TAL TECH

**DOCTORAL THESIS**

# Bandwidth Reduction DoS Attacks in Multi-Tenant NoC-based MPSoCs: Detection and Avoidance Strategies

Cesar Giovanni Chaves Arroyave

TALLINNA TEHNIKAÜLIKOOL
TALLINN UNIVERSITY OF TECHNOLOGY
TALLINN 2023

TALLINN UNIVERSITY OF TECHNOLOGY
DOCTORAL THESIS
8/2023

# Bandwidth Reduction DoS Attacks in Multi-Tenant NoC-based MPSoCs: Detection and Avoidance Strategies

CESAR G. CHAVES A.

TAL TECH PRESS

TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies
Department of Computer Systems

**The dissertation was accepted for the defence of the degree of Doctor of Philosophy in Computer and Systems Engineering on 13 March 2023**

**Supervisor:** Prof. Dr.-Ing. Thomas Hollstein,
Department of Computer Systems, School of Information Technologies,
Tallinn University of Technology,
Tallinn, Estonia
Faculty 2: Computer Science and Engineering,
Frankfurt University of Applied Sciences,
Frankfurt am Main, Germany

**Co-supervisor:** Dr. Johanna Sepúlveda,
Airbus Defence and Space,
Ottobrunn, Germany

**Opponents:** Dr. Ir. Poona Bahrebar,
Universiteit Gent,
Ghent, Belgium

Prof. Dr.-Ing. Thilo Pionteck,
Otto von Guericke Universität Magdeburg,
Magdeburg, Germany

**Defence of the thesis:** 31 March 2023, Tallinn

**Declaration:**
*Hereby I declare that this doctoral thesis, my original investigation and achievement, submitted for the doctoral degree at Tallinn University of Technology, has not been submitted for any academic degree elsewhere.*

Cesar G. Chaves A. _____
signature



Deutscher Akademischer Austauschdienst
German Academic Exchange Service


ISSN 2585–6898 (publication)
ISBN 978-9949-83-960-5 (publication)
ISSN 2585–6901 (PDF)
ISBN 978-9949-83-961-2 (PDF)
Printed by Koopia Niini & Rauam

TALLINNA TEHNIKAÜLIKOOL
DOKTORITÖÖ
8/2023

# Ribalaiuse vähendamise DoS-rünnakud mitme rentnikuga NoC-põhiste MPSoC-de puhul: tuvastamise ja vältimise strateegiad

CESAR G. CHAVES A.

TAL TECH
KIRJASTUS

To my family...

# Contents

# List of Publications

The present Ph.D. thesis is based on the following publications that are referred to in the text by Roman numbers.

I Cesar G. Chaves, Siavoosh P. Azad, Thomas Hollstein, and Johanna Sepúlveda, "A Distributed DoS Detection Scheme for NoC-based MPSoCs," in *Circuits and Systems (NorCAS), 2018 IEEE Nordic Conference on*, IEEE, October 2018

II Cesar G. Chaves, Siavoosh P. Azad, Thomas Hollstein, and Johanna Sepúlveda, "DoS Attack Detection and Path Collision Localization in NoC-Based MPSoC Architectures," *Journal of Low Power Electronics and Applications*, vol. 9, no. 1, 2019

III Cesar G. Chaves, Siavoosh P. Azad, Johanna Sepúlveda, and Thomas Hollstein, "Detecting and Mitigating Low-and-Slow DoS Attacks in NoC-based MPSoCs," in *14th International Workshop on Reconfigurable Communication-Centric Systems-on-Chip (ReCoSoC)*, IEEE, July 2019

IV Cesar G. Chaves, Johanna Sepúlveda, and Thomas Hollstein, "Lightweight Monitoring Scheme for Flooding DoS Attack Detection in Multi-Tenant MPSoCs," in *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5, May 2021

## Other related publications (Workshops)

V Cesar G. Chaves, Siavoosh P. Azad, Thomas Hollstein, and Johanna Sepúlveda, "Diagnosing DoS Attacks in NoC-based MPSoCs," in *Testmethoden und Zuverlässigkeit von Schaltungen und Systemen (TuZ 2019)*, February 2019

VI Cesar G. Chaves, Johanna Sepúlveda, and Thomas Hollstein, "Lightweight Packet Monitoring for Flooding DoS Attack Detection in NoC-based MPSoCs," in *Testmethoden und Zuverlässigkeit von Schaltungen und Systemen (TuZ 2021)*, February 2021

## Author's Contributions to the Publications

I In Publication I, as the main author, designed and implemented a distributed DoS detection scheme for NoC-based MPSoCs measuring end-to-end communication delays. Also proposed and developed a router architecture that enables the localization of the point where malicious traffic disrupts a monitored path. Additionally, carried out the simulations and the analysis of the results. Finally, wrote the manuscript.

II In Publication II, as the main author, enhanced the distributed DoS detection scheme for NoC-based MPSoCs presented in Publication I so that, apart from the router in which malicious traffic disrupts the monitored path, it also reported the input and output ports used by the malicious packets. By doing so, the attack source or a list of possible attack sources can be determined, which helps when attempting to mitigate DoS attacks. Additionally carried out the simulations and the analysis of the results. Finally, wrote the manuscript.

III In Publication III, as the main author, proposed and implemented two DoS attacks that had not been studied before in the context of NoC-based MPSoCs. Furthermore, proposed and implemented a mechanism able to mitigate both presented attacks. Additionally carried out the simulations and the analysis of the results. Finally, wrote the manuscript.

IV In Publication IV, as the main author, proposed and implemented a router architecture that enables packets to report the source of traffic that disrupted their communication the most. Additionally carried out the simulations and the analysis of the results, showing that the most commonly researched DoS attack is not so effective when using a fair packet arbitration. Finally, wrote the manuscript.

### Author's contributions to other related publications (Workshops)

V In Publication V, as the main author, proposed and implemented the presented distributed DoS detection scheme for NoC-based MPSoCs. Additionally proposed and developed the identification of the router in where malicious traffic disrupts a monitored path. Additionally carried out the simulations and the analysis of the results. Finally, wrote the manuscript.

VI In Publication VI, as the main author, proposed and implemented the presented distributed DoS detection scheme for NoC-based MPSoCs, which enables packets to report the execution location of applications reducing the NoC's bandwidth the most. Additionally carried out the simulations and the analysis of the results. Finally, wrote the manuscript.

## Abbreviations

| | |
|---|---|
| 2D | Two-Dimensional |
| ADAD | Active DoS Attack Detection |
| APCG | Application Characterization Graph |
| ASIC | Application-Specific Integrated Circuit |
| BCP | Bandwidth Control Policies |
| BCPE | Bandwidth Control Policies Enforcement |
| BRDoS | Bandwidth Reduction Denial of Service |
| BWT | Buffer Waiting Time |
| CD | Collision Degree |
| CDT | Communication Disruption Tattletaling |
| CPDD | Collision Point Direction Detection |
| CPR | Collision Point Report |
| CPRD | Collision Point Router Detection |
| CR | Collision Router |
| CTG | Communication Task Graph |
| CTP | Communication Testing Packets |
| CW | Collision Wait |
| DLC | Destination Latency Curves |
| DoA | Degree of Adaptivity |
| DoS | Denial of Service |
| E2E | End-to-End |
| EPI | European Processor Initiative |
| FDoS | Flooding Denial of Service |
| FIFO | First-In-First-Out |
| FIR | Flit Injection Rate |
| Flit | Flow control unit |
| FSM | Finite State Machine |
| GM | Global Manager |
| HT | Hardware Trojan |
| IFD | Inter-Flit Delay |
| IFI | Inter-Flit Interval |
| IP | Intellectual Property |
| $IP_A$ | Input Port of Attacker |
| $IP_C$ | Input Ports of Competitors |
| IPD | Inter-Packet Delay |
| IPT | Incomplete Packet Transmission |
| JFDV | Jellyfish Delay Variance |
| JFIDV | Jellyfish Inter-flit Delay Variance |
| LBDR | Logic-Based Distributed Routing |
| LIGO | Laser Interferometer Gravitational-Wave Observatory |
| LSDAA | Low-and-Slow DoS Attack Avoidance |
| LSDoS | Low-and-Slow Denial of Service |
| LSM | Local Security Manager |
| MPSoC | Multi-Processor System-on-Chip |
| MTCaaS | Multi-tenant Computing as a Service |
| NEWSL | North - East - West - South - Local |
| NI | Network Interface |
| NoC | Network-on-Chip |

| | |
|---|---|
| $OP_A$ | Output Port of Attacker |
| OCP | Open Core Protocol |
| PAC | Packet Arrival Curves |
| PE | Processing Element |
| PIR | Packet Injection Rate |
| PMF | Packet Missing Flits |
| PPL | Packet Payload Length |
| PAS | Possible Attack Sources |
| RA | Routing Algorithm |
| RR | Round Robin |
| RTS | Request to Send |
| RTL | Register Transfer Level |
| $S_A$ | Source of Attack |
| SEM | Structural Equation Modeling |
| SLA | Service Level Agreement |
| SMC | Squared Multiple Correlation |
| SSD | Sample Standard Deviation |
| SNN | Spiking Neural Network |
| SoC | System-on-Chip |
| VCO | Virtual Channel Occupancy |

# 1 Introduction

Aiming to tackle more complex problems, achieve higher performance levels, and/or reduce power consumption, Application-Specific Integrated Circuits (ASICs) have evolved into Multi-Processor Systems-on-Chip (MPSoCs), incorporating several Processing Elements (PEs) or Intellectual Property (IP) cores from different providers connected by a Network-on-chip (NoC), which are able to execute applications from different concurrent users. Multi-Tenant Computing-as-a-Service (MTCaaS) provides the basis for integrated cloud/fog systems [7, 8]. Moreover, MPSoCs with a heterogeneous array of PEs provide programmability and parallelism, yielding flexibility, processing performance, and power efficiency, which can be leveraged for minimizing the latency of communication links of the edge cloud [9]. Therefore, from a performance point of view, the incorporation of MPSoCs is beneficial for MTCaaS, however, from a security perspective, making MPSoCs available to a large number of diverse users exposes them to attacks that have not yet been thoroughly researched.

On the one hand, hardware security authors have presented the viability of inserting hardware trojans into IP cores for attempting Side-channel attacks that discover secret information of a system [10], as well as mechanisms for detecting such trojans [11]. On the other hand, hardware trojans and attacks originating at the NoC have also been researched [12]. Sarihi et. al. in [13] classify NoC security threats as attacks to confidentiality, integrity, and availability, a.k.a., Denial of Service (DoS) attacks. With the introduction of MPSoCs into MTCaaS, DoS attacks on MPSoCs can now also be originated by software, where malicious users submit applications that attempt to overwhelm the shared resources, forcing their service to be denied to tasks being run by concurrent tenants of the same MPSoC.

## 1.1 Motivation

In order to provide higher quality and security service levels in Multi-Tenant Computing-as-a-Service (MTCaaS), the execution of an application submitted by one of its tenants should not be impacted by the others.

In order to solve this problem when adding NoC-based MPSoCs to MTCaaS, it is important to identify: i) the DoS attacks that could be implemented by an application running on one of the MPSoC's processing elements (PEs) to disrupt the execution of applications from other tenants; ii) directives that a designer of multi-tenant NoC-based MPSoCs can follow to avoid, detect, localize, and/or mitigate the identified DoS attacks; and iii) how those attacks can be generated so that their impact be assessed.

This thesis provides a classification of DoS attacks that target the communication within MPSoCs, where Bandwidth Reduction DoS attacks were identified as one of the most common and powerful techniques to attack NoC-based MPSoCs. They aim at overwhelming the on-chip communication structure so that the execution of concurrent applications be delayed or stopped. Additionally, this document summarizes all the approaches taken by other authors to tackle DoS attacks in NoC-based MPSoCs and presents new proposals that go beyond the state of the art. Finally, based on the characteristics of the identified attacks, a model for their generation and for testing the resilience of different MPSoC designs is detailed.

## 1.2 Problem formulation

The focus of this Ph.D. thesis has been given to answer the following research questions:

RQ1 In the context of multi-tenant NoC-based MPSoCs, which DoS attacks could be implemented by an application running on one of its PEs to disrupt the execution of applications from other tenants?

RQ2 What directives can a designer of multi-tenant NoC-based MPSoCs follow to avoid, detect, locate, and/or mitigate the identified DoS attacks?

RQ3 How can the identified DoS attacks be generated and their impact on a given multi-tenant NoC-based MPSoC assessed?

## 1.3 Contributions of the thesis

The main contributions of the thesis are:

- As an answer to research question RQ1, a taxonomy of DoS attacks in the context of Multi-tenant NoC-based MPSoCs, is presented in Section 3. Despite related work focusing only on Flooding DoS attacks in the form of Packet Injection Rate (PIR)-based DoS attacks and some considering also Packet Payload Length (PPL)-based DoS attacks, this thesis additionally details two Low-and-slow DoS attacks, namely Jellyfish and Slowloris, which despite being from computer networks, they can also be implemented in the context of this thesis.
- As an answer to research question RQ2, four approaches were developed for DoS attack detection and two for DoS attack avoidance. Former approaches went beyond the state of the art by not focusing merely on PIR-based DoS attacks and the latter approaches are able to avoid the four identified Bandwidth Reduction DoS (BRDoS) attacks. Such approaches are listed below:
  - Collision Point Router Detection (CPRD): Based on the communication performance of the NoC, it identifies the presence of flooding DoS attacks, as well as the router where malicious traffic disrupts legitimate traffic (Section 6.1, Publication I, and Publication V).
  - Collision Point Direction Detection (CPDD): Extends CPRD, to detect the exact source of the attack in some cases and for others provides a list of possible attack suspects (Section 6.2 and Publication II).
  - Active DoS Attack Detection (ADAD): Extends CPDD enabling the location of the exact source of attack for the cases that CPDD cannot (Section 6.3).
  - Communication Disruption Tattletaling (CDT): Achieves the detection of the exact source of attack for PPL-based DoS attacks and relies on Round-robin arbitration for avoiding PIR-based DoS attacks (Section 6.4, Publication VI, and Publication IV).
  - Low-and-slow Dos Attack Avoidance (LSDAA): Eliminates the vulnerability of NoC-based MPSoCs to Jellyfish and Slowloris Low-and-slow Dos Attacks (Section 7.1, Publication III).
  - Bandwidth Control Policies Enforcement (BCPE): A first variant enables the Global Manager (GM) of an NoC-based MPSoC to prevent Bandwidth Reduction DoS attacks. On top of that, a second variant also reports to the GM any attempt at policy violation so that a new Service Level Agreement can be established (Section 7.2).

- As an answer to research question RQ3:
  - An algorithm for generating Bandwidth Reduction DoS attacks is presented. It is explained how its parameters can be set for generating not only the attacks found in literature, which are focusing mostly on PIR-based attacks only, but also PPL-based attacks as well as slow-loris and Jellyfish type attacks. Additionally, parameters also allow the generation of combined attacks (Section 9.2).
  - A methodology for assessing the impact of DoS attacks on any given NoC-based MPSoC is presented in Chapter 9.

## 1.4 Thesis Organization

The rest of the thesis is organized as follows: Chapter 2 provides definitions and explanations of the terms used throughout this document. Such topics include NoC-based MPSoC, Network on Chip, Project Bonfire, application workflows, and task scheduling. Chapter 3 answers research question RQ1, covering DoS attacks in NoC-based MPSoCs. Chapter 4 presents the targeted threat model, including the attack scenario, steps for executing the attack, and the attack's success conditions. Chapter 5 explains the process followed for answering the research questions RQ1 and RQ2. It summarizes the state of the art regarding Bandwidth Reduction DoS attacks in NoC-based MPSoCs while classifying related work as DoS attack avoidance or DoS attack detection. Additionally, it explains how the approaches presented in this thesis go beyond the state of the art. Chapter 6 and Chapter 7 detail the contributions made during the Ph.D. for answering the research question RQ2, through DoS attack detection and DoS attack avoidance, respectively. Chapter 8 describes how the MPSoC's software and hardware come together, eliminating the vulnerability to Bandwidth Reduction DoS attacks. Chapter 9 answers the research question RQ3. Finally, Chapter 10 concludes the thesis.

# 2 Background

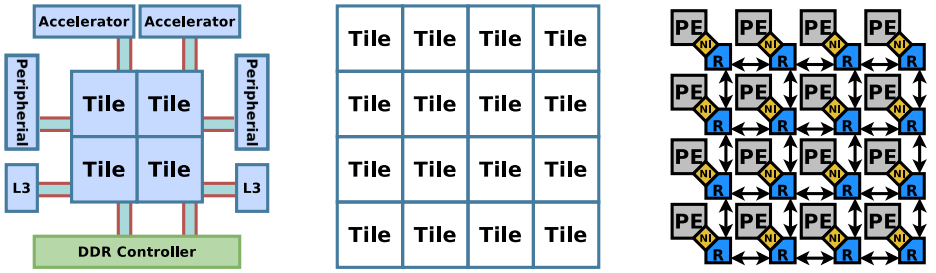
## 2.1 NoC-based MPSoCs

Multi-processors Systems-on-Chip (MPSoCs) are integrated circuits that contain a complete computational system on a single die. They are composed of two main structural types of components:

(i) the computational structure, consisting of Processing Elements (PEs) such as processors, hardware accelerators, memories, peripherals, and other Intellectual Property (IP) hardware cores to process and store information; and

(ii) the MPSoC internal communication structure, performing data exchange between PEs. Depending on the design, it is implemented as point-to-point connections, buses, crossbars, and/or Networks-on-Chip.

Three common diagrams used for representing NoC-based MPSoCs including a NoC with mesh topology are depicted in Figure 1. In Figure 1a, a cluster of tiles in the center connected to other IP cores is shown [14, 15]. Tiles include a NoC router and usually one or more processing cores with L1/L2 cache memory, while the other IPs are peripherals, hardware accelerators, cryptography cores, and memory controllers, among others. NoC-based MPSoCs are also sometimes depicted as an array on tiles as shown in Figure 1b [16–22]. As in the previous diagram, tiles contain a NoC router but now may also include any type of PE. Another option used for representing NoC-based MPSoCs is as an array of PEs which, by the use of network interfaces (NIs), connect to the NoC, as illustrated by Figure 1c [23–28]. This last option was selected for this document because it shows elements separately.




(a) Tiles + Intellectual Property (b) Array of Tiles (c) Connected Processing Elements

*Figure 1: MPSoC Diagrams*

Furthermore, examples of commercial NoC-based MPSoCs include the knights landing Intel Xeon Phi processor [29], IBM's Power9 processor [30], Marvell's ThunderX2 [31], and the Rhea exascale processor [32] from the European Processor Initiative (EPI) [33].

### 2.1.1 Network Interface

In NoC-based MPSoCs, NIs are in charge of encapsulating data into packets at the source of a transmission, as well as, decapsulating it at its destination. Such packets are composed of flow control units (Flits), which are the number of packet bits that are transmitted together on a single clock cycle. This amount is defined at design time, as well as the packet structure that will be understandable by the routers in the NoC.

An illustrative example of the NI's operation is presented in Figure 2. For example, PEs are considered to communicate using 32-bit words. One format that can be implemented for a PE to send data to an NI is depicted in Figure 2a. Data transmission is requested by providing the address of the desired destination and the payload length of the data to be transmitted. Subsequent valid flits will then carry the payload. Moreover, the destination address for each operation would be either provided when the application task was mapped for execution, or by an operation translation made by the firmware of the PE. Figure 2b shows a data structure after being encapsulated by the NI into a packet, a structure that is forwarded through routers until reaching an NI at the destination. The most significant bits help the identification of the flit type while the less significant bit enables single error identification. A *Header Flit* signals the beginning of a transmission and contains the information used for taking routing decisions. Such information corresponds at least to the destination address, but can also include other information relevant to routers or even the NI at the destination. A *Packet Info Flit* (a.k.a. the first body flit) contains an ID for reordering packets at the destination, as well as the packet payload length. One or more *Payload Body Flits* follow, carrying the data exchanged between PEs. Finally, a *Tail Flit* signals the end of the packet. In many designs, the *Tail Flit* carries the last part of the payload, however, it may carry any other required information. A structure for decapsulated data given from an NI to a PE at the destination is presented in Figure 2c. The first flit contains the source address, packet ID, and the packet's payload length. Subsequent flits carry the payload.

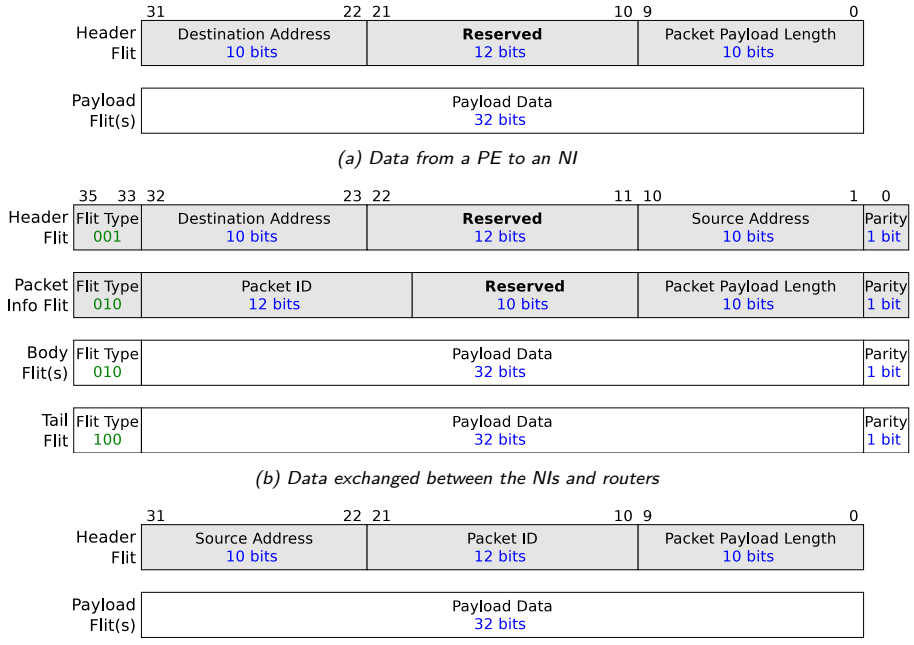


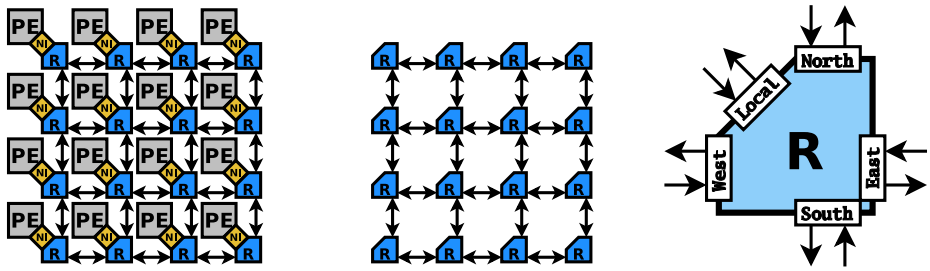

*Figure 2: Example of Data Structures*

### 2.1.2 Global Manager

In MPSoCs a central Global Manager (GM), is a piece of software that can be mapped to run on one of the PEs. Such a manager provides essential services as task scheduling [34–39], detailed assessment of the system's performance, fault detection, and instrument management as well as maintains health and resource maps [40, 41].

## 2.2 Network-on-Chip

Networks-on-Chip (NoCs) are used as the communication structure of MPSoCs that integrate a high number of PEs. Moreover, in order to achieve optimized pipelined data transmission characteristics, a mesh-based NoC topology is typically chosen. An MPSoC where a 2D mesh-based NoC was selected as its communication structure is illustrated by Figure 3a. In such an MPSoC, each PE is connected to one of the routers. Moreover, the 2D mesh-based NoC, which transmits data via routers and communication links, is depicted in Figure 3b. Furthermore, 2D mesh topologies are composed of a set of routers with a number of ports normally varying from three to five (Local, North, East, South, West) as shown in Figure 3c, however, other MPSoC topologies may connect more than one PE per router, adding more ports.




*(a) NoC-based MPSoC* *(b) NoC* *(c) Router*

*Figure 3: From NoC-based MPSoC to Router*

### 2.2.1 Router Components

#### 2.2.1.1 Input Buffers

In many router implementations, buffers store flits at the input ports until they can be forwarded, *i.e.* until the required output port becomes available. However, initial flits of a packet need to wait also while the routing decision is being made. Once the output port has been allocated and is ready, flits leave the buffer following a first-in-first-out (FIFO) methodology.

#### 2.2.1.2 Routing Module

Upon the arrival of a Header flit, the routing module determines the output port or ports through which the packet can be forwarded to continue the path toward its destination. Such a decision is taken based on a routing table, a preconfigured routing algorithm, and/or as established by the source in case source routing is being used. Once a routing decision has been made, a request for the selected output port is sent to the allocator which will grant access if/when such a port is available.

#### 2.2.1.3 Allocator

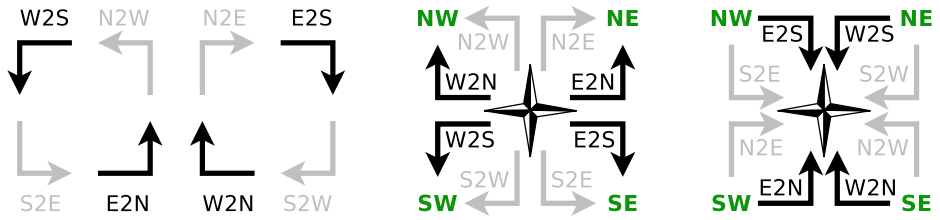
The Allocator is in charge of granting access to output ports. It selects which of the input ports requesting output access should be allowed to forward its packets through the requested output port. In cases where an input port requests more than one output, the allocator will also decide which request to consider.

#### 2.2.1.4 Crossbar

The Crossbar or Xbar is composed of multiplexers, one per output port. Each multiplexer connects all of the input ports to a single output port and it is controlled by the Allocator.

### 2.2.2 Routing Algorithms and Turn Models

Routing algorithms establish the output ports through which packets can be forwarded to reach their destinations. They can be classified as deterministic or adaptive, the former provided that packets have a single route option from source to destination, while the latter is otherwise. Turn models as the ones illustrated by Figures 4, 5, 6, and 7 can be used for describing routing algorithms. In 2D mesh, eight turns can occur; N2E, N2W, E2N, E2S, W2N, W2S, S2E, and S2W. A turn labeled as N2E is understood as the one made by a packet going North that turns East. A turn model presents which of the eight possible turns are allowed or disabled by the described routing algorithm. In this manuscript, the permitted and prohibited turns are marked as black and gray, respectively. Nevertheless, in other documents, they might be drawn as solid and dashed lines. Furthermore, the number of turns allowed by a routing algorithm is proportional to the adaptiveness degree of routing algorithms as explained in [42]. Moreover, the models in the subfigures labeled *(a)* are commonly used and show possible NoC paths in a way that eventual deadlocks can be identified. However, the compass rose models in subfigures labeled *(b)* are useful for taking routing decisions, showing if a packet is able to reach its destination after being routed through a specific output port. Finally, given a specific router, the compass rose models in subfigures labeled *(c)* illustrate the turns that packets from a specific quadrant can take to reach it.

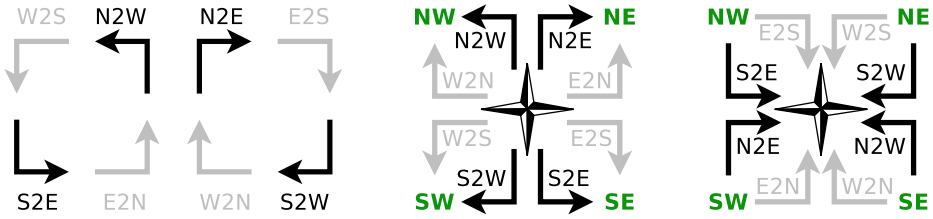



*(a) Common Representation* *(b) Router to Quadrant* *(c) Quadrant to Router*

*Figure 4: Turn Model Representations for XY Routing Algorithm*

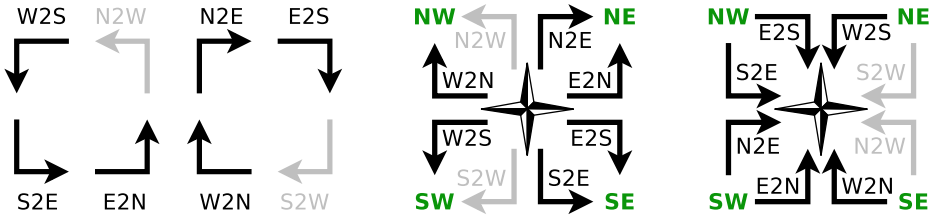
It is worth mentioning that paths taken by packets going in any direction, when entering a router, continuing in the same direction are not considered as turns but as straight. Additionally, straight paths are not restricted by routing algorithms but by the topology of the NoC. Also, packets requesting to be forwarded through the Local output port are also not restricted by routing algorithms.

Furthermore, as mentioned in [43], a routing algorithm for mesh NoCs must provide full connectivity and ensure deadlock freeness. The former means being able to route packets from any PE to any other PE in the MPSoC, while the latter refers to avoiding cyclic dependencies between packets that prevent them from reaching their destination.



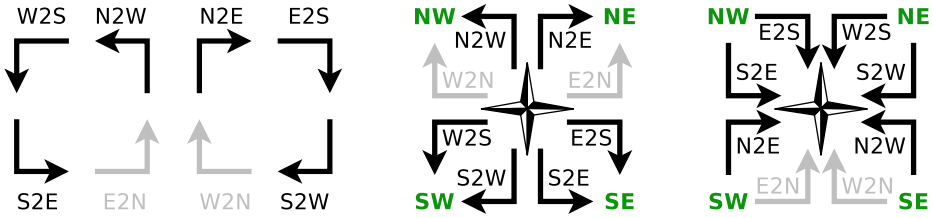

*(a) Common Representation* *(b) Router to Quadrant* *(c) Quadrant to Router*

*Figure 5: Turn Model Representations for YX Routing Algorithm*




*(a) Common Representation* *(b) Router to Quadrant* *(c) Quadrant to Router*

*Figure 6: Turn Model Representations for West-first Routing Algorithm*

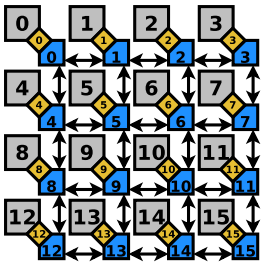



*(a) Common Representation* *(b) Router to Quadrant* *(c) Quadrant to Router*

*Figure 7: Turn Model Representations for North-first Routing Algorithm*

### 2.2.3 Logic Based Distributed Routing

Logic-based distributed routing (LBDR) is a mechanism for taking routing decisions presented in [44]. It relies on a distributed mechanism in which routing decisions are taken on each hop of the packet's path. They are based on the coordinates of the current and destination routers, as well as on two registers, namely $C_x$ and $R_{xy}$. Equivalence between the MPSoC representation and tiles with coordinates is presented in Figure 8, where a coordinate is composed of the row and column where a tile/PE is located. Moreover, within a router, register $C_x$ denotes whether its output ports are connected or not to other routers. Each of its four bits represents an output port according to $C_x : [North\ East\ West\ South]$, where the North output port is represented by the most significant bit. On the other hand, $R_{xy}$ stores the configured routing algorithm in the form of a turn model. Each of its eight bits denotes if a turn is allowed or not as follows: $R_{xy} : [N2E\ N2W\ E2N\ E2W\ W2N\ W2S\ S2E\ S2W]$, where N2E is represented by the most significant bit. The value of $R_{xy}$ for different routing algorithms and their turn model in a compass rose representation is illustrated in Figure 9. It is worth noticing that routing algorithms based on turn models do not restrict straight paths, However, it can be done by setting register $C_x$ accordingly.

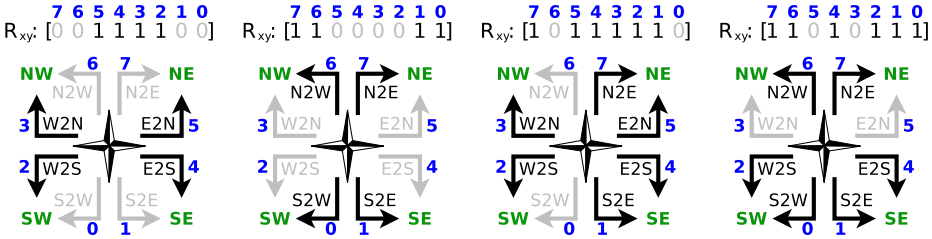



(a) MPSoC - Elements labeled with IDs

| | | | |
|---|---|---|---|
| 0 (0,0) | 1 (0,1) | 2 (0,2) | 3 (0,3) |
| 4 (1,0) | 5 (1,1) | 6 (1,2) | 7 (1,3) |
| 8 (2,0) | 9 (2,1) | 10 (2,2) | 11 (2,3) |
| 12 (3,0) | 13 (3,1) | 14 (3,2) | 15 (3,3) |

(b) MPSoC - Tiles labeled with coordinates $(y, x)$

*Figure 8: MPSoC IDs and Coordinates*




(a) XY routing (b) YX routing (c) West-first routing (d) North-first routing

*Figure 9: NoC routing algorithms - compass rose representation*

Inside each of the NoC routers, routing decisions of packets are taken based on three sets of logic formulas, listed from (1) to (12). The first set, from (1) to (4), is for answering the question: "In which direction is the destination located?". For the 2D-mesh topology and aiming to reach the destination with the less possible number of hops, such a question has three possible answers: i) the packet is already at its destination, given that non of the equations return a true value; ii) the destination is straight ahead in the direction with the true value, provided that only one of the equations is true; and iii) the destination is in one of the quadrants relative to the current router, which implies that the packet could potentially be forwarded through two possible output ports if two of the equations return true values.

$$N' = \quad Y_{dst} > Y_{cur} \tag{1}$$
$$E' = \quad X_{dst} > X_{cur} \tag{2}$$
$$W' = \quad X_{dst} < X_{cur} \tag{3}$$
$$S' = \quad Y_{dst} < Y_{cur} \tag{4}$$

For the cases where the destination has been identified to be in a quadrant and two ports have been selected as possible outputs, the second set of equations, from (5) to (8), is for answering the question: "If the packet is forwarded through the output ports determined in the previews step, will the routing algorithm allow it to make the required turn towards its destination?". If the implemented algorithm assures the connectivity of all the PEs, at least one of the two ports will still obtain a true value.

$$N'' = N' \cdot \overline{E'} \cdot \overline{W'} + N' \cdot E' \cdot R_{ne} + N' \cdot W' \cdot R_{nw} \quad (5)$$

$$E'' = E' \cdot \overline{N'} \cdot \overline{S'} + E' \cdot N' \cdot R_{en} + E' \cdot S' \cdot R_{es} \quad (6)$$

$$W'' = W' \cdot \overline{N'} \cdot \overline{S'} + W' \cdot N' \cdot R_{wn} + W' \cdot S' \cdot R_{ws} \quad (7)$$

$$S'' = S' \cdot \overline{E'} \cdot \overline{W'} + S' \cdot E' \cdot R_{se} + S' \cdot W' \cdot R_{sw} \quad (8)$$

As an example of how to use the second equations set, from (5) to (8), suppose that when attempting to forward a packet, the evaluation of the previous step (*i.e.* first equations set) only equations (1) and (2) return true values. In such a case, the LBDR in the router currently processing the packet must answer two questions: *If the packet is forwarded through the North output port, will it be able to turn East later?* and *If the packet is forwarded through the East output port, will it be able to turn North later?*, respectively equations (5) and (6). Furthermore, such answers can also be found by inspecting the compass rose representation for the adopted routing algorithm in Figure 9. As explained in Section 2.2.2, for this manuscript the permitted and prohibited turns are marked as black and gray, respectively.

