Security Enhancements for Building Saturation-free, Low-power NoC-based MPSoCs

Kyprianos Papadimitriou[1], Polydoros Petrakis[1], Miltos D. Grammatikakis[1], and Marcello Coppola[2]

[1]Technological Educational Institute of Crete, GR
[2]STMicroelectronics, FR

***Abstract*—In the future almost every consumer electronics device will be connected to an ecosystem of third-party partners providing services such as payment, streaming content, and so on. Present work aims to expose the foundations of a secure environment by ensuring security on the edge devices. MPSoCs are widely used in edge devices due to their capability to execute multiple applications in single-chips. To achieve the targeted security level against physical adversaries, all communications between the MPSoC and its environment must be protected.**

In an MPSoC multiple processing elements such as CPUs send requests to different on-chip memories. Network-on-chip has been proposed for MPSoC design, aiming at increasing performance and reducing power compared to on-chip buses. Tailoring the NoC to application(s) takes place usually at design-time. The selection of NoC parameter values affects both performance and power, while configuring them unwisely can result in unnecessary area overhead and chip cost. In the present work we concentrate on a commercial interconnect called STNoC from STMicroelectronics. We keep the NoC parameters fixed, and we explore the effects from the variation of other parameters, such as the injection rate of the packets transmitted by the CPUs, and the activation/deactivation of a security mechanism integrated in the network interface of the NoC, for multiple traffic scenarios with each one representing different amount of legal and malicious requests, for different mappings, and for different node setups. Experimental results, reveal the conditions under which the NoC starts experiencing saturation phenomena.

***Index Terms*—MPSoC, network-on-chip, gem5, saturation, NoC firewall.**

I. Introduction

A Network-on-Chip is typically configured at design-time by selecting the most suitable parameters, based on the requirements of the application-at-hand and the conditions the system is expected to face during operation. These specifications are provided by the customer to the NoC vendor.

We configure the STNoC with limited resources, and perform experimentation with large amount of traffic running for millions of simulation cycles. In several cases this makes the resource-limited STNoC incapable of serving the traffic, which eventually experiences saturation phenomena leading to enormous delays. The experiments were carried out using our gem5-based framework enhanced with a security mechanism placed at the network interface of NoC [1], [2]. This allowed us to explore the implications from enabling security for different parameters. Our contributions are:

- performance and power evaluation for different node setups and traffic scenarios, with and without enabling security;
- examination of the saturation-point of the on-chip network for different system parameters; we do this mainly for a small-scale node setup, and then we provide some initial experiments for more intricate networks.

The paper is organized as follows. Section II overviews related work on NoC performance analysis, as well as on security mechanisms. Section III presents the framework we employed to perform our experiments, and reports the system parameters. Section IV has the different scenarios we deployed to examine the effect of system parameters on performance and power consumption. In Section V, we examine the impact of parameter values in the delivery time of packets and the power consumption of the network, and discuss the saturation phenomena. Finally, Section VI concludes the paper.

II. Background and Motivation

Considerable research has been devoted on analyzing the performance of network-on-chips. One of the problems arising when multiple elements share a network-on-chip is network congestion [3]. The authors in [4] and [5] studied the latency of messages in relation with the rate they are injected by the nodes. The authors of [6] developed a latency model for network-on-chip performance analysis, and in an earlier work they presented an algorithm for adapting the routing on a NoC according to the traffic [7]; both works demonstrate that adjusting the NoC to the application is an important task. The authors of [8] compared the Ring, Spidergon, and 2D Mesh NoC topologies, under uniform load and realistic load assumptions, and showed that Spidergon is a good trade-off between performance, scalability, small energy and area requirements for SoCs.

Our novelty lies in that we model an STNoC instance - its router, link and network interface - in a cycle-accurate way in the gem5 environment, by designing a 2-stage pipeline switch based on gem5's garnet fixed pipeline router. We also take into account the security mechanism as an integral part of the system, which appears to be an interesting research subject [9], [10]. Our results incorporate the overhead of the NoC firewall integrated with the network interface, presented in detail in [1], [2]. These works showed that security can reduce the network load by preventing malicious requests from entering the network. The present paper extends them by discussing the way we explore the parameter selection

space. To perform exhaustive experimentation we selected a controlled-environment, with only a few nodes, which allowed us to examine all potential configurations by varying selected parameters. Our aim is to assist the NoC architect and researchers in making the proper design choices prior to proceeding with the actual STNoC customization; in the near future we will do this with the evaluation of MPSoCs with more nodes and the design of relevant tools.

