Software-Defined Networking for Many-cores

Marcelo Ruaro, Kevin Martin, Fernando G Moraes

Software-Defined Networking for Many-cores

Marcelo Ruaro*†, Kevin J. M. Martin*, Fernando G. Moraes†
*Univ. Bretagne-Sud, UMR 6285, Lab-STICC, Lorient, France – marcelo.ruaro@univ-ubs.fr , kevin.martin@univ-ubs.fr
†PUCRS – School of Technology, Porto Alegre, Brazil – fernando.moraes@pucrs.br

***Abstract*—In the Software-Defined Networking (SDN) paradigm, routers are generic and programmable forwarding units that transmit packets according to a given policy defined by a software controller. Recent research has shown the potential of such a communication concept for Network on Chip (NoC) management, resulting in hardware complexity reduction, management flexibility, real-time guarantees, and self-awareness. This work briefly introduces the SDN concepts for many-cores and shows different studies addressing SDN, including a comparison between centralized and distributed SDN approaches, as well as the use of SDN to self-adaptive management of QoS and faults, and to provide secure application communication at runtime.**

I. Introduction

Software-Defined Networking (SDN) makes the NoC simple by removing the communication control logic from the hardware level (router), bringing it to the software level. An SDN Controller (Controller) manages NoC path definition, enabling the NoC to support new communication services, as QoS, power management, and fault-tolerance (**FT**), by implementing them as controller's software rules rather than on dedicated hardware designs. Figure 1(a) shows an example of a typical Processing Element (PE) architecture of a many-core [1]. The NoC implements a Multiple Physical Network (**MPN**) concept, with three disjoint subnets, and consequently, three routers to implement each subnet. One Packet-Switching (PS) router allows link sharing and is suitable for non-priority flows. SDN Routers (**SR**) implement Circuit-Switching (CS) [2], supporting dedicated paths configured by the controller, and making them suitable for real-time flows. In this sense, PS paths act like roads, allowing sharing best-effort packets, while CS paths (managed by SDN) act like rails, supporting timing constrained packets. The number of SRs is configurable. The SRs of MPN have a low design cost with one SR requiring, on average, 25% of the area and power of a typical PS router [3].

Figure 1(b) shows the steps to establish a dedicated path in an SDN-based many-core. The controller can either run in dedicated PE [4], or as a high priority task [3]. The controller abstracts the network management to a system manager (**M**), which performs application admission and task mapping. Due to such abstraction, when M needs a path between a source (**S**) and a target (**T**) communicating task, it performs a path request to the controller (**step 1**). The controller searches the path guided by the network status and its current management policies (**step 2**). The example in Figure 1(b) explores a theoretical scenario where the controller searches the shortest path between S and T, avoiding hot-spot areas (in red) and faulty routers (router 3x3). After finding a path, the controller physically configures the path (**step 3**) by sending a configuration packet to the respective SR. The configuration packet uses the PS subnet to reach the desired PE where the SR is located. The NI (Network Interface) of that PE receives the configuration packet by the SDN configuration logic (SCL), which handles the packet and configures the respective SR. The configuration ends when the controller configures all SR bellowing to the path. After the configuration, the controller sends an acknowledgment packet to M (**step 4**), which sets S and T to use the path (**step 5**) effectively.

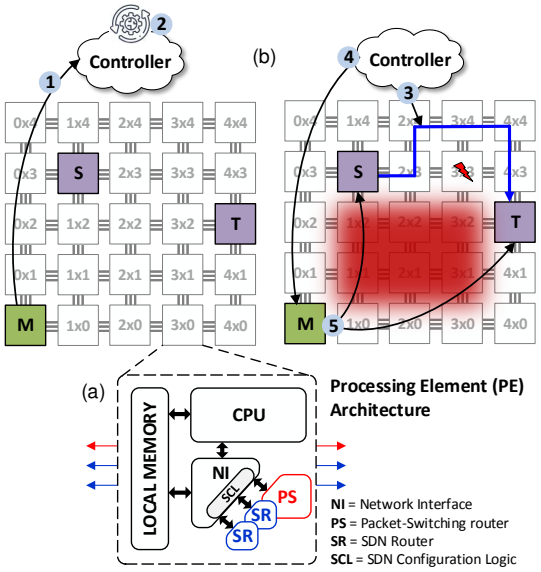

Fig. 1. (a) SDN-based communication in an MCSoC. (b) Overview of the Processing Element. Source: [1].