Finally, the third set of equations, from (9) to (12), is for validating paths in irregular topologies as the one presented in [44] by answering the question: "Is/are the required output port(s) connected to a neighbor?".

$$N = N'' \cdot C_n \quad (9)$$

$$E = E'' \cdot C_e \quad (10)$$

$$W = W'' \cdot C_w \quad (11)$$

$$S = S'' \cdot C_s \quad (12)$$

After evaluating the three sets of logic equations requests are sent to the arbiter. However, two possible outputs may happen to be true. In such a case, the arbiter will select which request to honor based on some other information such as the one with the fewer concurrent requests from other input ports.

### 2.2.4 Project Bonfire

#### 2.2.4.1 Bonfire

Bonfire is an open-source framework for developing dependability mechanisms for NoC-based Systems-on-Chip (SoCs). The target NoC architecture is a 2D mesh topology. Each network tile consists of a wormhole switching router equipped with fault tolerance mechanisms and a NI, which is connected to the local resource. The routers use credit-based flow control and support any turn-model-based minimal path adaptive routing algorithm [41, 45].

#### 2.2.4.2 Secure Bonfire

The Secure Bonfire is a branch of Project Bonfire that aims to abolish or mitigate some vulnerabilities of NoCs that could be leveraged by malicious users to disrupt the performance of the system [46].

Figure 10 shows the packet format used by the Secure Bonfire variant of the Bonfire framework. This packet format is designed to be compatible with the Open Core Protocol (OCP) and each body flit carries a payload of 28 bits.

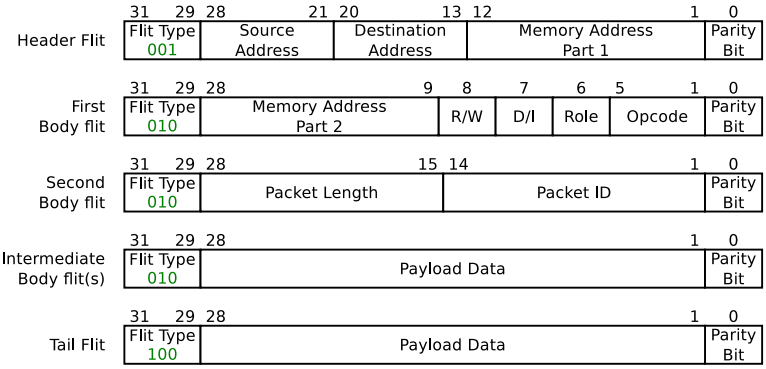



*Figure 10: Packet Structure - Secure Bonfire [46]*

Figure 11 shows a block diagram of the router found in Secure Bonfire. This router contains input buffers and LBDR routing units for each input port, one switch allocator, and a crossbar switch for each output port. It implements credit-based flow control, where each upstream router keeps track of empty buffers in the downstream router and sends the flits with the assumption that the downstream router can receive them. The FIFO unit utilizes a 4-flit deep circular buffer and is in charge of issuing a credit signal to the upstream router.

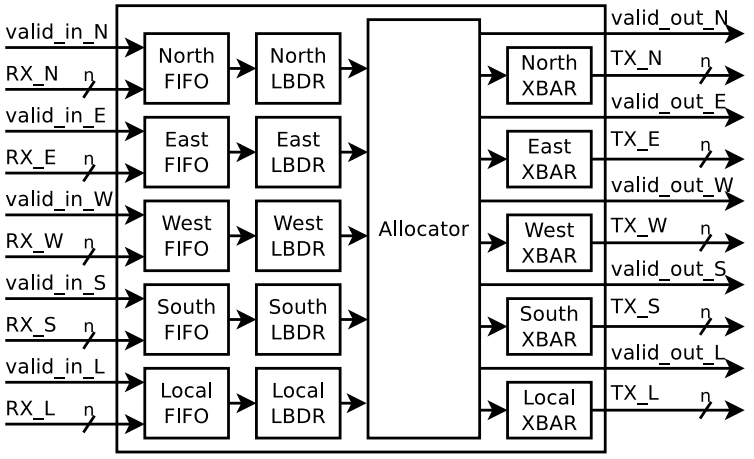



*Figure 11: Router Architecture - Secure Bonfire [46]*

The routing logic is implemented using the LBDR mechanism (Section 2.2.3) which is a lightweight distributed routing mechanism that supports any turn model-based routing algorithm and provides the possibility of an in-system reconfiguration of the routing algorithm.

The allocator unit utilizes a Round Robin arbitration, where each input direction can hold access to an output port for the length of a single packet. This arbitration mechanism has an important impact on the resilience of the network to DoS attacks, hence it prevents packets from one input port to block the transmission of packets entering the router through other input ports.

## 2.3 Task Scheduling and Mapping

Scheduling is a decision-making process that deals with the allocation of resources to tasks over a given time period [47]. Furthermore, resources and tasks refer to different things according to the scenario. Some scenario examples are: i) an airport, where take-off and landing tasks must be scheduled to runways, in order to reduce waiting periods; ii) Teaching activities that must be assigned to teachers and classrooms to reduce gaps in teacher schedules and avoid them in students' schedules; iii) Package deliveries assigned to mail carriers, in order to reduce delivery delays and costs; iv) Workflow tasks mapped to resources in an execution environment typically to reduce execution time, power consumption, or usage fees.

Several approaches have been proposed for scheduling application workflows in NoC-based MPSoCs. Consequently, surveys regarding this subject have been published over the years [39, 48–50]. Moreover, some approaches target scheduling of applications during runtime, where decisions regarding applications are taken when submitted on a first come first serve basis [34–36, 38, 51, 52].

It is worth mentioning that: i) the execution delay of any task is directly proportional to its processing cost and inversely proportional to the processing capacity of the PE to which it was mapped; ii) the makespan of an application is the time taken to execute all the tasks in its workflow (i.e., the overall execution time of an application); iii) the execution of a task with dependencies cannot begin until the data from its predecessors is available; iv) data exchanged between PEs may be divided into packets, which makes its end-to-end delay dependent on the allowed PIR, the PPL and the end-to-end delay of each packet, which in turn is proportional to a) encapsulation delay at the source NI; b) decapsulation delay at the destination NI; c) routing delay in each router in the path from source to destination; d) delays caused by congestion in each hop; and e) packet payload length.

Section 2.3.1 describes how application workflows are represented for scheduling. Section 2.3.4.1 gives an example of how an application workflow is scheduled on a NoC-Based MPSoC, while Section 2.3.4.2 explains scheduling on multi-tenant execution environments where applications arrive at different times.

### 2.3.1 Application Workflows

In order to speed up the execution of applications submitted to high-performance systems, they are divided into tasks and described by Communication Task Graphs (CTGs) in the form of Directed Acyclic Graphs (DAGs) [53]. In such types of graphs, as the one depicted in Figure 12, a workflow without cycles is detailed, where nodes represent tasks and arcs the dependencies between them. Tasks are considered as a set of instructions that are executed sequentially, on the same processor, without preemption [54]. In the context of this document, a processor refers to a PE, which in turn can be a processing core, a hardware accelerator, a cryptography core, a memory controller, a peripheral, or any other intellectual property core. Furthermore, values in nodes represent the execution cost of each task, while the weight of arcs is the amount of data to be transferred between connecting tasks.

The Embarrassingly Parallel workflow (Figure 12) is the simplest distributable application found in literature [55]. It contains a single level of tasks that have no data dependencies between them. Examples of other real-world scientific workflow applications are presented in Figure 14, which are commonly used for scheduling research in Cloud Computing [56–58]. The Montage application (Figure 14a) is used for processing various input images and stitching them together to make a mosaic of

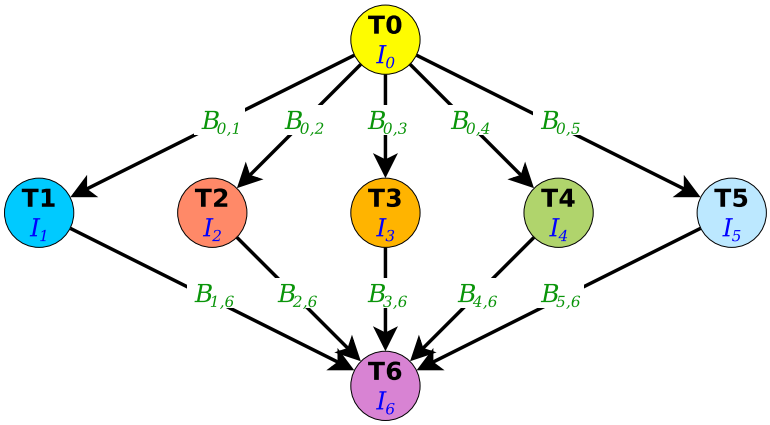



*Figure 12: CTG of an Embarrassingly Parallel Application with 7 Tasks and 10 Dependencies*

the sky. The Epigenome workflow (Figure 14b) is executed for mapping the epigenetic state of human cells. Seismic hazard maps for earthquake detection are generated with the CyberShake workflow (Figure 14c). The Laser Interferometer Gravitational-Wave Observatory (LIGO) workflow from Figure 14d is used for the detection of gravitational waves in the universe [59].

Moreover, CTG $= G(T, D)$ and, in this manuscript, is represented with the following notation:

- $n$: number of tasks, where $n \in \mathbb{N}$;
- $T$: set of tasks, where $T = \{t_i \mid 0 \leq i < n\}$;
- $D$: set of dependency arcs, where $D = \{(i, j) \mid i < j$ and there exists an arc from vertex $i$ to vertex $j$ in the CTG representing a data transmission from task $i$ to task $j\}$;
- $I_i$: execution cost of the $i^{th}$ task;
- $B_{i,j}$: number of data units transmitted between the $i^{th}$ task and the $j^{th}$ task, where $B_{i,j} \in \mathbb{R}_+$;

Furthermore, in order to force a scheduler to map more than one task on a single PE, the CTG can be modified to cluster tasks as if it were one. In [60], Tian et al. identify communication-intensive tasks in the CTG of applications and either cluster them or duplicate their executions on different PEs. Clustered task graphs are also used in [37] for dependable task deployment on systems with mixed-criticality constraints. One type of clustered task graph is the Application Characterization Graph (APCG) which similar to CTGs, is directed but allows cycles [39]. In APCGs, tasks are grouped according to characteristics that may require their execution on a specific type of PE. Moreover, depending on the amount cores in the PE and the dependencies of the clustered tasks, some executions can be in parallel. Figure 13 shows an APCG example of the CTG presented in Figure 12.

In an APCG $= G(C, D)$, each vertex $c_i \in C$ represents a cluster or an IP core (clusters can also contain single tasks), while the arc dependencies depict data transmissions between clusters, following the same formulation as CTGs.

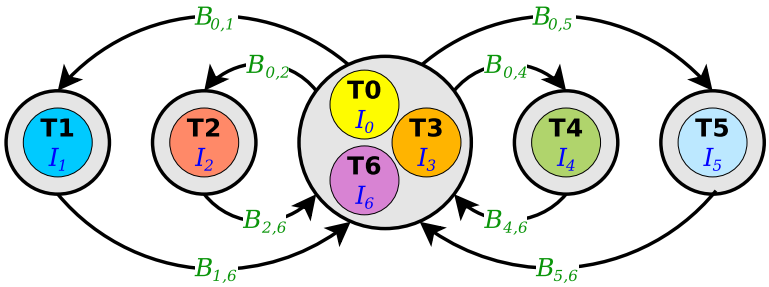



*Figure 13: APCG of an Embarrassingly Parallel Application with 7 Tasks and 10 Dependencies*

### 2.3.2 Modeling - Mesh NoC-based MPSoC Architecture

Considering a Mesh NoC-based MPSoC as an array of tiles as shown in Figure 8, its architecture can be modeled by the Architecture Graph AG $= G(T', L)$ which in this manuscript, is represented with the following notation:

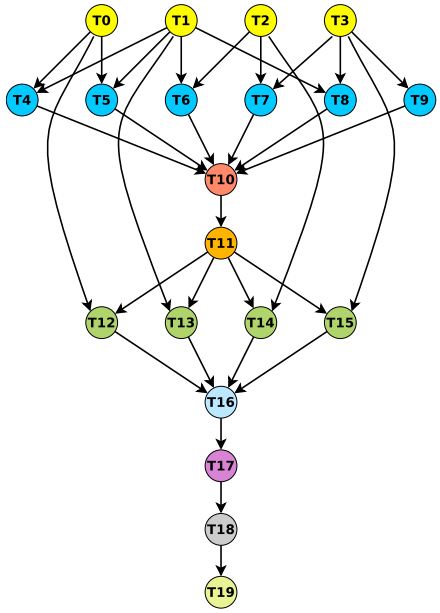
- $n'$: number of rows in the array of tiles, where $n' \in \mathbb{N}$;
- $m$: number of columns in the array of tiles, where $m \in \mathbb{N}$;
- $T'$: set of MPSoC tiles, where $T' = \{t'_{y,x} \mid 0 \leq y < n',\ 0 \leq x < m\}$;
- $CC_{y,x}$: core count of tile $t_{y,x}$, where $CC_{y,x} \in \mathbb{N}$;
- $TI_{y,x}$: average execution capacity of each core in tile $t'_{y,x}$, where $TI_{y,x} \in \mathbb{R}_+$;
- $GM_{y,x}$: location of the tile executing the global manager software, where $GM_{y,x} = t'_{y,x} \mid t'_{y,x} \in T'$ and the global manager software is running on $t'_{y,x}$;
- $L$: set of directional links connecting neighbor tiles, where $L = \{\ (t'_{y1,x1}, t'_{y2,x2}) \mid$ tile $t'_{y1,x1}$ is linked to tile $t'_{y2,x2}\}$. Additionally, $(t'_{y,x}, t'_{y,x}) \in L\ \forall\ t'_{y,x}$ and $(t'_{y1,x1}, t'_{y2,x2}) \in L$ does not imply that $(t'_{y2,x2}, t'_{y1,x1}) \in L$;
- $TB_{y1,x1,y2,x2}$: available bandwidth of the link connecting tile $t'_{y1,x1}$ with tile $t'_{y2,x2}$, where $TB_{y1,x1,y2,x2} \in \mathbb{R}_+$;
- $P_{y1,x1,yv,xw}$: set of paths connecting tile $t'_{y1,x1}$ with tile $t'_{yv,xw} \mid p_{y1,x1,yv,xw} \in P_{y1,x1,yv,xw}$, $p_{y1,x1,yv,xw} = \{(y1, x1),\ (y2, x2),\ \ldots\ (yv, xw)\}$ and each pair of consecutive elements in $p_{y1,x1,yv,xw} \in L$.

### 2.3.3 Modeling - Mapping of an Application to a NoC-based MPSoC

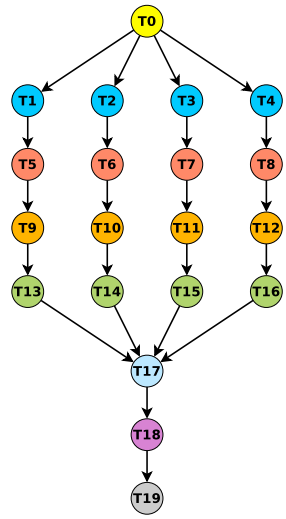
In the context of this thesis, the mapping process consists of two parts: i) assigning the execution of tasks to specific Tiles of an MPSoC, namely $MAP$. This could either be:

- APCG $\longrightarrow$ AG
- CTG $\longrightarrow$ APCG $\longrightarrow$ AG

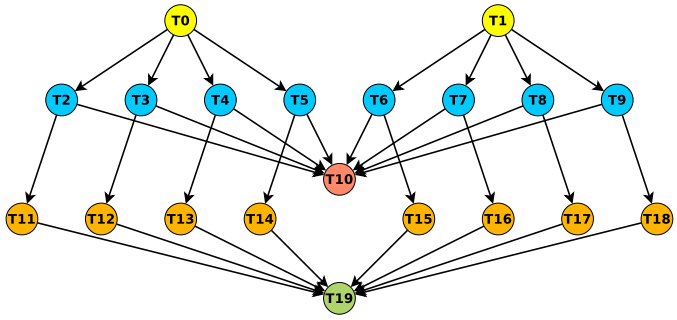
and ii) Mapping of inter-task communication (edges of task graph) on network links (edges of an AG), namely $MAP'$.



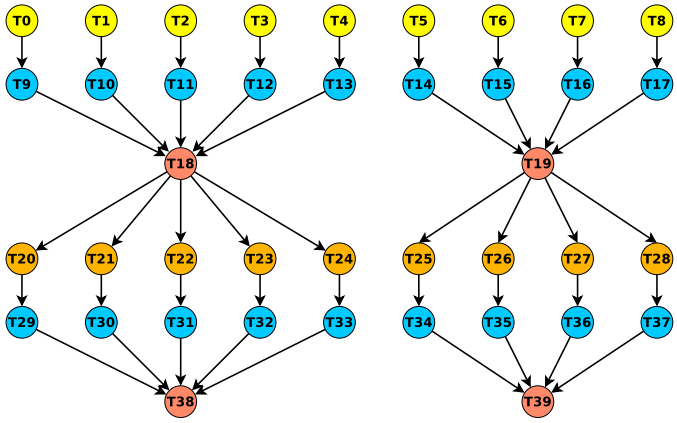

(a) Montage




(b) Epigenomics




(c) CyberShake




(d) LIGO

*Figure 14: Real-world Scientific Workflow Applications [59]*

Considering the notation presented in Section 2.3.1 for describing application workflows, as well as the one presented in Section 2.3.2 for representing the architecture of an NoC-based MPSoC:

- $MAP_{i,y,x,c}$ expresses that the execution of task $t_i$ was mapped to tile $t'_{y,x}$ on core ID $c$, $MAP_{i,y,x,c} \in \{0,1\}$;
- $MAP'_{i,j,y1,x1,y2,x2}$ expresses that the number of data units $B_{i,j}$, transmitted between the $i^{th}$ task and the $j^{th}$ task, will consume a certain part of the available bandwidth $TB_{y1,x1,y2,x2}$ of the link connecting tile $t'_{y1,x1}$ with tile $t'_{y2,x2}$, $MAP'_{i,j,y1,x1,y2,x2} \in \{0,1\}$.

Moreover, such mappings must respect the following mapping constraints:

MC1: $\forall\ t_i \in T\ \exists!\ MAP_{i,y,x,c} \mid t'_{y,x} \in T',\ c \in \mathbb{N}_0,\ 0 \leq c < CC_{y,x}$
All tasks of a workflow must be mapped once and to one of the cores of a tile.

MC2: $\forall\ (i,j) \in D\ \exists\ MAP_{i,y1,x1,c1}\ \wedge\ MAP_{j,y2,x2,c2} \mid P_{y1,x1,y2,x2} \neq \emptyset,\ c1 \in \mathbb{N}_0,\ 0 \leq c < CC_{y1,x1},\ c2 \in \mathbb{N}_0,\ 0 \leq c2 < CC_{y2,x2}$
All pairs of tasks that have a dependency are mapped to cores of tiles that have a communication path between them.

MC3: $\forall\ (i,j) \in D\ \exists!\ p_{y1,x1,y2,x2} \mid \forall$ pairs of consecutive elements {(yv,xv), (yw,xw)} in $p_{y1,x1,y2,x2}:\ MAP'_{i,j,yv,xv,yw,xw}$ is true.
For all pairs of tasks that have a dependency, exists a path in which all its links are mapped for the transmission of $B_{i,j}$.

Furthermore, scheduling must respect the following scheduling constraints at any given time:

SC1: $\sum_{t_i \in T} \sum_{c=0}^{CC_{y,x}-1} M_{i,y,x,c} \leq CC_{y,x},\ \forall\ t'_{y,x} \in\ T', M_{i,y,x,c} \in \{0,1\}$
A tile cannot execute simultaneously more tasks than its amount of cores.

SC2: $\sum_{i=0}^{n} \sum_{j=i+1}^{n} \Delta B_{i,j} \times MAP'_{i,j,y1,x1,y2,x2} \leq Q\ \forall\ (t'_{y1,x1}, t'_{y2,x2}) \in L$, where $Q$ is a maximum theoretical value of the amount of data that can flow through any given network link during a defined time interval.

### 2.3.4 Illustrative Example - Scheduling

#### 2.3.4.1 Single Application

For a better understanding of the scheduling process on an MPSoC, an illustrative example of an Embarrassingly Parallel Application is presented in Figure 15. The application contains seven tasks from which: $T0$ is a data distribution task that forwards data to five parallelizable tasks; $T1$ to $T5$ are independent processes; and $T6$ is a data aggregation task that generates the overall result of the application. Furthermore, Figure 16 presents a possible mapping of those tasks to a $4 \times 4$ NoC-based MPSoC for three different times of the schedule. Additionally, for the example, it was assumed that the Global Manager runs on PE12 and that all PEs have the same processing capacity.

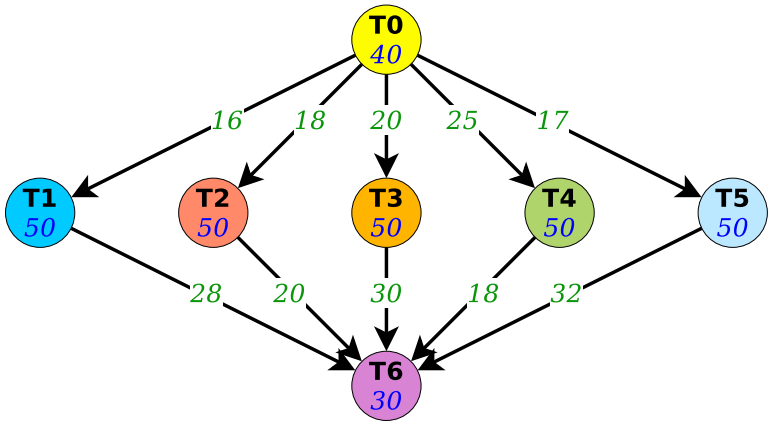



*Figure 15: Embarrassingly Parallel Application Task Graph with 7 Tasks and 10 Edges*


*(a) Time 0* *(b) Time 1* *(c) Time 2*

*Figure 16: Scheduling Example for Multi-tenant NoC-based MPSoCs*

### 2.3.4.2 Multi-tenant Execution Environment

The focus of this work is Multi-tenant Execution Environments in which NoC-based MPSoCs are part of the shared resources. Moreover, as soon as an MPSoC-powered device is identified as available, the environment's scheduler [56] can assign to it, applications that can leverage its PEs. Such applications may be of different characteristics, submitted by different users, and arrive at any time. Nevertheless, they must all be scheduled and executed as soon as possible so that new applications can also be served. Figure 17 presents an example of different applications arriving at different times (labeled from A to F). Furthermore, Figure 18 shows one scheduling possibility, where task IDs are accompanied by the application label.

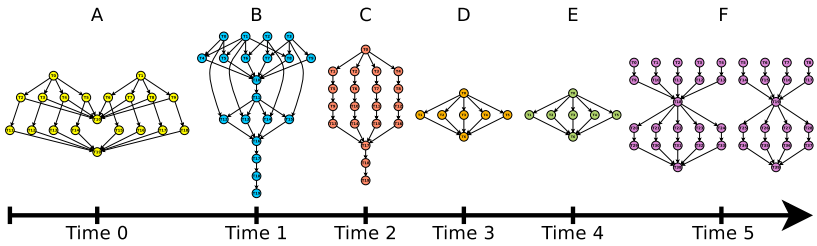



*Figure 17: Applications Arriving at Different Times*


*Figure 18: Scheduling Example for Multi-tenant NoC-based MPSoCs*

In the schedule illustrated by Figure 18, the processing cost was assumed equal for all tasks, as was the processing capacity of all PEs for simplicity reasons. Additionally, applications were mapped to the best available PEs, aiming to reduce the number of hops required by packets flowing between communicating tasks. In Figure 18d and Figure 18e, PEs 3, 7, and 15 were reserved for further execution of the Montage Workflow (application B) since they stored data from its first level tasks that is required for the execution of tasks in level 5.

# 3 DoS Attacks in NoC-based MPSoCs

Denial of Service (DoS) attacks attempt to disrupt the performance of a system. In Cloud Computing they can be classified as i) Physical disruption, ii) Data corruption, iii) Resource exhaustion, iv) Connectivity attacks, and v) Bandwidth attacks [61]. Such classification can also be adapted to NoC-based MPSoCs as follows:

- Physical disruption: it is related to tampering, for which an attacker requires physical access to the device;
- Data corruption: usually implemented as a man-in-the-middle attack carried out by a Trojan inside a NoC router. It modifies some bits of data packets, violating the integrity of data. If bits from a destination address are changed and this is not detected, it prevents a packet from reaching its intended destination and may cause congestion in an unexpected segment of the NoC. On the other hand, if any change to the packet is detected, the packet is dropped and retransmission may be requested, causing additional traffic to flow through the NoC [13, 62, 63];
- Resource exhaustion: consists in consuming all the capacity of a resource. In the case of a memory IP, it translates to filling up the memory so that no other PE can use it. On the other hand, if the target is a processing core, the attacker will try to execute tasks with long makespans keeping the PE busy and unable to execute tasks from other users [64];
- Connectivity attacks: in systems where a connection needs to be established before exchanging data, an attacker floods the NoC with excessive requests to a target until the server is unable to provide services to legitimate users [16];
- Bandwidth attacks: also known as Bandwidth Reduction Denial of Service (BR-DoS) attacks are those in which an attacker attempts to consume all or most of the bandwidth. By doing so, legitimate traffic is unable to flow through the NoC or takes longer than usual from its source to its destination. Such a type of attack is the focus of this thesis and is detailed in Section 3.1.

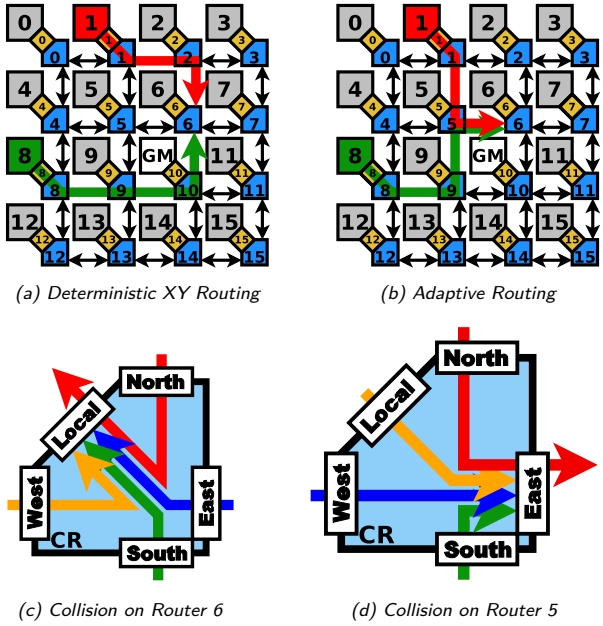
As explained in Chapter 4, the focus of this Thesis is on attacks that users of a Multi-tenant execution environment including NoC-based MPSoCs can attempt by submitting applications for execution. In this sense, neither physical disruption nor data corruption is considered for this Thesis since users will not have direct access to the system, preventing tampering and hardware trojan injection. On the other hand, even though resource exhaustion and connectivity attacks would be possible, the focus of this Thesis is on BRDoS attacks since, as stated in [65], they were identified to create a greater impact.

The remaining of this section is organized as follows: Section 3.1 explains BRDoS attacks in NoC-based MPSoCs. Section 3.2 presents a causal analysis of variables affecting the end-to-end delay of packets in a NoC-based MPSoC. Section 3.3 describes the setup of experiments and explains the obtained results.

## 3.1 Bandwidth Reduction DoS Attacks in NoC-based MPSoCs

BRDoS attacks create a greater impact to the NoC-based MPSoC since they do not only affect the destination, but also all other tasks wanting to communicate through any of the links being used by the attacker [65], and by doing so, increasing their end-to-end communication delays.

Figure 19a and Figure 19b illustrate two 4×4 NoC-based MPSoC scenarios implementing different routing algorithms, but in both, PE1 attempts a BRDoS attack by sending malicious traffic to PE6. Such malicious traffic collides with legitimate traffic originating at PE8 and heading to the same destination. Furthermore, Figure 19c and Figure 19d depict the worst-case presented within the collision routers of both scenarios. In such routers, all traffic streams compete for the same output port (considering even two additional traffic streams). Once one of them gets granted access to such a port, others will have to wait until the grant is given to them. Hence, the efficacy of a BRDoS attack is proportional to the amount of time that the attacking stream is able to maintain the grant.



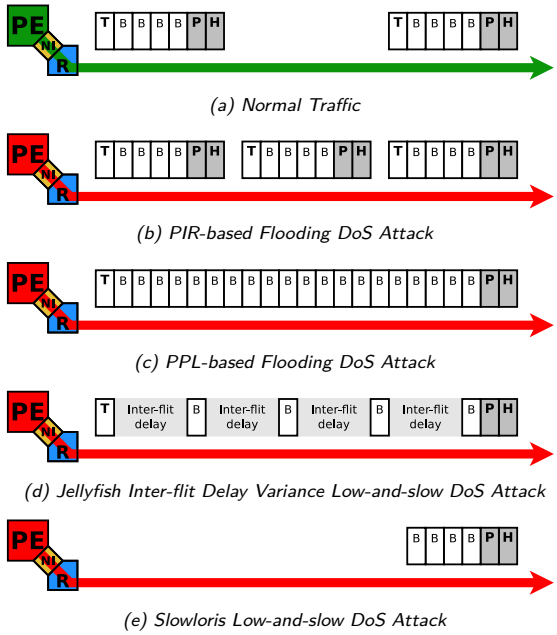

*(a) Deterministic XY Routing* *(b) Adaptive Routing*

*(c) Collision on Router 6* *(d) Collision on Router 5*

*Figure 19: Example of DoS Attack in MPSoCs*

**Flooding DoS Attacks:** A malicious task executing in one of the PEs is said to attempt a Flooding DoS (FDoS) Attack when it seeks to acquire and maintain the output grants of routers in a path by injecting a high amount of data [4]. Two types of FDoS attacks are depicted in Figure 20. The first, namely the PIR-based FDoS attack, injects data segmented into many consecutive packets [4, 66]. On the other hand, the second, namely the PPL-based FDoS attack, segments data into long packets which are not necessarily contiguous [4].

**Low-and-slow DoS Attacks:** In Low-and-slow DoS (LSDoS) Attacks, a malicious task injects a low amount of data, but similar to FDoS attacks seeks to acquire and maintain the output grants of routers in a path [3]. Two types of LSDoS attacks are depicted in Figure 20. The first, namely the Jellyfish Inter-flit Delay Variance (JFIDV) LSDoS, injects average-sized packets with unnecessary inter-flit delays. The second, namely Slowloris or Incomplete Packet Transmission (IPT) LSDoS attack, injects incomplete packets, lacking at least the Tail Flit.




*(a) Normal Traffic*

*(b) PIR-based Flooding DoS Attack*

*(c) PPL-based Flooding DoS Attack*

*(d) Jellyfish Inter-flit Delay Variance Low-and-slow DoS Attack*

*(e) Slowloris Low-and-slow DoS Attack*

*Figure 20: NoC Traffic Patterns*

## 3.2 Causal Analysis of End-to-end Delay in NoC-based MPSoCs

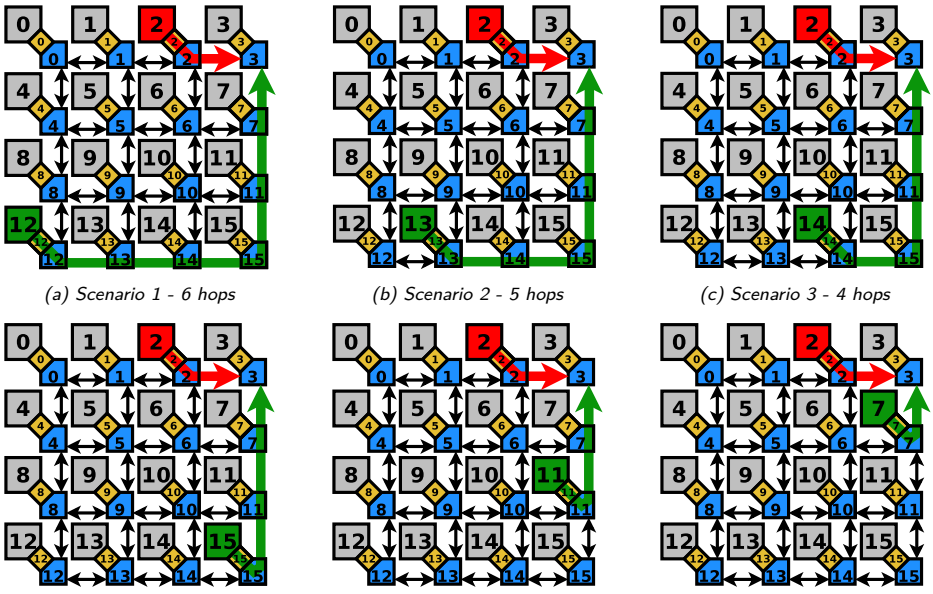
As explained in Section 3.1, BRDoS attacks in NoC-Based MPSoCs are caused when malicious tasks, running on one of the PEs, inject traffic to the NoC in a manner to prevent packets from legitimate tasks to flow or increase their end-to-end delay. Structural Equation Modeling (SEM) [67] was done to measure the covariance of traffic configurations for monitored and malicious packets that cause the end-to-end delay of monitored packets to increase. By doing so, a set of relationships between the end-to-end delay (dependent variable), and traffic generation parameters (independent variables) can be studied. Independent variables are usually considered as either predictor or causal variables because they predict or cause the dependent variables (the response or outcome variables). Table 1 lists the variables considered for the path analysis presented in this section.

### 3.2.1 Simulation Setup for Data Collection

BRDoS attacks were simulated within the Secure Bonfire framework (Section 2.2.4.2). As depicted in Figure 21, the adopted scenarios correspond to $4 \times 4$ mesh NoC-based MPSoC architectures. For all the scenarios, PE3 was set as the destination of the monitored as well as malicious traffic. Furthermore, as summarized in Table 2, experiments were considered for monitored packets with different payload lengths and injection rates, generated from different sources: PE12, PE13, PE14, PE15, PE11, and PE7, which respectively correspond to communication paths of 6, 5, 4, 3, 2, and 1 hops (see Figure 21). Simulations were executed with and without attack for different packet payload lengths and packet injection rates. In all scenarios, PE2 was set as the source of the attack and PE3 as the destination (i.e. attack path: PE2→PE3).

