Implementation and Evaluation of a Congestion Aware Routing Algorithm for Networks-on-Chip

Leonel Tedesco
FACIN-PUCRS
Porto Alegre – BRASIL
leonel.tedesco@pucrs.br

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

Fabien Clermidy
CEA-LETI-MINATEC
Grenoble - FRANCE
fabien.clermidy@cea.fr

Ney Calazans
FACIN-PUCRS
Porto Alegre – BRASIL
ney.calazans@pucrs.br

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

ABSTRACT

The major part of the state of art routing proposals have a limited view of the NoC congestion, since each router takes decisions based on few neighbors' status. Such local decision may lead packets to other congested regions, therefore being inefficient. The goal of this work is to propose and evaluate an adaptive source routing algorithm, where the path between source and target PEs may be modified due to congestion events. The proposed method requires QoS session establishment and traffic monitoring. A QoS session establishes a connection between two IPs, applying application constraints. Traffic monitoring carries congestion information to the target, leading to a global view of the routing path. Evaluated performance figures include latency, traffic distribution and the delay to switch to a new path. For hot-spot traffic scenarios, the average latency is reduced by 10%. The proposed routing method also achieved a better network occupation.

Categories and Subject Descriptors

B.7.1 [Integrated Circuits]: Types and Design Styles – advanced technologies, algorithms implemented in hardware, VLSI (very large scale integration).

General Terms

Design, Experimentation, Measurement, Performance, Theory, Verification.

Keywords

Networks on Chip, Traffic Monitoring, Dynamic Routing, Quality of Service.

1. INTRODUCTION

The heterogeneity of applications on the emerging embedded systems is an important feature to consider on the MPSoC design. Processing elements, which perform distinct functions and produces different traffic patterns, bring dynamicity and unpredictability to the overall chip communication events. In addition, different QoS requirements and levels are found [1] which are usually specified in terms of throughput, latency and deadlines [2]. Due to this dynamic behavior, the treatment of unpredictable events and QoS constraints should be carried out at run time.

Monitoring units, which collect statistical traffic information, are largely employed in NoCs. 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. Examples of actions driven by monitoring include: data injection control [3]-[5], dynamic priorities for QoS packets [6], dynamic buffer allocation [7], adaptive routing algorithms [8]-[13].

The major part of the state of the art proposals for adaptive routing algorithms considers local traffic monitoring, i.e., each router analyses their immediate neighbors to choose the output port. This approach presents a limited view of the network traffic, being suitable for traffic workloads with a high degree of locality, where nodes communicate with those that are closer to them. In traffic patterns with lower locality, the local monitoring may induce the routing of a given traffic to other congested areas.

The main contribution of this work is a method that uses information collected on the source-target path to execute adaptive source routing. To avoid hot spots produced by dynamic traffic events, the proposed method is summarized as follows: (*i*) traffic monitoring, done in a distributed way, with information collected on the routing path; (*ii*) transmission of congestion information to the traffic source; (*iii*) modification of the path to the target at the source, if a congestion path is detected.

The remaining of the paper is organized as follows. Section 2 presents related works in adaptive routing. Section 3 details the monitoring processes and congestion detection. Section 4 focuses on the proposed adaptive routing algorithm. Section 5 presents the experimental setup and results. The last Section summarizes the contributions of this work.

2. RELATED WORKS

Works [8] and [9] adopt buffer occupation statistics to evaluate if the current router is congested. If congestion is detected, the output port changes. DyXY routing [8] verifies the alignment on X or Y axis with the target router. If there is no alignment, the packet is adaptively routed, according to buffer fillings on the neighboring routers. A buffer history is used to help the input arbiter to select input requests for routing. DyAD [9] analyses congestion values, corresponding to the input buffers occupation. Congestion flags information is exchanged between neighbor routers. If the router neighbors are not congested, the DyAD router works on deterministic mode, otherwise, the adaptive mode is used.