Table I presents related works on SDN for many-cores. There is a noticeable increasing academic interest and recognition of this subject. The lines filed in blue show the authors' contributions for this subject. The main motivation for investigating the SDN paradigm for on-chip communication comes from its advantages over traditional hardware-centered implementations [3], [5]–[7]:

Hardware complexity reduction: each SDN router implements a simple hardware logic used only to support the router's configuration and packet forwarding [3]. There is no need to implement the logic for routing and arbitration of packets or another kind of control like power, QoS, and FT. Less hardware means less area, power, and lower temperature, helping against the dark silicon effect;

Management flexibility: due to software implementation, the SDN allows for changing and updating policies that define the routing path without the necessity to design a new circuit [8];

Real-time guarantees: SDN combined MPNs, allows for a dedicated path for each real-time application flow, thus ensuring QoS through communication isolation [9];

Multi-objective management: SDN allows multi-objective management due to its panoramic overview of the NoC status. Different adaptive and management goals can be implemented and managed by the controller, which can decide to configure the NoC at runtime to fulfill real-time application needs of system budgets [10].

II. Distributed SDN

One of the big challenges of SDN is regarding the scalability to search and configure paths at runtime. In [14], the authors proposed a cluster-based approach, where one controller is in charge of a cluster of PEs instead the whole systems. Controllers work in parallel for local (intra-cluster) paths. For global (inter-cluster) paths, the controllers execute a synchronization protocol inspired by VLSI routing, with global and detailed routing phases.

TABLE I
STATE-OF-THE-ART WORKS OF SDN FOR MANY-CORES.

Work	Year	Organization	Objective	Model
[11]	2014	At router level	Architecture	Noxim
[12]	2016	Centralized	Architecture	Noxim
[3]	2017	Centralized	Architecture	RTL
[8]	2017	Centralized	Architecture	OMNET++
[13]	2017	Centralized	Architecture	RTL
[6]	2018	At PE level	Power	Mininet
[5]	2018	Centralized	QoS	RTL
[7]	2018	Centralized	Architecture	RTL
[9]	2019	Centralized	QoS	RTL
[14]	2019	Distributed	Scalability	RTL
[1]	2020	Distributed	Security	RTL
[15]	2020	Distributed	Security	RTL
[10]	2020	Distributed	Faults+QoS	RTL

Figure 2 evaluates the scalability of the approach, by varying the system size and the number of SDN routers (CS_n) inside a PE. Figure 2(a) presents the *total latency*: time to establish the maximum number (worst-case) of simultaneous paths into the system. The distributed SDN (D-SDN) values are normalized in relation to centralized (C-SDN). In summary, the C-SDN presents a better total latency for smaller systems, on average, lower than 256 PEs, while D-SDN outperforms C-SDN for the larger system sizes.

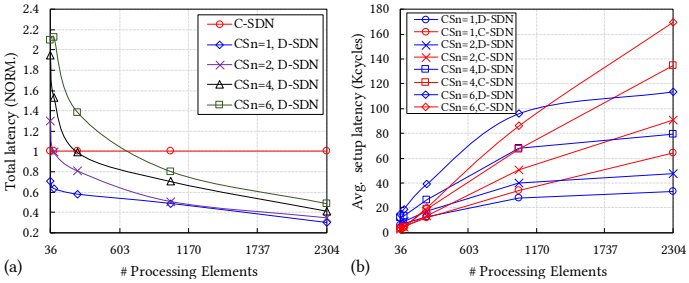

Fig. 2. Comparison between D-SDN and C-SDN.

Figure 2(b) presents the *average setup latency* (ASL) per path. For system sizes below 1024 PEs on average, it is possible to observe that the D-SDN presents a higher ASL. Note that the C-SDN ASL increases with the system size, due to the larger path search space. This result unveils that SDN can be adopted in larger systems by distributing the management role/load among controllers.

III. SDN FOR QOS AND FAULTS

This experiment evaluates the multi-objective adaptation using SDN, handling QoS and fault-tolerance simultaneously. Figure 3(a) shows an MPEG application mapping, where blue arrows represent the communication between MPEG tasks. Red arrows represent disturbing flows and the two red rays faulty CS routers. Figure 3(b) presents the MPEG iteration latency measured at the *output* task. The communication between tasks starts in PS mode. As there is no disturbing traffic, the NoC can meet latency threshold. At 800,000 *cc*, disturbing flows start, generating an increase in the iteration latency ($t1$ in Figure), and consequently, triggering a CS setup for the affected MPEG flows At 936,116 *cc*, the MPEG flows communicate using CS mode, restoring its QoS constraint.

