A message-level monitoring protocol for QoS flows in NoCs

Leonel Tedesco
UNISC
Santa Cruz - BRASIL
leoneltedesco@unisc.br

Thiago Rosa
FACIN - PUCRS
Porto Alegre - BRASIL
thiago.rosa@acad.pucrs.br

Fernando Moraes
FACIN - PUCRS
Porto Alegre - BRASIL
fernando.moraes@ pucrs.br

Abstract— **This work proposes a traffic monitoring scheme to be applied to NoCs. Current schemes transmit monitoring traffic to one or more MSA (Monitoring Service Access Point), and such MSAs are responsible to take some action, as task migration and fault detection. The proposed monitoring method is flow-oriented, i.e, it works at the message-level, and traffic congestion is evaluated all along the source-target path. It can be applied to different adaptation methods as adaptive routing, task migration and dynamic task mapping. Traffic information of QoS flows is stored in tables located at each router. Traffic initiators indicate the routers of the QoS flow path that should be monitored. MSAs are placed in targets of QoS traffic, and notify to the traffic source the occurrence of congestion. DATA packets are used to transmit both application data and monitoring traffic. Presented results evaluate the method in terms of area overhead, and precision/reactivity to congestion events, associated to an adaptive source routing algorithm.**

I. INTRODUCTION

An important trend observed in MPSoCs is the growing number of processing elements integrated in a single chip. This leads also to an increasing number of applications to be supported, which may have different features in terms of functions and traffic patterns. This brings to the overall chip communication events a dynamic nature.

As applications are dynamically inserted into the MPSoC, the occurrence of hot-spots is highly probable. Therefore, the consequences of the generated hot-spots must be detected and treated as soon as possible. Monitoring units, which collect statistical traffic information, are largely employed in NoCs [1][2][3][4], communication structure already used in MPSoCs [5][6]. Such units can notify some modules of the system that abnormality in the network traffic has occurred. If the monitored parameter is out of its bounds, some action is taken.

A monitoring process is composed basically by two actions: information collection and data processing. Probes are attached to system components, and collect traffic information, which are sent to one or more MSAs (Monitoring Service Access Point). A MSA is a control system that process monitoring information and decides how to treat a detected problem [3].

Two approaches for monitoring processing may be applied. In the centralized monitoring, all probes send information to one MSA. The convergence of traffic towards the MSA may generate a congestion point in the network, being this approach suitable to small networks. In the distributed monitoring, information is collected in specific regions of the network. This approach reduces monitoring traffic and is more scalable, compared to the centralized method, being suitable to larger networks.

The transmission of the monitored information is an important issue, because there is a trade-off between the area overhead and performance. A specific monitoring network has higher performance, due to the absence of application traffic concurrence. It is also possible to obtain a higher number of monitored values. Dedicated wires are another approach, where monitoring and application packets share arbitration and routing processing. However, these methods can be prohibitive in terms of area cost. Finally, monitoring and application packets can be multiplexed in the same network. This reduces the area utilization, at the cost of performance decreasing, which can be minimized with the utilization of a priority mechanism.

This work proposes a monitoring scheme where traffic initiators define the routers of the QoS path to be monitored. MSAs located at the target NI evaluate before each message transmission the congestion status of the path, characterizing a distributed approach. Application data are also used to transport monitoring information. Therefore, there is no need of specific monitoring packets for congestion status transmission. Tables located in NoC routers support this protocol are responsible to store QoS flows information.

II. RELATED WORK

This session presents previous work in traffic monitoring for NoCs. They differ in terms of monitoring parameters, number of MSAs and structure for data transmission.

Marescaux et al [1] adopt the use of monitoring units located in all NoC routers. Congestion is monitored by counting how many packets are stored in buffers. If there are more packets than a threshold value, it means that the current router is congested. A notification message is then mounted and sent to the traffic source through a dedicated network. This one configures its traffic shaper to adjust the packet generation rate.