*Table 1: Data variables*

| Variable | Description |
|---|---|
| Packet Payload Length | Number of payload flits in the packets that are being monitored. |
| Packet Injection Rate | Frequency with which the monitored packets are injected into the network. The inverse of the number of clock cycles between the generation of each header flit. |
| Path Length | Number of hops between the source and the destination of the monitored packets. |
| Attacker's Packet Payload Length | Number of payload flits in the malicious packets. |
| Attacker's Packet Injection Rate | Frequency with which the malicious packets are injected into the network. The inverse of the number of clock cycle between the generation of each malicious header flit. |
| Delay at Collision Point | Number of clock cycles that a monitored packet waits inside a router, while other traffic is being transmitted through the required output port. |
| Delay End to End (or End-to-End delay) | Number of clock cycles elapsed since the network injection of a header flit of a monitored packet and until such flit arrives at its destination. |



*(a) Scenario 1 - 6 hops* *(b) Scenario 2 - 5 hops* *(c) Scenario 3 - 4 hops*

*(d) Scenario 4 - 3 hops* *(e) Scenario 5 - 2 hops* *(f) Scenario 6 - 1 hop*

*Figure 21: Simulation Scenarios for Path Analysis Data Collection*

*Table 2: Configuration of Experiments*

| Variable | Adopted Values |
|---|---|
| Packet Payload Length | 10, 20, 30, and 40 flits |
| Packet Injection Rate | 0.01 (one packet each 100 clock cycles),<br>0.02 (one packet each 50 clock cycles),<br>0.04 (one packet each 25 clock cycles) |
| Path Length | 6, 5, 4, 3, 2, and 1 hop |
| Attacker's Packet Payload Length | 0 (scenarios with no attack), 10, 20, 30, and 40 flits |
| Attacker's Packet Injection Rate | 0 (scenarios with no attack),<br>0.01 (one packet each 100 clock cycles),<br>0.02 (one packet each 50 clock cycles),<br>0.04 (one packet each 25 clock cycles). |

### 3.2.2 Path Analysis

For the path analysis, four models were proposed and analyzed. Such models are depicted in Figures 22, 23, 24, and 25. As can be seen in the figures, all models consider 6 predictors for the end-to-end communication delay. However, the last two regression models include an indirect path. The weight of the arrows from predictors to the dependent variable (i.e., Delay End to End), known as regression estimate, represents the degree to which changes in predictor values affect the dependent variable. Moreover, the value on the top-right of dependent variables represents the Squared Multiple Correlation (SMC) [67] which is used for showing how much the predictors explain the variance of the dependent variable. Furthermore, odd-numbered models consider covariances between the predictors. Such covariances are represented by the bidirectional arrows on the left of the figures and their weights show the degree to which the two predictors are linearly associated.

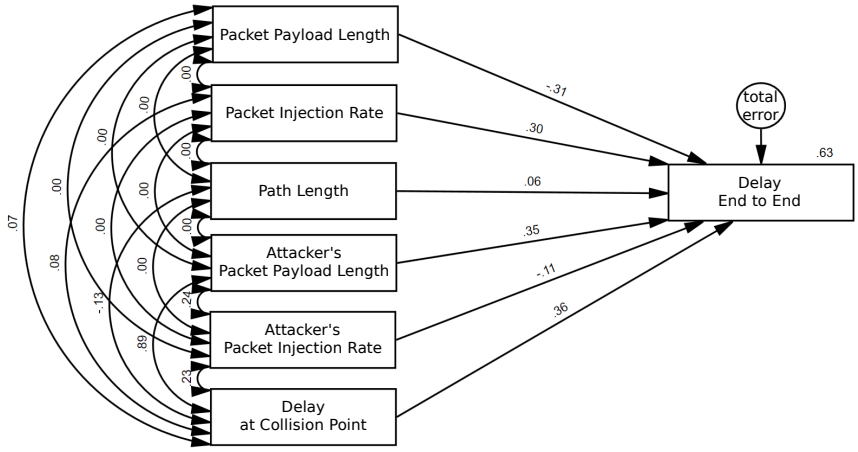



*Figure 22: Model 1*

Table 3 lists the path analysis results obtained with the four analyzed models, including their SMC value and all the predictors ordered according to their prediction strength (i.e. based on the standardized regression estimates). Furthermore, results show that even though the regression estimates are different for all the models, the prediction strength order is the same for all of them. It can also be observed that Model 4 achieves the best variance prediction of the NoC end-to-end communication delay (i.e., a higher SMC value).

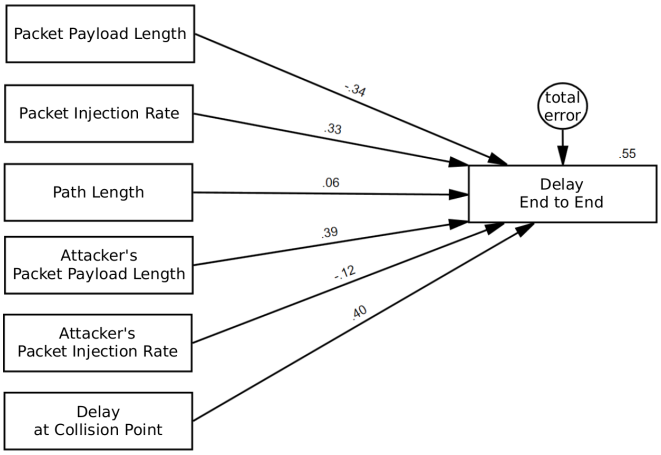



*Figure 23: Model 2*

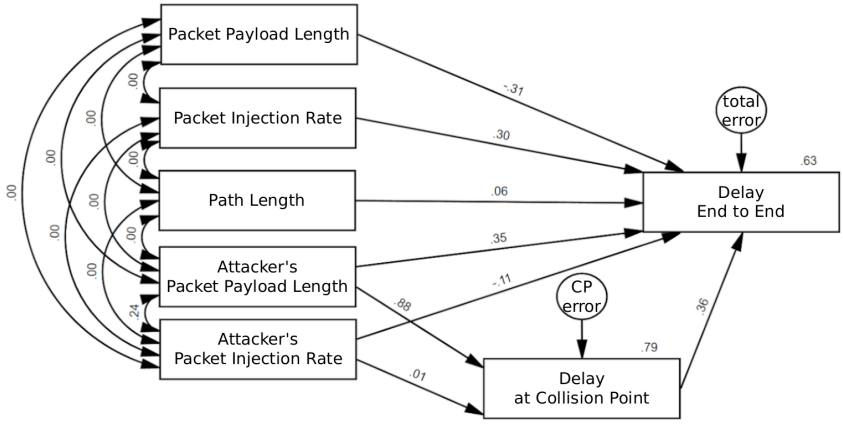



*Figure 24: Model 3*

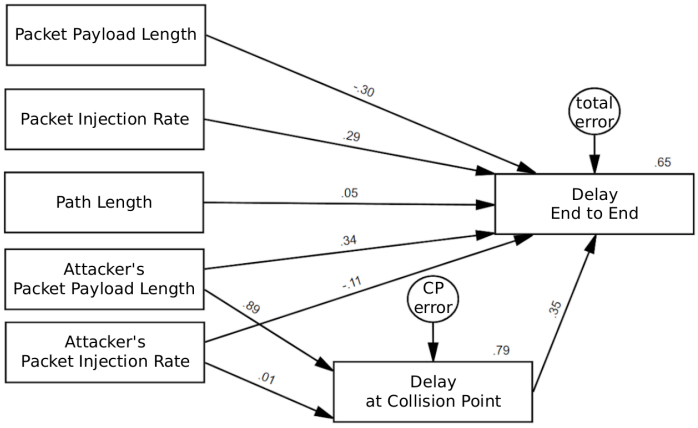



*Figure 25: Model 4*

*Table 3: Results of the Path Analysis*

| | **Model 1** | **Model 2** | **Model 3** | **Model 4** |
|---|---|---|---|---|
| SMC | .63 | .55 | .63 | .65 |
| Delay at Collision Point | .36 | .40 | .36 | .35 |
| Attacker's Packet Length | .35 | .39 | .35 | .34 |
| Packet Injection Rate | .30 | .33 | .30 | .29 |
| Path Length | .06 | .06 | .06 | .05 |
| Attacker's Packet Injection Rate | -.11 | -.12 | -.11 | -.11 |
| Packet Length | -.31 | -.34 | -.31 | -.30 |

Data obtained by simulations ($n = 640$)

Additionally, it can be noticed from Table 3 that the strongest predictors for the NoC end-to-end communication delay are: i) *Delay at Collision Point*, ii) *Attacker's Packet Payload Length*, and iii) *Packet Injection Rate*. These results can be explained as follows: i) the end-to-end delay is composed of the sum of delays in each hop, constant delays for buffering and routing decisions as well as a variable forwarding time that depends on how long the required output port is being used by other traffic. Moreover, the variable part is greater within the router identified as the collision point. ii) Once a malicious packet obtains the grant for the use of an output port, it will have it until its Tail flit has been forwarded, consequently preventing monitored packets from using the same output port and increasing their end-to-end delay. iii) Since Secured Bonfire's arbitration method (i.e., Round Robin) alternates only one packet from each competing input, packets from the same stream will remain buffered, hence increasing their end-to-end delay.

Regarding the validity of the presented models, Table 4 lists values for a more detailed level of model analysis. According to this analysis, the AMOS software [68] reported that Model 1 did not fit the data obtained with the communication simulations. Furthermore, by analyzing the Incremental Fit Measures, we can see that Model 2 does not meet the 0.90 level requirement either for the Normed Fit Index (NFI) or for the Comparative Fit Index (CFI). Finally, regarding models 3 and 4, both meet the 0.90 level requirement for NFI and CFI, however, Model 4 achieved a Tucker-Lewis Index (TLI) closer to 1; Therefore, we can conclude that Model 4 is the most appropriate for path analysis of end-to-end communication delays in NoC-based MPSoCs.

*Table 4: Goodness-of-fit Value Comparison for End-to-End Communication Delay Models*

| | Model 1 | Model 2 | Model 3 | Model 4 |
|---|---|---|---|---|
| Measures of Absolute Fit | | | | |
| $X^2$ | 0 | 1099.59 | 91.64 | 129.48 |
| $df$ | 0 | 15 | 3 | 13 |
| $X^2/df$ | — | 73.30 | 30.55 | 9.96 |
| $p$ | — | $> .001$ | $> .001$ | $> .001$ |
| $RMSEA$ | — | .341 | .218 | .120 |
| 90 percent C.I. | — — | .324 .358 | .181 .257 | .102 .139 |
| Incremental Fit Measures | | | | |
| $NFI$ | — | .362 | .947 | .925 |
| $CFI$ | — | .363 | .948 | .932 |
| $TLI$ | — | .108 | .635 | .889 |

Data obtained by simulations ($n = 640$)

Finally, it can be seen in Model 4 (Figure 25) that the *Attacker's Packet Payload Length* is a predictor not only for the *Delay at End to End* but also of the *Delay at Collision Point*; this because of the achieved 79% SMC by the .89 regression estimate

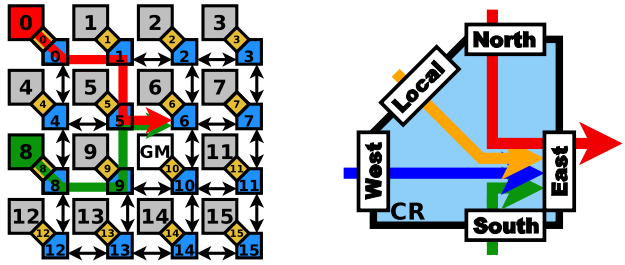
between the two predictors. Consequently, it can be concluded that when using a fair arbitration scheme such as Round Robin, a PPL-based DoS attack (Figure 20c) creates a greater impact on the NoC end-to-end communication delay than the PIR-based DoS Attack (Figure 20b). This analysis motivated the development of CDT, presented in Section 6.4.

## 3.3 Impact Analysis of PIR- vs PPL-based Attacks

In this Section, two communication parameters are exploited for attempting FDoS attacks on NoC-based MPSoCs, the PIR (used in most state-of-the-art approaches) and the packet's payload length (PPL). This section presents an effectiveness comparison of both attacks.

### 3.3.1 Simulation Scenarios

Figure 26a illustrates a scenario in which sensitive packets flow from PE8 to PE6 while malicious traffic does it from PE0 also to PE6. In such a scenario, both traffic flows collide inside router R5 and compete for its East output port. Figure 26b depicts a worst-case scenario in which apart from the two mentioned traffic flows, two additional flows also enter the competition. Since only one flow is granted access to the output at a time, it is the goal of the malicious traffic to win the access, making the others wait for their turn, hence increasing their end-to-end communication delay.




*(a) Attacked MPSoC* *(b) Collision Router*

*Figure 26: Setup of Simulation Scenarios*

Two types of FDoS attacks attempt to gain control of output ports, preventing legitimate traffic from being forwarded: PPL- and PIR-based DoS attacks. Five NoC traffic patterns were presented in Section 3.1: the one in Figure 20a, considered normal traffic, is composed of packets with an expected packet injection rate and within an acceptable size range. The PIR-based FDoS attack attempts to flood the network by increasing its packet injection rate, while the PPL-based FDoS attack by increasing the packet's payload length. The other two flows represent Low-and-slow DoS attacks, which are not the focus of the approach introduced in this section.

In order to compare both FDoS attacks, the Secure Bonfire platform (Section 2.2.4.2) was adapted so that it is compatible with the packet structure depicted in Figure 53. Subsequently, the scenario illustrated by Figure 26a was implemented. In such a scenario, PE0 and PE8 send data exclusively to PE6, to ensure that the path of both sources collides, at least at the destination, regardless of the paths taken. Furthermore, all the attacks executed in the experiments originated at PE0 and additional traffic was randomly generated from all the other PEs (i.e. traffic generators included in Secure Bonfire). Normal traffic was configured with a PPL of 10 flits and a PIR of 0.01 (i.e.

a rate of one packet every 100 clock cycles). Malicious traffic, on the other hand, was configured differently for two sets of experiments: i) PIR-based FDoS attack: the attacker's PPL was fixed to 10 flits and its PIR varied from 0.01 to full burst transmission (i.e. 10 flits of payload every 10 clock cycles); and ii) PPL-based FDos Attack: the attacker's PIR fixed to 0.01 and its PPL varied from 10 to 100 (i.e. a maximum of 100 flits every 100 clock cycles). Furthermore, the transmission buffer of NIs was extended so that attacks could completely maximize their parameters.

### 3.3.2 Effect Comparison of Both Flooding DoS Attacks

The effect of the PIR- and PPL-based FDoS attacks is presented in Figure 27, on the left and right graphs respectively. Both graphs show the effect caused by a single colliding path (i.e. PE8 $\rightarrow$ PE6) as well as the entire NoC. Such an effect is observed by the increase in the mean end-to-end delay of packets since they are transmitted by the PE until they reach their destination, while the attack is intensified.

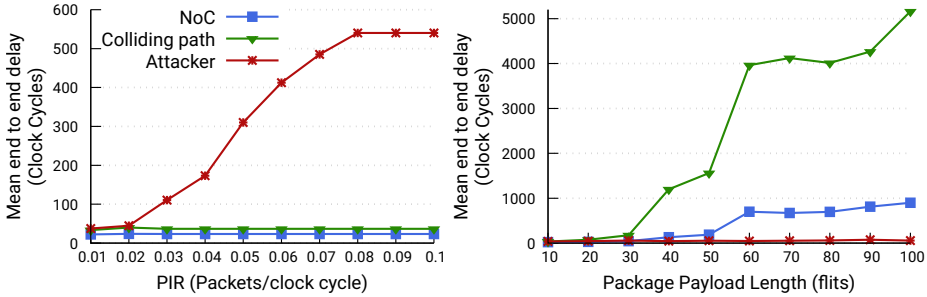



*Figure 27: Comparison of Flooding Dos Attacks*

As shown in Figure 27 (left), a PIR-based FDoS attack is ineffective in a NoC using a fair arbitration mechanism such as Round-robin which segments bursts while other packets are transmitted, consequently, increasing only the end-to-end delay of the malicious packets buffered in the NI. In contrast, Figure 27 (right) shows the effective communication disruption caused by the PPL-based FDoS attack to the NoC and even more to the colliding path. Hence FDoS attack detection and mitigation efforts should be focused on this scenario.

## 3.4 Conclusion

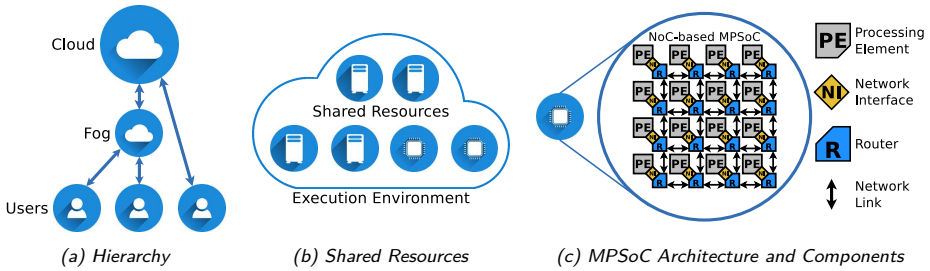
Since BRDoS attacks attempt to increase the end-to-end delays of legitimate traffic, a causal analysis was done to measure the effect of different traffic configurations on the end-to-end delay of monitored packets. Results showed that when using a Round Robin arbitration, the success of BRDoS attacks was related to the payload length of malicious packets rather than their injection rate. Further experiments were done for comparing PIR- and PPL-based BRDoS attacks on a network equipped with Round Robin arbitration and despite the focus of related work presented in Chapter 5 being mostly given to PIR-based attacks, PPL-based BRDoS attacks showed to cause a greater disruption to NoC-based MPSoCs.

# 4 Threat Model

The use of MPSoCs with a heterogeneous array of PEs can minimize the latency of communication links of the edge cloud, by providing programmability and parallelism, yielding flexibility, processing performance, and power efficiency [9]. However, connecting such devices to a bigger network exposes NoC-based MPSoCs to new security threats, such as BRDoS attacks (Section 3.1). The characteristics of such a scenario and its vulnerability to BRDoS attacks will be explained in this section as well as the steps followed by an attacker to execute a BRDoS attack and its success conditions.

## 4.1 Attack Scenario

This Ph.D. thesis resolves around heterogeneous multi-tenant execution environments. Examples of the target environments are Fog Computing and Cloud Computing, or even a combination of both as depicted in Figure 28. It is important to note that shared resources will not only include powerful multi-purpose computers but also heterogeneous NoC-Based MPSoCs. Moreover, such environments are able to accept applications submitted in parallel by different users. Additionally, based on the requirements of application tasks, they may be scheduled to a single MPSoC, which will concurrently execute tasks from different users. However, one or more of these tasks may disrupt the execution of others by being specifically engineered for such purpose or by incorporating a programming fault. In the scenario illustrated in Figure 28, users/clients submit applications for execution; subsequently, a Fog + Cloud scheduler (such as the one presented in [69]) divides them into tasks and maps their execution to some of the shared resources which include devices powered by NoC-based MPSoCs. Once an application or at least one of the tasks reaches the MPSoC, a Global Manager (GM) maps their execution to PEs according to the requirements of the task as well as the availability of the PEs (Section 2.3). This is done for all the received applications/tasks, allowing concurrent executions from different users. Moreover, as soon as the execution of all the tasks from a given application workflow is finished, the results are gathered following the Fog + Cloud schedule. Finally, when the entire application execution is over, results are sent back to the respective user/client.




*(a) Hierarchy* *(b) Shared Resources* *(c) MPSoC Architecture and Components*

*Figure 28: The enhanced Fog + Cloud execution environment*

## 4.2 Attack Steps

During a BRDoS attack, an attacker seeks to disrupt the services of a system by degrading its communication medium's performance. In the scope of this work, the system is represented by a Multi-tenant NoC-based MPSoC.

In order to perform a BRDoS attack in a NoC-based MPSoC, such as the one illustrated by Figure 28, the attacker, as any other client/user, submits an application to the execution environment. Such an application contains at least one task that, according to its characteristics/requirements, will be scheduled to run in one of the available NoC-based MPSoCs by a Fog + Cloud scheduler (such as the one presented in [69]).

Once in a NoC-based MPSoC, the execution of the malicious task is mapped to one of the PEs (as described in Section 2.3). Subsequently, when the execution of the malicious task starts, it transmits data packets to another PE attempting to reduce or even cancel the availability of NoC links between the two PEs. In order to maximize the probability of disruption, malicious tasks should include instructions that the scheduler or firmware may interpret as a transmission to a common destination, such as a shared memory, a hardware accelerator, a peripheral, or any other Intellectual Property (IP) component that will also be required by other applications.

Depending on the restrictions of the system, tasks are free to choose the way they transmit their data. Transmission characteristics may include Packet Injection Rate (PIR) or Inter-packet delay (IPD), Packet Payload Length (PPL), Inter-flit delay (IFD), and even the completeness of the packets (if the transmission of a packet begins before it is completely available). Such values can be hard coded into the tasks by the malicious user or can be adjusted by the task during runtime. Furthermore, a cautious attack may begin with moderate parameter values and will increase/decrease them gradually, as the malicious process perceives that the network is getting congested. In some cases, this can be done by monitoring the Effective PIR of the malicious traffic [70]. Finally, when an appropriate attack configuration is found, packets can continue to be constantly transmitted with such values or during alternating periods.

## 4.3 Attack Success Conditions

In order to impact a NoC-based MPSoC with a DoS attack, the following conditions must be met:

- A malicious task is able to be executed in a PE inside the NoC-based MPSoC; As any other user, applications with MPSoC execution requirements submitted by an attacker will be mapped to a PE of an available Multi-tenant NoC-based MPSoC, unknowingly of the malicious intentions.
- The malicious task transmits data through the NoC following a traffic behavior not expected by the MPSoC's Global Manager at the time of scheduling. The traffic transmission can be triggered by a memory store instruction, an instruction that requests the use of an IP commonly used in the system, or even a peripheral. Therefore, not only the service of the destination would be denied to legitimate tasks (i.e., blocking the output port of the router connected to the destination), but also disrupting the transmission of other packets attempting to use at least a segment of the path allocated to the malicious packets.
- The system allows a task to determine the value of parameters such as PIR or IPD, PPL, IFD, and/or the completeness of the packets.

In this work, specific PEs are selected to be transmitter or receiver nodes of monitored, malicious or random traffic. The destination of monitored and malicious packets is usually configured to be the same, forcing them to compete at least for the local output port of the destination, if not before.

# 5 Related Work

In order to answer the proposed research questions (Section 1.2), it is necessary to understand Networks on Chip (NoCs) and Multiprocessor Systems on Chip (MPSoCs) as well as how Denial of Service (DoS) attacks affect them. Such concepts are explained in Chapter 2. Furthermore, this chapter presents how other authors present DoS attacks in such a context and their proposals for overcoming such a threat. It is worth mentioning that the literature review presented in this chapter is filtered to target the attack scenario presented in Chapter 4. Therefore, considering only DoS attacks that could be implemented by an application running on a PE of a multi-tenant NoC-based MPSoC, and aiming to disrupt the execution of applications from other tenants.

The literature review was done using four scientific databases i.e., IEEE [71], ACM [72], SPRINGER [73] and ELSEVIER [74]. Such databases are known to store the most relevant publications in the targeted field. Additionally, no time constraint was set so that all the publications related to the targeted subject could be reached.

Table 5 lists the syntax of the search terms used for the literature review as well as the results obtained for each term in each database. Moreover, search terms were grouped into filtering phases according to their restriction degree. The first phase was done without the use of search operators. For the second phase, quotes (" ") were added to search for an exact phrase. Finally, in the third phase, search terms were joined by an *AND* to establish that the results must contain either DoS attacks and NoCs or DoS attacks in MPSoCs. Such review was done several times during the Ph.D. period so that the proposals could be up to date. The numbers shown in Table 5 correspond to search results obtained on January 22nd, 2022. Quotes were not used for *NoC* nor *MPSoC* so that papers presenting approaches with these terms as part of their name were not filtered out.

*Table 5: Literature review - Search terms and results*

| | | Search Term | IEEE [71] | ACM [72] | Springer [73] | Elsevier [74] | Total |
|---|---|---|---|---|---|---|---|
| Phase 1 | 1 | Denial of Service Attack | 5969 | 238279 | 48915 | 18045 | 311208 |
| | 2 | DoS Attack | 3442 | 483517 | 27340 | 19100 | 533399 |
| | 3 | Network on Chip | 41896 | 312980 | 130505 | 115749 | 601130 |
| | 4 | NoC | 6241 | 3572 | 27199 | 19164 | 56176 |
| | 5 | Multiprocessor System on Chip | 5675 | 561496 | 7734 | 5853 | 580758 |
| | 6 | MPSoC | 2208 | 1449 | 1178 | 756 | 5591 |
| | | **Phase Total** | 65431 | 1601293 | 242871 | 178667 | **2088262** |
| Phase 2 | 7 | "Denial of Service Attack" | 2379 | 1616 | 10265 | 5011 | 19271 |
| | 8 | "DoS Attack" | 1401 | 1725 | 8512 | 4765 | 16403 |
| | 9 | "Network on Chip" | 7183 | 2343 | 2800 | 2207 | 14533 |
| | 10 | "NoC" | 5850 | 3419 | 27199 | 19164 | 55632 |
| | 11 | "Multiprocessor System on Chip" | 856 | 331 | 600 | 386 | 2173 |
| | 12 | "MPSoC" | 2032 | 1214 | 1178 | 756 | 5180 |
| | | **Phase Total** | 19701 | 10648 | 50554 | 32289 | **113192** |
| Phase 3 | 13 | "Denial of Service Attack" *AND* "Network on Chip" | 14 | 11 | 57 | 25 | 107 |
| | 14 | "Denial of Service Attack" *AND* NoC | 12 | 15 | 100 | 25 | 152 |
| | 15 | "DoS Attack" *AND* "Network on Chip" | 12 | 18 | 62 | 21 | 113 |
| | 16 | "DoS Attack" *AND* NoC | 13 | 20 | 89 | 25 | 147 |
| | 17 | "Denial of Service Attack" *AND* "Multiprocessor System on Chip" | 0 | 1 | 8 | 11 | 20 |
| | 18 | "Denial of Service Attack" *AND* MPSoC | 7 | 2 | 22 | 18 | 49 |
| | 19 | "DoS Attack" *AND* "Multiprocessor System on Chip" | 1 | 2 | 8 | 4 | 15 |
| | 20 | "DoS Attack" *AND* MPSoC | 6 | 6 | 24 | 8 | 44 |
| | | **Phase Total** | 65 | 75 | 370 | 137 | **647** |

Table 6 lists the number of search results selected in each of the filtering phases as well as the number of rejected results and the filtering or rejection criteria. As explained before, the first three filtering phases were done using search operators so that results matched the research topic as much as possible. The results from the final eight search terms and for the four scientific databases were then combined into a single list. Subsequently, in the fourth phase, duplicated results were eliminated from such a list. The remaining results were then filtered, in a fifth phase, based on their title. Finally, the remaining papers were studied and the most relevant selected.

*Table 6: Literature review - Filtering process of search results*

| Phase | Found | Rejected | Selected | Criteria |
|---|---|---|---|---|
| 1 | 2088262 | — | 2088262 | Search-Based Selection |
| 2 | 2088262 | 1975070 | 113192 | Use of Quotes for Search |
| 3 | 113192 | 105756 | 647 | Combined Research Terms |
| 4 | 647 | 353 | 294 | Rejection per Repetition |
| 5 | 294 | 235 | 59 | Rejection per Title |
| 6 | 59 | 42 | 16 | Rejection per Content |

Based on the papers resulting from the literature review, it was identified that when aiming to disrupt the execution of applications from other tenants, the best option for a malicious application running on a PE of a multi-tenant NoC-based MPSoC, is to implement Bandwidth Reduction DoS (BRDoS) attacks. This answers research question RQ1. However, it was noticed that most authors mainly target a single type of BRDoS attack (i.e. PIR-based Flooding DoS attack). BRDoS attacks were explained in Section 3.1.

Research question RQ2 can be answered by the different approaches presented by the authors of the reviewed papers aiming either to avoid or detect BRDoS attacks. Some of them attempted additionally to locate the attack source and even manage to mitigate the attack or outline directives for their mitigation. However, some space for improvement was identified, which motivated the design and development of the new approaches presented in this thesis.

The remainder of this chapter is organized as follows: Section 5.1 presents approaches aiming to avoid DoS attacks in NoC-based MPSoCs. Section 5.2 presents approaches aiming to detect DoS attacks in NoC-based MPSoCs. Section 5.3 explains the beyond-state-of-the-art contributions achieved with the approaches presented in this Thesis.

## 5.1 DoS Attack Avoidance

Approaches for DoS attack avoidance focus on providing an infrastructure that prevents vulnerabilities to such types of attacks. In successful cases, an attack attempt done by a malicious application will not be able to disrupt the performance of the protected application executions.

Evain et al. in [15] propose a division of the MPSoC resources into two areas, secure and unsecured. Additionally, the physical links of the NoC forward data through two separate virtual channels. One of the virtual channels, tagged as secure, transports traffic exchanges within the secure area while the one tagged as low security, data originated within the unsecured area. The secure area manages both types of virtual channels, prioritizing always packets flowing through the high-security virtual channel. Even though this approach manages to prevent malicious traffic from one area to disrupt one from the other, it allows disruption caused by other traffic originating in the same area. Furthermore, the use of virtual channels results in considerable hardware overhead, thus each router input requires an additional buffer for each virtual channel as well as control logic to select from which of the buffers the data is forwarded.

In [75], Sepulveda et al. propose an avoidance strategy based on a hybrid switching routing mechanism. Data tagged as critical is forwarded through the NoC using circuit switching while non-critical uses packet switching. A small control packet is sent from a PE willing to initiate the communication of critical data to its required destination. With such an approach, adding a small area overhead, a control packet reserves a communication path exclusively for the critical data, which successfully prevents other traffic from disrupting it. Consequently, data packets transporting critical data are able

to arrive in the correct order and with predictable latencies. However, if this approach were to be implemented in the context of multi-tenant NoC-based MPSoC, it would only be applicable to a few tenants. This is because allocating network links to a single tenant reduces the number of shareable resources for the others. Hence, the DoS attack avoidance mechanism would deny communication services to traffic not labeled as critical.

Later, in [26], Sepulveda et al. attempt to avoid DoS attacks by considering the dynamic allocation of communication resources in the MPSoC. To this end, NoC routers implement random arbitration and adaptive routing. With such an approach, data entering a router through different input ports and competing for the same output wait for their turn to be randomly granted, thus preventing a single traffic source takes indefinite control over an output port. Additionally, packets transmitted between a pair of PEs may have more than a single path available, hence allowing them to avoid some congested routes. This approach proved to be effective against its targeted BRDoS attack which is the PIR-based flooding DoS attack, and also requires a small area overhead. However, it would be useless against the other three BRDoS attacks targeted by this Thesis.

In [17], Boraten et al. propose non-interference-based adaptive routing to secure NoCs from side channel and DoS attacks. Following this approach, traffic is spread all over the NoC within three levels of routing. Such levels forward packets simultaneously along the NoC implementing a different routing algorithm. Level one considers dimensional order routing (XY routing); Level two uses orthogonal one turn (O1TURN) routing, either XY or YX routing; and the third level, reserved for traffic with the highest priority, uses randomized oblivious minimal multi-phase (ROMM) routing. The latter offers the highest path diversity by taking a routing decision randomly on each hop. However, the authors state that in order to prevent deadlocks in XY routing, a single domain requires at least one virtual channel. For domains that are elevated to O1TURN routing, a minimum of two virtual channels is required, one for XY and another for YX routes. Additionally, for ROMM, the virtual-channel requirement is dependent on the number of routing hops that packets can take before they reach their destination. Given the mentioned virtual-channel requirements, this approach implies a great area overhead when introducing the additional buffers needed for each virtual channel.

The evaluation of the related work shows that approaches found in the literature aiming DoS attack avoidance are not suitable for multi-tenant NoC-based MPSoC. Approaches presented in [15] and [75] target only a few resources, not being able to secure all tenants. The approach in [26] is only effective against PIR-based DoS attacks. Finally, the approach in [17] does protect a NoC-based MPSoC against the four BRDoS attacks considered for this thesis at the cost of a very high area overhead.

## 5.2 DoS Attack Detection

Approaches for DoS attack detection attempt to identify the occurrence of such attacks so that they can subsequently be mitigated.

In [23], Diguet et al. tackle deadlock and livelock DoS attacks in NoCs. They obtain deadlock-freeness by implementing connection-oriented transactions with end-to-end flow control in the transport layer as well as source routing in the network layer. This is complemented with livelock-freeness by placing the number of hops required to reach the destination in the header of the packets. Such value is decremented on each hop expecting that it reaches zero on its destination, otherwise, the packet is deleted. However, source routing does not only increase the complexity of data transmission due

to the calculation of routes but also the overhead of the packet size by adding routing instructions for each router in the path. Issues that are overcome with the use of logic-based distributed routing as explained in Section 2.2.3. Additionally, livelock-freeness can also be achieved by restricting logic-based distributed routing to use minimal routing. Finally, the approach proposed by Diguet et al. in [23] is not applicable for Flooding nor Low-and-slow DoS attacks, which are the focus of this thesis.

In [66], Fiorin et al. present buffer occupancy monitoring at the NI as a way for detecting PIR-based Flooding DoS attacks. With the use of expected-buffer-occupancy thresholds defined during design time, deviating behavior is considered anomalous and reported to a centralized security manager. Such a manager takes care of all the warning signals coming from the different monitors in the system. However, the authors define occupancy monitoring as "the number of elements in the buffers" but do not specify if it refers to communication requests, packets, or flits. Additionally, threshold values are defined during design time which would not work on a multi-tenant NoC-based MPSoC since it should be able to execute applications with different traffic generation footprints. Finally, alerts are sent to a centralized security manager but it is not clear how it classifies a reported behavior as an actual attack.

Later, in [24], Fiorin et al. propose another monitoring system for detecting security violations carried out against a NoC-based system. Probes embedded in NIs monitor traffic when inserted by PEs. Bandwidth is calculated over a time window whose length can be set not only by the designer at design time but also by a network security manager at run-time. The Network Security Manager is in charge of collecting security alerts coming from probes as well as elaborating appropriate countermeasures to attacks and problems detected, which can be stopped or limited based on the location of the probes. Since detection relies on the ability of a malicious execution to inject more traffic than a threshold, it can be canceled by a congested NoC using fair arbitration such as round robin. This is because the buffer in the NI may overflow due to the reduced forwarding rate of malicious packets in the route.