SBCCI'10, September 6–9, 2010, São Paulo, Brazil.

To minimize the local congestion view of the previous approaches, Lotfi-Kamran et al. [10] employ the concept of critical routers. Such routers monitor their congestion status, transmitting this information to routers close to traffic sources. Routing decisions are made using this information, with the goal to evenly distribute the load inside the network. Many monitoring messages to groups of routers may be generated, which increases the overall traffic load. In [11] is proposed a routing technique where congestion information is taken in parts of the network beyond adjacent routers. This information is propagated across a specific network, separated to the one used for application data transmission, bringing to this approach a global view of congestion. The availability of an output port is evaluated based on the weight of a path between a given current and destination node.

Ascia et al. [12] propose a selection mechanism strategy, which chooses between a set of admissible ports returned by adaptive routing algorithms. The congestion view of this approach is extended to two hops, i.e., the output ports available are evaluated based on values computed for each output port of each neighbor. Faruque et al. [13] present an adaptive communication scheme that dynamically allocates paths from a given flow to meet QoS requirements. Intermediate routers take decisions in a local way depending on the available bandwidth at each direction to the neighboring routers and on the distance between current and target routers. This availability is evaluated based on the weight of a path between a given current and destination node.

One particular feature found in works [10]-[13] is the fact that traffic is monitored in routers that not belong to the source-target path. The overhead to analyze immediate neighbors is small, but increases when the algorithms try to evaluate congestion far from the current router.

Comparing the present work to the state of the art, the following differences can be pointed out: (*i*) flow oriented, there is a session establishment phase before data transmission, identifying the flow, enabling monitoring in the source-target path; (*ii*) adaptation at the source router, an alarm packet signalizes the source router that the path is congested and a new path is computed. The goal is to reduce the router complexity. There is no need to verify the neighbor status, store their occupation, and take decisions based in such values. The router of the proposed method monitors itself, and includes the monitored values in data packets when it is necessary.

3. MONITORING AND CONGESTION DETECTION

Two packet classes are employed: DATA and ALARM packets. ALARM packets are used to notify the source router the congestion points in the source-target path. DATA packets carry information used by applications.

Three tables store congestion information: (*i*) SCT (Source Congestion Table), inserted at the source NI; (*ii*) RCT (Router Congestion Tables), available in all routers of the NoC; (*iii*) TCT (Target Congestion Table), inserted at the target NI. The SCT contains, for each hop, 3 fields: *hop* number; *path*, which identifies the output port taken at each hop; *cong*, which indicates if the hop is congested or not. The RCT has a set of entries, each one for a given QoS flow passing through the router. Each entry stores the flow identification (*flow number*), the hop number (*identification*), and the metric used to identify congestion. The TCT stores the congestion level at each router of the source-target path.

Each traffic session contains one or more messages, and each message is composed by fixed size packets. The first packet of a message establishes a session between the source-target pair. Although a session is established, there is no resource reservation, as in circuit switching. This first 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 hop identification value (varying from 1 to the number of hops in the path). When a router in the path receives a DATA packet, it verifies the identification field, and if its matches with the router address, the router occupation value is inserted into to packet. The occupation field is filled with the value of the parameter adopted for QoS evaluation that can be, for example, the amount of used buffer slots, the output data rate or the average time spent by packets to traverse the router.

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. As an example, consider the target NI (router 7), in Figure 1. It is possible to observe that a congestion level equals 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, and back-propagated to the source router.

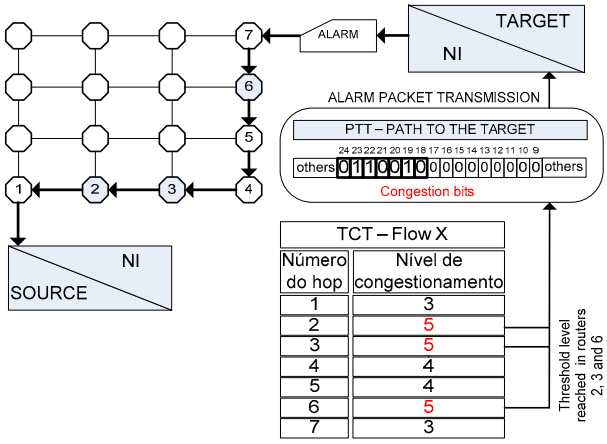

Figure 1 – Congestion detection example.