III. Use of a Versatile Experimental Framework

To evaluate the effect of system parameters we employed a framework combining CPUs and shared memory, interconnected with a network-on-chip topology, augmented with a firewall placed at each network interface (NI) of the NoC [1]. Within this framework we measure delivery times starting from the time a packet enter the NI queue, and power consumption at the network-layer.

A. Building-up the Framework

The framework combines ARM v7 CPU technology and STNoC point-to-point interconnect. STNoC can be customized according to given specifications to interconnect MPSoC components, i.e. CPUs, memories, and peripherals [11]. From the network point of view, each component is a node, and information - instructions and data - is transmitted across the network in STNoC formatted packets. At the STNoC interface, prior to transmission, a packet is broken down in smaller units called flits that in turn are released into the network.

To conduct experiments we described the above system in the gem5 simulator [12], a platform widely used in computer system architecture research, encompassing processor and system-level microarchitecture. It is highly-configurable and includes support for multiple ISA and diverse CPU models including ARM, with detailed memory systems and interconnect models. Within the gem5 environment, we created an STNoC-like network model instance, which we time-annotated from cycle-accurate design specifications of the STMicroelectronics backbone architecture; it includes the STNoC router, the network interface and a synchronous link. Also, we implemented a gem5 transaction-level model of the NoC firewall and integrated it in the network interfaces of STNoC. Then, we time-annotated the NoC firewall model based on a cycle-accurate design we implemented in HDL. More details on the framework and firewall are beyond the scope of present work, and can be found in [1], [2]. It should be noted that the firewall adds an overhead of 5 cycles maximum in order to perform rule checking; this overhead is taken into account within our framework.

B. System Parameters

We explored parameters that can be customized, as we are interested in studying the response of the system when changing the parameter values. The link-width of the NoC in combination with the size of the packet affects the amount of flits into which the packet is segmented prior to transmitting it. We kept both sizes constant, thus the amount of flits per packet remained constant throughout our experiments. We varied the number of nodes that are connected to the NoC, i.e. CPUs and memories. We also have the capability to change the status of the firewall (on/off); when it is activated it adds overhead, but at the same time it relieves the network from malicious content, which incurs unnecessary load. In order to evaluate its use and examine our system's behaviour we generated malicious packets along with legal ones; we did the same in [1], [2]. Finally, we varied the injection rate of the packets generated by the CPUs.

IV. Experimental Scenarios

We experimented with different number of nodes connected to the STNoC. Figure 1 illustrates the way the nodes are interconnected for two cases; 4-nodes and 8-nodes. Present work includes experiments with up to 32-nodes, however most of them concern a 4-node setup. For each case we produced different combinations with regard to the number of legal (we also refer to them as safe) and malicious processes. We even deployed scenarios in which almost all network traffic is malicious. Although this constitutes an extreme situation it is quite realistic; due to that a virus or malicious program can replicate itself fast, there is no limit on the malicious traffic that can be generated to memory. Recent DDoS attacks at network gateways have reached bandwidths of over 300 Gbps [13].

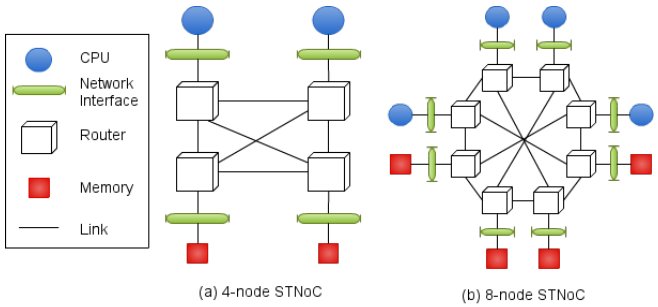

Fig. 1. We used different configurations to study the effect of security, and performed experiments with up to 32-nodes. Case (a) corresponds to two different configurations; either both CPUs or only one CPU can be active.

Table I has the parameters of the simulation framework. We ran simulations with 1 CPU/2 Memories (actually 2 CPUs but 1 is inactive), 2 CPUs/2 Memories, 4 CPUs/4 Memories, 8 CPUs/8 Memories, and 16 CPUs/16 Memories. We tested different scenarios with a varying rate of safe vs. malicious processes running in the CPUs; in Table I we represent this as $xSyM$, where x is the number of safe processes and y the number of malicious processes. The process requesting access to memory is selected with a random policy, and it can be either a "write" or "write-execute" request. A request is performed to a random address, which belongs to a certain pre-allocated rule-protected memory segment. We performed the same experiments by enabling and then disabling the NoC firewall. When enabled, the NoC firewall denies access to every single request from malicious processes since all

requests are destined to memory segments that are protected and can be used by safe processes only. Therefore, malicious requests are rejected at the network interface, which results in prohibiting unnecessary load from entering the NoC. On the other hand, when the NoC firewall is disabled, malicious requests are entering the NoC.