In [76], Grammatikakis et al. target distributed DoS attacks via a firewall in control of configurable access rules in the NI. The security risk is evaluated by the product of frequency and magnitude of losses (by dropping the packets at the NI). Although such a firewall does not allow the DoS attack to be effective, it was designed to protect the destination PE (i.e., the on-chip memory), thus it does not detect the source of the attack, nor does it kill the attack-related traffic congesting the network.

In [27], Achballah et al. divide MPSoC resources into two zones as Evain et al. in [15] (i.e. secured and non-secured). However, instead of isolating traffic from both zones using virtual channels, they place firewalls on the physical network links that join both zones. Such firewalls, apart from preventing unauthorized access to secured resources, monitor the occupation time of the physical link where they are located. By doing so, if the occupation time of the physical link exceeds a maximum threshold of successive usage, the traffic is halted and the system administrator is informed. Even though malicious traffic generated at the non-secure zone would be banned from entering the secured zone, it is not prevented from disrupting traffic within the same zone. Furthermore, implementing such an approach in the context of multi-tenant NoC-based MPSoCs becomes impractical because applications are not identified as malicious before being mapped to a PE, allowing a malicious application to disrupt the executions of other tenants in the same zone.

In [16], Madden et al. focus on detecting PIR-based attacks in NoCs where communication between two PEs is started by transmitting a request-to-send (RTS)

message to the desired destination. They apply a spiking neural network (SNN) for identifying traffic patterns, aiming to differentiate normal from abnormal amounts of communication requests. They assume that each NoC router is connected to a core and state that a specific packet transmission rate can be observed for creating a normal system profile. Consequently, if there are more RTS signals in a certain time period than there would normally be, it can signify an attack. Their approach is tested by executing a video-playing Multi-Media System. The SNN was trained offline for approximately two hours and then it was able to process a single dataset in five minutes. Even though they report good results for the adopted test scenario, their approach would not be applicable in the threat model presented in this thesis in Chapter 4 due to the expected continuously varying traffic profiles generated by different applications submitted by different users.

In [18], Charles et al. target PIR-based DoS attacks. Network traffic is analyzed during design time to create communication patterns. With this, an upper bound of packet arrival curves (PAC) at each router and destination packet latency curves (DLC) at each IP are identified. During run-time, traffic monitors included in routers evaluate if the established communication bounds are exceeded to determine the presence of an attack. Once an attack has been detected, the IP connected to the detecting router sends additional packets to the routers in the path of delayed packets attempting to locate the attack's source. Such work was later enhanced considering multi-source scenarios in [19]. Despite the presented effectiveness results, such an approach is also not applicable to the threat model presented in this thesis in Chapter 4. Similar to the solution presented by Madden et al. in [16], the establishment of communication bounds is required. However, the nature of the target multi-tenant execution environment is to be dynamic, continuously starting and finishing application executions from different users, hence no long-lasting communication bounds can be established.

In [28], Yao et al. reference [1] and present an approach for detecting Hardware Trojans (HTs) in third-party PEs of MPSoCs. Such HTs are assumed to execute PPL-based DoS attacks targeting the communication performance of packets generated by other PEs. As in [1], their approach also monitors the end-to-end communication delay of packets and attempts to detect the router where the attack enters the sensitive communication path. For the detection of such HTs, they use machine learning techniques based on random forest. Despite their approach presenting a precision detection range above 79%, as other machine learning approaches, it requires the identification of a normal traffic profile so that anomalies would be tagged as attacks. However, such a normal traffic profile is not considered to be present in Multi-tenant NoC-Based MPSoCs.

In [20], Sinha et al. present a framework called Sniffer. As in this thesis, they consider that malicious code running on an IP block can attack other on-chip modules through the shared NoC. In order to mitigate this issue, their framework employs a machine learning-based approach that tunes multiple network parameters to appropriate thresholds, attempting not only to attack detection but also accurate localization of malicious PEs. Three features were considered: i) Buffer Waiting Time (BWT), ii) Inter-Flit-Interval (IFI), and Virtual Channel Occupancy (VCO). Moreover, they propose a collective decision-making strategy by considering the neighbors' opinions at every intermediate node during the localization process to increase the accuracy of localization. Despite achieving an average of 96.754% detection accuracy, experiments consider the execution of a single application which would not be the case for Multi-tenant NoC-Based MPSoCs.

In [21], Pan et al. propose the use of machine learning for detecting PIR-based DoS attacks. They use hardware performance counters as well as trace buffers and perform on-chip network traffic analysis. Since the extracted features are in a flit level instead of a packet level, such as the number of flit arrivals, their approach would also be applicable for detecting PPL-based DoS attacks. They report good results for the adopted experiment's setup. Nevertheless, similar to other published approaches based on machine learning, it requires a normal behavior to be present for training during design time. As shown in Chapter 4, this thesis targets environments with constantly changing traffic profiles due to multi-tenancy.

In [22], Sudusinghe et al. introduce a machine learning-based runtime monitoring mechanism aiming to detect PIR-based DoS attacks in NoCs. They explore different machine learning models and evaluate several traffic features of patterns generated by various application mappings. Machine learning models are trained during design time. Subsequently, trained models are stored in a dedicated IP denoted as the Security Engine. Despite considering various application mappings to evaluate their approach, such mappings were done for a single application. Moreover, the use of machine learning requires that a normal behavior can be identified and learned. However, this is not the case for multi-tenant execution environments, the focus of this thesis, due to the randomness of users and the submitted applications at any given time.

## 5.3 Beyond State of the Art

Sections 5.1 and 5.2 summarize related work found with the conducted literature review. Such papers are related to BRDoS attacks in NoCs and/or MPSoCs, an attack type that was reported as the main attack a malicious user can attempt to disrupt executions from other tenants while running on a multi-tenant NoC-based MPSoC. Table 7 lists these papers pointing out their main characteristics: publication year, intended action, targeted attack type, proposed approach, and an evaluation with pros and cons. Moreover, actions include Avoidance (A), Detection (D), Localization (L), and Mitigation (M). Additionally, targeted attacks are numbered as follows:

1. Packet Injection Rate based (PIR-based) Flooding DoS (FDoS);
2. Packet Payload Length based (PPL-based) Flooding DoS (FDoS);
3. Jellyfish Inter-flit Delay Variance Low-and-slow DoS (LSDoS);
4. Slowloris or Incomplete Packet Transmission (IPT) Low-and-slow DoS (LSDoS).

Furthermore, a check mark (✔) symbolizes either that the attempted action could be achieved in a multi-tenant NoC-based MPSoC or that the attack was successfully taken care of. On the other hand, a dot (●) states that the attempted action was not completely achieved or that the attack was not completely covered in the context of multi-tenant NoC-based MPSoCs.

Based on the format of Table 7, Table 8 lists the characteristics of the approaches proposed in this thesis for answering research question RQ2 (refer to Section 1.2).

Since the main impact caused by a BRDoS attack on paths in a NoC is to delay its traffic, the first two proposed approaches were based on communication degradation monitoring and are explained in Sections 6.1 and 6.2 for Collision Point Router Detection (CPRD) and Collision Point Direction Detection (CPDD), respectively. Their simplicity gives them an advantage when compared to the related works. However, most related works may only be used for detecting PIR-based Flooding DoS attacks, while CPRD and

*Table 7: Related Work Summary*

| Ref. | Year | Action* | | | | Target Attack** | | | | Approach | Evaluation |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | A | D | L | M | 1 | 2 | 3 | 4 | | |
| [15] | 2005 | ● | | | | ● | ● | ● | ● | Virtual channels | ✚ Traffic in the secured zone is not disrupted by external packets<br>− Attacks are not neutralized<br>− Incremented hardware complexity of NoC routers |
| [23] | 2007 | | ✔ | | | | | | | Hops credit at packets' header (Livelock)<br>+ Source Routing (Deadlock) | − Uses Source Routing<br>− Only targets DoS attack enabled by non-minimal routing |
| [66] | 2007 | | ✔ | ✔ | | ✔ | | | | Monitoring buffer occupancy at the NI<br>+ centralized security manager | ✚ Distributed monitoring<br>− Buffer occupancy thresholds set at design time |
| [24] | 2008 | | ✔ | ✔ | ● | ✔ | | | | Bandwidth monitoring at the NI<br>+ centralized network security manager | ✚ Distributed monitoring<br>✚ Possibility of mitigating DoS attacks<br>− Fair arbitration at the router affects detection |
| [75] | 2010 | ● | | | | ● | ● | ● | ● | Path isolation (critical packets use circuit switching while others packet switching) | ✚ Prevents malicious packets from disrupting critical packets<br>− Protects traffic only from a few selected sources |
| [26] | 2015 | ✔ | | | | ✔ | | | | Random arbitration + adaptive routing | ✚ Attacks do not disrupt specific paths<br>− Effective only against PIR-based flooding DoS attacks |
| [76] | 2015 | | | | ✔ | ● | ● | ● | ● | Firewall at the NI of traffic initiator nodes | ✚ Can block DoS attacks at the source<br>− No detection mechanism is presented |
| [27] | 2017 | | ● | | ● | ✔ | ✔ | | | Monitoring the occupation time of NoC links | ✚ Neutralization of abnormal transmissions<br>− Created to monitor a few links not all |
| [17] | 2018 | ✔ | | | | ✔ | ✔ | ✔ | ✔ | Virtual channels<br>+ concurrent routing algorithms | ✚ Parallelizes traffic from different sources<br>− Big area overhead due to several virtual channels |
| [16] | 2018 | | ✔ | | | ✔ | | | | Monitoring transmission requests<br>+ spiking neural networks | ✚ Can detect PIR-based DoS attacks when normal behavior is known<br>− Not suitable for Multi-tenant NoC-Based MPSoCs |
| [18] | 2019 | | ✔ | ✔ | | ✔ | | | | Monitoring packet arrival periodicity<br>+ request information from other routers | ✚ Can detect PIR-based DoS attacks when normal behavior is known<br>− Not suitable for Multi-tenant NoC-Based MPSoCs |
| [19] | 2020 | | ✔ | ✔ | | ✔ | | | | Monitoring packet arrival periodicity<br>+ request information from other routers | ✚ Can detect PIR-based DoS attacks when normal behavior is known<br>− Not suitable for Multi-tenant NoC-Based MPSoCs |
| [28] | 2020 | | ✔ | ✔ | | ✔ | ✔ | | | Monitoring communication degradation<br>+ machine learning (Random Forest) | ✚ Can detect DoS attacks when normal behavior is known<br>− Not suitable for Multi-tenant NoC-Based MPSoCs |
| [20] | 2021 | | ✔ | | | ✔ | | | | Virtual channels<br>+ random forest machine learning | ✚ Detects anomalies to known behavior<br>− Works only for the trained traffic profile |
| [21] | 2021 | | ✔ | | | ✔ | ✔ | | | Hardware performance counters<br>+ embedded trace buffers<br>+ on-chip network traffic analysis | ✚ Detects anomalies to known behavior<br>− Works only for the trained traffic profile |
| [22] | 2021 | | ✔ | | | ✔ | | | | Distributed traffic monitoring<br>+ centralized machine learning | ✚ Detects anomalies to known behavior<br>− Works only for the trained traffic profile |

*** Action:** **A**-Avoidance, **D**-Detection, **L**-Localization, **M**-Mitigation
**** Target Attack:** **1**-PIR-based DoS, **2**-PPL-based DoS, **3**-Jellyfish, **4**-Slowloris

*Table 8: Proposed Work Summary*

| Name | Year | Action* | | | | Target Attack** | | | | Approach | Evaluation |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | A | D | L | M | 1 | 2 | 3 | 4 | | |
| CPRD Section 6.1 | 2018 | | ✔ | ● | | ✔ | ✔ | ✔ | | Monitoring communication degradation | ✚ Lightweight solution<br>✚ Determines collision point<br>− Does not detect the exact source of attack |
| CPDD Section 6.2 | 2019 | | ✔ | ● | | ✔ | ✔ | ✔ | | Monitoring communication degradation | ✚ Lightweight solution<br>✚ Generates a list of possible attack sources<br>− Does not detect the exact source of attack |
| LSDAA Section 7.1 | 2019 | | ✔ | ✔ | ✔ | | | ✔ | ✔ | Monitoring emptiness of router FIFO buffers | ✚ Detects and mitigates LSDoS attacks<br>✚ Very lightweight solution<br>− Only against LSDoS attacks |
| CDT Section 6.4 | 2021 | ✔ | ✔ | ✔ | | ✔ | ✔ | | | Round robin arbitration on packet level<br>+ Reporting communication disruptions | ✚ Avoids PIR-based DoS attacks<br>✚ Detects the exact source of PPL-based DoS attacks<br>− Only effective against Flooding DoS Attacks |
| ADAD Section 6.3 | 2022 | | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | | Monitoring communication degradation<br>+ Software-defined NoC | ✚ Detects the exact source of attack<br>− Temporarily restricts adaptive routing of NoC zones |
| BCPE Section 7.2 | 2022 | ✔ | | | | ✔ | ✔ | ✔ | ✔ | Enforcing bandwidth control policies | ✚ Lightweight solution<br>✚ Prevents NoC disruption by BRDoS attacks |

*** Action:** **A**-Avoidance, **D**-Detection, **L**-Localization, **M**-Mitigation
**** Target Attack:** **1**-PIR-based DoS, **2**-PPL-based DoS, **3**-Jellyfish, **4**-Slowloris

CPDD can be used for PIR- and PPL-based Flooding DoS attacks as well as Jellyfish Low-and-slow DoS attacks. Furthermore, with CPRD, an MPSoC can take a step forward toward the localization of the attack source for its mitigation. This is because it helps to determine the router where malicious data disrupted a monitored path. The CPDD approach goes even one step further than CPRD, by providing information useful for detecting the exact attack source or at least a list of possible attack sources.

Subsequently, in order to complement CPDD, Active DoS Attack Detection (ADAD) was proposed. When the information provided by CPDD is not sufficient for detecting the exact source of the attack, but a list of suspects, this approach makes use of software-defined NoCs and small communication packets to locate the exact source of the attack. Details are given in Section 6.3.

Furthermore, with the experiments executed along the development of approaches that give a better answer to research question RQ2 regarding flooding DoS attacks, it was realized that routers implementing a fair arbitration scheme mitigate the effect of single-source PIR-based DoS attacks. Thus for Flooding DoS attack detection, efforts should be focused on PPL-based DoS attacks (untouched by most related works). Therefore, it was determined that an alternative against Flooding DoS attacks was to implement Round-Robin arbitration to avoid PIR-based DoS attacks in conjunction with Collision Disruption Tattletaling (CDT), enabling MPSoCs to detect also PPL-based DoS attacks as well as to locate the attack source for its mitigation.

During a study of DoS attacks done out of the scope of NoC-based MPSoCs, two attacks were identified as portable to such context. Since related works only focuses on flooding DoS attacks, they expect that an attack is caused either by a greater-than-normal transmission of communication requests or data packets. Therefore, MPSoCs would remain vulnerable to Low-and-Slow DoS attacks which are able to cause the same or greater disruption with normal or less-than-normal transmission of communication requests or data packets. In order to tackle this, approaches against flooding DoS attacks can be complemented with the proposed Low-and-Slow DoS Attack Avoidance (LSDAA) scheme presented in Section 7.1.

As can be seen in Table 7, DoS attacks in NoC-based MPSoCs gained a greater relevance starting in 2018. Also, most approaches focus on detection rather than avoidance. Moreover, half of those approaches propose machine learning for learning a “normal behavior" of traffic profiles so that anomalies can be later detected. Considering that in a multi-tenant environment, as the one considered for this thesis, any user is able to submit any type of application, the traffic flowing through the NoC at a given time will be any combination created by the current users. Consequently, no permanent traffic profile could be labeled as normal or malicious. Therefore, machine learning as a means for DoS attack detection in multi-tenant NoC-based MPSoCs was discarded. As explained in Section 7.2, bandwidth control policies can be derived from application scheduling contracts, which when enforced for each transmission, can guarantee the avoidance of the four targeted BRDoS attacks.

# 6 DoS Attack Detection

Approaches presented in this chapter aim to provide better answer to research question RQ2 than the ones found in related work attempting detection of BRDoS attacks in NoC-based MPSoCs, summarized in Section 5.2. The first approach, named Collision Point Router Detection (CPRD), is presented in Section 6.1. An enhancement towards the localization of an attack source, named Collision Point Direction Detection (CPDD), is presented in Section 6.2. As a final step in the same direction, an Active DoS Attack Detection phase is presented in Section 6.3. Also, a different approach seeking the same objective, named Communication Disruption Tattletaling, is presented in Section 6.4. A comparison of such approaches with related works is presented in Section 5.3.

## 6.1 Collision Point Router Detection

This section presents the first approach seeking to answer research question RQ2 (Section 1.2), namely CPRD, which attempts to detect DoS attacks in NoC-based MPSoCs. CPRD is a distributed DoS detection scheme that is able to measure the performance degradation of monitored data and detect the router where malicious packets collide/disrupt the communication path (a.k.a. Collision Router - CR). As explained in Section 3.1, a Flooding DoS attack is effective in a NoC when the malicious packets constantly collide with other packets, i.e. when they compete for a router's output and the legitimate packets are prevented from being forwarded.

When the CPRD method was designed, only Flooding DoS attacks (i.e. PIR- and PPL-based DoS attacks) had been identified as BRDoS attacks that could be executed by a user of a multi-tenant NoC-based MPSoC. However, it was determined later that CPRD is equally effective for detecting the Jellyfish Inter-flit Delay Variance (Jellyfish) Low-and-slow DoS attack because of the disruption equivalence of the Jellyfish and the PPL-based DoS attacks (Sections 3.1 and 7.1.2.2).

The remainder of this section is organized as follows: Section 6.1.1 presents the proposed architecture for implementing the CPRD scheme. Section 6.1.2 details the setup of experiments and discusses the obtained results. Section 6.1.3 presents an overhead evaluation of the proposed mechanism in terms of area, critical-path delay, and power. Finally, Section 6.1.4 concludes the section.

### 6.1.1 Proposed Architecture

In order to achieve a scalable NoC Architecture that is not only able to detect a DoS attack, but also the router where it disrupts the communication of legitimate packets, this section presents three small modifications to the Secure Bonfire platform (Section 2.2.4.2). The first two of these modifications are regarding the structure of the communication packets and the third relates to the architecture of the routers.

As shown in Figure 29, one of the modifications to the structure of the communication packets refers to the definition of the *Last Body Flit*, which will now carry a time-stamp of the packet's generation time. The time stamp is introduced for enabling the system to calculate the end-to-end latency of each packet, and based on it, detect if a DoS attack is taking place. The second modification is related to the *Tail Flit* which now carries the address of the router where the packet waited for the most for other packets to be forwarded through a required output port (i.e. *Max Latency Router Address*), and the number of clock cycles elapsed while waiting (i.e. *Max Latency Value*). This information is evaluated in every router by a proposed DoS Monitor and updated in case the waiting time in the current router is longer than the one stored in the packet.

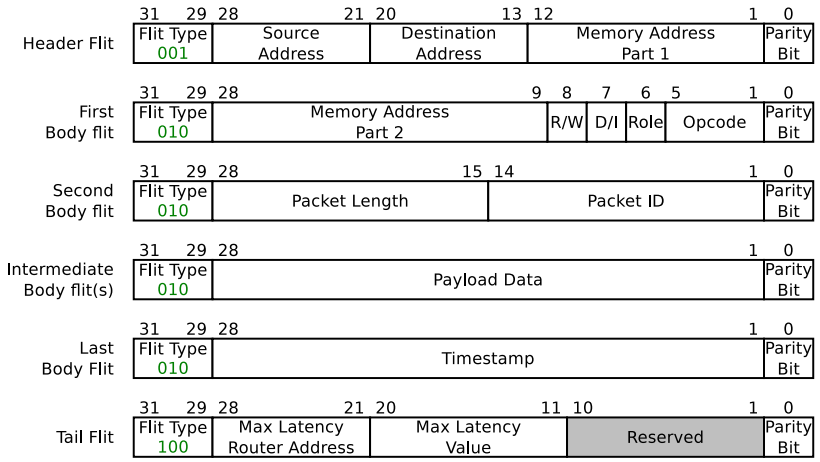



*Figure 29: Packet Structure for CPRD*

As mentioned earlier and shown in Figure 30, the third modification is to the router's architecture, where a proposed *DoS Monitor* is added to the beginning of the router's data path (i.e. before each of the FIFO buffers). Such a monitor enables the NoC to distributively accuse the router where malicious traffic disrupts the path of monitored communication. And by following such a distributed approach, it is easily scalable for bigger NoC-based MPSoCs. The internal architecture of the proposed *DoS Monitor* is depicted in Figure 31.

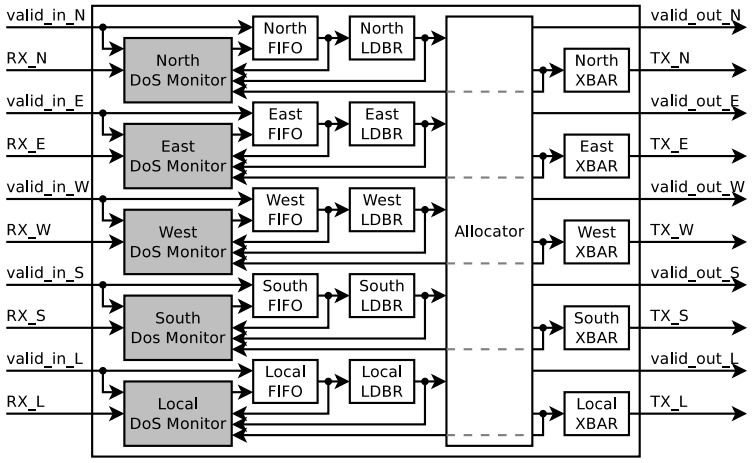



*Figure 30: Router architecture for CPRD*

The proposed *DoS Monitor*, illustrated by Figure 31, starts a 10-bit counter every time a new Header flit arrives to the input of the FIFO and stops it when the Header flit leaves the FIFO's output. The counter is incremented on each clock cycle while other packets are forwarded through their required output port. This is done by monitoring the output grants provided by the router's Allocator circuit filtered by the request sent by the LDBR circuit. Furthermore, once the tail flit of the packet arrives, the counter value is compared against the *Max Latency Value* stored in the tail flit. If the counter

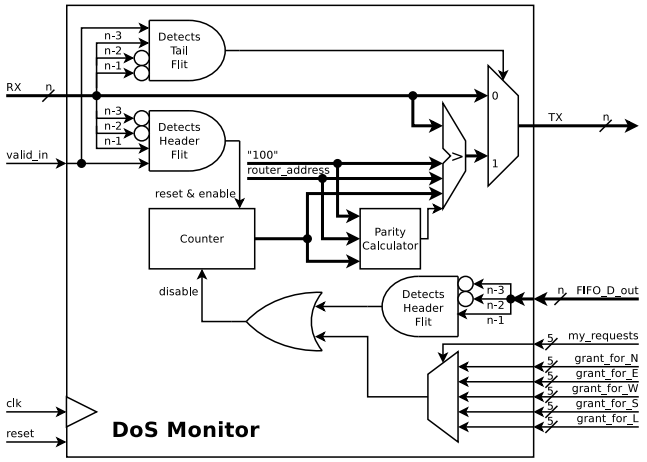



*Figure 31: DoS Monitor Architecture for CPRD*

value is smaller or equal, the tail flit will be forwarded as it is, otherwise, the monitor replaces the stored value with the value of the counter. Additionally, the *Max Latency Router Address* is updated with the current router address, and a new parity for the tail flit is calculated and updated.

Once monitored packets reach their destinations, their end-to-end transmission latency can be calculated based on the packet's time stamp. Subsequently, such value can be compared to that of a maximum expected, and if it is beyond the acceptable limit, the *Max Latency Router Address* and the *Max Latency Value* can also be retrieved from the packet and analyzed together with previous DoS suspicion reports. Finally, a Collision Point Report (CPR) can be sent to the MPSoC's Global Manager (GM) so that it can decide on further action.

### 6.1.2 Experimental Work

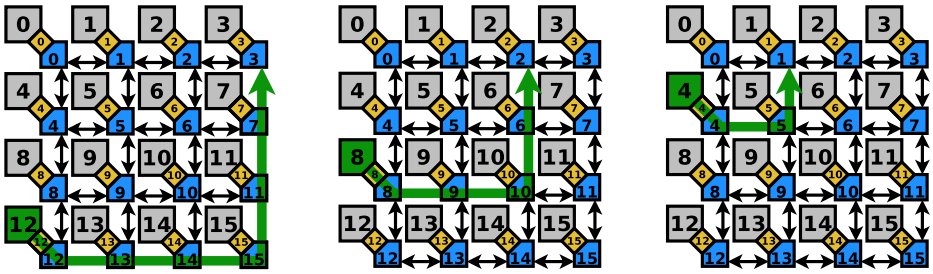
The DoS Monitor, presented in Section 6.1.1, was implemented in RTL level and integrated into the Secure Bonfire framework. Subsequently, the proposed NoC architecture with the distributed DoS attack detection system was simulated using *Modelsim* from *Mentor Graphics* [77].

The remainder of this section is organized as follows: Section 6.1.2.1 details the configuration of the simulation scenarios. Section 6.1.2.2 presents the results obtained on the experiments evaluating the effect of the attacker's PIR, while Section 6.1.2.3 discusses the effect of the attacker's PPL and its location.

#### 6.1.2.1 Simulation Scenarios

In order to evaluate the performance of the proposed Distributed DoS Detection system, simulations were executed on a $4 \times 4$ mesh NoC-based MPSoC architecture. Three monitored paths were considered and disrupted from all possible attack sources. Figure 32 depicts the three scenarios, one for each path. Traffic generators included in the Bonfire platform were leveraged for simulating normal traffic and DoS attacks. Moreover, the network routers (applying an XY-routing algorithm) use a credit base flow control with fair Round-Robin arbitration (on packet level) and utilize Wormhole switching with 4-flit deep FIFOs. Furthermore, the PIR, PPL, and location of the monitored and malicious packets vary according to the purpose of each experiment.

The adopted PIR values were either $0.003$, $0.01$ (i.e., one packet every $100$ clock cycles), or $0.03$, while the PPL values were $10$, $20$, $30$, or $50$ flits. Additionally, random traffic packets were transmitted with a PIR of $0.01$ and a PPL of $10$ flits, generated for $20$ pseudo-random simulation start seeds.



*(a) Scenario 1* *(b) Scenario 2* *(c) Scenario 3*

*Figure 32: Simulation Scenarios*

Figure 32a shows the first scenario considered for the experiments. The monitored path is represented by the green arrow, which begins in PE 12 and ends in PE 3, going sequentially through routers 12, 13, 14, 15, 11, 7, and 3. Figure 32b describes a scenario where the source and destinations of the monitored path are PE 8 and PE 2, respectively. Hence, monitored packets go through routers 8, 9, 10, 6, and 2. Finally, Figure 32c depicts a scenario where monitored packets are sent by PE 4 and received by PE 1, thus going through routers 4, 5, and 1. It is worth noticing that longer paths provide more points of disruption.

#### 6.1.2.2 Effect of the Attacker's PIR

The experiments presented in this section target the analysis of the DoS attack from two aspects: i) from the point of view of the attacker in order to maximize the effectiveness of the attack, and ii) from the point of view of network protection, evaluating different metrics that can be leveraged for detecting an attack and qualifying the effectiveness of the CPRD mechanism. For the twofold evaluation, the scenario from Figure 32a was adopted, adding an attacker in PE 15 and the attack path 15→3, as illustrated by Figure 33. Moreover, the malicious and monitored packets collide within router $15$ while attempting to be forwarded to their destinations through the north output port of the collision router (CR).

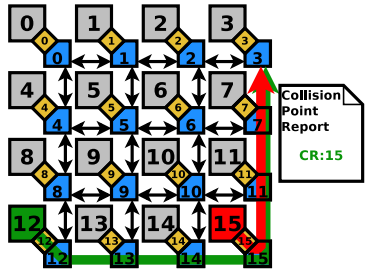



*Figure 33: Scenario 1 + attack path 15→3, collision in router 15*

Within this analysis targeting the effect of the attacker's PIR ($A_{PIR}$), simulations were done considering monitored packets with a PPL of 10 flits (i.e., $M_{PPL} = 10$) and malicious packets with a PPL of 30 flits (i.e., $A_{PPL} = 30$).

As mentioned in [70], an attacker may be able to identify the success/effectiveness of the attack by measuring its own throughput when attempting a PIR-based BRDoS attack. As shown in the first and third column of Table 9, when the intended PIR value of the attacker ($A_{PIR}$) is incremented, an increasing gap with the Attacker's Effective PIR can be seen. An $A_{PIR}$ value of $0.01$ is shown to be over the limit of the network's capacity for the adopted configuration since trying to send a packet with a PPL of $30$ flits ($A_{PPL}$) every $100$ clock cycles resulted in an effective $A_{PIR}$ of one packet every $140$ clock cycles, approximately. Such difference, expressed as a growth percentage related to the intended $A_{PIR}$, is presented in the fourth column of Table 9. In this work, a PIR deviation of $10\%$, seen by the attacker is considered an effective attack.

*Table 9: Attack effectiveness ($A_S$: 15, $A_{PPL}$: 30, $M_{PPL}$: 10, $R_{PPL}$: 10, $R_{PIR}$: 0.01).*

| Attacker PIR_A | Monitored PIR_M | Attacker Effective PIR_A | Attacker PIR_A Deviation (%) | Attack Effective |
|---|---|---|---|---|
| 0.003 | 0.003 | 0.003 | 0 | ✖ |
| 0.003 | 0.01 | 0.003 | 0 | ✖ |
| 0.003 | 0.03 | 0.003 | 0 | ✖ |
| 0.01 | 0.003 | 0.0071 | 29 | ✔ |
| 0.01 | 0.01 | 0.0071 | 29 | ✔ |
| 0.01 | 0.03 | 0.0070 | 30 | ✔ |
| 0.03 | 0.003 | 0.0086 | 71.33 | ✔ |
| 0.03 | 0.01 | 0.0085 | 71.67 | ✔ |
| 0.03 | 0.03 | 0.0086 | 71.33 | ✔ |

As mentioned in Section 6.1.1, it is proposed that each monitored packet carries a time-stamp generated when the packet entered the NoC. Such a time stamp can be used for calculating the end-to-end communication delay upon the packet's arrival. The mean end-to-end delay of the monitored packets for the $20$ pseudo-random traffic simulation seeds in a scenario with no attacker is listed in Table 10 for each $M_{PIR}$ value. This mean value and the sample standard deviation (SSD) are used for detecting DoS attacks if an end-to-end delay goes above a threshold $T$, calculated with (13).

$$T = mean\_delay + 0.5 \times SSD \tag{13}$$

*Table 10: End-to-End delay under no attack ($M_{PPL}$: 10, $R_{PPL}$: 10, $R_{PIR}$: 0.01).*

| Monitored PIR_M | Mean End-to-End Delay (Clock Cycles) | Sample Standard Deviation (Clock Cycles) | Detection Threshold (Clock Cycles) |
|---|---|---|---|
| 0.003 | 88.9056 | 17.3167 | 97.5640 |
| 0.01 | 162.0356 | 43.6219 | 183.8466 |
| 0.03 | 915.5529 | 405.9398 | 1118.5228 |

Furthermore, Table 11 lists the results of the experiments from the network protection's point of view for the same scenario where the attacker is located at router $15$ (Figure 33). Since a DoS attack was present, the end-to-end delay reached values greater than the threshold $T$ (see Equation (13) and Table 10), enabling the diagnosis mechanism to detect the attack. However, for an $A_{PIR}$ of $0.01$, the collision point was not determined. This is because the traffic entering from other inputs and requiring the same output was more or less equal to the one from the attacker. On the other

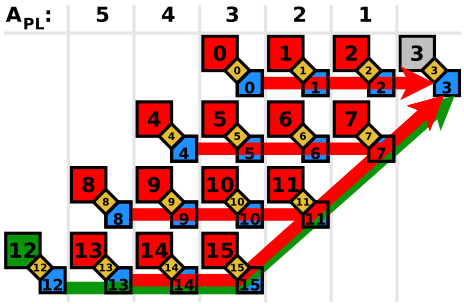
hand, for $A_{PIR}$ configurations where the attacker noticed success by monitoring its own throughput (i.e., $A_{PIR} \geq 0.01$), the proposed DoS CPRD mechanism managed not only to detect the attack but also the point where the malicious traffic intercepted the monitored path. For the scenarios where the collision point router was found, the last column of the table lists the detection confidence.

*Table 11: Attack detection ($A_S$: 15, $A_{PPL}$: 30, $M_{PPL}$: 10, $R_{PPL}$: 10, $R_{PIR}$: 0.01).*

| Attacker PIR_A | Sensitive PIR_M | End-to-End Delay (Clock cycles) | Attack Detected by Attacker | Attack Detected with CPRD | Collision Point Detection Confidence |
|---|---|---|---|---|---|
| 0.003 | 0.003 | 181.331 | ✖ | ✔ | ✖ |
| 0.003 | 0.01 | 765.047 | ✖ | ✔ | ✖ |
| 0.003 | 0.03 | 2679.22 | ✖ | ✔ | ✖ |
| 0.01 | 0.003 | 656.859 | ✔ | ✔ | 1 |
| 0.01 | 0.01 | 3346.42 | ✔ | ✔ | 0.8 |
| 0.01 | 0.03 | 4336.19 | ✔ | ✔ | 0.7 |
| 0.03 | 0.003 | 308.823 | ✔ | ✔ | 1 |
| 0.03 | 0.01 | 2937.8 | ✔ | ✔ | 1 |
| 0.03 | 0.03 | 4318.1 | ✔ | ✔ | 0.95 |

### 6.1.2.3 Effect of the Attacker's Packet Payload Length and Location

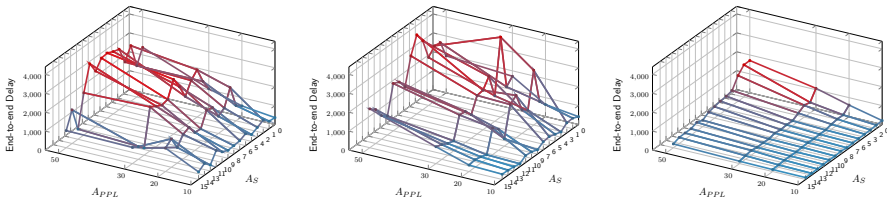
In order to find configurations that would maximize the success of an attack, experiments were done for the three scenarios presented in Section 6.1.2.1 considering four different $A_{PPL}$ values (i.e., 10, 20, 30, and 50 flits). For such experiments, $M_{PPL}$ and $R_{PPL}$ were set to 10 flits and all the PIRs were set to 0.01 (i.e., $A_{PIR} = M_{PIR} = R_{PIR} = 0.01$). Additionally, while maintaining the source and destination of each scenario, the attack source was simulated in all other PEs, which as shown in Figure 34 for Scenario 1, translates to different Attack Path Lengths ($A_{PL}$) and different Collision Routers (CRs). However, in cases where the attack source is directly in the monitored path, the attack source and the CR are the same.




| Attack Source | Collision Router |
|---|---|
| 0, 1, or 2 | 3 |
| 4, 5, 6, or 7 | 7 |
| 8, 9, 10, or 11 | 11 |
| 15 | 15 |
| 14 | 14 |
| 13 | 13 |

*Figure 34: Attack path lengths and collision routers for different attack sources - Scenario 1*

Figure 35 presents the end-to-end delay simulation results of the monitored packets for the combination of the considered $A_{PPL}$ and attack sources ($A_S$), for each of the three selected scenarios. As expected, results show that the end-to-end delay of the monitored packets is proportional to the $A_{PPL}$ value, achieving the highest mean delay values for the 50-flit $A_{PPL}$ configurations. This is because a longer packet manages to retain the grant of a router's output port longer, preventing other packets from being forwarded. Regarding the $A_S$, it can be seen that the highest end-to-end transmission delays for monitored packets were caused as a result of injecting the malicious traffic into the routers closest to the monitored destination.