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, a new path is computed (if it is necessary) and used for the next message. No packet reordering mechanism is necessary since (*i*) all packets of a given message use the same path; (*ii*) packets belonging to different messages are not mixed, because an ALARM packet must be received before the transmission of a new.

4. PATH COMPUTATION ALGORITHM

The definition of the new path adopts the following assumptions: (*i*) minimal routing, all paths have the same number of hops to the target; (*ii*) the new path should use, if it is possible, routers near to the oldest path to minimize the impact of the new flow in other

existing flows; (*iii*) the *x* direction has higher priority when computing a new path; (*iv*) the search in the *x* direction continues until a *y* column in the present *x* position is congestion free.

Noc_cong matrix and cong vector are the two main data structures used by the proposed algorithm. The *noc_cong* matrix stores the congestion history of the routers that were part of the paths computed by the routing algorithm. As adaptive minimal routing is used, these routers are those inside the rectangle defined the by addresses of the source-target pairs. Figure 2 presents an example of a *noc_cong* matrix, where each position assumes a value that can be '1' (congested point) or '0' (non-congested point). The *cong* vector contains the bits received in the ALARM packet.

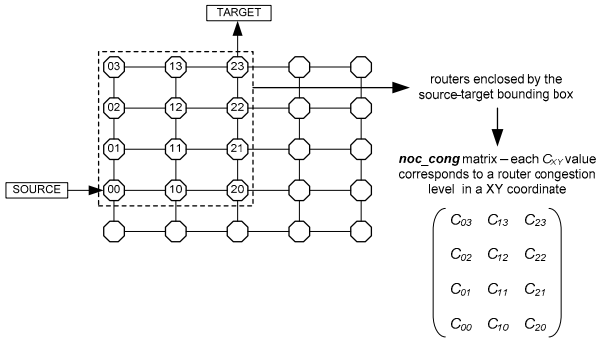

Figure 2 - Relationship between the routers addresses and the *noc_cong* matrix.

The reception of an ALARM packet with congested routers fires the execution of the algorithm presented in Figure 3. When an ALARM packet is received, the SCT is updated from the *cong* vector (*fill_SCT* function).

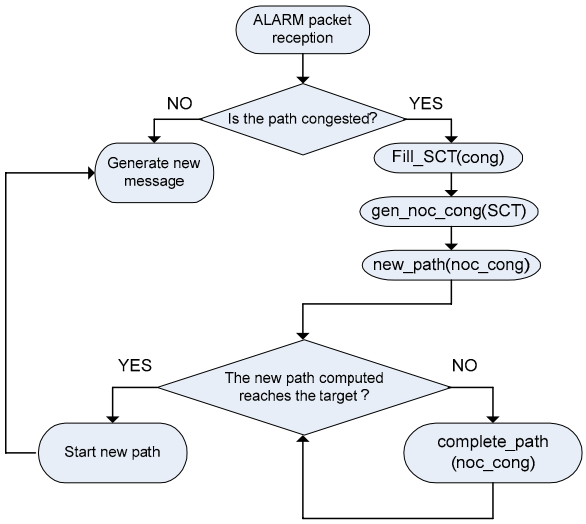

Figure 3 - Steps executed to treat a received packet at the source NI.

The example in Figure 4 is a path with 9 hops, and congestion in hops 2, 6 and 8. After updating the SCT, the function *gen_noc_cong* inserts the new congested addresses in the *noc_cong* matrix.