TABLE I
SIMULATION PARAMETERS

Parameter	Type/Value
Interconnect topology	Spidergon STNoC
CPU type	ARM v7 Cortex A-9
Nodes setup	1CPU/2MEM; 2CPU/2MEM; 4CPU/4MEM; 8CPU/8MEM; 16CPU/16MEM
Processes per CPU (Safe,Malicious)	all Safe; 8S1M; 4S1M; 3S1M; 2S1M; 1S2M; 1S3M; 1S4M; 1S8M
NoC link-width	8 Bytes
Message type	w; wx
Packet size in Bytes	136 Bytes (header=8; body=128)
Packet size in flits	17
Status of firewall	on; off
NI, Router, Link delay (in cycles)	{1, 2, 0}
Injection rate	varied from 0.01-1 cc, with a pace of 0.05

The challenging part was to setup the proper values so as to expose the cases under which the system experiences saturation. We selected a small link-width so as to build a system with limited resources, which allowed for exhibiting saturation early. The packet is relatively large, thus it is segmented into several flits, i.e. $136 \div 8 = 17$ flits per packet. It is worth noting here that in contemporary systems packets are segmented into as less as possible flits - even only 1 flit per packet. On the other hand, in our case the selected values are for sake of experimentation and served our scope well. As we will later see it allowed us to demonstrate cases in which limited-resource NoC-based systems run in steady-state and with low-power (8 Bytes consumes lower power than 16 or 32 Bytes).

In our traffic scenarios, we are interested in examining NoC behaviour for different amount of requests, safe and malicious ones, rather than in comparing its behavior for different percentages of malicious requests over total ones. In fact, examining how the system is affected by setting up experiments with different percentages of malicious requests over the total ones does not fall within the scope of present study. For example, $1S3M$ scenario corresponds to 75% malicious requests within a time-window of sending 4 packets, while $1S8M$ scenario corresponds to 88.8% malicious requests within a time-window of sending 9 packets. In both scenarios, when NoC firewall is enabled, only 1 packet is released to the NoC within their corresponding time-window. Also, consider the $1S1M$ and $10S10M$ scenarios; these generate the same percentage of malicious requests, i.e. 50%. Their difference is the following: $1S1M$ corresponds to a time-window of 2 packets in which safe and malicious requests are generated in an 1-by-1 alternate sequence, i.e. 1 safe, 1 malicious, then again 1 safe and so on. $10S10M$ corresponds to a time-window of 20 packets in which a chunk of 10 safe requests is first produced, followed then by a chunk of 10 malicious requests, then again 10 safe requests, and so on. When comparing these scenarios, the number of packets that are eventually transmitted when the NoC firewall is enabled will be exactly the same for a given simulation time period - in our case 2 million clock cycles. However, their major difference lies in the amount of safe packets arriving at the NoC in succession. Having big chunks of safe packets in succession rather that small ones, imposes high demands from the side of NoC, i.e. the NoC should be able to sustain a high serving rate. For instance, the more demanding case from the aforementioned ones is the $10S10M$ scenario. In specific, in the $16CPU/16MEM$ setup, there are time-windows in which all 16 CPUs produce concurrently 10 safe requests in a sequence; this stands for each CPU, and ideally all should be served by the NoC. After the safe traffic, the malicious traffic follows, which is perceived again as a big chunk of malicious content. However, this will be stopped by the NoC firewall. On the other hand, when the NoC firewall is disabled, being concerned about the percentage of malicious requests is meaningless as in all scenarios all packets are allowed to enter the NoC, e.g. in the $10S10M$ scenario all 20 packets are transmitted to the NoC, and this is repeated until completing the simulation runtime. In conclusion, in our study it is much more interesting to examine the system behaviour in relation to the amount of packets that are eventually released to the NoC. Later on, we show that this revealed cases in which the queues across the system cannot handle effectively large amount of traffic and the NoC becomes saturated.

By selecting different parameter values we performed experiments for several configurations. In specific, from Table I, 5 different node setups, 9 different combinations of safe and malicious requests, and 2 options for the security mechanism (enabled/disabled), give a total of $5 \times 9 \times 2 = 90$ different configurations. Regarding NoC parameters, the packet size of a write command is 136 bytes, while we configured a link-width of 8 Bytes, hence, each packet is split up in $136 \div 8 = 17$ flits.