(a) Scenario 1 (b) Scenario 2 (c) Scenario 3

*Figure 35: End-to-end delay vs attacker's packet payload length ($A_{PPL}$) for different attack sources ($A_S$) using XY Routing*

### 6.1.3 Area and Power Overhead

The proposed architecture was synthesized using the $0.18\mu m$ AMS library and Synopsys design vision at 200 $MHz$. Area overhead and critical path delay of the proposed architecture compared to the baseline architecture (Secure Bonfire) are reported in Table 12. The critical path delay overhead of the proposed method is negligible and the proposed monitors only add $17\%$ area overhead to the minimalist router area (each monitor only adds $3.4\%$ overhead to the router's area). The main contribution to area overhead is caused by the counter registers of each DoS Monitor.

*Table 12: Area and critical path delay overhead.*

| Router | Area | | | | Critical Path Delay (ns) |
|---|---|---|---|---|---|
| | Sequential ($\mu m^2$) | Combinational ($\mu m^2$) | Total ($\mu m^2$) | Overhead (%) | |
| Baseline | 48378.7 | 42669.0 | 91047.8 | – | 4.82 |
| CPRD | 52033.7 | 55129.9 | 107163.7 | 17.7% | 4.8 |

Power analysis of the proposed method has been performed for random uniform traffic with a packet injection rate of $0.01$ for the baseline architecture and the proposed architecture (without the presence of an attacker, see Figure 32). The results of these experiments are reported in Table 13. Results show that the proposed approach adds $5\%$ power overhead to each router that includes five monitors.

*Table 13: Power overhead.*

| Router | Switching Power (mW) | Internal Power (mW) | Leakage Power (mW) | Total Power (mW) | Total Overhead (%) |
|---|---|---|---|---|---|
| Baseline | 0.151 | 3.663 | 0.261 | 3.814 | – |
| CPRD | 0.273 | 3.374 | 0.327 | 4.008 | 5% |

### 6.1.4 Conclusion

This section presented the first approach proposed for detecting DoS attacks in multi-tenant NoC-based MPSoCs and answer research question RQ2, namely Collision Point Router Detection (CPRD) [1]. It is a distributed DoS detection scheme that measures the performance degradation of monitored data transmissions under flooding BRDoS attacks, which can also be used for detecting the router where the malicious packets disrupted the monitored path. An exploration of the effect of different attack configurations was also presented. Such configurations include different packet injection rates, packet payload lengths, and attack sources. Experimental work showed that malicious packets

with a longer payload and intercepting the monitored path closer to its destination, cause a greater communication disruption; additionally, that a combination of two delay metrics can be leveraged for not only detecting a DoS attack but also identifying its entry point into the monitored path with high accuracy.

As explained in Section 5.3, the CPRD is able to detect more types of attacks than most of the approaches presented in related works. Moreover, it is shown in Section 6.1.3 that small area and power overheads are incurred for its implementation.

As a step forward towards detection and localization of BRDoS attacks in addition to locating a collision router, Section 6.2 presents an approach able to either find the exact attack source or generate a list of suspects.

## 6.2 Collision Point Direction Detection

The approach presented in this section is named Collision Point Direction Detection (CPDD) which is an improvement of the CPRD scheme detailed in Section 6.1. The aim of this approach is not only to find the router where malicious traffic disrupts the monitored communication (i.e., the collision router - CR), but also to determine a list of Possible Attack Sources (PAS - i.e., the PEs where the attack might be originated). To this end, the input and output ports of the CR used by the malicious traffic are also determined. By combining such information, results show that the list of PAS can be reduced by a maximum of 69%, thus lowering the required effort of an MPSoC's Global Manager for locating the source of a BRDoS attack.

This section is organized as follows: Section 6.2.1 presents the proposed architecture for implementing the CPDD scheme. Section 6.2.2 details the Collision Point Report (CPR). Section 6.2.3 explains how a list of PAS can be determined based on a CPR, showing the improvement of CPDD when compared to CPRD. Section 6.2.4 details the setup of experiments and discusses the obtained results. Section 6.2.5 presents an evaluation of the overhead of the proposed mechanism in terms of area, critical-path delay, and power. Finally, Section 6.2.6 concludes the section.

### 6.2.1 Proposed Architecture

The CPDD architecture is similar to that of the CPRD (Section 6.1.1), but extends it to report the direction of the malicious traffic through the CR (i.e., the input and output ports of the CR). As shown in Figure 36, the packet format has been slightly changed to include the inputs competing to enter the monitored path and the output for which they competed. The proposed architecture of the router for CPDD remains the same as CPRD (Figure 30). However, the architecture of the DoS monitor is changed to report the direction of competing packets, as shown in Figure 37. Now, if the latency value stored in the packet's tail flit is less than the value of the local latency counter, the monitor updates the tail flit not only with the router address and the waiting time but also with the competitors and the output for which they competed.

For adding the competitors in the *Tail Flit* of monitored packets, a 5-bit competitors log was added to the DoS monitor. Such log records all the input ports that acquired a forwarding grant to the required output while the monitored packet waits its turn.

Once a monitored packet reaches its destination, similar to the CPRD approach, the firmware of the MPSoC-powered device calculates the end-to-end delay based on the timestamp contained in the packet. Provided that the calculated value exceeds the threshold, the collision point is determined. However, for the CPDD approach, the firmware will also extract the direction information to narrow down a list of PAS, which can later be used for localizing the compromised PE.

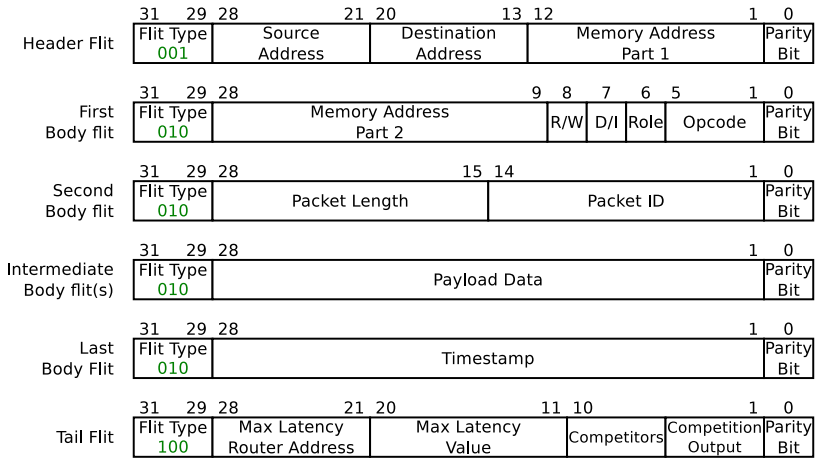



*Figure 36: Packet Structure CPDD*

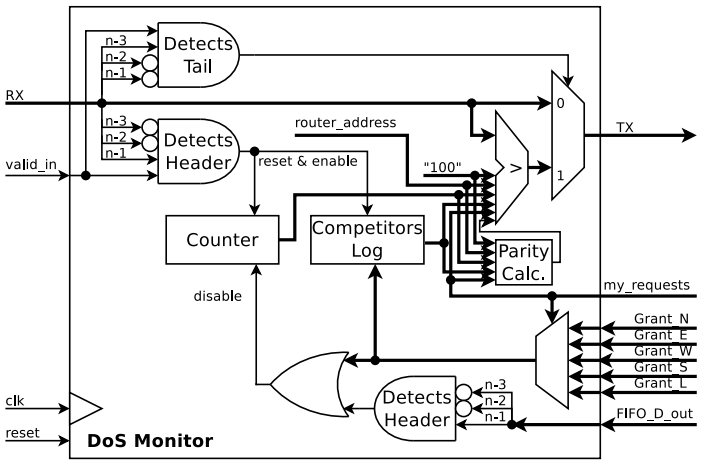



*Figure 37: DoS Monitor Architecture CPDD*

### 6.2.2 Collision Point Report

As presented in [2] and also summarized in this section, by introducing communication-monitoring information into the *Tail Flit* (see Figure 36) during the transmission of packets and retrieving it when they are received, it is possible for the destination NI or firmware to generate a CPR from where a list of PAS can be generated.

A CPR contains: i) the router in which the packet had its worst collision (CR), determined by: ii) the greatest amount of time the packet waited in the CR while the required output port was busy (CW); iii) the input ports that won the competition for the required output port while the packet waited in the CR ($IP_C$); and iv) the output port for which the traffic flows competed ($OP_A$). Both, $IP_C$ and $OP_A$ are expressed as 5-bit binary words following a *NEWSL* port convention (i.e. North, East, West, South, Local). Furthermore, the single-packet collision report can be processed by the destination NI as follows:

A competition or collision degree (CD) can be calculated by identifying the number of competitors reported at the CR (i.e. the number of ones in $IP_C$). According to

the load of the NoC, such a degree will obtain values from 0 to 3 (considering that no U-turns are allowed and that the input of the reporting packet cannot compete with itself). By comparing the CD and the CW values to the thresholds established by the GM, the NI can label the evaluated single-packet collision report as an anomaly and store it for further analysis.

In the second phase of the analysis, when the NI has gathered related anomalies determined by the GM, the related anomalies are evaluated together to find the input port reported the most as a competitor, which is considered as the port through which malicious traffic entered the CR ($IP_A$). Thus, in a successful DoS attack, the packets of a legitimate source will constantly collide with the malicious traffic flowing through its path, rather than with other discontinued legitimate bursts of packets.

At this point, the NI has identified the router where the monitored and malicious traffic collided (aka CR), the input port through which the malicious traffic entered the CR (aka $IP_A$), and the output for which the two traffic competed (aka $OP_A$). Such information, from here on after referred to as the Collision Point Report (CPR), can be processed for generating a list containing the $PAS$. The process of identifying the PAS is explained next in Section 6.2.3.

### 6.2.3 List of Possible Attack Sources

As explained before, when an NI detects recurrent NoC communication anomalies, it can generate a CPR containing a router labeled as CR as well as the input and output ports of the CR used by traffic tagged as malicious (i.e. $IP_A$ and $OP_A$, respectively). As presented in [2] and in this Section, based on such information gathered during the normal communication over the NoC, a list of PAS can be created. An illustrative example is presented in Section 6.2.3.1. Additionally, formal definitions are presented in Section 6.2.3.2 where the dimensions of the NoC and the current routing constraints are also taken into consideration for reducing the PAS.

Figure 38 shows the groups of routers that can be tagged as PAS relative to the CR. Such groups are labeled according to their location in the NoC as well as the location of the CR. Labels correspond either to directions (i.e., North, East, West, and South) or quadrants (i.e., North-East, North-West, South-East, and South-West).

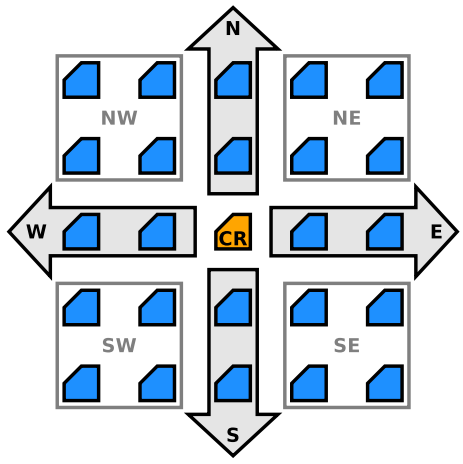
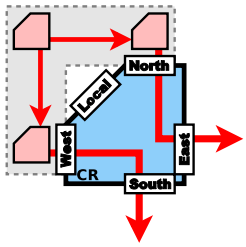
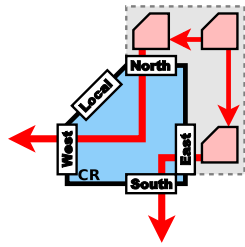



*Figure 38: Directions and quadrants of a NoC relative to a collision router*

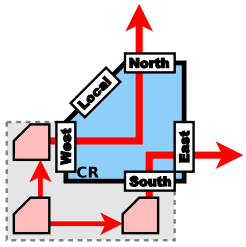
Furthermore, routers in directions and/or quadrants are labeled as PAS depending on the path taken by malicious traffic inside the CR. Figure 39 illustrates the PAS if the path followed by the malicious packets inside the CR corresponds to a turn. On the other hand, Figure 40 depicts the PAS when the malicious traffic was forwarded to the Local output port of the CR or if the path corresponds to a straight path.



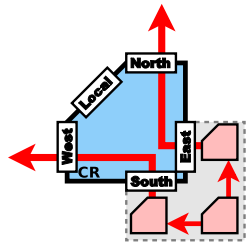

(a) PAS = (N + NW) or (W + NW)




(b) PAS = (N + NE) or (E + NE)

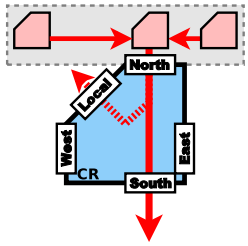



(c) PAS = (W + SW) or (S + SW)

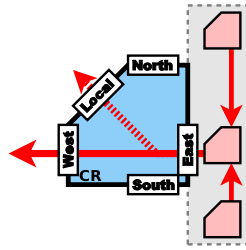



(d) PAS = (E + SE) or (S + SE)

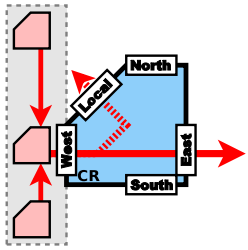
*Figure 39: List of Possible Attack Sources - turns*



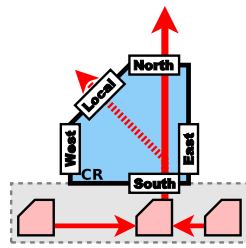

(a) PAS = NW + N + NE




(b) PAS = NE + E + SE




(c) PAS = NW + W + SW




(d) PAS = SW + S + SE

*Figure 40: List of Possible Attack Sources - straight path or destination*

#### 6.2.3.1 Illustrative Examples

This section explains the relation between the CPR and the PAS via simple examples based on the XY routing algorithm [78]. However, as shown later in this chapter, experimental results using adaptive routing are also provided. In order to notice the enhancement done to the CPRD through the CPDD, the illustrative examples adopt the scenarios presented in Section 6.1.2.1.

Figure 41 shows the first scenario considered for this exercise. The monitored path is represented by the green arrow, which begins in router 12 and ends in router 3, going also through routers 13, 14, 15, 11, and 7. For such a case, it is considered that neither router 12, nor 3 can attack such a path.

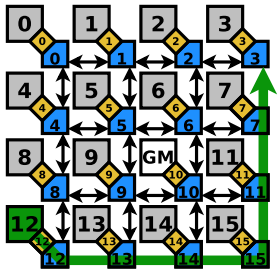



*Figure 41: Example 1 - Scenario*

In case Router 13 is labeled as the CR in the CPR, there would be only one possible attack source, namely PE 13, since the traffic of no other PE could request the East output port of Router 13, when implementing XY routing. However, in more sophisticated cases, the set of candidates can be larger (as listed in Table 14). It is important to note that the GM excludes the nodes that have a higher chance of collision in their own router. For example, if the reports label Router 14 as the CR, the GM (Router 10) will not include PE 13 as a suspect since it has a better chance of interference on the East output port of Router 13. Additionally, the location of the GM will also not be considered as a suspect.

*Table 14: Possible Attack Suspects for Example 1*

| Collision Router | Suspects using CPRD [1] | $IP_A$ | $OP_A$ | Suspects using CPDD [2] |
|---|---|---|---|---|
| 13 | 13 | L | E | 13 |
| 14 | 14 | L | E | 14 |
| 15 | 15 | L | N | 15 |
| 11 | 8, 9, 10, 11 | W | N | 8, 9, 10 |
| | | L | N | 11 |
| 7 | 4, 5, 6, 7 | W | N | 4, 5, 6 |
| | | L | N | 7 |
| 3 | 0, 1, 2 | W | L | 0, 1, 2 |

Figure 42 describes a scenario where the source and destinations of the monitored path are nodes 8 and 2, respectively. This example is different from the previous example in one important aspect: the monitored path can be disrupted from both sides. This in turn will increase the PAS for a given CR. As can be seen in Table 15, CR 10 is a descriptive example of such a situation. However, knowing the direction of the interference will further decrease the list of PAS (from 6 PEs to a number of 4 PEs in the worst case, and in the best case scenario down to a single PE).

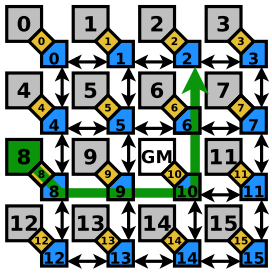



*Figure 42: Example 2 - Scenario*

*Table 15: Possible Attack Suspects for Example 2*

| Collision Router | Suspects using CPRD [1] | $IP_A$ | $OP_A$ | Suspects using CPDD [2] |
|---|---|---|---|---|
| 9 | 9 | L | E | 9 |
| 10 | 10, 11, 12, 13, 14, 15 | E | N | 11 |
| | | S | N | 12, 13, 14, 15 |
| | | L | N | 10 |
| 6 | 4, 5, 6, 7 | E | N | 7 |
| | | W | N | 4, 5 |
| | | L | N | 6 |
| 2 | 0, 1, 3 | E | L | 3 |
| | | W | L | 0, 1 |

The scenario depicted in Figure 43 shows a more extreme case than Scenario 2, where a CPR stating Router 5 as the CR would lead to a list of 11 possible attack suspects (refer to Table 16). Further investigation shows that knowing the direction of interference reduces the number of suspects, in the worst case to 8 PEs and in the best case to 1 PE (3.6 PEs on average).

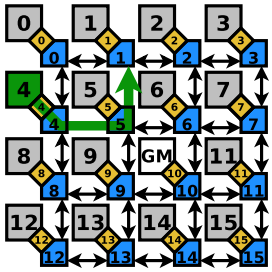



*Figure 43: Example 3 - Scenario*

*Table 16: Possible Attack Suspects for Example 3*

| Collision Router | Suspects using CPRD [1] | $IP_A$ | $OP_A$ | Suspects using CPDD [2] |
|---|---|---|---|---|
| 5 | 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15 | E | N | 6, 7 |
| | | S | N | 8, 9, 10, 11, 12, 13, 14, 15 |
| | | L | N | 5 |
| 1 | 0, 2, 3 | E | L | 2, 3 |
| | | W | L | 0 |

#### 6.2.3.2 Definition by Intention

Using groups theory, this section defines the concepts of Quadrant, Direction, and List of PAS for an $n \times m$ NoC-based MPSoC. The PAS is derived from the provided CPR (Section 6.2.2) and the implemented routing algorithm (RA).

- Routing Algorithm (RA): Given that $R_{xy}$ is an 8-bit word representing the permitted and prohibited turns by the NoC's implemented routing algorithm in the form of $R_{xy} : [N2E\ N2W\ E2N\ E2W\ W2N\ W2S\ S2E\ S2W]$, as presented in Section 2.2.3, the RA is defined as the set of allowed turns as follows:

$$RA = \{x2y \mid R_{xy} = 1 \text{ for } x2y\} \tag{14}$$

- Directions:

$$D_N(CR) = \{r \mid 0 \leq r_y < CR_y,\ r_x = CR_x\} \tag{15}$$
$$D_E(CR) = \{r \mid r_y = CR_y,\ CR_x < r_x < m\} \tag{16}$$
$$D_W(CR) = \{r \mid r_y = CR_y,\ 0 \leq r_x < CR_x\} \tag{17}$$
$$D_S(CR) = \{r \mid CR_y < r_y < n,\ r_x = CR_x\} \tag{18}$$

- Quadrants:

$$Q_{NW}(CR) = \{r \mid 0 \leq r_y < CR_y,\ 0 \leq r_x < CR_x\} \tag{19}$$
$$Q_{NE}(CR) = \{r \mid 0 \leq r_y < CR_y,\ CR_x < r_x < m\} \tag{20}$$
$$Q_{SW}(CR) = \{r \mid CR_y < r_y < n,\ 0 \leq r_x < CR_x\} \tag{21}$$
$$Q_{SE}(CR) = \{r \mid CR_y < r_y < n,\ CR_x < r_x < m\} \tag{22}$$

- PAS if the path followed by malicious packets within the CR corresponds to a turn:

  * $IP_A = N\ \wedge\ OP_A = E$:

$$PAS = \begin{cases} D_N(CR)\ \cup\ Q_{NW}(CR) & E2S \in RA \\ D_N(CR) & E2S \notin RA \end{cases} \tag{23}$$

  * $IP_A = W\ \wedge\ OP_A = S$:

$$PAS = \begin{cases} D_W(CR)\ \cup\ Q_{NW}(CR) & S2E \in RA \\ D_W(CR) & S2E \notin RA \end{cases} \tag{24}$$

  * $IP_A = N\ \wedge\ OP_A = W$:

$$PAS = \begin{cases} D_N(CR)\ \cup\ Q_{NE}(CR) & W2S \in RA \\ D_N(CR) & W2S \notin RA \end{cases} \tag{25}$$

  * $IP_A = E\ \wedge\ OP_A = S$:

$$PAS = \begin{cases} D_E(CR)\ \cup\ Q_{NE}(CR) & S2W \in RA \\ D_E(CR) & S2W \notin RA \end{cases} \tag{26}$$

* $IP_A = W \ \wedge \ OP_A = N$:

$$PAS = \begin{cases} D_W(CR) \ \cup \ Q_{SW}(CR) & N2E \in RA \\ D_W(CR) & N2E \notin RA \end{cases} \quad (27)$$

* $IP_A = S \ \wedge \ OP_A = E$:

$$PAS = \begin{cases} D_S(CR) \ \cup \ Q_{SW}(CR) & E2N \in RA \\ D_S(CR) & E2N \notin RA \end{cases} \quad (28)$$

* $IP_A = E \ \wedge \ OP_A = N$:

$$PAS = \begin{cases} D_E(CR) \ \cup \ Q_{SE}(CR) & N2W \in RA \\ D_E(CR) & N2W \notin RA \end{cases} \quad (29)$$

* $IP_A = S \ \wedge \ OP_A = W$:

$$PAS = \begin{cases} D_S(CR) \ \cup \ Q_{SE}(CR) & W2N \in RA \\ D_S(CR) & W2N \notin RA \end{cases} \quad (30)$$

- Possible Attack Suspects if the path followed by malicious packets within the CR corresponds to a straight path or if they were forwarded to the Local output port of the CR:

* $IP_A = N \ \wedge \ (OP_A = S \ \vee \ OP_A = L)$:

$$PAS = \begin{cases} D_N(CR) \cup Q_{NW}(CR) \cup Q_{NE}(CR) & E2S \in RA \ni W2S \\ D_N(CR) \cup Q_{NW}(CR) & E2S \in RA \not\ni W2S \\ D_N(CR) \cup Q_{NE}(CR) & E2S \notin RA \ni W2S \\ D_N(CR) & E2S \notin RA \not\ni W2S \end{cases} \quad (31)$$

* $IP_A = E \ \wedge \ (OP_A = W \ \vee \ OP_A = L)$:

$$PAS = \begin{cases} D_E(CR) \cup Q_{SE}(CR) \cup Q_{NE}(CR) & N2W \in RA \ni S2W \\ D_E(CR) \cup Q_{SE}(CR) & N2W \in RA \not\ni S2W \\ D_E(CR) \cup Q_{NE}(CR) & N2W \notin RA \ni S2W \\ D_E(CR) & N2W \notin RA \not\ni S2W \end{cases} \quad (32)$$

* $IP_A = W \ \wedge \ (OP_A = E \ \vee \ OP_A = L)$:

$$PAS = \begin{cases} D_W(CR) \cup Q_{SW}(CR) \cup Q_{NW}(CR) & N2E \in RA \ni S2E \\ D_W(CR) \cup Q_{SW}(CR) & N2E \in RA \not\ni S2E \\ D_W(CR) \cup Q_{NW}(CR) & N2E \notin RA \ni S2E \\ D_W(CR) & N2E \notin RA \not\ni S2E \end{cases} \quad (33)$$

* $IP_A = S \ \wedge \ (OP_A = N \ \vee \ OP_A = L)$:

$$PAS = \begin{cases} D_S(CR) \cup Q_{SW}(CR) \cup Q_{SE}(CR) & E2N \in RA \ni W2N \\ D_S(CR) \cup Q_{SW}(CR) & E2N \in RA \not\ni W2N \\ D_S(CR) \cup Q_{SE}(CR) & E2N \notin RA \ni W2N \\ D_S(CR) & E2N \notin RA \not\ni W2N \end{cases} \quad (34)$$

- Further reduction of the PAS: Since the PE running the Global Manager will normally not run user applications, its location can be removed from the PAS:

$$PAS' = PAS \setminus \{GM\} \tag{35}$$

#### 6.2.3.3 Further Discussion

When applying the same method to different routing algorithms, the gain of the proposed mechanisms becomes clearer. Figures 44, 45, and 46 depict the minimum, maximum and average number of PAS for the three examples presented in Section 6.2.3.1, for different minimal-path turn-model based routing algorithms. In this analysis, the following routing algorithms have been considered: XY [78], YX [79], West-First [80], East-First [81], North-First [82], North-Last [80], Negative First [80], and South-First [82]. In this work, the $SocDep^2$ [37] framework was used for modeling the routing graphs for the above-mentioned turn models.

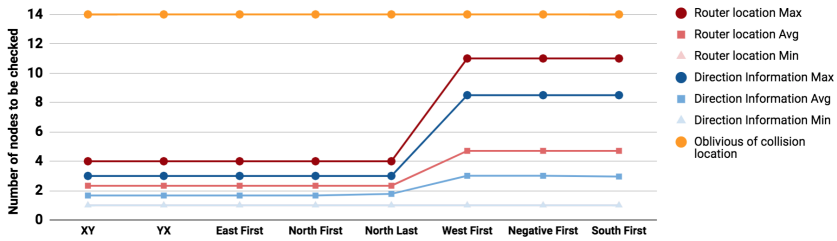



*Figure 44: Average number of PAS under different routing algorithms (Example 1)*

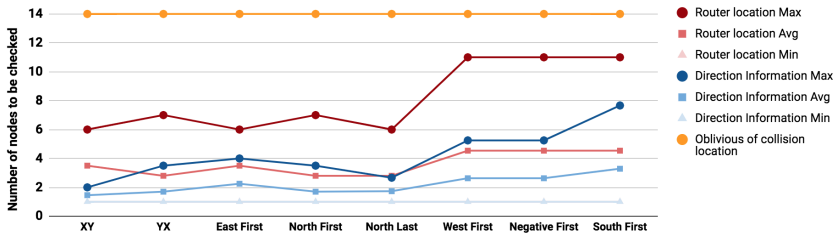



*Figure 45: Average number of PAS under different routing algorithms (Example 2)*

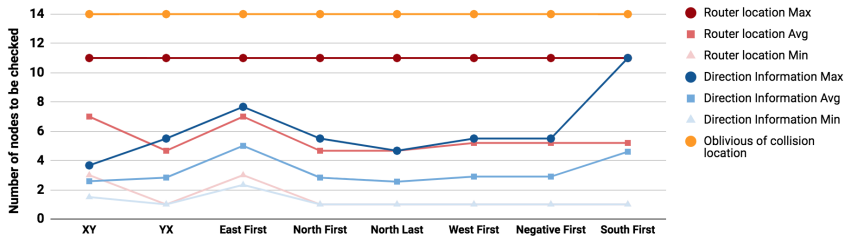



*Figure 46: Average number of PAS under different routing algorithms (Example 3)*

Compared to oblivious search (marked in orange in Figure 44, Figure 45, and Figure 46, the use of router locations will reduce the PAS in the worst case effort by 52%, 42%, and 21% for the monitored paths of the three examples presented in Section 6.2.3.1. Similarly, the use of directions will further reduce the worst-case effort by 63%, 69%, and 56%. The use of collision direction in locating the attacker source reduces the maximum effort of the location-based solution by 24%, 48%, and 44% respectively for the three examples and the average effort by 30.6%, 40%, and 39%.

### 6.2.4 Experimental Work

To evaluate the proposed CPDD scheme presented in Section 6.2.1, it was implemented on RTL level and integrated into the Secure Bonfire framework (Section 2.2.4.2), and compared to the CPRD scheme. Traffic generators included in the Bonfire platform were leveraged for simulating normal traffic and DoS attacks on $4 \times 4$ mesh NoC-based MPSoC scenarios. Moreover, the network routers (applying either XY or West First routing algorithm) use a credit base flow control with fair Round-Robin arbitration (on packet level) and utilize wormhole switching with 4-flit deep FIFOs. Furthermore, random traffic was transmitted with a PPL of $10$ flits and a PIR of $0.01$ (i.e., one packet every $100$ clock cycles). The PPL and PIR of the monitored and attacker packets, as well as their source and destination, vary according to the purpose of each scenario. The remainder of this section details the scenarios considered for the simulations in Section 6.2.4.1 and discusses the obtained results in Section 6.2.4.2 and Section 6.2.4.3.

#### 6.2.4.1 Simulation Scenarios

Section 6.2.3 explained the benefit of identifying the port through which malicious traffic disrupts the monitored path. This section presents the direction detection success and misses of the CPDD mechanism for the scenarios presented as illustrative examples in Section 6.2.3.1. Simulations were executed considering a PIR_A value of $0.03$ and PIR_M = PIR_R = $0.01$. Each experiment was performed for $20$ pseudo-random simulation seeds to provide uniform results. Additionally, four attacker PPLs were adopted (i.e., 10, 20, 30, and 50 flits), while the network's random traffic and the monitored node's traffic generated packets with the PPL of $10$ (i.e., PPL_R = PPL_S = $10$). Moreover, experiments were executed considering not only a deterministic routing algorithm but also an adaptive routing algorithm. Results for each routing algorithm are presented separately in Sections 6.2.4.2 and 6.2.4.3, respectively.

#### 6.2.4.2 Results for XY Routing

Tables 17, 18, and 19 summarize respectively the obtained detection results for scenarios depicted in Figures 41, 42, and 43 under XY routing. Each row lists the detection effectiveness of the CPR and the input direction of the malicious packets for every possible attack source. Also, each pair of columns groups the router and direction detection results for each of the attacker packet length values.

Results show that CPDD detected the input direction for almost all the scenarios where the CR was found. Also, once again, due to the wormhole switching, as the PPL of the attacker was increased, it became easier to detect the collision point router.

#### 6.2.4.3 Results for West-First Routing

A similar analysis to the one shown in the previous section was performed by adopting the West-First adaptive routing algorithm. Tables 20, 21, and 22 summarize respectively the obtained detection results for scenarios depicted in Figures 41, 42, and 43 under

*Table 17: XY Routing - Example 1*

| | APL = 10 | | APL = 20 | | APL = 30 | | APL = 50 | |
|---|---|---|---|---|---|---|---|---|
| AS | CPRD | CPDD | CPRD | CPDD | CPRD | CPDD | CPRD | CPDD |
| 0 | ✖ | – | ✖ | – | ✔ | ✔ | ✔ | ✔ |
| 1 | ✖ | – | ✖ | – | ✔ | ✔ | ✔ | ✔ |
| 2 | ✖ | – | ✖ | – | ✔ | ✔ | ✔ | ✔ |
| 3 | – | – | – | – | – | – | – | – |
| 4 | ✖ | – | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 5 | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 6 | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 7 | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 8 | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 9 | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 10 | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 11 | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 12 | – | – | – | – | – | – | – | – |
| 13 | ✖ | – | ✖ | – | ✔ | ✔ | ✔ | ✔ |
| 14 | ✖ | – | ✖ | – | ✔ | ✔ | ✔ | ✔ |
| 15 | ✖ | – | ✖ | – | ✔ | ✔ | ✔ | ✔ |

*Table 18: XY Routing - Example 2*

| | APL = 10 | | APL = 20 | | APL = 30 | | APL = 50 | |
|---|---|---|---|---|---|---|---|---|
| AS | CPRD | CPDD | CPRD | CPDD | CPRD | CPDD | CPRD | CPDD |
| 0 | ✖ | – | ✖ | – | ✔ | ✔ | ✔ | ✔ |
| 1 | ✖ | – | ✖ | – | ✔ | ✔ | ✔ | ✔ |
| 2 | – | – | – | – | – | – | – | – |
| 3 | ✖ | – | ✖ | – | ✔ | ✔ | ✔ | ✔ |
| 4 | ✖ | – | ✖ | – | ✔ | ✔ | ✔ | ✔ |
| 5 | ✖ | – | ✖ | – | ✔ | ✔ | ✔ | ✔ |
| 6 | ✖ | – | ✖ | – | ✔ | ✔ | ✔ | ✔ |
| 7 | ✖ | – | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 8 | – | – | – | – | – | – | – | – |
| 9 | ✖ | – | ✖ | – | ✖ | – | ✔ | ✔ |
| 10 | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 11 | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 12 | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 13 | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 14 | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 15 | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |

*Table 19: XY Routing - Example 3*

| | APL = 10 | | APL = 20 | | APL = 30 | | APL = 50 | |
|---|---|---|---|---|---|---|---|---|
| AS | CPRD | CPDD | CPRD | CPDD | CPRD | CPDD | CPRD | CPDD |
| 0 | ✖ | – | ✖ | – | ✔ | ✔ | ✔ | ✔ |
| 1 | – | – | – | – | – | – | – | – |
| 2 | ✖ | – | ✖ | – | ✔ | ✔ | ✔ | ✔ |
| 3 | ✖ | – | ✖ | – | ✔ | ✔ | ✔ | ✔ |
| 4 | – | – | – | – | – | – | – | – |
| 5 | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✖ |
| 6 | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 7 | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 8 | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 9 | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 10 | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 11 | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 12 | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 13 | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 14 | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 15 | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |

West-First routing. Each row lists the detection effectiveness of the CR and input port of the malicious packets for every possible attack source. Also, each pair of columns groups the router and direction detection results for each of the attacker PPL values.