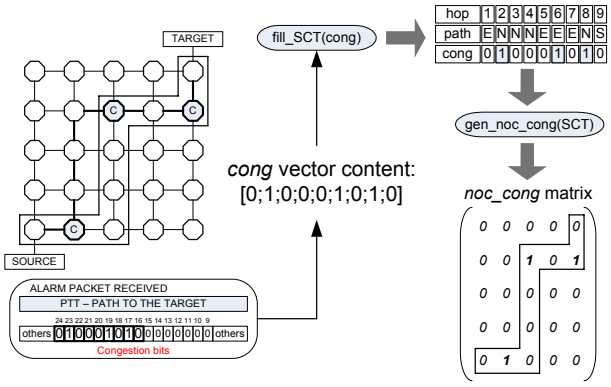

Figure 4 - Data structures used for new path computation.

The function *new_path* is the main part of the method herein proposed, and it is detailed in the pseudo-algorithm presented in Figure 5. The objective is to obtain a partial path with lines and columns of the mesh without congested routers. The algorithm starts seeking non-congested routers in the *x* direction (as in the XY routing algorithm). The *y* direction is taken when a congested neighbor in the *x* direction is congested.

```
 1:new_path (noc_cong){
 2:  while ((x_actual!=x_target) &&
           (y_actual!=y_target)){
 3:    if (noc_cong[x_actual+1][y_actual] == true){
 4:       y_actual++;
 5:       add_in_path(new_path,y_direction);
 6:    }
 7:    else{
 8:      for (i=x_actual+1 ; i<=x_target ; i++){
 9:        if (noc_cong[i][y_actual]==true)
10:          break;
11:      }
12:      for (i_aux=x_actual+1 ; i_aux < i ; i_aux++){
13:        free_path=true;
14:        for (j=y_actual ; j<=y_target ; j++){
15:          if (noc_cong[i_aux][j]==true){
16:            free_path=false;
17:            break;
18:          }
19:          if (free_path==true){
20:            last_x_free=i_aux;
21:            x_free_found=true;
22:          }
23:      }
24:      if (x_free_found==true){
25:        for (m=x_actual ; m<last_x_free ; m++){
26:          add_in_path(new_path,x_direction);
27:          x_actual++;
28:        }
29:        for (m=y_actual ; m<y_target; m++){
30:          add_in_path(new_path,y_direction);
31:          y_actual++;
32:        }
33:      }
34:      else{
35:        y_actual++;
36:        add_in_path(new_path,y_direction);
37:      }
38:    }
39:  }
40:}
```

Figure 5 - New path definition algorithm.

The loop to find an uncongested path executes until the *x* or *y* coordinates of the target router are reached (line 2). In line 3 it is verified if the next neighbor in the *x* direction is congested. If so, the next hop must be the next located in the *y* direction and a new iteration is executed. This is illustrated in the Figure 6(a). If not, routers in the *x* direction are analyzed (lines 7 to 38). Lines 8 to 11 search the first congested router in the current *y* address. Such step defines the search space for the present *y* address. In Figure 6(b) all routers in *y*=2 are not congested.

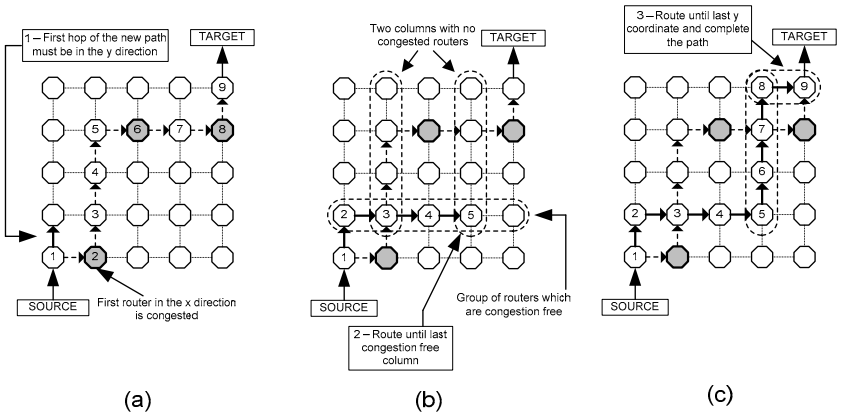

Figure 6 - An example of new path to target computation.