Another parameter that affected the behavior of the experiments was the packet injection rate, i.e. the rate at which packets are released from the application layer in each CPU. We experimented with up to 20 different injection rates. Variations in the injection rate - even small ones - showed that the end-to-end delivery times of the packets are affected tremendously due to that the NoC starts experiencing congestions. We repeated the experiments numerous times in order to verify such occurrences. In fact, we did this for large simulation times, starting from a few hundred up to 2 million clock cycles.

V. EXPERIMENTAL RESULTS

In the present Section we evaluate the NoC-based system by studying the impact of security and we illustrate the relation between the end-to-end delivery time and power at the network-layer. Then, we analyze exhaustively the saturation

cases for a small-scale MPSoC, and perform some initial experiments for MPSoCs with 8 and 16 nodes.

A. *Effect of Security on Network Transmission*

Our framework enables measuring the end-to-end delivery times at the network level, and the power consumption of the routers and links. We measure two types of delay; the time a packet is waiting in the NI queue before it is actually broken down into flits and released to the network (called NI queue delay), and then, after the flit leaves the NI queue, the time spent to traverse the network until it reaches its destination. The latter delay is referred to as network traversal delay, and it is the time required for the flit to pass through all the routers on its path and eventually reach the target network interface. Table II has the results for a specific configuration when NoC firewall is disabled or enabled. The last column shows the impact from activating the firewall. Negative values indicate improvement, thus in all cases activation of security affects positively the network.

TABLE II
EFFECT OF SECURITY ON QUEUE DELAY, NETWORK DELAY AND POWER AT THE STNOC LEVEL. SCENARIO CONCERNS A 4CPU/4MEM SETUP WITH 1S2M REQUESTS. LINK-WIDTH OF STNOC IS 16 BYTES.

Metric	without firewall	with firewall	+/-(%)
NI queue delay	28.0905 cycles	6.1824 cycles	-77.99%
Network traversal delay	10.5503 cycles	8.3907 cycles	-20.47%
Router power (total)	0.5364 Watt	0.3746 Watt	-30.16%
Router clock power	0.1917 Watt	0.1917 Watt	0%
Router static power	0.1189 Watt	0.1189 Watt	0%
Router dynamic power	0.2258 Watt	0.0640 Watt	-71.66%
Link power	0.0297 Watt	0.0084 Watt	-71.68%

The chosen case concerns the $4CPU/4MEM$ node setup, with 1 safe and 2 malicious CPU processes requesting memory access. It is obtained that when the NoC firewall is active, both the traversal delay and the delay in NI queues decrease. This is due to that malicious requests are prevented from entering the network, thus fewer packets are released, resulting in smaller network traffic and less busy queues. Moreover, the total power of routers decreases by 30.16%, mainly due to the drastic decrease in the dynamic power. Finally, the use of firewall reduces the power consumed at the network links by almost 72%.

The above analysis indicates that a security mechanism relieves the network from unnecessary load in the presence of malicious processes. The values in Table II are average ones and concern all CPUs. We performed experiments for two type of messages, e.g. "write" and "write-execute". For these specific experiments the amount of flits transmitted was 2280 flits when firewall was on, and 7315 flits when firewall was off. The above use case demonstrates the way the framework is used to evaluate the effect of security both on delay and on power consumption at the NoC layer.

In Figure 2 we analyse the relation between the network traversal delay and the power consumed at the routers, when firewall is on/off. In specific, we illustrate the relation between

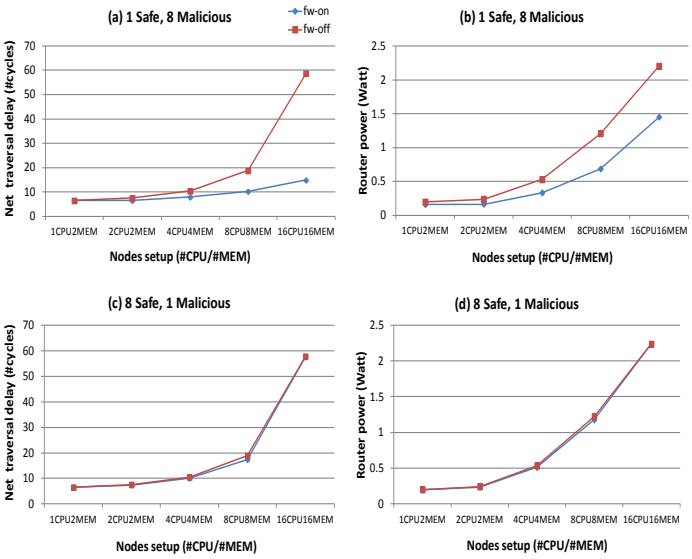

Fig. 2. Relation between network traversal delay (after NI) and router total power consumption for two different scenarios. Each chart shows both cases, when security is enabled (fw-on) and disabled (fw-off). Top side, i.e. (a)-(b), has the $1S8M$ scenario, and bottom side, i.e. (c)-(d), has the $8S1M$ scenario. Link-width of STNoC is 16 Bytes.