*Table 20: West-First Routing - Example 1*

| | APL = 10 | | APL = 20 | | APL = 30 | | APL = 50 | |
|---|---|---|---|---|---|---|---|---|
| AS | CPRD | CPDD | CPRD | CPDD | CPRD | CPDD | CPRD | CPDD |
| 0 | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 1 | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 2 | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 3 | – | – | – | – | – | – | – | – |
| 4 | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 5 | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 6 | ✔ | ✖ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 7 | ✔ | ✖ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 8 | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 9 | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 10 | ✔ | ✖ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 11 | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 12 | – | – | – | – | – | – | – | – |
| 13 | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 14 | ✔ | ✖ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 15 | ✔ | ✖ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |

*Table 21: West-First Routing - Example 2*

| | APL = 10 | | APL = 20 | | APL = 30 | | APL = 50 | |
|---|---|---|---|---|---|---|---|---|
| AS | CPRD | CPDD | CPRD | CPDD | CPRD | CPDD | CPRD | CPDD |
| 0 | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 1 | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 2 | – | – | – | – | – | – | – | – |
| 3 | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 4 | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 5 | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 6 | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 7 | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 8 | – | – | – | – | – | – | – | – |
| 9 | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 10 | ✔ | ✖ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 11 | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 12 | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 13 | ✔ | ✔ | ✔ | ✖ | ✔ | ✔ | ✔ | ✔ |
| 14 | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 15 | ✔ | ✖ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |

*Table 22: West-First Routing - Example 3*

| | APL = 10 | | APL = 20 | | APL = 30 | | APL = 50 | |
|---|---|---|---|---|---|---|---|---|
| AS | CPRD | CPDD | CPRD | CPDD | CPRD | CPDD | CPRD | CPDD |
| 0 | ✔ | ✖ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 1 | – | – | – | – | – | – | – | – |
| 2 | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 3 | ✔ | ✔ | ✖ | – | ✔ | ✔ | ✔ | ✔ |
| 4 | – | – | – | – | – | – | – | – |
| 5 | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 6 | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 7 | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 8 | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 9 | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 10 | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 11 | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 12 | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 13 | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 14 | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 15 | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |

The results for this set of experiments show a better detection efficiency of the attacker's location compared to a system using XY routing. However, since traffic between two routers may have more than one minimal path, the number of PAS is much larger (refer to Figures 44, 45, and 46).

### 6.2.5 Area and Power Overhead

The proposed architectures (CPRD and CPDD Routers) were synthesized using the $0.18$ $\mu$m AMS library and Synopsys design vision at 200 MHz. Area overhead and critical path delay of the proposed architectures compared to the baseline architecture are reported in Table 23. The critical path delay overhead of the CPRD method is negligible and the proposed CPRD monitors only add $17\%$ area overhead to the minimalist router area (each CPRD monitor only adds $3.4\%$ overhead to the router's area). It is important to note that the main contribution to area overhead is due to the inclusion of the counter register. The width of the counters can be adjusted based on the application. The CPDD router applies 23.2% area overhead to the baseline router. In order to put the area overheads into perspective, we will consider the overheads on a 4 mm$^2$ chip. For an SoC using a $4 \times 4$ mesh NoC, the CPRD and CPDD would impose 0.4% and 0.5% overhead to the system respectively.

*Table 23: Area and critical path delay overhead.*

| Router | Area | | | | Critical Path Delay (ns) |
|---|---|---|---|---|---|
| | Sequential ($\mu$m$^2$) | Combinational ($\mu$m$^2$) | Total ($\mu$m$^2$) | Overhead (%) | |
| Baseline | 48378.7 | 42669.0 | 91047.8 | – | 4.82 |
| CPRD | 52033.7 | 55129.9 | 107163.7 | 17.7% | 4.8 |
| CPDD | 58313.6 | 53873.9 | 112187.5 | 23.2% | 4.79 |

*Power analysis:* the power consumption of the proposed methods and the baseline architecture were evaluated for random uniform traffic with a PIR of $0.01$ (without the presence of an attacker). The results of these experiments are reported in Table 24. The results show that the CPRD approach induces $5\%$ power overhead and the CPDD approach adds 9.4% to the baseline router.

*Table 24: Power overhead.*

| Router | Switching Power (mW) | Internal Power (mW) | Leakage Power (mW) | Total Power (mW) | Total Overhead (%) |
|---|---|---|---|---|---|
| Baseline | 0.151 | 3.663 | 0.261 | 3.814 | – |
| CPRD | 0.273 | 3.374 | 0.327 | 4.008 | 5% |
| CPDD | 0.269 | 3.905 | 0.346 | 4.174 | 9.4% |

### 6.2.6 Conclusion

Network-on-Chip solutions have become the central communication infrastructure of the modern MPSoCs. However, DoS attacks have been shown as an important threat to NoC integrity. Hence, it is of utmost importance to detect the occurrence of such attacks in the system, and also to locate the attacker in order to neutralize its effects. To this end, this section presented an analysis of the benefit when in addition to finding the collision router, the path taken by malicious traffic is also determined. Showing that in some cases the actual attack source can be found. Additionally, a minimalist hardware architecture was presented, which measures the performance degradation of monitored data transmissions under BRDoS attacks and detects the collision point as well as the path taken by malicious packets during communication disruption. Experimental results show the effectiveness of the proposed approach.

As explained in Section 5.3, the CPDD is able to detect more types of attacks than most of the approaches presented in related works. Moreover, it is shown in

Section 6.1.3 that small area and power overheads are incurred for its implementation. As a final step towards the detection and identifying the location of BRDoS attacks, Section 6.3 presents a full procedure including the CPDD.

## 6.3 Active DoS Attack Detection

The proposed DoS attack detection mechanisms, namely CPRD and CPDD (respectively, Sections 6.1 and 6.2), are from here on referred to as passive DoS attack detection mechanisms since they aim to detect the attacks during the normal communication process of the NoC-based MPSoC. Both methods were able to generate a CPR. However, CPDD included additional information that can be used for generating a reduced list of PAS, which for some cases could even be the actual attack source. This section presents a two-action Active DoS attack detection scheme for locating the actual source of attack when passive DoS attack detection is unable to do so.

This section is organized as follows: Section 6.3.1 summarizes Passive DoS attack detection classifying it into possible cases.

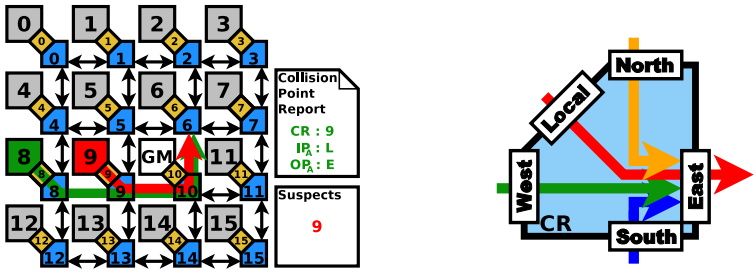
Section 6.3.2, explains one of the actions taken during the Active DoS Attack Detection phase where the GM sends network requests to rearrange traffic routes of the PAS. Section 6.3.3 presents the other action in which communication testing packets (CTPs) are transmitted aiming to intercept malicious traffic at its source, thus locating the infected PE.

### 6.3.1 Passive DoS Attack Detection Cases

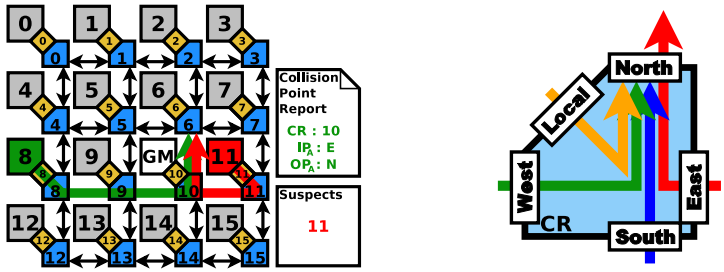
As explained in Section 6.2.2, during the normal functioning of the NoC, an NI is able to detect NoC communication disruptions, and after recurrent disruptions, create and send a CPR to the GM, Such a report contains a router ID labeled as CR as well as the input and output ports of the CR used by the traffic flow tagged as malicious (i.e. $IP_A$ and $OP_A$, respectively). Subsequently, as explained in Section 6.2.3, based on such information, together with the location of the GM, the dimensions of the NoC and the current routing constraints, a list of possible attack sources can be created. Figure47 depicts four NoC-based MPSoC scenarios in which all of the possible path types that malicious packets can take inside the CR are illustrated.

Furthermore, when GM receives a CPR, it also considers the NoC size and current routing constraints (given by the current routing algorithm and possible additional configured constraints), ending up in one of the following Passive DoS Attack Detection cases:

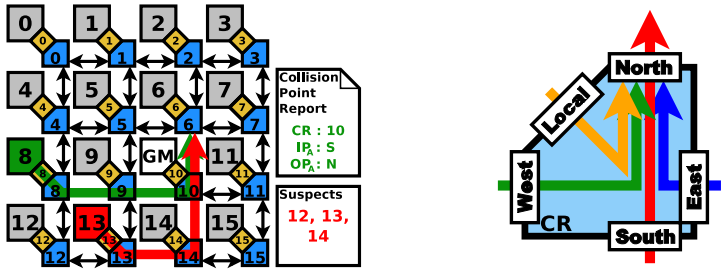
- *Case 1*: there is a single $S_A$ suspect. This is the case when $IP_A = Local$, as depicted in the scenario of Figure 47a where the collision occurred at the source of the attack meaning that the attack is originated from the PE locally connected to the CR. The scenario illustrated by Figure 47b falls into this case as well. Considering the size of the NoC and the restrictions provided by the XY routing algorithm, only traffic originating in P11 can access R10 through the West port and exit through the North port.
- *Case 2*: $IP_A$ and $OP_A$ reflect a turn inside the CR, as illustrated by Scenario- 2 in Figure 47b (It is worth mentioning that Case 1 has priority over Case 2. Thus Scenario 2 falls into Case 1, but that is not the case of all the scenarios that fit into Case 2). In this case, $S_A$ is usually unknown, but a list of possible attack sources can be derived. Such a list containing the PEs relatively located in one of the CR's quadrants and in the direction the packet entered. Figure 39 illustrates all



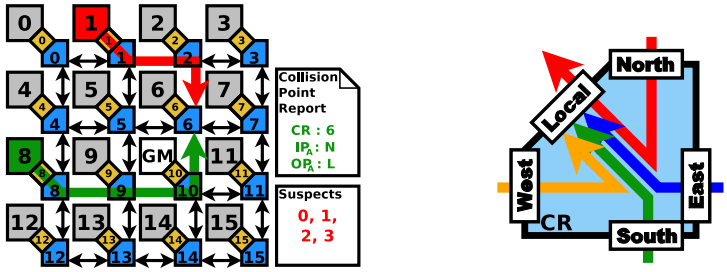

(a) Scenario 1: $IP_A$ = Local




(b) Scenario 2: Malicious Packets Turn Inside the CR




(c) Scenario 3: Malicious Packets Go Straight Inside the CR




(d) Scenario 4: $OP_A$ = Local

*Figure 47: Examples of possible paths of malicious packets inside a Collision Router*

the viable turns a packet can take inside a router and where $S_A$ would be located if any of the turns were to be reported. This is formalized in Section 6.2.3.2.

Take as an example a collision report informing a turn inside the CR where malicious packets entered through the West port and exited through the South port (i.e. $IP_A$ = West and $OP_A$ = South) as depicted in Figure 39a. Two possible situations are: i) $S_A$ is in the same row as the CR to the West; or ii) $S_A$ is North-West of the CR implying that the malicious packets were traveling South and turned East before reaching the CR. Even though the west input port of the CR can also be reached by packets originating South-West of the CR, such a possibility is excluded due to minimal path routing characteristic of the considered routing algorithms as a turn returning to the south would not be allowed.

- *Case 3*: $IP_A$ and $OP_A$ are in the opposite directions or $OP_A$ = *Local*, as illustrated by Scenarios 3 and 4 of Figure47, respectively. In this case, $S_A$ is mostly unknown (unless it falls into Case 1) and the list of possible attack suspects will show the PEs relatively located to the CR in three of the eight cardinal directions, comprising 2 quadrants. Figure 40 depicts all the $IP_A$ to $OP_A$ combinations considered as straight paths and the corresponding location region of the $S_A$.

  Take for example Figure 40c where the collision report states that the attacker entered the CR from West to East or West to Local (i.e. $IP_A$ = West and $OP_A$ = East or Local). The three possible situations are: i) $S_A$ is in the same row as the CR (to the West); ii) $S_A$ is North-West of the CR and malicious packets traveling South turned East before reaching the CR; or iii) $S_A$ is South-West of the CR and malicious packets traveling North turned East.

In Figure 47, all the scenarios depict a $4 \times 4$ mesh network implementing the XY deterministic routing algorithm and monitored traffic originating in PE8 is forwarded to PE6 (colored green). Table 25 lists the attack suspects for each possible CR in the path of the monitored flow. The second column is based on the CPRD approach, while the last column is based on the CPDD approach, which considers $IP_A$ and $OP_A$ for creating smaller lists. Even though for some cases of the given example, the CPDD manages to identify a unique suspect, in some others it identifies up to four suspects. Moreover, the number of suspects becomes larger once the network utilizes a routing algorithm with a higher degree of adaptivity (where more turns are allowed). This is the case of the West-First (WF) routing algorithm which would provide three possible routes for packets transmitted from PE8 to PE6, each with a different set of attack suspects as listed in Table 26. Furthermore, if the route taken by the monitored traffic is unknown which is the case for non-deterministic routing, the list of attack suspects for a given CR is the sum of suspects in all possible routes which can also be filtered considering $IP_A$ and $OP_A$. However, in some cases, lists of more than one suspect will be obtained, hence further processing is required for locating the source of the attack. Section 6.3.2 and Section 6.3.3 describe subsequent steps that can be taken for detecting the actual attack source which are referred to as Active DoS attack detection.

### 6.3.2 Traffic Rearrangement

In the context of this dissertation, Traffic Rearrangement is defined as the action of restricting the NoC's traffic so that it flows through a deterministic path. By doing so during a DoS attack, a new CPR can be generated, providing additional information that can be used for reducing the list of PAS. Such action can be performed by carefully

*Table 25: Attack Suspects for Path 8→6 - XY Routing*

| Collision Router | Suspects using CPRD [1] | $IP_A$ | $OP_A$ | Suspects using CPDD [2] |
|---|---|---|---|---|
| 8 | - | - | - | - |
| 9 | 9 | L | E | 9 |
| 10 | 10, 11, 12, 13, 14, 15 | E | N | 11 |
| | | S | N | 12, 13, 14, 15 |
| | | L | N | 10 |
| 6 | 0, 1, 2, 3, 4, 5, 7 | N | L | 0, 1, 2, 3 |
| | | E | L | 7 |
| | | W | L | 4, 5 |

*Table 26: Attack Suspects for Path 8→6 - WF Routing*

| | Collision Router | Suspects using CPRD [1] | $IP_A$ | $OP_A$ | Suspects using CPDD [2] |
|---|---|---|---|---|---|
| Route 1 | 8 | 0, 4, 12 | N | E | 0, 4 |
| | | | S | E | 12 |
| | 9 | 0, 1, 4, 5, 9, 12, 13 | N | E | 0, 1, 4, 5 |
| | | | S | E | 12, 13 |
| | | | L | E | 9 |
| | 10 | 10, 11, 12, 13, 14, 15 | E | N | 11 |
| | | | S | N | 12, 13, 14, 15 |
| | | | L | N | 10 |
| | 6 | 0, 1, 2, 3, 4, 5, 7, 9, 12, 13 | N | L | 0, 1, 2, 3 |
| | | | E | L | 7 |
| | | | W | L | 0, 1, 4, 5, 9, 12, 13 |
| Route 2 | 8 | 9, 10, 11, 12, 13, 14, 15 | E | N | 9, 10, 11 |
| | | | S | N | 12, 13, 14, 15 |
| | 4 | 0, 4 | N | E | 0 |
| | | | L | E | 4 |
| | 5 | 0, 1, 5, 9, 12, 13 | N | E | 0, 1 |
| | | | S | E | 9, 12, 13 |
| | | | L | E | 5 |
| | 6 | 0, 1, 2, 3, 7, 9, 10, 11, 12, 13, 14, 15 | N | L | 0, 1, 2, 3 |
| | | | E | L | 7 |
| | | | S | L | 9, 10, 11, 12, 13, 14, 15 |
| Route 3 | 8 | 0, 4, 12 | N | E | 0, 4 |
| | | | S | E | 12 |
| | 9 | 9, 10, 11, 12, 13, 14, 15 | E | N | 10, 11 |
| | | | S | N | 12, 13, 14, 15 |
| | | | L | N | 9 |
| | 5 | 0, 1, 4, 5, 12 | N | E | 0, 1 |
| | | | W | E | 0, 4, 12 |
| | | | L | E | 5 |
| | 6 | 0, 1, 2, 3, 7, 9, 10, 11, 12, 13, 14, 15 | N | L | 0, 1, 2, 3 |
| | | | E | L | 7 |
| | | | S | L | 9, 10, 11, 12, 13, 14, 15 |

adding restrictions to the routing algorithm without compromising its functionality. This can be done by introducing additional routing constraints in the header flit of packets and modifying the routers so that they can interpret them.

As explained in Section 6.3.1, the GM's DoS detection process is triggered by the reception of a CPR and then generates a list of PAS. Furthermore, if such a list fits into Case 1 of Section 6.3.1, passive DoS attack detection is enough for locating the source of the attack and no further DoS attack detection mechanism has to be followed. On

the other hand, if it fits into any of the other two cases, the GM proceeds to check if the first action of the Active DoS Attack Detection can be executed (aka Traffic Rearrangement). In order to be able to execute the Traffic Rearrangement, one of the following two conditions must be met:

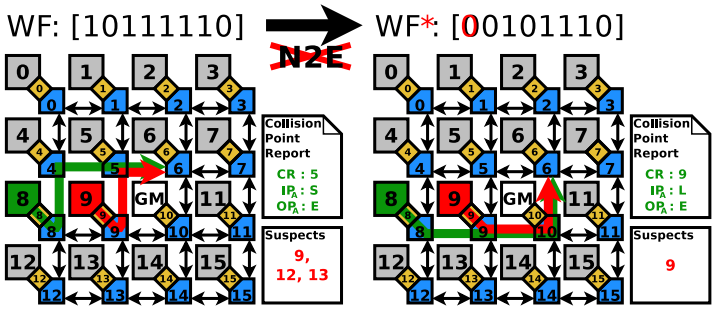
- *Condition 1*: The CPR fits only into Case 2 of Section 6.3.1 and suspects in the identified quadrant have two available turns for reaching the CR. In this situation, the path taken by the malicious traffic inside the CR corresponds to a turn and based on it, the GM can determine the quadrant where the attack source is located (Figure 39 and Section 6.2.3.2). Furthermore, regardless of the current routing constraints, packets originating in a such quadrant can take more than one type of turn for reaching the CR following minimal routing (refer to Figure 4c to Figure 7c). When this condition is met, the GM proceeds to restrict the reported turn from the identified quadrant. However, turns detected inside a router are expressed following an inner notation (e.g., a packet coming from the East input port and turning to the North output port), while routing restrictions are made using an outer notation (e.g., a packet traveling West that turns North).
- *Condition 2*: The CPR fits only into Case 3 of Section 6.3.1 and suspects of at least one of the two detected quadrants have two available turns for reaching the CR (refer to Figure 4c to Figure 7c). In this situation, the malicious packet either followed a straight path from $IP_A$ to $OP_A$ or reached its destination, i.e. $OP_A = Local$ (Figure 40). Therefore, the list of PAS includes two quadrants. Consequently, for each quadrant with two available turns, the same procedure of Condition 1 is followed for a reported turn with the same $IP_A$.

If the received CPR meets any of the two conditions, the suspects have some level of routing adaptivity. Consequently, the GM will be able to impose additional routing restrictions to them. Otherwise, only the second action of ADAD will be taken (a.k.a., Insertion of Communication Testing Packets, explained in Section 6.3.3).

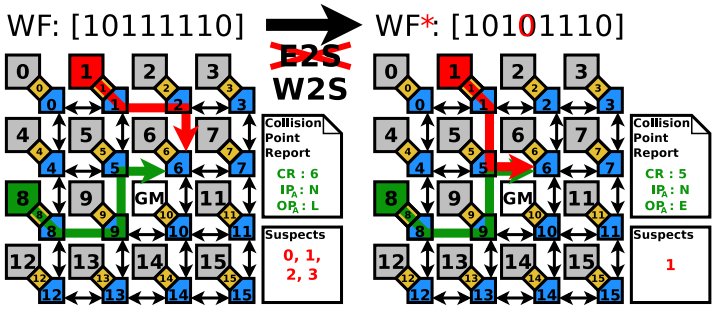
Once a router receives a traffic rearrangement request, it continues the transmission of the current packet (if any) and subsequently sends the following packets with a header flit containing a different value of *Routing Constraints*. Subsequently, each time a packet with the additional routing constraints reaches a NoC router, it will be routed not only considering the restrictions imposed by the current routing algorithm, but also the ones found in the header flit of the packet.

Figure 48 illustrates four examples where only Traffic Rearrangement is used. In such examples, 16 PEs of an MPSoC are connected by a $4 \times 4$ mesh NoC that implements the West-first adaptive routing [80]. Additionally, monitored traffic flows from PE8 to PE6 (colored green) with different attack paths in each example (colored red).

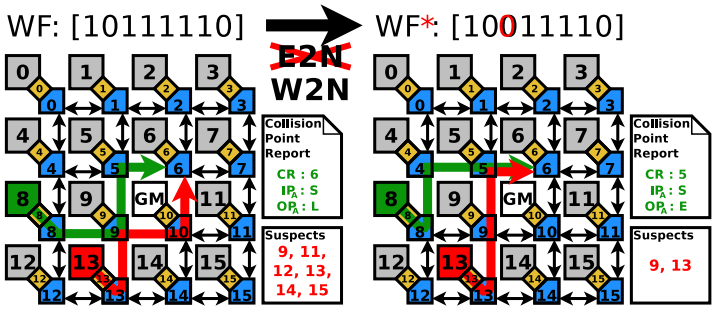
In the first example, depicted in Figure 48a, malicious traffic flows from PE9 to PE6 (colored red). Also, a CPR tags R5 as the CR where the malicious traffic collided with the legitimate traffic, entering through the South input port and competing for the East output port. Since such report fits Case 2 of Section 6.3.1, the GM knows that the source of the attack is in the South-west quadrant of the CR or South to it (refer to Figure 39c). Consequently, the GM generates the list of PAS: $\{9, 12, 13\}$ and decides to narrow down such a list further. Apart from the N2E turn, the current routing constraints also allow the suspects to reach the CR with the E2N turn (Figure 6c), hence the CPR meets the Condition 1 for Traffic Rearrangement. Therefore, all the routers connected to suspects are requested to restrict the turn equivalent to the one detected, i.e. from North to East. Afterward, with the new routing constraints, a CPR



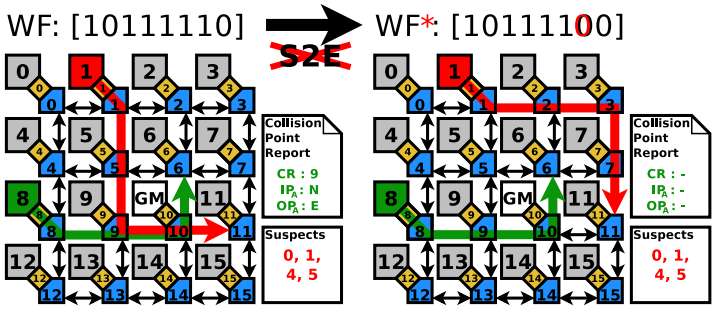

*(a) Example 1: Attack path 9→6*




*(b) Example 2: Attack path 12→6*




*(c) Example 3: Attack path 13→6*




*(d) Example 4: Attack path 1→11*

*Figure 48: Examples of Traffic Rearrangement*

tags PE9 as the source of the attack, hence the CPR fits into Case 1 since the new collision in R9 (i.e. CR=R9) and the malicious traffic entering the CR through the Local input port.

In the second example, illustrated by Figure 48b, the malicious traffic flows from PE1 to PE6 (colored red) and the CPR tags R6 as the CR with packets entering through the North input port and competing for the Local output port which fits into Case 3 of Section 6.3.1. Consequently, the GM generates the list of PAS: {0, 1, 2, 3}. Furthermore, now the GM knows the source of the attack is north of R6, including the North-West and North-East quadrants as illustrated by Figure 40a, it checks the available turns in each of those quadrants (Figure 6c). It then finds that according to the current routing constraints, the CPR meets Condition 2 for Traffic Rearrangement since suspects in the North-West quadrant are allowed to take E2S and S2E turns, while suspects in the North-East quadrant are only allowed to make W2S turns (S2W is disabled by the WF routing algorithm [80]). Therefore, the GM can rearrange traffic by requesting only the North-West quadrant to disable East-to-South (E2S) turns. With the new routing constraints, PE1 is tagged as the source of the attack, hence the report fits Case 1 of Section 6.3.1, since the new collision occurred in R5 (i.e. CR=R5) and despite the malicious traffic entering the CR through the North port and not the Local port, based on the size of the network and the current routing constraints, PE1 is the only core which is able to generate traffic capable of entering R5 through the North port and exit through the East port.

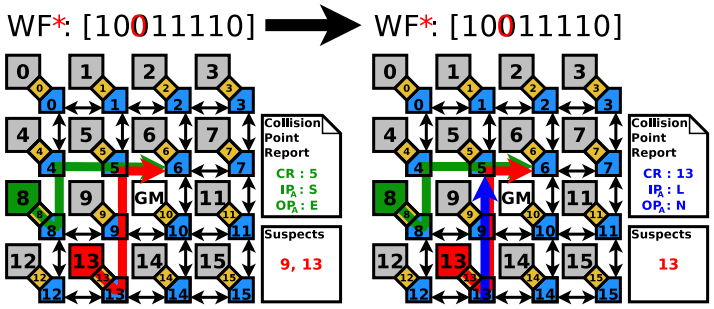
In the third example, illustrated by Figure 48c, the malicious traffic flows from PE13 to PE6 (colored red). As in the second example, the CPR tags R6 as the CR. However, the malicious traffic now entered through the South port and exited through the Local port. Nevertheless, the collision report also fits into Case 3. So the GM generates the list of PAS: {9, 11, 12, 13, 14, 15}. It is worth noticing that for this example, PE10 is not considered a suspect because the GM is running on it, thus it does not execute applications from the users. Furthermore, now that the GM knows that the source of the attack is south of R6, in order to assess the routing adaptivity allowed to the suspects for reaching the CR, it classifies them into two quadrants. Subsequently, it realizes that despite the suspects in the South-East quadrant having only the W2N turn available according to the current routing constraints, suspects in the South-West quadrant are allowed two turns: E2N and N2E (Figure 6c), meeting Condition 2 for Traffic Rearrangement. Consequently, the GM requests the South-West quadrant to disable turns when going East and turning North (E2N). With the new routing constraints, a new CPR reaches the GM stating R5 as the CR, and that the malicious traffic entered through the South and exited through the East. Such a situation fits into Case 2, establishing that the source of the attack is South of the CR which narrows the list of PAS to {9, 13}. Since no more routing restrictions can be imposed on the current suspects, Condition 1 for Traffic Rearrangement is not met, requiring the second action of Active DoS Attack Detection (a.k.a., Insertion of Communication Testing Packets) which is explained in Section 6.3.3.

Finally, in the fourth example, illustrated by Figure 48d, the malicious traffic flows from PE1 to PE11 (colored red). The CPR tags R9 as the CR, where the malicious traffic entered through the North port and exited through the East port. Since such a report fits Case 2, the GM knows that the source of the attack is in the North-West quadrant of the CR or North to it (refer to Figure 39a). Consequently, the GM generates the list of PAS: {0, 1, 4, 5}. Knowing that traffic generated in the North-West quadrant has two available turns to reach the reported CR (Figure 6c), the GM decides to narrow

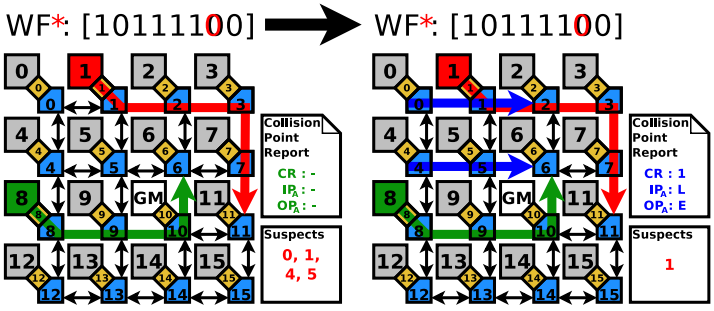
the list of PAS further. Consequently, routers are reconfigured so that they introduce an additional routing restriction to their local traffic, disabling turns of packets going South turning East (S2E). As shown in Figure 48d (right), after applying the additional routing constraints, it is possible that, if only the traffic rearrangement action is taken, no further disruption is reported. In such a situation, the GM may be configured to finish the detection process until another traffic flow generates a new CPR, or execute the second action of the Active DoS Attack Detection (a.k.a., Insertion of Communication Testing Packets) which is explained in Section 6.3.3.

### 6.3.3 Insertion of Communication Testing Packets

As seen in the previous section, traffic rearrangement is not enough for all scenarios. In order to find the exact Source of the Attack ($S_A$), a second action of the active DoS attack detection is proposed. In this step, communication testing packets (CTPs) are transmitted. Such packets sweep specific segments of the network and find the point where malicious packets enter the NoC. CTPs are small so that they are able to assess the competition of the output ports in the path without imposing too much delay on the other packets. However, CTPs are only useful if they flow through a deterministic path, where only one turn should be available for them to reach their destination. Therefore, CTPs are only sent if the implemented routing algorithm restricts one of the turns or together with the traffic rearrangement action. Figure 49 illustrates two examples of using CTPs to locate the $S_A$.




*(a) Example 1: Attack path 13→6*




*(b) Example 2: Attack path 1→11*

*Figure 49: Examples of communication testing packets insertion*

Figure 49a illustrates an example in which the insertion of CTPs is used for pinpointing the source of a DoS attack. Such an example follows the action of Traffic Rearrangement depicted in Figure 48b, which ended with the PAS {9, 13} after restricting the E2N

turn of the West-First routing algorithm. With this second action of the active DoS attack detection, the GM commands R13 to send CTPs to the CR, forcing them to flow through R13 and R9, transporting the required information so that R5 reports R13 as the CR. Additionally, that the malicious traffic entered the CR through the Local port, identifying it as the $S_A$.

On the other hand, Figure 49b also depicts an example of the insertion of CTPs, but as the continuation of the example from Figure 48d where the Traffic Rearrangement action was equally not sufficient for finding the exact source of Attack ($S_A$). The four suspects are located in two rows and two columns. Additionally, the introduced routing constraints force the traffic to initially flow horizontally when going from West to East regardless of the row of their destination. Therefore, the GM chooses R0 and R4 as the edge routers asked to introduce the CTPs. By doing so, the source of the attack is discovered by the CTPs generated at R0.

### 6.3.4 Complete DoS Attack-Source Detection Strategy

This section describes the general strategy for detecting the exact $S_A$. This process starts when the GM receives a CPR. Such a strategy combines Passive and Active DoS attack detection as detailed in Algorithm 1.

Once a collision is reported, the GM is able to generate a list of PAS as explained in Section 6.2.3.2. This is done based mainly on the input ($IP_A$) and output ($OP_A$) ports used by the malicious packets which the GM uses for identifying if the malicious packets made a turn or followed a straight path within the CR. Subsequently, if the list of PAS contains only one element, the search can be terminated.

Provided that the list of PAS contains more than one element, the GM attempts the first action of Active DoS Attack Detection (ADAD), namely Traffic Rearrangement. It first needs to identify the number of unrestricted turns the suspects can take to reach the CR. This is done with the aim of reducing the PAS by forcing the traffic generated by suspects to flow through deterministic paths.

A turn inside a router is defined as a ninety-degree path, as included in the following set: $\{ N \rightarrow E,\ N \rightarrow W,\ E \rightarrow N,\ E \rightarrow S,\ W \rightarrow N,\ W \rightarrow S,\ S \rightarrow E,\ S \rightarrow W \}$, using the notation $IP_A \rightarrow OP_A$. Identifying the quadrant following Figure 39 and the restricted turns for that quadrant according to Section 2.2.2, the GM will restrict one of two turns, if two turns are allowed.

On the other hand, if malicious packets were routed following a straight path within the CR as stated by the set: $\{ N \rightarrow S, N \rightarrow L, E \rightarrow W, E \rightarrow L, W \rightarrow E, W \rightarrow L, S \rightarrow N, S \rightarrow L \}$, the GM will check separately the restricted turns of each quadrant that could reach $IP_A$.

Finally, the GM will ask specific edge routers to send CTPs for sweeping the paths followed by the PAS. These packets will follow all the routing constraints, even the ones determined for traffic rearrangement, and update the routing constraints for suspects. Moreover, actions triggered directly by the received CPR end here. However, some of the requested CTPs will collide with the malicious traffic at its source where a new CPR will be sent to the GM, enabling it to determine the exact $S_A$.

### 6.3.5 Proposed Architecture

This section details the proposed architecture for implementing ADAD which is based on the architecture of the CPDD presented in Section 6.2.1. This section contains all the packet structures used in ADAD, as well as the proposed router architecture for its implementation.