The loop between the lines 12 and 23 analyses the columns for each router on the current *y* coordinate. For each router in an *x* position, it is verified all routers which belong to the correspondent *x* coordinate, and that can make part of the new routing path. If there is at least one congested router, the column related to its *x* coordinate is considered not congestion free. The goal is to find the closest congestion free column to the target router. Figure 6(b) illustrates that there are two columns that are congestion free, in the *x* coordinates 1 and 3.

Once an *x* coordinate congestion free is found, the path is incremented with the horizontal and vertical routing directions, until the y coordinate of the target is reached. In the example shown in Figure 6(b), the path is completed with 3 routes on the *x* direction. In Figure 6(c) is shown 3 routes in the *y* direction. This is done between the lines 24 and 33.

Once the partial path is completed, it is necessary to complete the path until the target node, if the target is not already reached, which may occur. This is done by the *complete_path* function (see Figure 3). Due to the fact that an alignment in *x* or *y* direction with the target router will be achieved after the execution of *new_path* function, only routing in the vertical or horizontal direction will be done, depending on the current state of the *new_path* variable. Figure 6(c) illustrates a path completion until the target, with one routing hop in the *x* direction (in the example, the *east* direction).

Before the adoption of the new path, a special DATA packet (containing a *clean* bit asserted in the *others* field) is sent to the target using the previous path to clean the congestion tables. The new path is than initialized with a DATA packet to open a new routing session, identifying the routers of this path. Considering that minimal routing is adopted, the TCT on the target remains with the same size.

According to the turn model [14], a partial-adaptive routing algorithm is considered deadlock free if some turns are prohibited. To be completely adaptive, 2 virtual channels are used, the first virtual channels adopts the west-first routing, and the second virtual channel a symmetric algorithm, the east-first. The source NI chooses one virtual channel according to the following rules:

1. if the traffic target is at the right side of the source, use lane 1;
2. if the traffic target is at the left side of the source, use lane 2;
3. if the traffic target is aligned in the vertical or horizontal direction to the source, use lane 1.

5. RESULTS

The set of experiments uses the asynchronous NoC platform presented in [15]. The implementation of the methods is in SystemC TLM. The main features of this communication structure are: (*i*) source routing; (*ii*) wormhole packet switching; (*iii*) flit size equals to 32 bits; (*iv*) end-to-end credit based flow control.

Figures 7 and 8 illustrate the spatial traffic distributions. A CPU, responsible to control the traffic generators, is placed in the NoC central position. The QoS flows, generated by S nodes, execute the method for adaptive routing. Traffic targets, labeled as T, analyze the congestion condition of the QoS paths and generate ALARM packets. In both traffic distributions QoS flows are S1→T1 and S2→T2, and the shadowed routers generate disturbing flows.

In the complement traffic distribution, Figure 7, the disturbing flows have a minimum of 5 and a maximum of 9 hops. Even if the network bisection is the region with more traffic, the complement traffic evenly distributes the load in the network.

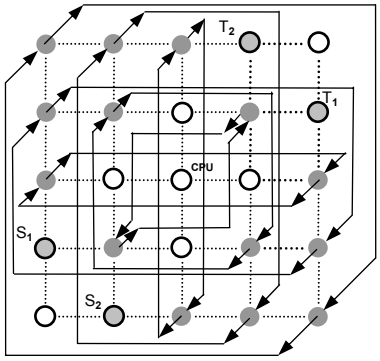

Figure 7 – Complement scenario.

In the hot-spot scenario, Figure 8, disturbing flows are labeled as H, being positioned at eight different network regions. The distance between each source-target disturbing flow is 2 hops, resulting in a higher locality degree in comparison to the complement distribution.