In the work presented by Van den Brand et al [2], the link utilization is used as monitoring parameter. According to the authors, the use of this metric is justified by its precision to evaluate network contention. Probes located in the overall network are used to collect link utilization. This information is transmitted over a guaranteed service channel, without the interference caused by concurrent application traffic.

The monitoring scheme proposed by Ciordas et al [3] is configured at run time by a MSA, which can be distributed or centralized. A traffic manager regulates the data

978-1-4244-8278-8/10/$26.00 ©2010 IEEE

transmission from the MSAs to probes (for probe configuration) and from probes to MSAs (monitoring information). All monitored information is modeled as *events*, and each event has the following parameters: *identifier*, which defines its class; *timestamp*, which defines the moment when the event occurs; *producer*, which indicates the event generator; *attributes*, which are specific parameters for each event type. Observed events are related to application traffic transmission, network alarms and loss of monitoring information.

Al Faruque et al. [4] also propose an event-based monitoring system, which offers adaptivity at both system and architecture levels. Adaptation at system level is applied by mapping applications. Adaptation at architecture level occurs by the execution of a routing algorithm and virtual buffer channels according to their demand. The monitored events are those related to routers operation. A *TTL-expire* event occurs when a packet exceeds the maximum number of routing hops. A *No-route-found* event occurs when there is a fail for a route search. A *No-buffer* event occurs when the arbiter cannot find a buffer to store an incoming packet. Finally, a *Buffer-full* event occurs when a buffer for an established connection becomes full.

The original contribution of this work is the monitoring scheme, executed only in the path of QoS flows, and not over all network routers. Also, the monitored information flow uses the same packets of the application data flow. Finally, the monitoring information processing at the MSAs is performed at the message-level, and not for each packet.

The advantages of the method are: (*i*) a global view of the congestion path; (*ii*) the traffic initiator may define the routers to be monitored; (*iii*) absence of specific monitoring packets, reducing the global traffic load in the network; (*iv*) the same network structure is used, which maintains the area overhead. The drawbacks of the method are the effect of message sizes in the amount of time to change to the new path for a congestion event and the increased complexity for the external NIs to support the proposed monitoring protocol.

III. DESCRIPTION OF THE PROPOSED MONITORING SYSTEM

This session presents the proposed monitoring method. Initially, protocol aspects are discussed, with a description of packet types and how traffic sessions are opened and closed. Also, a simple example explaining congestion detection and traffic source notification is presented. This session finishes detailing the router architecture with area results.

A. Monitoring storage structure and packet types

Two types of tables are used to store congestion information in the NoC: RCT and TCT. The RCT (*Router Congestion Table*) is available in all routers of the NoC. The RCT has a set of entries, one for each QoS flow passing through the router. Each entry stores the flow identification (flow number) and the hop number (identification). The TCT (*Target Congestion Table*) is inserted at the target NI, and stores the congestion level at each router of the source-target path.

Four packet classes are used: SREQ, DATA, ALARM and CLEAN. Each packet has a header, with two 32-bit flits, and optionally a payload part. The first header flit contains the path to the target (PTT field). The second header flit is used for control, with the two first bits indicating the packet type identification.

The SREQ packet establishes a QoS flow session, initializing all RCTs of the path, and the MSA located at the target router, with the number of hops to be monitored.

DATA packets (Figure 1(a)) are those that carry application information, and also congestion information updated by monitored routers. For monitoring proposes, the parameters used in the second flit of a DATA packet are the *ident* and *ocup*. The *ident* field is the router address responsible to collect the congestion status. The *ocup* field stores congestion status of the router specified in the field *ident*.

