi, L. Benini, and G. De Micheli, "Error control schemes for on-chip communication links: the energy-reliability tradeoff," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 6, pp. 818–831, 2005.

[5] T. Lehtonen, D. Wolpert, P. Liljeberg, J. Plosila, and P. Ampadu, "Self-adaptive system for addressing permanent errors in on-chip interconnects," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 18, no. 4, pp. 527–540, 2010.

[6] V. Rantala, T. Lehtonen, P. Liljeberg, and J. Plosila, "Analysis of monitoring structures for network-on-chip—a distributed approach," *IGI International Journal of Embedded and Real-Time Communication Systems*, vol. 2, no. 1, pp. 49–67, 2011.

[7] W. J. Dally and B. Towles, *Principles and Practices of Interconnection Networks*, Morgan Kaufmann, 2004.

[8] V. Nollet, T. Marescaux, and D. Verkest, "Operating-system controlled Network on Chip," in *Proceedings of the 41st Design Automation Conference*, pp. 256–259, 2004.

[9] R. Mouhoub and O. Hammami, "NoC monitoring hardware support for fast NoC design space exploration and potential NoC partial dynamic reconfiguration," in *Proceedings of the International Symposium on Industrial Embedded Systems (IES '06)*, pp. 1–10, October 2006.

[10] K. Goossens, J. Dielissen, and A. Rădulescu, "Æthereal network on chip: concepts, architectures, and implementations,"

IEEE Design and Test of Computers, vol. 22, no. 5, pp. 414–421, 2005.

[11] C. Ciordas, K. Goossens, T. Basten, A. Radulescu, and A. Boon, "Transaction monitoring in networks on chip: the on-chip run-time perspective," in *Proceedings of the International Symposium on Industrial Embedded Systems (IES '06)*, pp. 1–10, October 2006.

[12] J. van den Brand, C. Ciordas, K. Goossens, and T. Basten, "Congestion-controlled best-effort communication for Networks-on-Chip," in *Proceedings of the Design, Automation and Test in Europe (DATE '07)*, pp. 1–6, April 2007.

[13] M. Al Faruque, T. Ebi, and J. Henkel, "ROAdNoC: runtime observability for an adaptive Network on Chip architecture," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD '08)*, pp. 543–548, November 2008.

[14] T. Marescaux, A. Rångevall, V. Nollet, A. Bartic, and H. Corporaal, "Distributed congestion control for packet switched Networks on Chip," in *Proceedings of the International Conference ParCo*, pp. 761–768, 2005.

[15] P. Gratz, B. Grot, and S. Keckler, "Regional congestion awareness for load balance in networks-on-chip," in *Proceedings of the IEEE 14th International Symposium on High Performance Computer Architecture (HPCA '08)*, pp. 203–214, February 2008.

[16] V. Rantala, T. Lehtonen, P. Liljeberg, and J. Plosila, "Analysis of status data update in dynamically clustered network-on-chip monitoring," in *Proceedings of the 1st International Conference on Pervasive and Embedded Computing and Communication Systems (PECCS '11)*, March 2011.