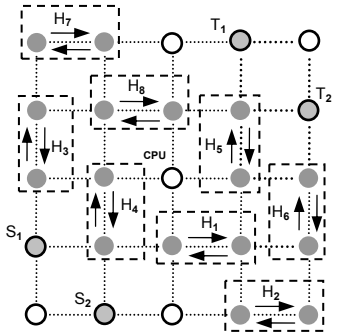

Figure 8 – Hot-spot scenario.

Initially, all flows execute the XY routing algorithm. These flows, controlled by the CPU, start at distinct moments of the simulation, to produce a dynamic traffic workload. Three simulation scenarios are applied to each traffic distribution, varying for each one the message size, as detailed in Table 1.

Table 1 - Traffic Scenarios.

Traffic type	*Description*	*Packet size (number of flits)*	*Message size (number of packets)*	*Number of packets*
QoS	Traffic with QoS requirements	16 flits	8, 16, 32	500
Disturbing	Traffic flows which disturb the QoS flows	16 flits	16	250

Each router monitors the time each flit waits in the input buffers before being injected into the network. Such spent time is defined as *flit time*. The latency of a flit is the addition of all flit times in the path.

At the target side, each QoS receptor periodically monitors its TCT to detect congestion on the path (as shown in Figure 1). A first experiment, without disturbing traffic, was executed and a flit time equals to 1 clock cycle was obtained at each hop. For the scenarios with disturbing traffic, a router with flits waiting more than 2 clock cycles to be injected into the network is considered congested. Therefore, a value greater than 2 clock cycles is the threshold used to signalize a congested router.

5.1 LATENCY AND FLIT TIMES EVALUATION

Table 2 presents the obtained average latency and standard deviation values, in clock cycles. The proposed routing method does not decrease the average latency values for the complement traffic distribution. Two reasons explain such behavior: traffic evenly distributed inside the network, leading to an absence of paths to be explored by the adaptive routing; overhead for path changing.

On the other hand, for the hot-spot scenario it is possible to observe that the average latency decrease is in average 10% and the latency standard deviation in average 7%. In this scenario, congestion on specific regions of the network where introduced, and the method proposed detected and avoided these regions. Figure 9 illustrates the latency for each packet for flow S2→T2. It is possible to note that from packet number 200, the latency reach its minimum value. In Figure 10 the packets that passed through congestion points arrive to the target node with variable latency values, also introducing jitter.

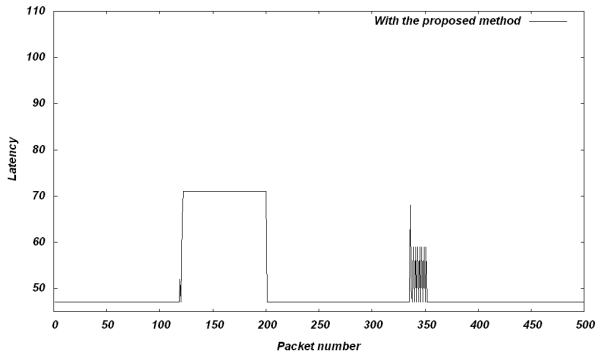

Figure 9 - Latency values for S2→T2 flows, with the proposed method (message with 8 packets).

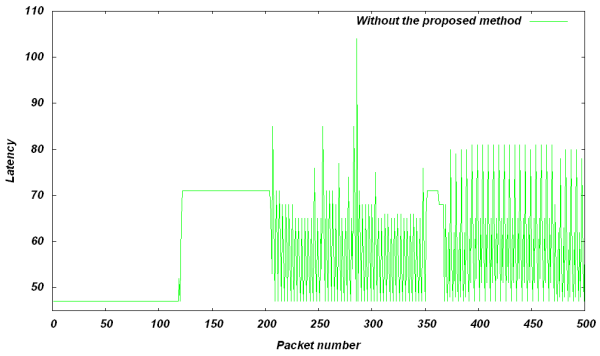

Figure 10 - Latency values for S2→T2 flows, without the proposed method (message with 8 packets).