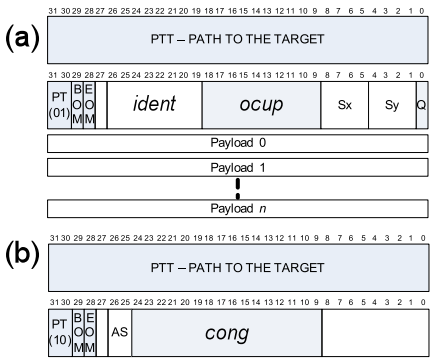

Figure 1. DATA (a) and ALARM (b) packet structures.

ALARM packets (Figure 1(b)) have three functions: (*i*) notify the traffic source if the QoS session requested (by the SREQ packet) has been initialized or not; (*ii*) synchronize the messages transmission, for a in-order packets reception at the target; (*iii*) notify the source router the congestion points in the source-target path. The field *cong* carries the path congestion status, where each bit indicates if a hop on the path is congested or not. Sixteen bits are available to store this information, which limits the QoS path to 16 hops.

CLEAN packets are sent to remove entries of routers internal tables, and may occur in two situations. The first situation arises when it is not possible to initialize internal tables for one or more routers of a QoS flow path. The second situation is related to some modification on the path, reflecting a decision of the source router taken after the reception of an ALARM packet indicating congestion.

B. Monitoring execution and congestion detection

This section describes an example detailing a monitoring session for a flow named ‘X’, with intermediate routers labeled from 1 to 7. The target router, using the information stored at the TCT, has a global knowledge of the path being used by the DATA packets of the flow ‘X’. A SREQ packet establishes a session between the source-target pair. Each session contains one or more messages, and each message are composed by fixed size packets. Although a session is established, there is no resource reservation, as in circuit switching. This packet initializes the field identification of all RCTs in the path, and the TCT.

After session initialization, each DATA packet is transmitted with a different *ident* value, chosen by the traffic generator. This feature makes this monitoring method flexible, since it is possible to monitor all the routers of a path, or a part of the path, located in a specific region. When a router in the path receives a DATA packet, it verifies the *ident* field, and if its matches with the router address, *ocup* information is inserted into to packet. This field is filled with the value of the parameter adopted for QoS evaluation.

Figure 2 illustrates a request to capture the occupation of the router 3 (*ident*=3). The buffer occupation of the input port receiving flow 'X' is 2, value forwarded to the target router by means of the *ocup* packet field. When the DATA packet arrives at the target router, the content of the *ocup* field is inserted in the TCT. The *ident* field addresses the TCT, sized according to the number of hops in the path. As shown in Figure 2, the table has 7 entries, since there are 7 hops in the path of flow 'X'. Note that the third entry on the table is updated.

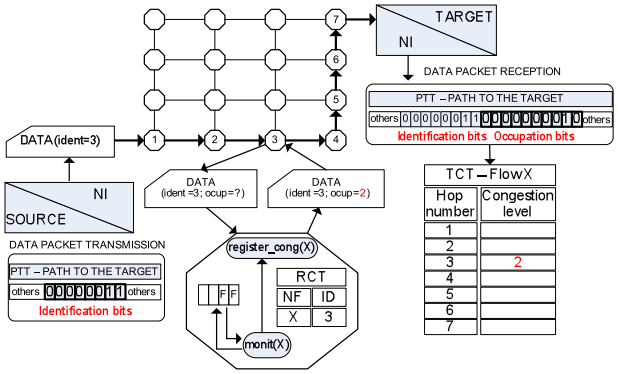

Figure 2. DATA(a) and ALARM (b) packet structures.

At the end of each message, the NI compares the TCT congestion levels to a given threshold, previously defined according to the QoS requirements of the application that generates the flow, back propagating an ALARM packet. If there is no congestion, all congestion bits of the ALARM packet are zero. Otherwise, the ALARM packet contains the address (hop number) of the congested routers. When the source router receives the ALARM packet, it knows the congested routers, and will take a decision to avoid congestion for the next message.