the delay of the flit after leaving the NI queue until reaching its destination, and the total power consumed at the routers. Figure 2 examines this correlation for two extreme cases, i.e. $1S8M$ and $8S1M$. In the first case, shown on the top side of Figure 2, (a) & (b), it appears that the total power consumed at the routers decreases with the decrease in the network traversal delay due to the activation of security, i.e. power consumption follows delay tendency. In particular, only the dynamic power is affected, as discussed earlier under Table II; neither the clock power nor the static power of the router are affected. The same behaviour is observed in the second case, shown at the bottom of Figure 2, (c) & (d). In this case power remains almost unchanged, following again the behaviour of network traversal delay.

B. *Saturation Analysis*

We have experimented with different injection rates, i.e. number of injected packets per cycle per CPU, for a 4-node system. We conducted extensive experiments with different parameters, i.e. mapping scenarios; amount of legal and malicious requests; NoC firewall status (enabled or disabled); and injection rates. The latter was kept the same across the different CPUs, in each experiment. Our aim was to find the injection rate beyond which the network reaches saturation in each case. We deployed the following mapping scenarios for the 4-node system:

(a) CPU#1 & CPU#2 send requests randomly both to MEM#1 & MEM#2;
(b) CPU#1 sends to MEM#1, CPU#2 & MEM#2 are idle;
(c) CPU#1 sends to both memories, CPU#2 is idle;
(d) CPU#1 sends to MEM#1, CPU#2 sends to MEM#2;

(e) CPU#1 & CPU#2 send to MEM#1 only, MEM#2 is idle.

The 5 different mapping scenarios, 9 different combinations of safe and malicious requests, 2 options for the security mechanism (on/off), and tests with roughly 20 different injection rates, result in a total of $5 \times 9 \times 2 \times 20 = 1800$ configurations.

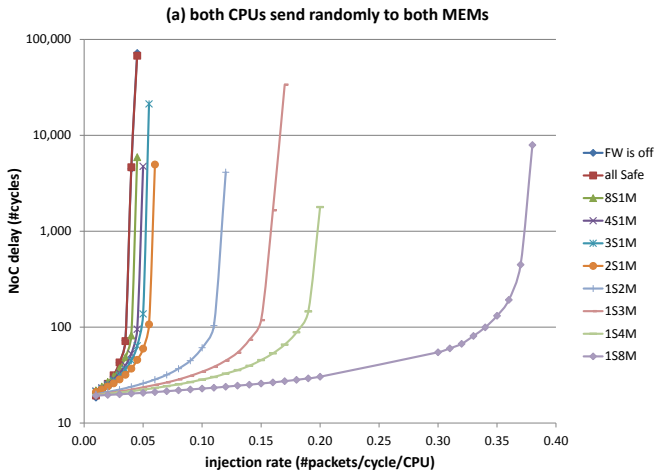

Fig. 3. NoC delay for different packet injection rates per cycle per CPU (scale is logarithmic), in a setup with 2 CPUs and 2 memories interconnected with STNoC, link-width=8 Bytes. Injection rate is the same across all CPUs.

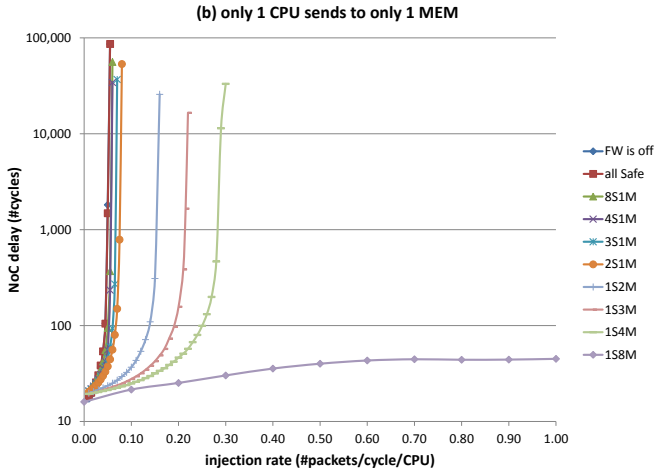

Fig. 4. NoC delay for different packet injection rates per cycle per CPU (scale is logarithmic), in a setup with 2 CPUs and 2 memories interconnected with STNoC, link-width=8 Bytes. Injection rate is the same across all CPUs.