**Algorithm 1:** Flooding DoS Attack Source Detection

**input :** *// Mandatory general arguments*
$(n,m)$: NoC size (number of rows and columns in the NoC)
$RA$ : Disabled turns in the current routing algorithm
*// Mandatory arguments, provided with a Collision Point Report*
$CR$ : Collision Router (Router where the malicious traffic collided with the monitored traffic)
$IP_A$ : Input Port of the CR used by the malicious traffic
$OP_A$ : Output Port of the CR used by the malicious traffic
*// Optional arguments on first execution*
$RC$ : Set of additional disabled turns (initially $RC = \emptyset$)
$PAS$ : Set of Possible Attack Sources (initially $PAS = \emptyset$)

**output :** $S_A$ : Source of DoS attack

*// $NI$ : Network Interface*
*// $CTPs$ : Communication Testing Packets*
*// $inv(dir)$ : Inverse of a direction.*
*// $\forall\ dir \in \{N,\ E,\ W,\ S\}\ \exists\ inv(dir) \in \{N,\ E,\ W,\ S\}$ so that:*
*// $inv(N) = S,\ inv(S) = N,\ inv(W) = E,\ inv(E) = W$*

$Turns \leftarrow \{\ N \rightarrow E,\ \ N \rightarrow W,\ \ E \rightarrow N,\ \ E \rightarrow S,\ \ W \rightarrow N,\ \ W \rightarrow S,\ \ S \rightarrow E,\ \ S \rightarrow W\ \}$;
$StraightPaths \leftarrow \{\ N \rightarrow S,\ E \rightarrow W,\ W \rightarrow E,\ S \rightarrow N, N \rightarrow L,\ E \rightarrow L,\ W \rightarrow L,\ S \rightarrow L\ \}$;
$S_A \leftarrow \emptyset$;
**if** $IP_A \neq \emptyset \wedge OP_A \neq \emptyset$ **then**
  *// A collision was reported. A CPR always triggers the algorithm.*
  Generate/update $PAS$ based on $n$, $m$, $CR$, $IP_A$, $OP_A$, and $(RA \cup RC)$;
  **if** *there is only one element in* $PAS$ **then**
    *// Case 1 of Passive DoS Detection: There is a single suspect*
    $S_A \leftarrow$ PAS(0); Clear all $RCs$;
    **return** $S_A$;
  *// Active DoS Attack Detection*
  **else**
    *// Action 1: Determine new restrictions for traffic rearrangement*
    **if** $\{IP_A \rightarrow OP_A\} \in Turns$ **and** $\{OP_A \rightarrow inv(IP_A)\} \notin (RA \cup RC)$ **then**
      *// Condition 1 for Traffic Rearrangement: $\{IP_A \rightarrow OP_A\}$ corresponds to a turn and $\{OP_A \rightarrow inv(IP_A)\}$ is not disabled*
      Add $\{inv(IP_A) \rightarrow OP_A\}$ to $RC$;
    **else if** $\{IP_A \rightarrow OP_A\} \in StraightPaths$ **and** $(\exists\ dir : \{inv(IP_A) \rightarrow dir\} \in Turns \wedge \{inv(IP_A) \rightarrow dir\} \notin (RA \cup RC) \not\ni \{dir \rightarrow inv(IP_A)\})$ **then**
      *// Condition 2 for Traffic Rearrangement: $\{IP_A \rightarrow OP_A\}$ corresponds to a straight path and at least one turn can be disabled for suspects*
      **forall** $dir : \{inv(IP_A) \rightarrow dir\} \in Turns \wedge \{inv(IP_A) \rightarrow dir\} \notin (RA \cup RC) \not\ni \{dir \rightarrow inv(IP_A)\}$ **do**
        Add $\{inv(IP_A) \rightarrow dir\}$ to $RC$;
    *// Action 2: Request transmission of CTPs with RC*
    Request the appropriate edge routers to insert CTPs that flow through all the routers in $PAS$, following and updating restrictions $(RA \cup RC)$;
    *// The Source of the attack has not been found yet, but it will be when a Collision Point Report is triggered by the CTPs*

The proposed packet structures used for this approach are depicted in Figure 50, where the three most significant bits of each flit indicate its type. Additionally, the less significant bit of each flit is a parity bit used for single error detection. As in other approaches presented in this thesis, the *Header Flit* of all the packets contains the destination and source addresses. However, for ADAD the *Header Flit* also contains eight bits for additional routing constraints and three bits to indicate the packet type. One of the packet type bits signals when the packet is related to CTPs, another when the packet is carrying a CPR, and the other when the packet was sent by the GM (bits 13, 12, and 11, respectively).

Figure 50a shows the structure of data packets where one or more *Payload Body Flits* carry the data exchanged between PEs. It is identified by having the three packet-type bits set to zero. Furthermore, since the destination of this type of packet is a PE, the first body flit contains other information regarding the packet which is referred to as the *Packet Info Flit*. It carries an ID for allowing packet reordering at the destination and the length of the packet for checking the packet to be complete. Finally, for enabling communication disruption monitoring, the *Tail Flit* carries the address of the router where the packet waited for the most to be forwarded, the number of clock cycles it waited while its required output was busy, the input ports that competed and won access to the required output, and the output port for which the competition took place.

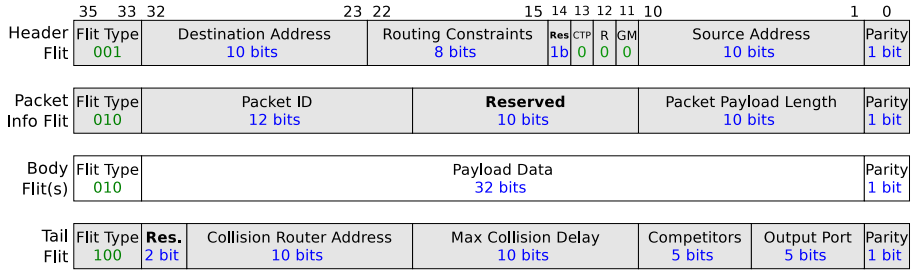
Figure 50b depicts the structure of packets sent by the GM to Local Security Managers (LSMs) for defining the reports threshold. It does not contain additional routing restrictions (i.e., only restricted by the routing algorithm) and since it is transmitted by the GM, the *GM* bit is set to one while the other packet type bits are set to zero. Additionally, since it only carries a new threshold, the packet only contains one *Header* and one *Tail* flit.

Figure 50c illustrates the structure of packets sent to the GM with a CPR. It does not contain additional routing restrictions (i.e., only restricted by the routing algorithm). The *R* bit is set to one while the other packet-type bits are set to zero. Since it will be delivered to the PE executing the GM, it includes the *Packet Info Flit*. It also contains one *Body* and one *Tail* flit which contain the information of the *Header* and *Tail* flits of the packet that triggered the report.

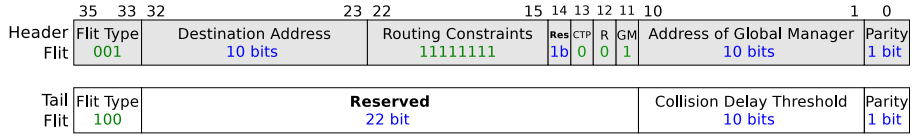
Figure 50d represents the structure of packets sent by the GM to specific edge routers when requesting the generation of CTPs. Their routing constraints are only given by the routing algorithm. Since this type of packet is sent by the GM and is related to CTPs, the *GM* and the *CTP* bits are set to one while the *R* bit is set to zero. Its *Tail Flit* is able to carry the necessary information, thus no *Body Flits* are required. It transports the destination where the CTPs should be sent, the routing constraints that must be followed and updated, and the number of CTPs that need to be generated.

Finally, Figure 50e presents the structure of CTPs where, from the packet type bits, only *CTP* is set to one. Furthermore, since this is a monitored packet, it may also contain additional routing constraints and a *Tail Flit* with information regarding the greatest disruption found in its path. Moreover, since CTPs have only two flits, their impact to the NoC's congestion is negligible.

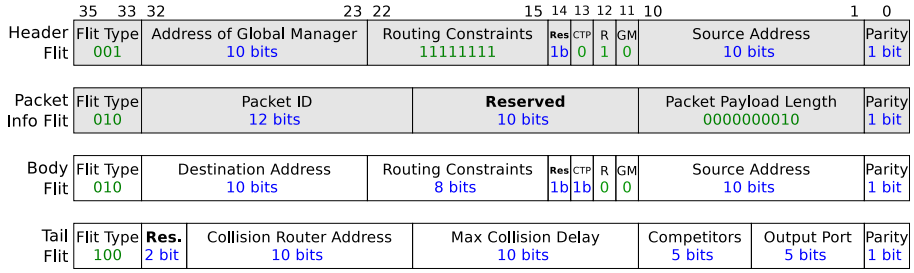
The proposed router architecture for implementing ADAD is depicted in Figure 51. As with the CPDD approach, input ports are equipped with traffic monitors that count the number of clock cycles a packet is waiting while its required output port is busy. Additionally, if the waiting time of a packet is longer than the previously recorded time,



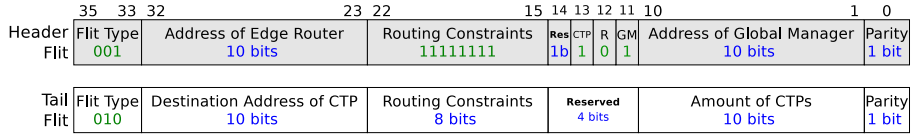

*(a) Structure of Data Packets*



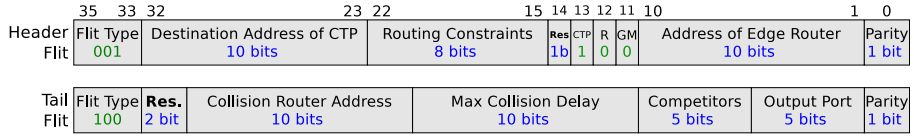

*(b) Structure of Threshold Configuration Packet Sent by the Global Manager*




*(c) Structure of Collision Point Report Packet Sent to the Global Manager*




*(d) Structure of CTP Request Packet Sent by the Global Manager*




*(e) Structure of CTP Packet*

*Figure 50: Packet Structures for Active DoS Attack Detection*

it updates the tail flit with the new values. In addition to the CPDD router sub-blocks, the ADAD introduces a Local Security Manager (LSM). The architecture of this new sub-block which is in charge of the following operations is illustrated in Figure 52:

- Read the header of arriving packets so that only data packets are forwarded to the NI (Extraction Filter). It is important that no information from the monitors reaches the applications run by users to avoid side-channel attacks.
- Interpret other arriving packets and update the reports threshold (Infraction Reporter). According to the scheduled load to the MPSoC, the GM can change the end-to-end delay threshold so that irrelevant reports are not transmitted.
- Send CPRs of monitored packets that arrive and had a collision greater than the threshold (Infraction Reporter). Such information is the basis for detecting BRDoS attacks.
- Sniff packets not destined to the local output port and update routing constraints when CTPs pass through (Injection Manager). Apart from patrolling NoC segments containing attack source suspects, CTPs propagate the additional routing constraints. Therefore, routers must be aware of changes and adjust new packets to them.
- Write additional routing restrictions on packets being injected into the NoC (Injection Manager). To ensure that CTPs collide with malicious packets, traffic flows generated by attack suspects must follow the specified deterministic path.

If the LSM is in an edge router, it also needs to provide the following features:

- Resolve requests for generating CTPs (CTPs Generator), updating routing constraints (Injection Manager);
- Generate CTPs (CTPs Generator).
- Calculate how long a CTP waits while a packet arriving from the local input port is being transmitted (CTP Traffic Generator).

### 6.3.6 Conclusion

This Section presented Active DoS Attack Detection (ADAD). Such an approach is the final step towards DoS attack detection following previous approaches, i.e. Collision Point Router Detection and Collision Point Direction Detection. With such a mechanism, the MPSoC's Global Manager is able to restrict the traffic generated by attack suspects to flow through deterministic paths in which other collisions are reported and the exact source of the attack is detected.

## 6.4 Communication Disruption Tattletaling

The related works address Flooding DoS (FDoS) attacks in NoC-based MPSoCs mainly considering them as an increase of the packet injection rate (PIR-based FDoS attack) as summarized in Chapter 5. However, as shown in Section 3.3, such an attack is not effective when using fair-arbitrated NoCs. In contrast, an FDoS attack by increasing the packet payload length (PPL-based FDoS attack) represents a real threat to NoC-based MPSoCs. This is the aim of the distributed scheme of monitors presented in this section for tracking the communication behavior of users in multi-tenant MPSoCs,

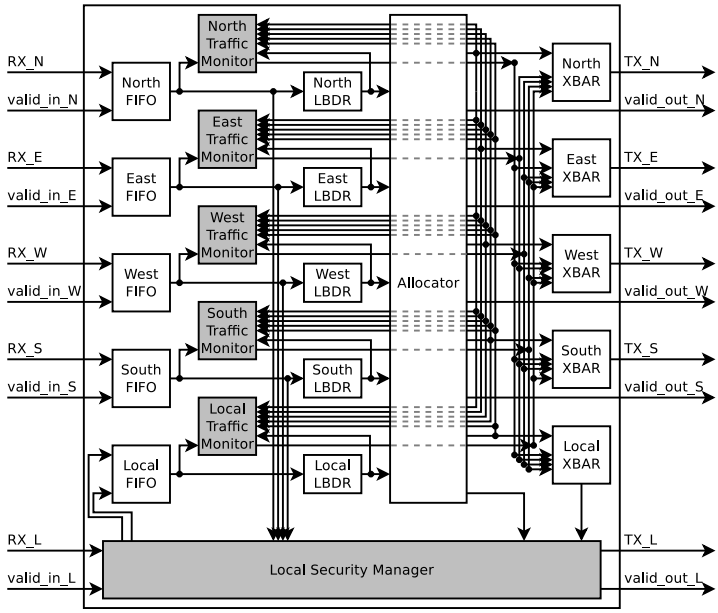
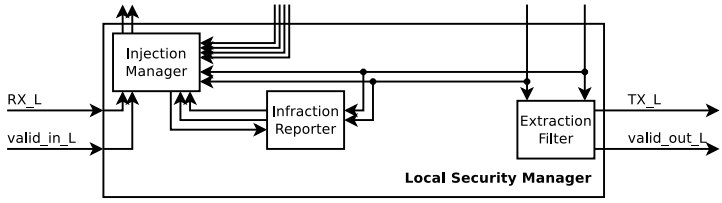
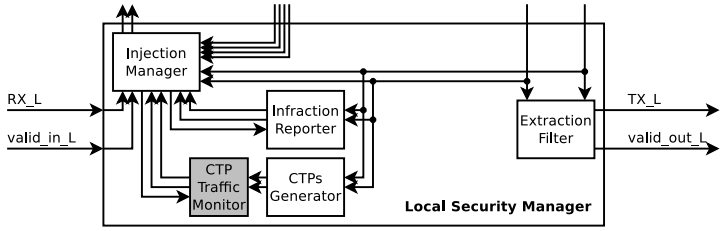



*Figure 51: Router architecture - Active DoS Attack Detection*




*(a) Non-edge Router*




*(b) Edge Routers*

*Figure 52: Architecture of the Local Security Manager*

named *Communication Disruption Tattletaling (CDT)*. The CDT enables monitoring information to piggyback on data packets to be analyzed at the destination, which can subsequently take direct action or report to the MPSoC's GM when the presence of an attack is suspected.

The remaining of this section is organized as follows: Section 6.4.1 details the proposal for CDT. Section 6.4.2 contains an overhead evaluation of the proposed mechanism in terms of area and power. Finally, Section 6.4.3 presents conclusions.

### 6.4.1 Proposed Architecture

This section details the proposed architecture for implementing CDT. The proposed packet structure is depicted in Figure 53 where the three most significant bits of each flit indicate its type. Additionally, the less significant bit of each flit is a parity bit used for single error detection. The *Header Flit* contains the destination and source addresses. The first body flit which contains information regarding the packet is referred to as the *Packet Info Flit*. It carries an ID for allowing packet reordering at the destination and the length of the packet to make sure that the packet is complete. One or more *Payload Body Flits* follow, carrying the data exchanged between PEs. Finally, the *Tail Flit* is used for DoS attack detection by the CDT approach. It transports the length of the longest packet encountered in the path as well as its source. Such information is evaluated in every router of the path and updated if necessary.

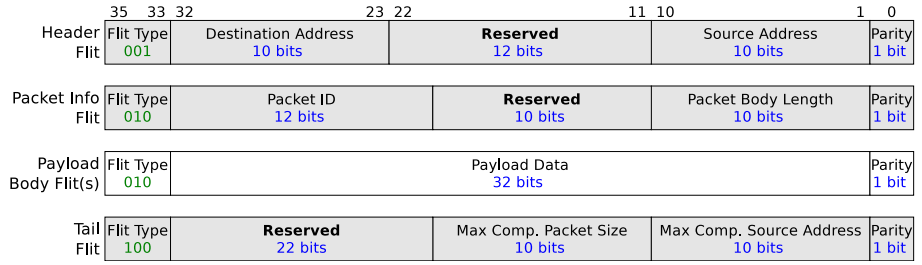



*Figure 53: Proposed Packet Structure*

The architecture of the routers that enable CDT is based on the *Baseline router* architecture of the Secure Bonfire platform, as presented in Figure 54. A *Packet Size Monitor* at each of the router's input ports extracts the source address and size of the last packet transmitted from such port (the less significant bits of the Header and Packet Info flits, respectively, without considering the parity bit). Furthermore, each input port is also equipped with a *Greatest Competitor Updater* which gathers information from other blocks as follows: i) requests the output ports that can be used to forward the packet in its monitored port from the *LBDR* Section 2.2.3) unit; ii) the grants are given by the *Allocator* for using the output ports, which by being matched with the requests, indicate the input ports that won the competition for the required output ports; and iii) information from the *Packet Size Monitors* for identifying the size and source of the packets that prevented the local packet from being forwarded. Finally, by updating the *Tail flit* with the information of the greatest competitor, each packet is able to tattletale on the longest packet found in its path together with its source. Consequently, the destination router, NI, or PE's firmware can be compared to a threshold defined during design or run time. Subsequently, as defined by the system's architect, a neutralizing action can be taken against the reported source or a report sent to the GM to identify if the found behavior was allowed or not.

### 6.4.2 Area and Power Overhead

This section presents synthesis results of the baseline router and the overhead when adding the proposed CDT. Synthesis was done with the *Genus Synthesis Solution* provided by Cadence [83], using a $40nm$ commercial CMOS technology and a clock frequency of $200MHz$. Table 27 shows area results and Table 28 power (using a value change dump), both for the worst-case corner (i.e. slow-slow, $0.99v$, and $125°C$). Furthermore, considering that corner, edge, and middle routers have a different amount

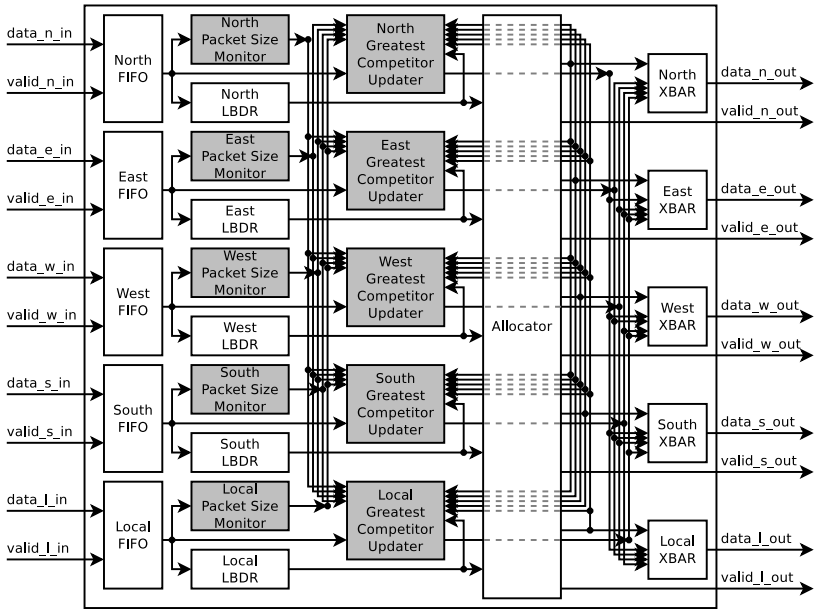



*Figure 54: Proposed Router Architecture*

of ports in a NoC following a mesh topology, results for each router type are presented. It is worth noticing that even though the reported overheads are greater than 30%, the baseline router is already a minimalist design.

*Table 27: Area overhead (40$nm$, 200$MHz$, 0.99$v$, 125°$C$)*

| | | **Area** | | | |
|---|---|---|---|---|---|
| Router | Cell Count | Sequential ($\mu m^2$) | Combinational ($\mu m^2$) | Total ($\mu m^2$) | Overhead (%) |
| Router with 3 ports (corner router - R15) | | | | | |
| Baseline | 1104 | 3636.192 | 1313.827 | 4950.019 | – |
| Proposed | 1741 | 4495.848 | 2065.291 | 6561.139 | 32.55% |
| Router with 4 ports (edge router - R14) | | | | | |
| Baseline | 1874 | 4875.696 | 2102.923 | 6978.619 | – |
| Proposed | 2971 | 6051.696 | 3411.106 | 9462.802 | 35.60% |
| Router with 5 ports (middle router - R10) | | | | | |
| Baseline | 2377 | 6114.259 | 3190.724 | 9304.982 | – |
| Proposed | 3913 | 7613.659 | 5337.629 | 12951.228 | 39.19% |

*Table 28: Power overhead (40$nm$, 200$MHz$, 0.99$v$, 125°$C$)*

| Router | Leakage Power ($mW$) | Internal Power ($mW$) | Switching Power ($mW$) | Total Power ($mW$) | Total Overhead (%) |
|---|---|---|---|---|---|
| Router with 3 ports (corner router - R15) | | | | | |
| Baseline | 0.0047 | 0.3243 | 0.0280 | 0.3571 | – |
| Proposed | 0.0063 | 0.4513 | 0.0735 | 0.5311 | 48.73% |
| Router with 4 ports (edge router - R14) | | | | | |
| Baseline | 0.0069 | 0.4478 | 0.0627 | 0.5174 | – |
| Proposed | 0.0095 | 0.6453 | 0.1653 | 0.8200 | 58.48% |
| Router with 5 ports (middle router - R10) | | | | | |
| Baseline | 0.0094 | 0.6175 | 0.1214 | 0.7482 | – |
| Proposed | 0.0134 | 0.9602 | 0.3159 | 1.2896 | 72.36% |

### 6.4.3 Conclusion

A novel approach for the detection of BRDoS attacks in NoC-based MPSoCs was presented in this section. With such a mechanism, routers register the source of arriving packets as well as their payload length. Additionally, as packets wait to be forwarded, their tail flit is updated with the information of their greatest competitor (provided that the competitor's payload is longer than others found in the path). This allows packets to "tell on" the packet that disrupted its end-to-end delay the most. Experimental results presented in Section 3.3 showed that PIR-based BRDoS attacks were avoided thanks to Round Robin arbitration, thereby, mechanisms such as CDT would be enough against Flooding DoS attacks, providing detection against PPL-based BRDoS attacks. The area and power overhead values of the proposed approach were also reported.

# 7 DoS Attack Avoidance

Similar to the approaches presented in Chapter 6, the techniques proposed in this chapter also aim to provide a better answer to research question RQ2 (Section 1.2) than those found in related works (Section 5.3). However, instead of detection, they attempt to avoid of BRDoS attacks in NoC-based MPSoCs. The first approach, presented in Section 7.1.1 and named LSDAA, targets Low-and-Slow BRDoS attacks by monitoring the local input port of routers. The second approach, presented in Section 7.2 and named BCPE, targets all the BRDoS attacks presented in Section 3.1. It ports the LSDAA approach to the NI for avoiding Low-and-Slow BRDoS attacks and includes other enforcers for Flooding BRDoS attacks. Configuration parameters of the enforcers are assigned by the Global Manager during the scheduling process and through bandwidth control policies.

## 7.1 Low-and-Slow DoS Attack Avoidance

This section proposes the use of hardware monitors at the router's local port aiming to avoid Low-and-Slow BRDoS attacks in NoC-based MPSoCs. These types of attacks include Jellyfish Inter-flit Delay Variance (Jellyfish) and Slowloris DoS attacks (Section 3.1). With such monitors, the flit injection rate and packet completeness can be observed.

The remainder of this section is organized as follows: Section 7.1.1 presents the architecture modifications done to the Secure Bonfire platform (Section 2.2.4.2) for implementing the LSDAA scheme. Section 7.1.2 details the setup of experiments and discusses the obtained results. Section 7.1.3 presents an evaluation of the overhead of the proposed mechanism in terms of area and critical-path delay. Finally, Section 7.1.4 concludes the section.

### 7.1.1 Proposed Architecture

Based on the router architecture of the Secure Bonfire open-source framework, a new architecture that can avoid LSDoS attacks in NoC-based MPSoCs is presented (depicted in Figure 55). As in the base architecture, the router contains i) input buffers (FIFOs) that store flits received through each input port until they can be forwarded; ii) routing units that implement a Logic Based Distributed Routing (LBDR) mechanism [44] which supports any turn-model based routing algorithm and provide the possibility of an in-system reconfiguration of the routing algorithm; iii) one switch allocator that arbiters data transmission from the FIFOs to the required output port; and iv) a crossbar that provides the connection between the FIFOs and the required output port. In the proposed architecture, the Low-and-Slow DoS monitor (depicted in Figure 56) sits between the local FIFO and the subsequent blocks. Such location allows the monitor to truncate malicious LSDoS packets and drop their remaining flits from the FIFO (if any).

Figure 57 depicts the Finite State Machine (FSM) that describes the functionality of the LSDoS monitor and Table 29 details the main assignments done during each state.

On reset, the LSDoS monitor goes to *Idle*, where the counter is set to the maximum acceptable inter-flit delay (which can be at set any time by the GM), from where it will transition to the *Bypass* state as soon as a header flit is outputted by the local *FIFO*. During the *Bypass* state, data will flow as if no LSDoS monitor were present. However, the counter will be reset every time a new flit is received, or decreased every clock cycle in which the FIFO remains empty and data is required. If a tail flit is received, the monitor will go back to the *Idle* state and wait for a new packet. Otherwise, if

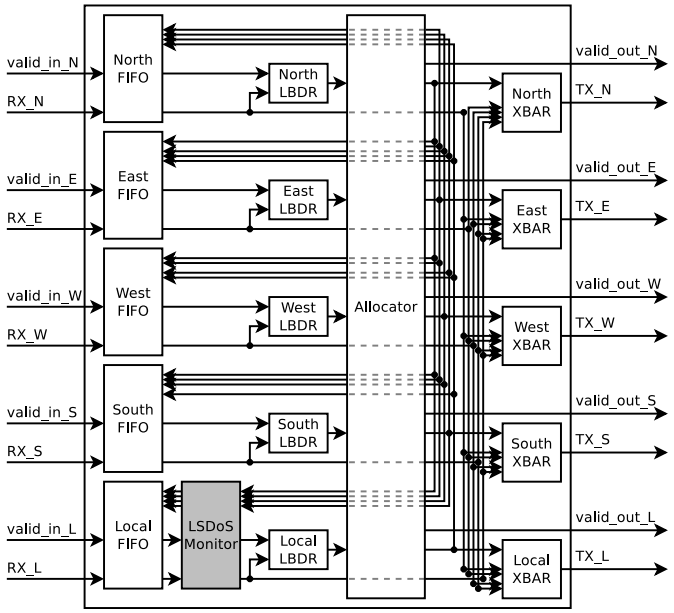



*Figure 55: LSDoS monitor - Router architecture*

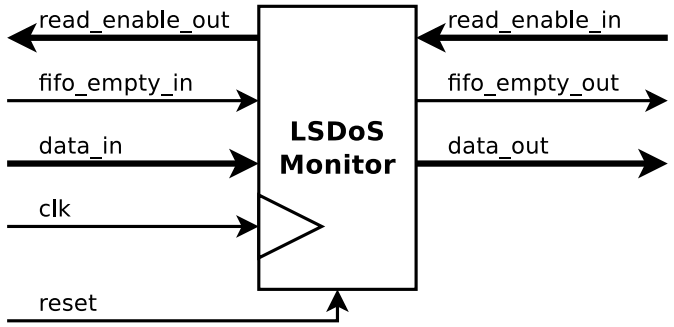



*Figure 56: LSDoS monitor - Black box diagram*

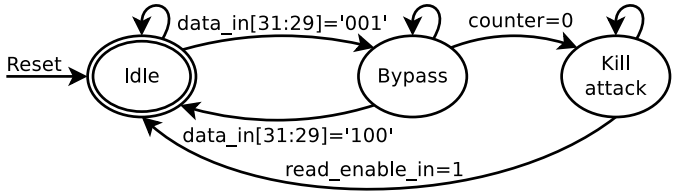



*Figure 57: LSDoS monitor - Finite State Machine (FSM)*

the counter times out, the presence of an LSDoS attack is assumed. Consequently, the monitor will transmit a predefined tail flit and go back to *Idle*. Thanks to this tail flit, incomplete packets generated by the Slowloris DoS attack attempt are terminated while packets generated by a Jellyfish DoS attack attempt are cropped, releasing the communication path. Consequently, at the destination, the received packet will be dropped, since the declared packet size will not match the actual size. Moreover, by

*Table 29: LSDoS monitor - FSM assignments*

| | |
|---|---|
| Idle | counter = MAX_VALUE - 1;<br>read_en_out = '1';<br>data_out = 0;<br>fifo_empty_out = '1'; |
| Bypass | read_en_out = read_enable_in;<br>data_out = data_in;<br>fifo_empty_out = fifo_empty_in;<br><br>if ( fifo_empty_in = '0' ) then<br>  counter = MAX_VALUE - 1;<br>elsif ( read_enable_in = '1' ) then<br>  counter = counter - 1;<br>end if; |
| Kill attack | counter = MAX_VALUE-1;<br>read_en_out = 1;<br>data_out = "1000 0000 0000 0000 0000 0000 0000 0001";<br>fifo_empty_out = '1'; |

returning to *Idle*, all the remaining flits from the malicious packet, in case of a Jellyfish DoS attack attempt, will be dropped, thereby neutralizing the attack. This is because the proposed LSDoS monitor only forwards flits while being at the *Bypass* state, to which it will only transition when detecting a header flit.

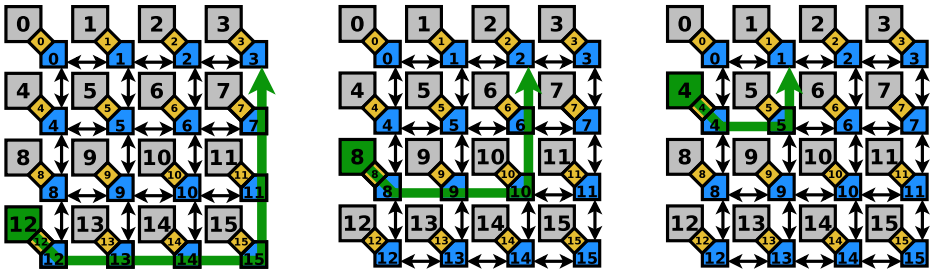
### 7.1.2 Experimental Work

In order to validate the impact caused to a NoC-based MPSoC by the introduced LSDoS attacks and the efficacy of the proposed LSDoS monitor, the implementation of the attack behavior and the detection mechanism have been done based on a VHDL RTL description, integrated into the Secure Bonfire NoC platform. Moreover, such an integration allowed the use of the platform's traffic generators to create sensitive, normal, and DoS attack traffic with different PIR values and PPLs. The traffic generators, however, were modified so that transmission delays could be added between the flits of the attack packets. Section 7.1.2.1 details the scenario configurations set for the experiments. Section 7.1.2.2 summarizes the obtained results regarding the comparison of FDoS and LSDoS attacks. Finally, Section 7.1.2.3 presents the results that prove the efficacy of the proposed LSDoS monitors.

#### 7.1.2.1 Simulation Scenarios

For the experiments detailed in this section, the same simulation scenarios used for testing CPRD and CPDD were adopted. Such scenarios are depicted in Figure 58 where in $4 \times 4$ mesh NoC-based MPSoCs, routers apply the XY-routing algorithm and credit-based flow control with fair Round-Robin arbitration (on packet level) as well as the wormhole switching with 4-flit deep FIFOs.

Moreover, as shown in Figure 58 each scenario has its own distinct source and destination of monitored traffic. For each scenario, 14 sets of experiments were performed. Each set of experiments uses a unique attacker location, covering all the nodes in the network (except for the monitored origin and destinations of the evaluated scenario). The attacker nodes in all scenarios, send their traffic to the monitored node's destination, ensuring a collision with the monitored traffic. It is important to note that each experiment was performed for $20$ pseudo-random traffic simulation seeds to provide uniform results.



*(a) Scenario 1* *(b) Scenario 2* *(c) Scenario 3*

*Figure 58: Simulation Scenarios*

#### 7.1.2.2 Flooding DoS vs Low-and-Slow DoS attacks

To evaluate the impact of LSDoS attacks on NoC-based MPSoCs, experiments were done using Scenario 1 (Figure 58a) where the sensitive traffic flows from PE12 to PE3. In each experiment, all traffic was transmitted with the same PIR value, either 0.003, 0.01 (one packet every 100 clock cycles), or 0.017. For both attacks, FDoS and LSDoS, the source is PE0 which sends the malicious packets to PE3, as well.

Figure 59 summarizes the results obtained with different configurations of LSDoS attacks and their equivalent configuration for FDoS as well as configurations where no attack was present (Horizontal dashed lines). Results are grouped by the inter-flit delay value of the LSDoS attack together with the equivalent FDoS packet length. Moreover, each group is composed of three pairs of bars, one for each PIR value, and the bars of each pair correspond to a different attack type; LSDoS and FDoS respectively from left to right. Additionally, the Y axis corresponds to the mean end-to-end communication delay of sensitive packets in clock cycles (transmitted from PE12 to PE3). As shown in Figure 59, the effect caused by each LSDoS attack has a corresponding FDoS configuration, regardless of the packet injection rate.