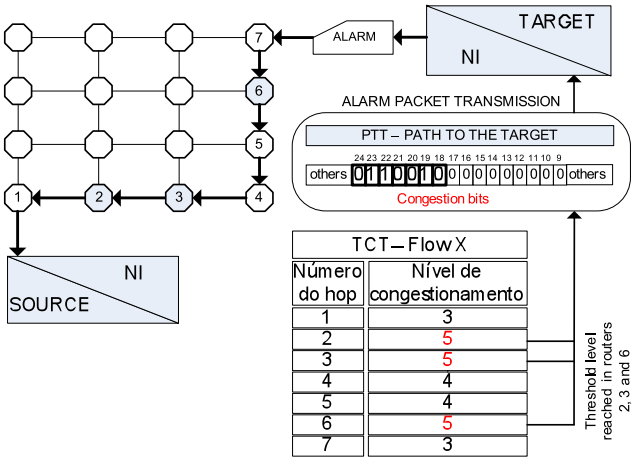

Figure 3. DATA(a) and ALARM (b) packet structures.

As an example, consider the target NI (router 7), in Figure 3. It is possible to observe that a congestion level equal to 5 is reached at routers 2, 3 and 6. At this moment, a new ALARM packet is configured with the address of the 3 congested routers asserted in the congestion information field (respectively in bits 23, 22 and 19, as detailed in Figure 4).

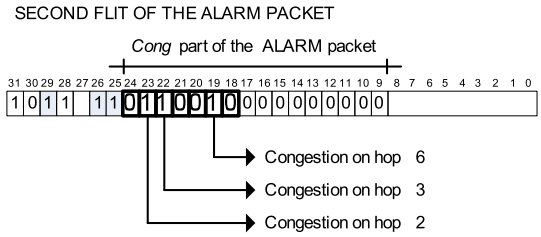

Figure 4. An example of congestion information carried in the second flit of an ALARM packet.

It is important to mention that the proposed method offers *soft* guarantees to QoS flows, since there is no resource reservation. One of the drawbacks of the proposed method is the monitoring approach, which carries one router's status per DATA packet. This feature may reduce the reactivity to congestion events. One possible solution is to increase the number of evaluated routers per DATA packet, at the cost of the packet size and latency. This is a suggestion for future research, and it is out of the scope of the present work.

C. *Hardware cost for the router*

The router supporting the proposed monitoring approach is based in the *Hermes* NoC [7], and is called *Hermes-PLR* (*Path Load Routing*). The main architectural differences to *Hermes* are: (*i*) absence of a routing module, once the NoC employs source routing; (*ii*) monitoring probes added at each router port; (*iii*) inclusion of a RCT module.

The *Hermes-PLR* router is described in VHDL RTL, targeting the xcvlx330-2ff1760 Xilinx FPGA. Table I presents the synthesis results for the central router, compared to the original *Hermes*, which employs distributed XY routing, 32-bit flits, 8-flit buffers, 2 virtual channels. Each central router for the *Hermes-PLR* router is composed by 5 input ports (each one with 2 virtual channels), an arbitration module, a RCT module, a monitor probe connected at each port and 5 output ports.

TABLE I. IMPLEMENTATION RESULTS FOR THE *HERMES-PLR* ROUTER, AND REFERENCE VALUES FROM ORIGINAL *HERMES*.

Component	*Size (in number of LUTs)*	
	Original *Hermes*	*Hermes-PLR*
Central router	***2.193***	***3.181***
▪ Input port	270	432
• Buffer	141	213
▪ Output port	163	205
▪ Routing and Arbitration	405	-
▪ Arbitration	-	253
▪ Monitor	-	88
▪ RCT	-	135

The area overhead of the proposed approach, compared to the original router, is 45%. On average, the NoC

area in an MPSoC is small, being reported in the literature values around 10% [10] when simple IPs are connected to routers. Therefore, this is an acceptable cost, because the new router received the logic to treat the source routing, monitor modules at each port, and the RCT.