At 1,800,000 *cc*, faults occur at CS routers in the cores executing $ivlc$ and $idct$ tasks, which makes the controller to switch the affected flows to PS mode. As the disturbing traffic is still active, the iteration latency increases, generating latency misses and making the controller to setup CS again ($t2$ in Figure). As the controller is aware about the faulty CS routers, it sets CS using different subnets than previously. After 1,984,991 *cc*, all affected flows of MPEG meet the QoS constraint latency by using CS.

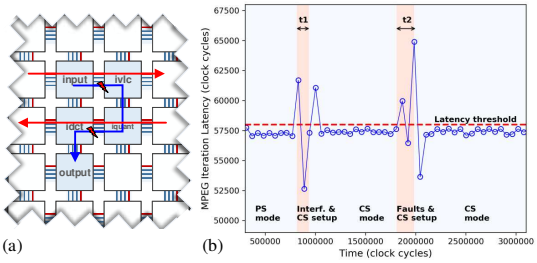

Fig. 3. (a) MPEG mapping with disturbing flows, and faulty CS routers; (b) MPEG iteration latency. Source: [10].

IV. SDN FOR SECURITY

Authors in [1] present a systemic and secure SDN framework (SDN-SS). The work describes the iteration between the hardware, operating system, and user's tasks to serve a data-sensitive application for dedicated and secure paths. The originality of SDN-SS includes (i) a step-by-step framework description addressing the phases required to support a secure SDN management; (ii) a secure SDN router configuration protocol; (iii) a protocol to change the subnet at runtime. Experimental results show the framework's capability to avoid DoS and spoofing attacks while presenting a low SDN router configuration overhead, corresponding up to 2% of a related work delay and a small impact over the user's task communication.

V. CONCLUSION

This paper presents an overview of the possibilities in exploiting SDN for many-cores. The SDN is a new open-topic and as demonstrated by results can contribute to reach a scalable and runtime multi-objective management at communication level.

REFERENCES

[1] M. Ruaro, L. L. Caimi, and F. G. Moraes, "A systemic and secure sdn framework for noc-based many-cores," *IEEE Access*, 2020.

[2] S. Liu, A. Jantsch, and Z. Lu, "MultiCS: Circuit switched NoC with multiple sub-networks and sub-channels," *Journal of Systems Architecture*, 2015.

[3] M. Ruaro, H. M. Medina, and F. G. Moraes, "Sdn-based circuit-switching for many-cores," in *ISVLSI*, 2017.

[4] S. Ellinidou, G. Sharma, T. Rigas, T. Vanspouwen, O. Markowitch, and J. Dricot, "Sspsoc: A secure sdn-based protocol over mpsoc," *Security and Communication Networks*, 2019.

[5] A. Kostrzewa, S. Tobuschat, and R. Ernst, "Self-aware network-on-chip control in real-time systems," *IEEE Design Test*, 2018.

[6] A. Scionti, S. Mazumdar, and A. Portero, "Towards a Scalable Software Defined Network-on-Chip for Next Generation Cloud," *Sensors*, 2018.

[7] M. Ruaro, H. Medina, A. Amory, and F. Moraes, "Software-Defined Networking Architecture for NoC-based Many-Cores," in *ISCAS*, 2018.

[8] K. Berestizshevsky, G. Even, Y. Fais, and J. Ostrometzky, "Sdnoc: Software defined network on a chip," *Microprocessors and Microsystems*, 2017.

[9] M. Ruaro, A. Jantsch, and F. G. Moraes, "Self-Adaptive QoS Management of Computation and Communication Resources in Many-Cores SoCs," *ACM Transaction on Embedded Computing Systems*, 2018.

[10] M. Ruaro and F. G. Moraes, "Multiple-objective management based on a distributed sdn architecture for many-cores," in *SBCCI*, 2020.

[11] L. Cong, W. Wen, and W. Zhiying, "A configurable, programmable and software-defined network on chip," in *WARTIA*, 2014.

[12] R. Sandoval-Arechiga, R. Parra-Michel, J. L. Vazquez-Avila, J. Flores-Troncoso, and S. Ibarra-Delgado, "Software Defined Networks-on-Chip for multi/many-core systems: A performance evaluation," in *ANCS*, 2016.

[13] A. Fathi and K. Kia, "A Centralized Controller as an Approach in Designing NoC," *International Journal of Modern Education and Computer Science*, 2017.

[14] M. Ruaro, N. Velloso, A. Jantsch, and F. G. Moraes, "Distributed sdn architecture for noc-based many-core socs," in *NOCS*, 2019.

[15] M. Ruaro, L. L. Caimi, and F. G. Moraes, "Sdn-based secure application admission and execution for many-cores," *IEEE Access*, 2020.