Table 3 presents the average and standard deviation for the flit time parameter, for the hot spot traffic distribution. A better network utilization is observed, due to the diversity of paths used by the proposed adaptive routing method, combined with the detection of congestion points produced by the hot-spot pattern, which did not occur in the complement distribution.

Table 2 - Packets latency results for QoS flows, in clock cycles (AV: Average, SD: Standard Deviation).

Flow	Message size for QoS flows (packets)	*Complement Traffic Distribution*				*Hot-spot Traffic Distribution*			
		Without the proposed method		**With the proposed method**		Without the proposed method		**With the proposed method**	
		AV	SD	**AV**	**SD**	AV	SD	**AV**	**SD**
S1 → T1	8 (S1)	65.9	19.3	**63.6 (-3.5%)**	**20.5**	59.2	9.5	**53.6 (-9.5%)**	**9.7**
	16 (S2)	66.2	19.3	**63.8 (-3.6%)**	**19.9**	59.2	9.5	**52.8 (-10.8%)**	**10.0**
	32 (S3)	65,7	19.2	**64.4 (-1.9%)**	**21.9**	59.3	9.5	**54.2 (-8.6%)**	**10.3**
S2 → T2	8 (S1)	47.1	1.4	**48.6 (+3.2%)**	**5.2**	58.4	11.5	**51.1 (-12.5%)**	**8.9**
	16 (S2)	47.2	1.9	**48.7 (+3.2%)**	**5.1**	57.9	11.2	**51.2 (-11.6%)**	**9.1**
	32 (S3)	47.3	2.6	**48.3 (+2.1%)**	**4.5**	57.9	11.2	**51.8 (-10.5%)**	**9.5**

Table 3 - Flit Times in routers, for the Hot-Spot traffic pattern, in ns (Av: Average, Sd: Standard Deviation).

Message size for QoS flows (packets)	*Without the proposed method*		*With the proposed method*	
	AV	SD	AV	SD
8	4.0	6.0	2.1	0.8
16	4.2	6.3	2.2	1.4
32	4.1	6.1	2.2	1.4

5.2 REACTION TO CONGESTION EVENTS

The hot-spot scenario is the one which allows a greater path exploration. The routing mechanism reaction is the amount of packets sent with an undesired QoS level. We evaluate this metric according to the number of packets inside each message, i.e, the granularity of the algorithm. The worst case is for QoS messages with 32 packets (compare Figure 9 to Figure 11). This shows that long messages leads to a later treatment of congestion events, i.e., the reaction of the algorithm to congestion events take a longer time. As shown in Figure 9, the shortest message reaches minimal latency in packet 200. However, some greater latency values are observes from packets 330 to 350. The use of longer messages, as shown in Figure 11, conducts to a slower reaction time. As the number of monitored events is also increased with longer messages, the algorithm is able to find a less congested path, suppressing the noise observed in Figure 10. Table 4 summarizes the reactivity results for both S1→T1 and S2→T2 flows, for the adopted message sizes.

Table 4 - Number of QoS packets with high latency before path modification

Flow	***Message size (number of packets)***		
	8	***16***	***32***
S1 → T1	56	62	75
S2 → T2	81	94	113

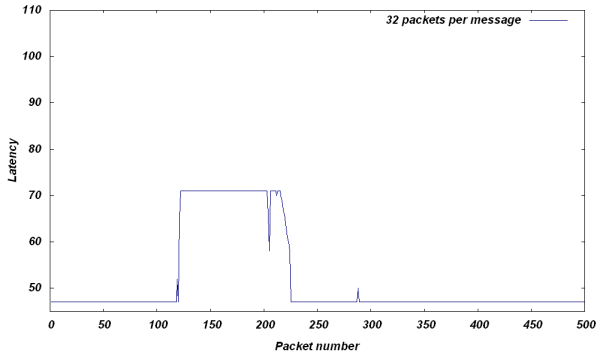

Figure 11 - Latency values for S2→T2 flows, with the proposed method (message with 32 packets).