IV. RESULTS

The simulated scenario uses a 5x5 *Hermes-PLR* mesh NoC. The spatial traffic distribution is illustrated in Figure 4. The flow initiated in the NI attached in the router **00** generates QoS packets to the NI attached to the router **44**. Eight traffic flows generate BE traffic to disturb the QoS flow, being each generator labeled as *S* and each receptor as *T*.

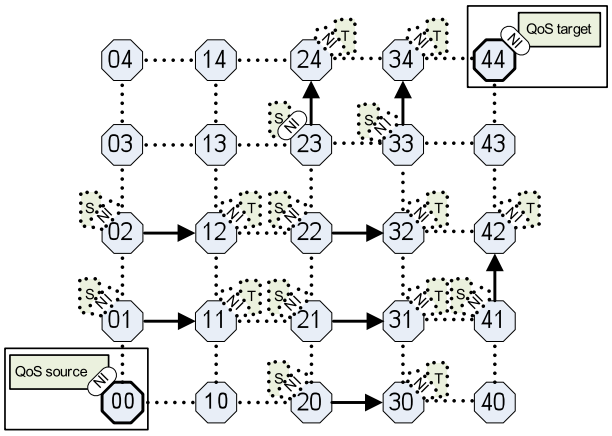

Figure 4. Spatial traffic distribution.

Each QoS traffic initiator generates 1200 22-flit packets at 240 Mbps. The first two flits of each packet compose the header. Since the total channel bandwidth is 1.6 Gbps, each traffic initiator generates a 20% traffic load. BE generators transmit three 400-packets messages, with a random interval between them, characterizing an ON-OFF distribution. Packets from these flows have also 22 flits. Each monitoring router captures the data rate for each input port as the metric for congestion evaluation at the target NIs. Therefore, a router with input data rate higher than 20% of the link capacity is considered a congested one.

Table II presents the average latency, average latency standard deviation, number of path modifications and the percentage of packets with minimum latency. The first three lines of the Table evaluate the proposed monitoring method, integrated within an adaptive routing [8], varying the message length (20, 80 and 160 packets per message). The fourth line compares the results to the deterministic XY routing algorithm, and the last line to the oblivious routing [9] (which changes packets paths randomly), both without any monitoring support.

The packet latency is the evaluated performance parameter. The minimum latency value that can be reached for a 22-flit packet of the QoS flow is 73 clock cycles. For each packet the latency is computed as follows: (*i*) arbitration: 2 to 3 cycles, depending on the turn executed by the packet in the current router; (*ii*) 1 clock cycle to connect the input port to the output port; (*iii*) 2 cycles to process the monitoring information (read the RCT and insert the monitored value into the packet). Therefore, the minimum number of clock cycles to process the header is 5 or 6 clock cycles.

TABLE II. RESULTS FOR QOS FLOWS.

Applied method	Average latency (ck cycles)	Standard deviation (ck cycles)	Path modifications	% of packets w. minimum latency
QoS message 20 packets	*86.5*	*17.6*	*42*	*43%*
QoS message 80 packets	***81.3***	***15.4***	***13***	***66%***
QoS message 160 packets	*92.7*	*22.8*	*7*	*45%*
No monitoring (XY routing)	*103.4*	*19.4*	*0*	*25%*
No monitoring (oblivius routing)	*85.4*	*13.3*	*1200*	*42%*

It is possible to observe the trade-off between message size, average latency and percentage of packets with minimum latency. When short messages are used, each router is poorly evaluated, since there are a reduced number of monitored events. This leads to a higher number of path modifications. Long messages leads to a later treatment of congestion events, i.e., the reaction of the algorithm to congestion events take a longer time. The best result is obtained when 80 packets compose each message. In this case, the analysis precision was higher, considering that each router was evaluated a higher number of times. However, when messages with 160 packets are employed, the reaction time for congestions events is higher, leading the packets to be transmitted in a congested flow for a long time.