Results of the above scenarios are shown in Figures 3, 4, 5, 6, 7 respectively. It appears that in all cases when the firewall is disabled the NoC becomes saturated for small values of the

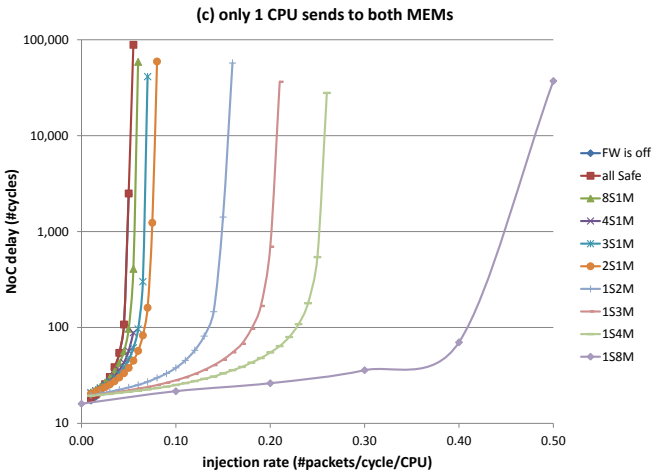

Fig. 5. NoC delay for different packet injection rates per cycle per CPU (scale is logarithmic), in a setup with 2 CPUs and 2 memories interconnected with STNoC, link-width=8 Bytes. Injection rate is the same across all CPUs.

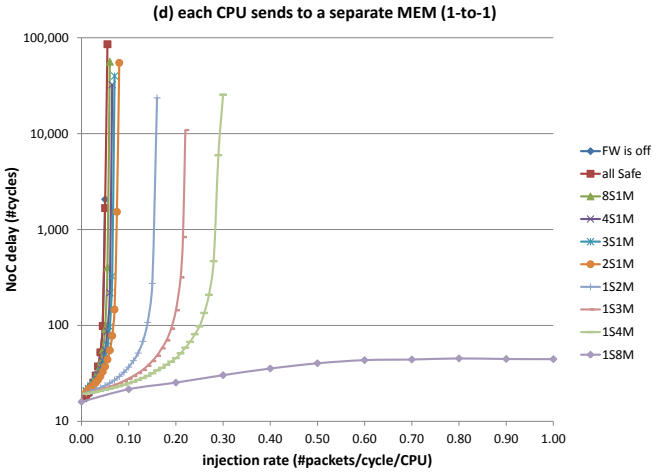

Fig. 6. NoC delay for different packet injection rates per cycle per CPU (scale is logarithmic), in a setup with 2 CPUs and 2 memories interconnected with STNoC, link-width=8 Bytes. Injection rate is the same across all CPUs.

injection rate, i.e. less than 0.05 $packets/cycle/CPU$. On the other hand, when the firewall is active, as the number of malicious requests increases over the safe ones, the injection rate that the system can serve without becoming saturated increases as well. We observe that in Figures 4 and 6, in the case of $1S8M$, the firewall prevents the system from becoming saturated. Finally, Figure 7 illustrates the scenario in which the network becomes saturated earlier than in the rest cases; this

TABLE III

RANGE OF PACKET INJECTION RATE - PER CLOCK CYCLE PER CPU - AT WHICH THE NOC STARTS BEING SATURATED. RESULTS CORRESPOND TO $2CPUs/2MEMs$ SETUP, WHEN BOTH SECURITY IS DISABLED, I.E. FW-OFF, AND ENABLED, I.E. ALL OTHER CASES.

mapping scenario	fw-off/all safe	8S1M	2S1M	1S2M	1S4M	1S8M
(a)	0.035-0.040	0.040-0.045	0.050-0.060	0.100-0.120	0.180-0.200	0.360-0.380
(b)	0.045-0.050	0.050-0.055	0.065-0.075	0.130-0.150	0.270-0.290	does not saturate
(c)	0.045-0.050	0.050-0.055	0.065-0.075	0.130-0.150	0.240-0.260	0.400-0.500
(d)	0.045-0.050	0.050-0.055	0.065-0.075	0.130-0.150	0.260-0.290	does not saturate
(e)	0.025-0.030	0.030-0.035	0.035-0.045	0.070-0.090	0.130-0.150	0.250-0.300