6. CONCLUSIONS AND FUTURE WORK

The original contribution of this research work is a new method for adaptive routing to be used in networks on chip. This method takes routing decisions based on the congestion path of each QoS flow, bringing to the routing algorithm a global view of the path being routed, not only neighbors routers status, as the state of the art proposals.

Results concerning packet latency and flit times in router were presented. The average and standard deviation on the packet latency show the effectiveness of the method when compared with a fixed routed flow. Reduction on the packet average latency and standard deviation was reached for the hot-spot traffic pattern. The use of messages of variable sizes also allowed the evaluation of the reactivity time of the algorithm. A better traffic distribution was also observed, with the analysis of average flit times in routers.

Future works include comparison of the proposed method with other routing algorithms (which can use source or distributed routing); evaluate the cost of the method in terms of area and power; conduct experiments using other traffic distributions and real traffic scenarios.

7. ACKNOWLEDGMENTS

This research was supported partially by CNPq (Brazilian Research Agency), projects 301599/2009-2 and 140043/2008-0.

8. REFERENCES

[1] Marculescu, R. et al. “Outstanding Research Problems in NoC Design: System, Microarchitecture, and Circuit Perspectives”. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, v. 28(1), 2009, pp 3-21.

[2] Tedesco, L. et al. “Application Driven Traffic Modeling for NoCs”. In: SBCCI’06, pp. 62-67.

[3] Ogras, U. Y.; Marculescu, R. “Analysis and Optimization of Prediction-Based Flow Control in Networks-on-Chip”. ACM Transaction on Design Automation of Electronic Systems, v.13(1), 2008, article 11, 28p.

[4] J. W. van den Brand et al. “Congestion-Controlled Best-Effort Communication for Networks-on-Chip”. In: DATE’07, pp. 948–953.

[5] Manolache, S. et al. “Buffer Space Optimization with Communication Synthesis and Traffic Shaping for NoCs”. In: DATE’06, pp. 1-6.

[6] Mello, A. et al. “Rate-based Scheduling Policy for QoS Flows in Networks on Chip”. In: VLSI-SOC 2007, pp. 140-145.

[7] Nicopoulos, C.A et al. " ViChaR: A Dynamic Virtual Channel Regulator for Network-on-Chip Routers". In: MICRO’06, pp. 333-346.

[8] Li, M. et al. “DyXY - A Proximity Congestion-aware Deadlock-free Dynamic Routing Method for Network on Chip”. In: DAC’06, pp. 849-852.

[9] Hu, J.; Marculescu, R. “Dyad – Smart routing for networks on chip”. In: DAC’04, pp. 260-263.

[10] Lotfi-Kamran, P. et al. “BARP-A Dynamic Routing Protocol for Balanced Distribution of Traffic in NoCs”. In: DATE’08, pp. 1408-1413.

[11] Gratz, P. et al. “Regional Congestion Awareness for Load Balance in Networks-on-Chip”. In: HPCA’08, pp. 203-214.

[12] Ascia, G. et al. “Implementation and Analysis of a New Selection Strategy for Adaptive Routing in Networks-on-Chip”. IEEE Transaction on Computers, v.57(6), 2008, pp. 809 – 820.

[13] Faruque, M. A. et al. “Run-time Adaptive On-chip Communication Scheme”. In: ICCAD’07, pp. 26-31.

[14] Glass, C.; Ni, L. “The Turn Model for Adaptive Routing” . Journal of the Association for Computing Machinery, v. 41(5), Sep. 1994, pp. 874-902.

[15] Lattard, D. et al. “A Reconfigurable Baseband Platform Based on an Asynchronous Network-on-Chip”. IEEE Journal Of Solid State Circuits, v. 43(1), 2008, pp 223-235.