The XY routing algorithm has, as expected, higher latencies values, since it is not possible to change the path when congestion arrives. On the other hand, oblivious routing achieves comparable results to the routing with monitoring, but there are a huge number of path changes, which may disturb other flows of the system.

Figure 5 details latency values for each transmitted packet, for 80-packet messages. When congestion is detected, a new path is computed, resulting in minimal latency values.

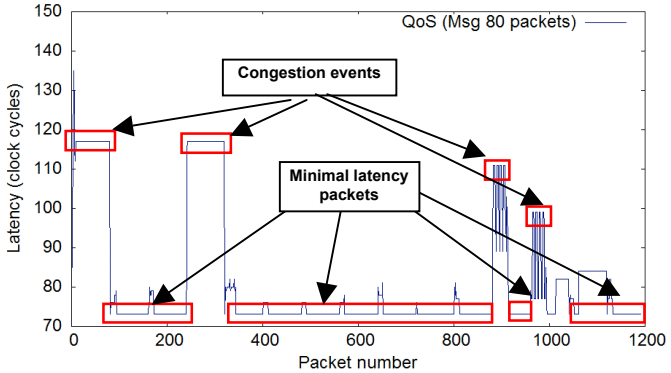

Figure 5. Packets latency when monitoring is applied (80-packet messages).

V. CONCLUSIONS AND FUTURE WORK

The original contribution of this work is a new method for monitoring to be used in NoCs, offering soft guarantees to QoS flows. This method evaluates the congestion path of each QoS flow, bringing a global view of the path being routed. Average and standard deviation latency values show the effectiveness of the method when compared to a scenario that does not employ monitoring. The use of messages with variable sizes also allowed the evaluation of the reactivity time and precision of the algorithm.

Future works include experiments using other traffic distributions, considering real application data, and the inclusion of the monitoring scheme for other adaptation methods, such as task migration and dynamic mapping. As the method is implemented in the NI, the evaluation NI area is an import future work.

ACKNOWLEDGEMENTS

This research was supported partially by CNPq (Brazilian Research Agency), project 301599/2009-2.

REFERENCES

[1] Marescaux, T. et al. “Distributed Congestion Control for Packet Switched Networks on Chip”. In: ParCo’05, pp. 761-768.

[2] van den Brand, J. et al. “Congestion-Controlled Best-Effort Communication for Networks-on-Chip”. In: DATE’07, pp. 1-6.

[3] Ciordas, C. et al. “An Event-based Network-on-Chip Monitoring Service”. In: 9th IEEE International High-Level Design Validation and Test Workshop, 2004, pp. 149-154.

[4] Al Faruque, M. et al. “ ROAdNoC: Runtime Observability for an Adaptive”. In: ICCAD’08, pp. 543-548.

[5] Intel Corporation. “Intel’s Teraflops Research Chip”. Captured: http://download.intel.com/pressroom/kits/Teraflops/Teraflops_Research_Chip_Overview.pdf, 2010.

[6] Clermidy, F. et al. “An Open and Reconfigurable Platform for 4G Telecommunication: Concepts and Application”. In: DSD’09, pp.449–456.

[7] Moraes, F. et al. “HERMES: an Infrastructure for Low Area Overhead Packet-switching Networks on Chip”. Integration, the VLSI Journal, Amsterdam, vol. 38-1, October 2004, pp. 69-93.

[8] Tedesco, L. et al. “A Monitoring and Adaptive Routing Mechanism for QoS Traffic on Mesh NoC architectures”. In: CODES+ISSS’09, pp. 109-117.

[9] Dally, W. J.; Towles, B. “Principles and Practices of Interconnection Networks”. Elsevier, San Francisco. 2004, 550p.

[10] Angiolini, F. et al. “A Layout-Aware Analysis of Networks-on-Chip and Traditional Interconnects for MPSoCs”. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 26-3, March 2007, pp. 421-434.