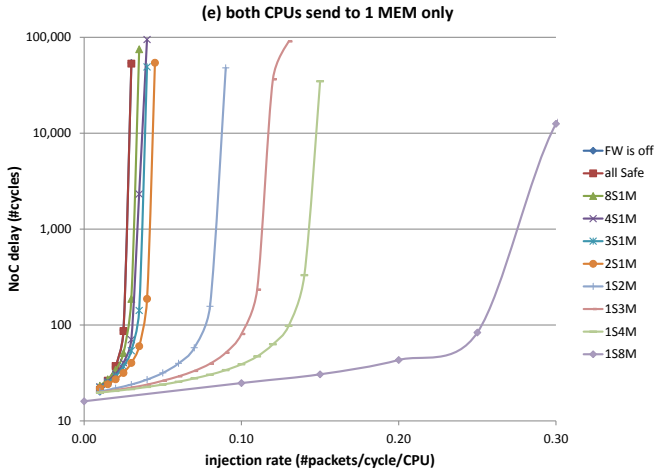

Fig. 7. NoC delay for different packet injection rates per cycle per CPU (scale is logarithmic), in a setup with 2 CPUs and 2 memories interconnected with STNoC, link-width=8 Bytes. Injection rate is the same across all CPUs.

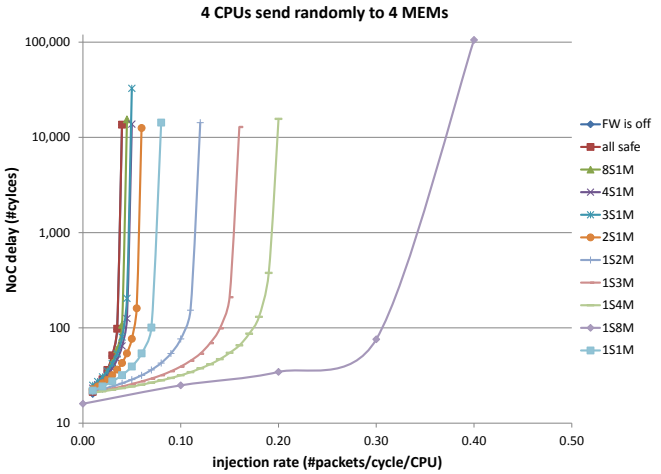

Fig. 8. NoC delay for different packet injection rates per cycle per CPU (scale is logarithmic), in a setup with 4 CPUs and 4 memories interconnected with STNoC, link-width=8 Bytes. Injection rate is the same across all CPUs.

is due to that both CPUs send requests to the same memory, which subsequently causes higher congestion as compared to the other cases. This is more clear in Table III, which shows the saturation-point in each scenario for different amount of legal and malicious requests per CPU. Obviously, enabling the firewall affects the NoC saturation-point value, i.e. the injection rate that can be served across the network increases; this is more clear when the number of malicious requests is much bigger than the safe ones.

We also performed some initial experiments with complex networks. These represent more realistic MPSoC devices. Figures 8 and 9 have the results for networks with 8 and 16 nodes respectively. The mapping scenario we ran is the one in which all CPUs raise requests to all memories in a random manner; this corresponds to mapping scenario (a). We observe the exact same behaviour with the experiments for the 4-node MPSoC when the firewall is active. As the number of malicious requests increases over the safe ones, the injection rate that the system can serve without becoming saturated, increases.

Saturation is related with the packet injection rate, and possibly with the incapability of the buffers at the network interfaces to serve it, while the routing infrastructure remains largely unaffected. While the packets are entering the network, the buffers are filled up, and the problem is propagated to the entire network, thus the physical links eventually become underutilized. When the link-width is set to 8 Bytes the system is not able to break down the packets in flits within a reasonable time, and consequently the network experiences long delays.

VI. CONCLUSIONS

We have analysed extensively the STNoC targeting mainly small-scale MPSoCs in order to expose saturation conditions for different injection rates, both when the firewall is on and off. We revealed the cases in which the NoC is in steady state, avoiding saturation. We then performed experiments of limited extend for networks with up to 16 nodes and we observed similar behaviour. We also showed the relation between delivery time and power consumption for setups with up to 32 nodes. In the near future we will perform experiments

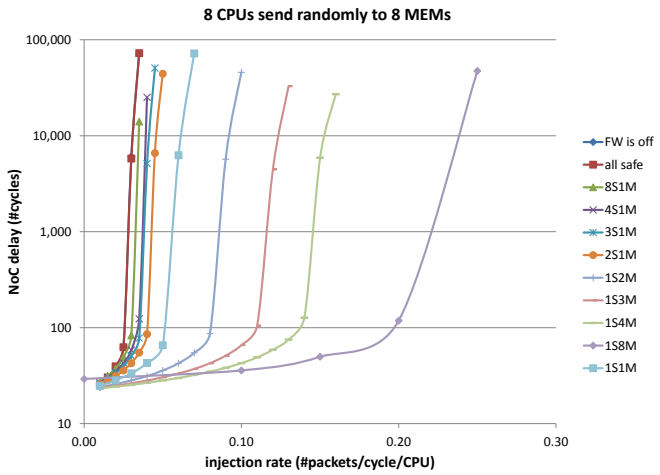

Fig. 9. NoC delay for different packet injection rates per cycle per CPU (scale is logarithmic), in a setup with 8 CPUs and 8 memories interconnected with STNoC, link-width=8 Bytes. Injection rate is the same across all CPUs.

for large devices with up to 64 nodes, and for wider links. This work is in progress, and our preliminary results demonstrate that extensive study can reveal the setups that do not lead the system to saturation conditions. This is useful in evaluating MPSoC edge devices serving high-demanding applications. Such examples are mpeg videos with real-time demands, and Internet of Things applications in which multiple devices such as sensors generate multiple streaming data processed by one or many NoC-based MPSoCs.

ACKNOWLEDGMENT

This work is supported by the EU through the FP7 project TRESCCA (GA No. 318036). The authors thank Antonis Papagrigoriou for modelling the firewall that was then implemented in hardware for measuring the actual time overhead.

REFERENCES

[1] M. D. Grammatikakis, K. Papadimitriou, P. Petrakis, A. Papagrigoriou, G. Kornaros, I. Christoforakis, and M. Coppola, "Security Effectiveness and a Hardware Firewall for MPSoCs," in *IEEE International Conference on High Performance Computing and Communications (HPCC)*, August 2014, pp. 1032–1039.

[2] M. D. Grammatikakis, K. Papadimitriou, P. Petrakis, A. Papagrigoriou, G. Kornaros, I. Christoforakis, O. Tomoutzoglou, G. Tsamis, and M. Coppola, "Security in MPSoCs: A NoC Firewall and an Evaluation Framework," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD) - Special Issue on Hardware Security and Trust*, vol. 34, no. 8, pp. 1344–1357, August 2015.

[3] G. Nychis, C. Fallin, T. Moscibroda, and O. Mutlu, "Next Generation On-Chip Networks: What Kind of Congestion Control Do We Need?" in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks (Hotnets)*, October 2010.

[4] M. Moadeli, A. Shahrabi, W. Vanderbauwhede, and M. Ould-Khaoua, "An Analytical Performance Model for the Spidergon NoC," in *21st IEEE Annual Conference on Advanced Networking and Applications (AINA)*, May 2007, pp. 1014–1021.

[5] A. E. Kiasari, Z. Lu, and A. Jantsch, "An Analytical Latency Model for Networks-on-Chip," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 21, no. 1, pp. 116–128, January 2013.

[6] Z. Qian, D.-C. Juan, P. Bogdan, C.-Y. Tsui, D. Marculescu, and R. Marculescu, "A Comprehensive and Accurate Latency Model for Network-on-Chip Performance Analysis," in *19th Asian and South Pacific Design Automation Conference (ASP-DAC)*, December 2014, pp. 323–328.

[7] Z. Qian, P. Bogdan, G. Wei, C.-Y. Tsui, and R. Marculescu, "A Traffic-aware Adaptive Routing Algorithm on a Highly Reconfigurable Network-on-Chip Architecture," in *International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, October 2012, pp. 161–170.

[8] L. Bononi and N. Concer, "Simulation and Analysis of Network on Chip Architectures: Ring, Spidergon and 2D Mesh," in *Design, Automation and Test in Europe (DATE)*, October 2006, pp. 154–159.

[9] M. LeMay and C. A. Gunter, ACM Computing Research Repository (CoRR), title = "Network-on-Chip Firewall: Countering Defective and Malicious System-on-Chip Hardware", month = April, year = 2014, Tech. Rep.

[10] J. Porquet, A. Greiner, and C. Schwarz, "NoC-MPU: a Secure Architecture for Flexible Co-hosting on Shared Memory MPSoCs," in *Proc. Design Automation and Test in Europe*, July 2011, pp. 591–594.

[11] G. Palermo, C. Silvano, G. Mariani, R. Locatelli, and M. Coppola, "Application-Specific Topology Design Customization for STNoC," in *Proc. 10th Euromicro Conference on Digital System Design Architectures, Methods and Tools (DSD)*, Aug 2007, pp. 547–550.

[12] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, and D. A. Wood, "The gem5 Simulator," *ACM SIGARCH Computer Architecture News*, vol. 39, no. 2, pp. 1–7, 2011.

[13] http://rt.com/news/biggest-ddos-us-cloudflare-557/.