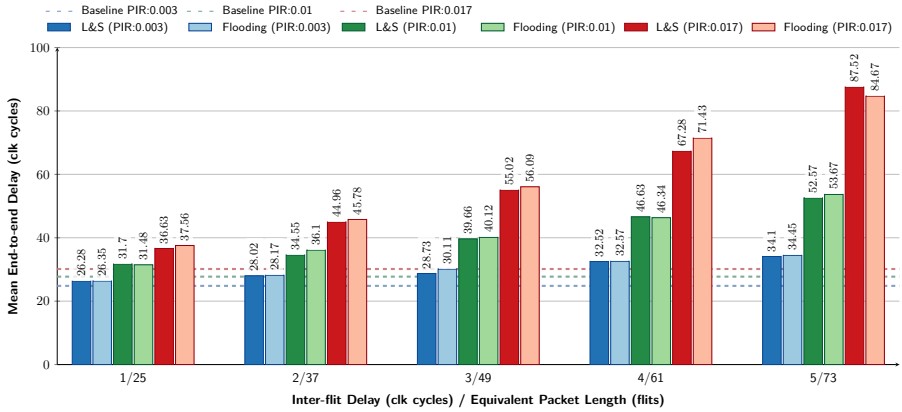



*Figure 59: Comparison of mean end-to-end delay of sensitive traffic under flooding and equivalent LSDoS attacks for different PIR values*

#### 7.1.2.3 Efficacy of the LSDoS monitor

The results of the experiments done to test the LSDoS Monitor for each of the three scenarios presented in Section 7.1.2.1 are summarized in Figure 60, Figure 61, and Figure 62, respectively. For all the figures, the X axis corresponds to the number of clock cycles that the attacking PE waits between the transmission of consecutive flits of the same packet. On the other hand, the Y axis corresponds to the mean end-to-end delay values of the monitored packets (also in clock cycles). Additionally, the blue circular markers identify the results when considering a PIR of 0.003 for all the packets flowing through the NoC (i.e. monitored, malicious and random). The yellow square markers correspond to a PIR of 0.01 and the red triangles to a PIR of 0.03. Furthermore, dotted lines show the proportionality between the inter-flit delay and the mean end-to-end delay of the monitored packets when the proposed countermeasure is disabled. In contrast, the solid lines show the impact of the LSDoS attack to the monitored packets in a NoC where the proposed countermeasure is implemented. As shown by each pair of lines (i.e. lines with the same color/marker), experiments with and without countermeasures achieve the same results until an inter-flit delay threshold is reached. This is due to the fact that the countermeasure is triggered by exceeding the threshold (for our experiments, we adopted an inter-flit delay threshold of 5 clock cycles, nevertheless, dynamic configuration of this value can be done by the GM).

| AL | Detected |
|---|---|
| 0 | ✔ |
| 1 | ✔ |
| 2 | ✔ |
| 3 | S. dest |
| 4 | ✔ |
| 5 | ✔ |
| 6 | ✔ |
| 7 | ✔ |
| 8 | ✔ |
| 9 | ✔ |
| 10 | ✔ |
| 11 | ✔ |
| 12 | S. src |
| 13 | ✔ |
| 14 | ✔ |
| 15 | ✔ |

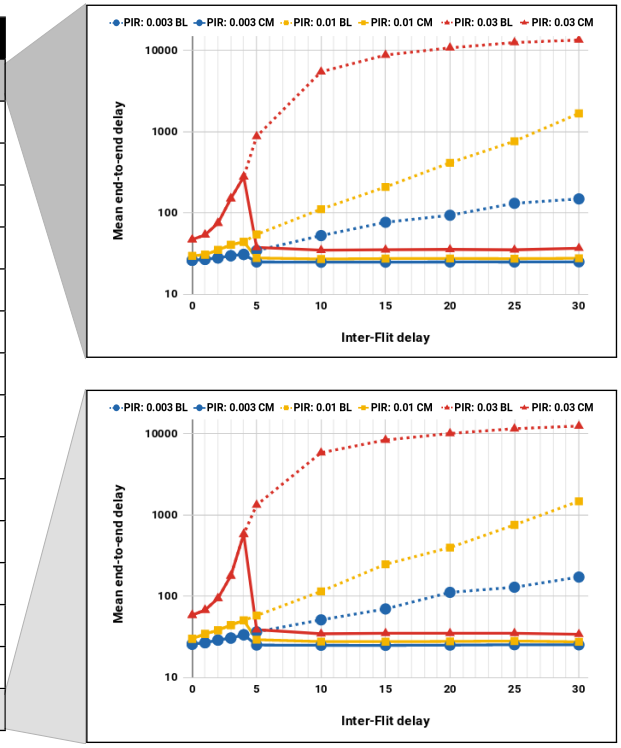



*Figure 60: Detection and counter-measure results for Scenario 1*

Additionally, the resulting latency after exceeding the adopted inter-flit delay threshold is smaller than that of normal conditions (without attack) because once the attack is detected, the proposed countermeasure truncates the malicious packet at the source, hence, the channel is freed earlier than if a normal-sized packet were to be transmitted.

| AL | Detected |
|---|---|
| 0 | ✔ |
| 1 | ✔ |
| 2 | S. dest |
| 3 | ✔ |
| 4 | ✔ |
| 5 | ✔ |
| 6 | ✔ |
| 7 | ✔ |
| 8 | S. src |
| 9 | ✔ |
| 10 | ✔ |
| 11 | ✔ |
| 12 | ✔ |
| 13 | ✔ |
| 14 | ✔ |
| 15 | ✔ |

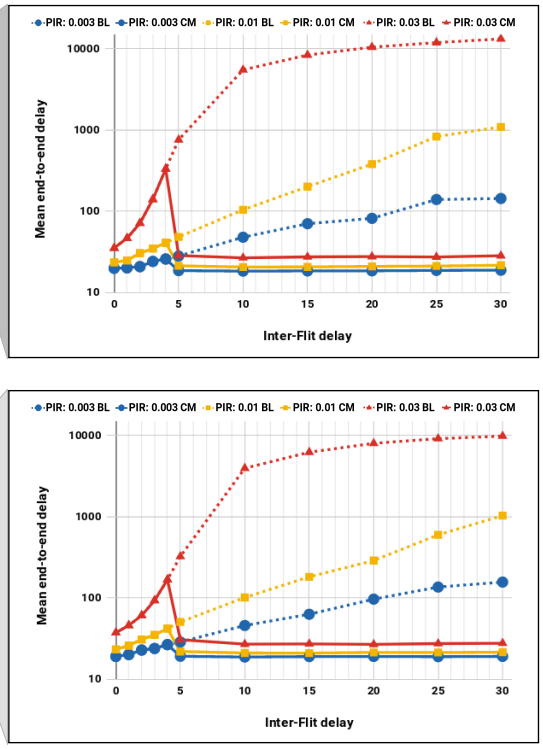



*Figure 61: Detection and counter-measure results for Scenario 2*

Even though experiments were done considering the 14 possible Attack Locations (ALs) (except for the source and destination of the monitored traffic in each scenario), figures show results of only two attacker locations for each scenario. Hence, as expected, all locations presented similar behavior.

### 7.1.3 Area and Timing

This section presents an overhead evaluation of the proposed mechanism on the router and the system. The proposed architecture was synthesized using the $0.18$ $\mu$m AMS library and Synopsys Design Vision at 200 MHz. Table 30 shows the area overhead of the proposed mechanism on the baseline router. However, even though the overhead of the proposed solution is just 2%, it is important to note that the baseline architecture is very minimalist. To put the results in perspective, for a 4 $mm^2$ chip with 16 cores and a SoC using a $4 \times 4$ mesh network, the area overhead would be around 0,06% which is completely negligible. Table 30 also shows that the overhead of the proposed monitor on the critical path delay of the system is negligible.

*Table 30: Area comparison of the proposed architecture and the baseline router*

| | Area | | | | Timing | |
|---|---|---|---|---|---|---|
| | Sequential ($\mu m^2$) | Combinatorial ($\mu m^2$) | Total ($\mu m^2$) | Overhead (%) | Critical-Path Delay ($ns$) | Overhead (%) |
| Baseline | 48562.21 | 48336.42 | 96898.63 | – | 4.98 | – |
| Proposed | 49411.75 | 49832.29 | 99244.05 | 2.4% | 4.97 | $\approx 0$ |

| AL | Detected |
|---|---|
| 0 | ✔ |
| 1 | S. dest |
| 2 | ✔ |
| 3 | ✔ |
| 4 | S. src |
| 5 | ✔ |
| 6 | ✔ |
| 7 | ✔ |
| 8 | ✔ |
| 9 | ✔ |
| 10 | ✔ |
| 11 | ✔ |
| 12 | ✔ |
| 13 | ✔ |
| 14 | ✔ |
| 15 | ✔ |

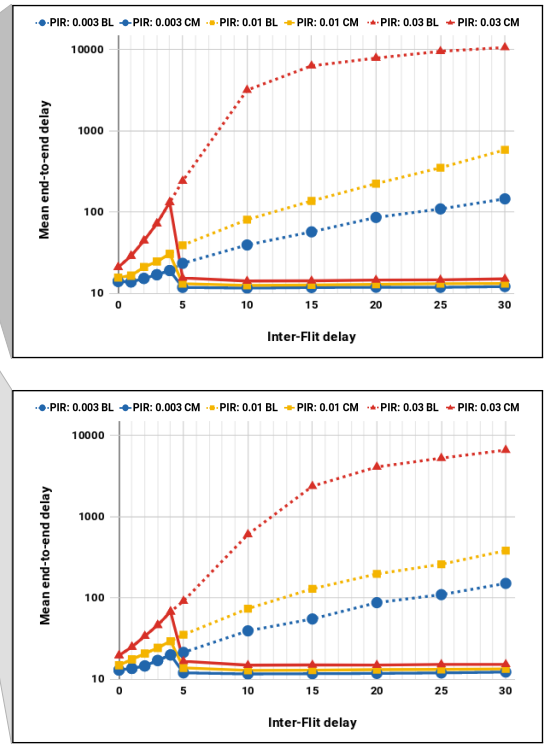



*Figure 62: Detection and counter-measure results for Scenario 3*

### 7.1.4 Conclusion

Wormhole routing makes NoC-based MPSoCs prone to LSDoS attacks. A comparative analysis with Flooding DoS attacks has been carried out, proving that LSDoS attacks can be as effective as Flooding DoS attacks requiring fewer data to be transmitted. The mitigation of single-source and multi-source attacks can be effectively handled at the connection point between the network interface and the local port of the NoC router with neglectable overhead compared to a minimalist router.

## 7.2 Bandwidth Control Policies Enforcement

A trend has been seen over the past few years for implementing different machine learning approaches attempting to learn the normal traffic patterns presented in NoC-based MPSoCs so that any different behavior can be tagged as a DoS attack [16, 20–22]. The detection algorithm is trained by analyzing traffic features of the NoC executing specific tasks or benchmarks on designated PEs, with and without the influence of an attack to learn both behaviors. Subsequently, the algorithm is tested with conditions similar to the ones experienced during the training phase to be classified as normal behavior or under attack. Extracted traffic features are normally related to PIR-based DoS attacks where a malicious source transmits a more-than-normal number of communication requests or dummy packets. Even though such approaches work in the mentioned specific configurations, they would not work on a multi-tenant execution environment like the one which is the focus of this thesis. This is because of the diversity of applications that can be submitted by the different users where the possible combinations of them

running simultaneously eliminate the probability of the so-called “normal behavior". Therefore, instead of trying to learn a “normal" traffic behavior of the entire Multi-tenant NoC-based MPSoC attempting to detect BRDoS attacks, such attacks could be avoided by estimating traffic profiles for individual tasks, assigning bandwidth control policies for each execution and enforcing them.

As will be seen in Section 8, Bandwidth Control Policies (BCPs) can limit packet injection rate (in the form of a minimum inter-packet delay limit), maximum packet payload lengths, and maximum inter-flit delays. Therefore, BRDoS attacks that attempt to disrupt the communication performance of Multi-tenant NoC-based MPSoCs could be avoided with strategies that enforce the rules established by such BCPs. Such strategies are listed in Table 31 together with the type of DoS attack that they target. Additionally, Round-robin arbitration is also considered a strategy against PIR-based DoS attacks due to the results shown in Section 3.3.

*Table 31: BRDoS Attacks and Avoidance Strategies*

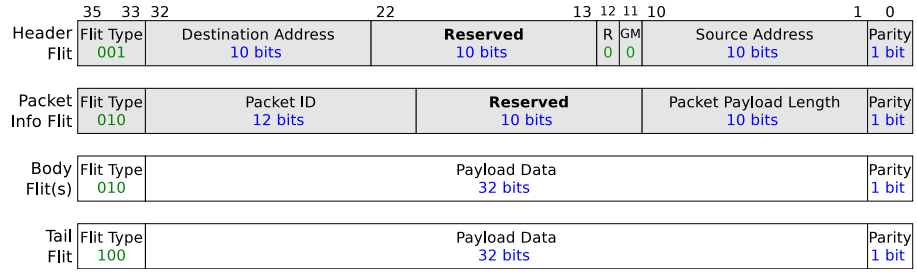
| | | **BRDoS Attack Types** | | | |
|---|---|---|---|---|---|
| **Avoidance Strategies** | | PIR-FDoS | PPL-FDoS | Slowloris | Jellyfish |
| *AS1* | Enforce a minimum inter-packet delay | ✔ | | | |
| *AS2* | Enforce a maximum packet payload size | | ✔ | | |
| *AS3* | Enforce a maximum inter-flit delay [3] | | | ✔ | ✔ |
| *AS4* | Implement Round-robin arbitration [66] | ✔ | | | |
| *AS5* | AS1 + AS2 + AS3 | ✔ | ✔ | ✔ | ✔ |
| *AS6* | AS2 + AS3 + AS4 | ✔ | ✔ | ✔ | ✔ |

### 7.2.1 Proposed Architecture

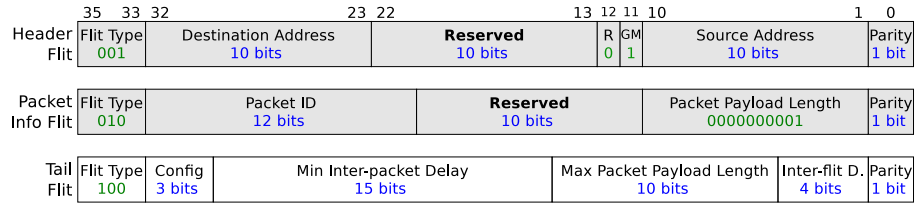
In this section, an architecture for the Bandwidth Control Policies Enforcement (BCPE) approach with two variants is presented. Both variants allow the MPSoC's GM to configure enforcers after establishing the Service Level Agreement (SLA) and determining the rules of the BCPs. In the first variant, enforcers are configured only once per task execution, before it starts. By doing so, violation attempts done by attackers are neutralized without other users experiencing any disruption. The second variant extends the previous in a way that violation attempts to any rule of a BCP are reported to the GM so that it can decide if, when, and to which extent BCPs should be modified. This may also lead to a renegotiation of the SLA, consequently generating an overcharge to users attempting to use more resources than what they had previously requested/paid for.

For both variations, data flowing through the MPSoC will follow the basic packet structures presented in Figure 2. However, as depicted in Figure 63, NIs will now also utilize one or two bits from the Header flit that were tagged as reserved before. The *GM* bit may only get a *1* value from the NI connected to the PE running the GM. Moreover, the *R* bit may only be set to *1* by NIs when generating a report. Other packets, generated by user tasks will be forwarded with these values set to *0* to avoid spoofing. Therefore, the combination of values from these two bits helps to identify which type of packet is being forwarded.

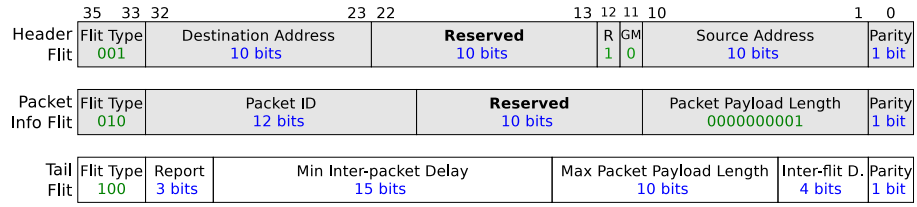
Both types of packets, for configuration and infraction reports, in Figure 63 use almost the same structure. Their Tail flit has 15 bits for minimum inter-packet delay, 10 bits for maximum packet payload length, and 4 bits for the maximum inter-flit delay. However, the configuration packet uses 3 bits of the Tail flit to define which of the




*(a) Structure of Data Packets*




*(b) Structure of Configuration Packet Sent by the Global Manager*




*(c) Structure of Infraction Report Packet Sent to the Global Manager*

*Figure 63: Packet Structures for Bandwidth Control Policies Enforcement*

three limits should be configured while the infraction-report packet those bits are used to define which of the three limits had a violation attempt.

Furthermore, to avoid network congestion, the avoidance strategies *AS1*, *AS2*, and *AS3* listed in Table 31 should be implemented as close as possible to traffic sources, i.e., inside the NI. Figure 64 shows a proposed Local Security Manager (LSM) at the input port of the NI, where the PE connects to it. Once again, it is worth noticing that by implementing a distributed solution, such as this one, it can be scaled to NoC-based MPSoCs of different sizes.

### 7.2.2 Conclusion

When targeting BRDoS attacks in NoC-based MPSoC, machine learning approaches that base their detection on identifying a “normal behavior" of the traffic flowing through the NoC, become useless when implementing multi-tenancy and not limiting the type of applications that could be executed. On the other hand, if a traffic profile of each application task is determined together with the Service Level Agreement, bandwidth control policies can be established and enforced, enabling the MPSoC's Global Manager to achieve the agreed quality of service regardless of DoS attack attempts.

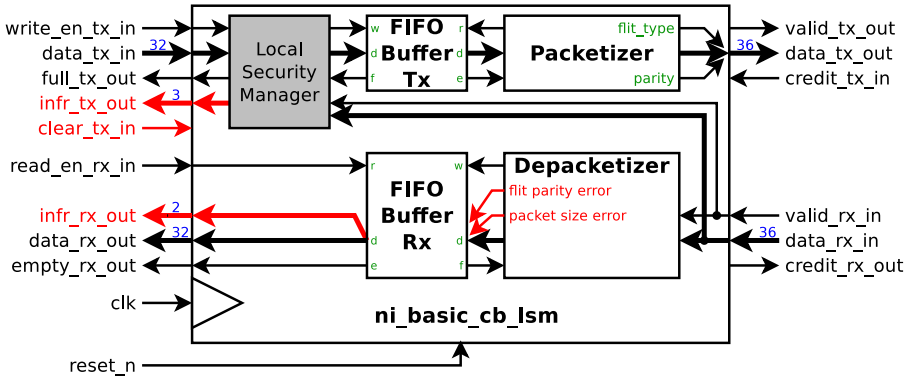



*Figure 64: Proposed Architecture - Network Interface with Local Security Manager*

# 8 System-level Service Management

In utility computing, application execution services are provided on demand in a pay-as-you-go manner as other utilities such as water, electricity, gas, and telephony [84]. This section describes a system-level view of an execution environment for utility computing where Multi-tenant NoC-Based MPSoCs can be found among the available resources. On top of that, it explains how the MPSoC's software and hardware come together, eliminating the vulnerability to BRDoS attacks, to which systems in multi-tenant environments are exposed. Figure 65 illustrates the main components of such MPSoCs.

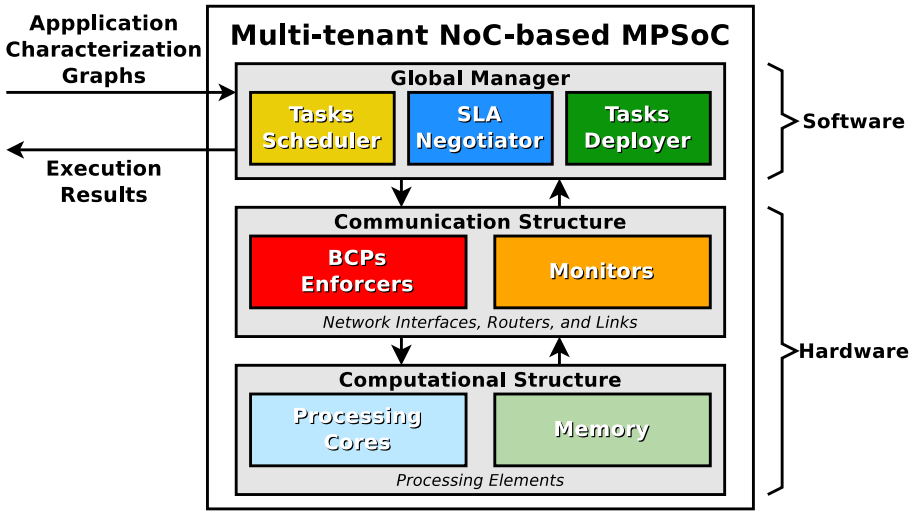



*Figure 65: MPSoC Application Execution - System Level*

As explained in Section 2.1, MPSoCs are composed of two hardware structures: i) Computational, which includes any type of Processing Elements (PEs), and ii) Communication, which in the case of NoC-based MPSoCs include Network Interfaces (NIs), Routers, and Links. Additionally, MPSoCs incorporate a Global Manager (GM) for controlling the entire system. Such a manager is usually implemented in software and during the system boot-up, is set to run on one of the MPSoC's PEs [34–41].

The following sections of this chapter describe the proposed features that should be provided by NoC-based MPSoCs when enhancing execution environments for utility computing (such as Fog and Cloud Computing) in order to tackle BRDoS attacks.

## 8.1 Cloud Connectivity

As explained in Section 2.3.1, when a user submits an application for its execution, a workflow profile of the application expressed as a communication tasks graph (including tasks dependencies as well as an estimation of the execution costs and the amount of data to be transmitted for each dependency) is provided apart from the executable code. Such information is given to a Fog + Cloud scheduler which selects the appropriate resources that should be used for the execution [69] and formats data as required by the selected resource. In the context of this thesis, such resources correspond to NoC-based MPSoCs and the format is the application code compiled for the specific MPSoC together with the Application Characterization Graph (APCG) describing it (Section 2.3.1). This information is received by the MPSoC's GM which from here

on takes care of coordinating the entire execution, and once it is finished, returns the results to be sent back to the user.

## 8.2 MPSoC Management

With the aim of using MPSoCs to enhance utility computing, the MPSoC's Global Manager (GM) is in charge of controlling the execution of the received applications. Based on the service deployment life cycle presented in [85], Figure 66 depicts the Application Execution Life Cycle followed by the GM.

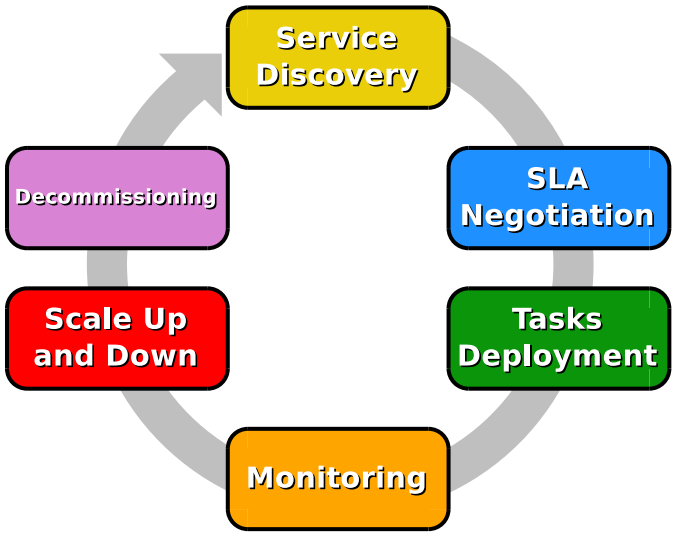



*Figure 66: Application Execution Life Cycle*

During the Service Discovery phase, the APCG of the application submitted by the user is used for discovering the best-suited available resources for its execution. This process is done by the *Tasks Scheduler* (Section 2.3). However, since the target execution environment must support multi-tenancy, the scheduler must provide detailed resource usage guidelines so that users take precautions against disrupting other tenants. It is proposed in this chapter that such guidelines include how an application should make use of the NoC bandwidth allocated for its execution. This can be done by extending the scheduler formulation to define Bandwidth Control Policies (BCPs) including the following three rules:

- $mIPD_{y,x}$: Minimum inter-packet delay for tile $t'_{y,x}$ (for controlling the Packet Injection Rate)
- $MPPL_{y,x}$: Maximum Packet Payload Length for tile $t'_{y,x}$
- $MIFD_{y,x}$: Maximum Inter Flit Delay for tile $t'_{y,x}$

At the Service Level Agreement Negotiation phase, the *SLA Negotiator* communicates the result obtained in the previous phase and together with the Fog/Cloud SLA Negotiator, forward such information, negotiating the monetary costs of executing the application and the quality of service (QoS) for the execution.

Once the user and the execution platform reach an agreement, application tasks are forwarded to their designated PEs. This phase is referred to as Task Deployment.

In order for an execution environment to make sure that it is honoring an SLA, it goes into a monitoring phase while an application is being executed. Thus, if the execution environment were to violate the SLA, it may be liable to monetary penalties and/or loss of clients. However, users/clients should also be penalized if they violate the SLA because they would be using resources contracted by other tenants. QoS and SLA violation attempts monitoring features are explained in Sections 8.3 and 8.4.

Provided that the monitoring phase signals that predefined thresholds have been reached, the GM may decide to increase or reduce the allocated resources to any given application execution which is done in the Scale Up and Down phase. Scaling up of resources is usually done for compensating previous low-performance levels so that deadlines can still be met. However, scaling down can also be used for limiting task executions that are using resources beyond what is stipulated by the SLA, or even before the execution starts to prevent disrupting other tenants.

Finally, during the decommissioning phase, the final processing of executions is done before erasing data used by an executed application that will not be required by other tasks. One operation is gathering results from parallel tasks from the same application so that they can be sent together to the user that requested the execution.

Since values in application profiles are not always exact, but estimated mean values, service providers may generate SLAs considering some flexibility, thus allowing application executions to deviate within a tolerable range. However, greater flexibility reduces the profit obtained by the service provider, while overlapping flexibility allowed to different users may lead to undesired QoS.

With this in mind, SLAs offered to users shall communicate an established Bandwidth Control Policy (BCP) for the application's execution. Furthermore, such policy should not contain rules with single bandwidth amounts representing a minimum assured value, a maximum allowed value, and/or an average value. The values should instead be decomposed into three values: i) the minimum inter-packet delay; ii) the maximum packet payload length; and iii) the maximum inter-flit delay. Furthermore, such values can be used not only for improving the scheduling quality but also for configuring BCPEs that prevent BRDoS attacks on other users of a Multi-tenant MPSoC.

## 8.3 Quality of Service Monitoring

As mentioned before, an execution environment is liable to penalties if it fails to meet what is established in the SLA. Therefore, the execution of its applications should be monitored. Furthermore, by assessing the QoS experienced by one application execution, the GM may decide to grant it more resources to make up for deviations from the schedule. However, such action may harm another execution, which by also being monitored, will help the GM to take another decision before the problem is beyond repair. To enable QoS monitoring, a NoC-based MPSoC can include CPRD from Section 6.1 or CPDD from Section 6.2, which enable each packet to report the worst disruption found in its path. Even better, ADAD from Section 6.3 or CDT from Section 6.4 for in addition to reporting the disruption point, the source that caused the disruption should also be identified.

## 8.4 SLA Violation Attempts Monitoring

It has been proposed in Section 8.2 that a task scheduler should provide BCPs so that users have the required information for trying not to disrupt the execution of other tenants with their own. However, given the existence of malicious users who may

attempt to execute DoS attacks, an execution environment cannot expect that all users follow its utilization guidelines. Therefore, BCPs should not only be followed by the execution environment but also enforced for application executions. BCP enforcers are presented in Section 7.2 together with SLA violation attempts monitors. Such monitors are able to report to the GM if any of the scheduled tasks attempt to usufruct resources beyond the SLA. Consequently, with such information, the GM can decide if a new SLA should be negotiated, increasing the execution costs to SLA-violating users.

## 8.5 Conclusion

With the System-level Service Management presented in this chapter, together with the Bandwidth Control Policies Enforcers from Section 7.2, DoS detection becomes unnecessary in Multi-tenant NoC-based MPSoCs. This is due to the fact that regardless of the malicious attempts of attackers to implement BRDoS attacks, they will only be able to make use of the communication structure as established in the Service Level Agreement (SLA) trough Bandwidth Control Policies (BCPs).

Additionally, considering that in utility computing, application execution services are provided on demand in a pay-as-you-go manner, if an application attempts to use more resources than what was granted in the SLA, the user who submitted the application will receive a more expensive offer for upgrading their SLA, provided that more resources are available. Furthermore, if the user refuses the new SLA, their application will continue to be forced to honor the initial SLA without disrupting others. On the other hand, if the user agrees to pay more, the enforcement will be relaxed according to the new SLA, still without disrupting others since the granted resources were available for demand.

# 9 BRDoS Attack Impact Assessment

In order to guarantee that a NoC-based MPSoC design is not vulnerable to Bandwidth Reduction DoS (BRDoS) attacks, the performance of the communication structure must be observed based on traffic expectations resulting from the task scheduling and then compared to the performance when malicious traffic is present. To this end, test benches should be implemented recreating scenarios like the ones depicted in Figure 19a and Figure 19b where malicious packets attempt to disrupt legitimate packets. Furthermore, the performance of the MPSoC's communication structure should be assessed for the traffic patterns presented in Figure 20.

Towards answering research question RQ3 (Section 1.2), Section 9.1 details a proposed BRDoS attack impact assessment model for NoC-based MPSoCs. Section 9.2 explains a simple way to generate traffic within the proposed model, mimicking either legitimate or malicious traffic.

## 9.1 Model

Figure 67 depicts a proposed DoS attack impact assessment model for NoC-based MPSoCs. It includes bandwidth-test modules that can be configured as i) source: to generate traffic or ii) destination: to calculate the end-to-end delay of received packets.

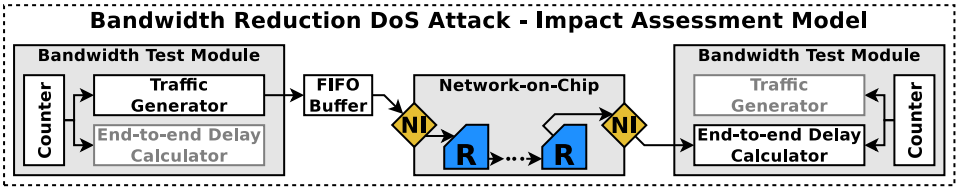


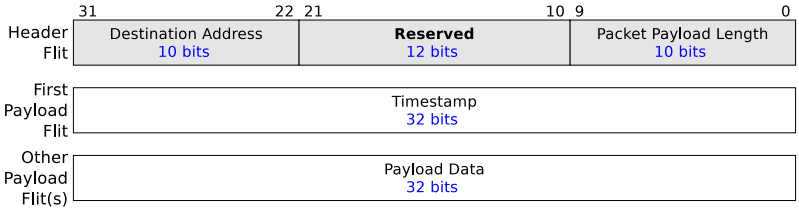

*Figure 67: Bandwidth Reduction DoS Attack - Impact Assessment Model*

By introducing a timestamp into the transmitted packets (Figure 68) and subtracting its value from the current system time upon arrival, the end-to-end communication delay can be calculated. Subsequently, the communication degradation of monitored packets can be assessed for different traffic configurations.

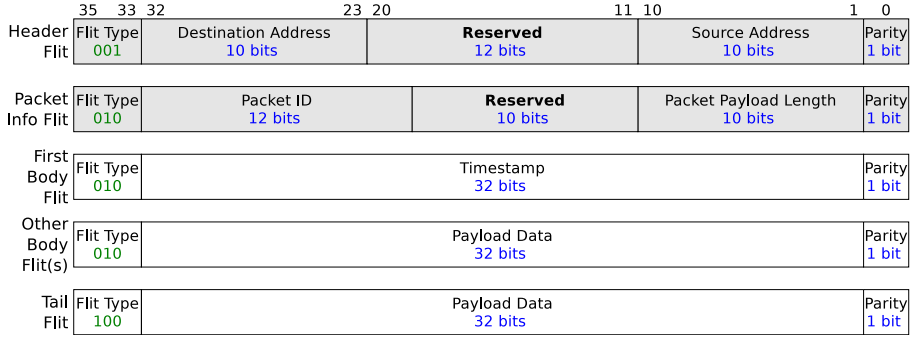
## 9.2 Simple Traffic Generation

Traffic can be generated following Algorithm 2. Once the normal parameters have been set for the application targeted by the MPSoC design, the BRDoS Attacks presented in Section 3.1 can be implemented as follows: i) PIR-based FDoS attacks by adjusting high Packet Injection Rates ($PIR$); ii) PPL-based FDoS attacks by configuring large packet payload lengths ($PPL$); iii) Jellyfish Inter-flit Delay Variance (JFID) LDoS attack by providing high inter-flit delays (IFD); and iv) Slowloris LDoS attacks by setting the Packet Missing Flits ($PMF$) parameter to a value greater than zero.

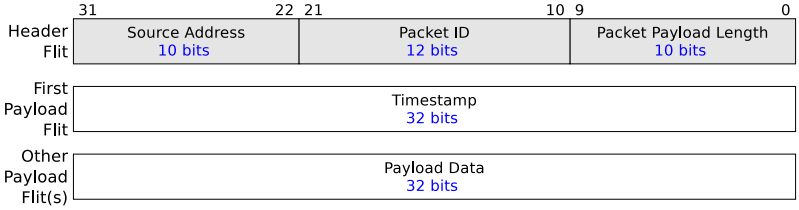
Since only the communication structure of the NoC-based MPSoC needs to be evaluated, the PEs can be replaced by either: i) Processors executing Algorithm 2 for generating packets and a procedure for calculating the end-to-end delay of monitored packets upon their arrival; or ii) Bandwidth Test Modules as the one illustrated in Figure 67, also capable of generating the described traffic patterns, as well as calculating the end-to-end delay of packets, but which incorporate a counter for generating and evaluating timestamps. The counters of all Bandwidth Test Modules must be connected to the same clock tree and started on system boot-up.




*(a) Data From the Traffic Generator to the NIs - with Timestamp*




*(b) Data Exchanged Between the NIs and Routers - with Timestamp*




*(c) Data From the NIs to the End-to-end Delay Calculator*

*Figure 68: Data Structures for End-to-end Delay Calculation*

**Algorithm 2:** Simple traffic generation method

**input :**
$DST$ : Destination Address
$PIR$ : Packet Injection Rate *// Packets per clock cycles*
$PPL$ : Packet Payload Length *// Number of Flits ( > 0 )*
$IFD$ : Inter-flit Delay *// Number of clock cycles*
$PMF$: Packet Missing Flits *// Condition: (PPL-PMF) > 0*

// *ipd: Inter-packet delay in clock cycles*
// **get_time**(): *Retrieves system time value*
// **send_data**($data$): *Sends data to the NI*
// **sleep**($n$): *Halts the transmission $n$ clock cycles*

1 $ipd \leftarrow \lceil 1/PIR \rceil - (PPL+1) - PPL * IFD$
2 **if** $PIR > 0$ **and** $PPL > 0$ **and** $ipd \geq 0$ **then**
3 $generation\_enabled \leftarrow$ **True**
4 **while** *generation_enabled* **do**
5 **if** $PMF > 0$ **and** $PPL - PMF > 0$ **then**
6 $payload\_counter \leftarrow PPL - PMF$
7 $generation\_enabled \leftarrow$ **False**
8 **else**
9 $payload\_counter \leftarrow PPL$
10 $header\_timestamp \leftarrow$ **get_time**()
11 **send_data**($[DST\ PPL]$)
12 **sleep**($IFD$)
13 **send_data**($header\_timestamp$)
14 **while** $payload\_counter > 1$ **do**
15 **sleep**($IFD$)
16 **send_data**($0$)
17 $payload\_counter \leftarrow payload\_counter - 1$
18 **sleep**($ipd$)

# 10 Conclusion

Adding new devices to current execution environments exposes each of them to the security vulnerabilities of the other. Such is the case when introducing NoC-based MPSoCs to Cloud/Fog Computing. This thesis presented four types of Denial of Service (DoS) attacks that users from a Fog/Cloud execution environment can use for Multi-tenant NoC-based MPSoCs. The aim of the four presented attacks is to impact the NoC's performance and by doing so, prevent packets generated by other tenants to use the MPSoC's communication structure. The first approach which is referred to as PIR-based, floods the network with a higher-than-normal packet injection rate aiming to clog the network. Such a type of attack is the most common approach in the state-of-the-art. However, it has been shown in this thesis that by implementing a fair arbitration scheme such as Round-robin, this type of attack can be avoided.

The second type of attack, namely the PPL-based DoS attack, has been considered by a few authors. It floods the network by sending lots of information encapsulated into packets with a long payload length. The third and fourth attack types are called low-and-slow DoS (LSDoS) attacks, namely Jellyfish and Slowloris. These two attacks had never before been studied in the context of NoC-based MPSoCs, nor can they be detected/avoided by the approaches presented in the related work. Detection mechanisms found in the literature usually count the number of packets being transmitted, the transmission requests, or consumed bandwidth in an NoC link. However, as detailed in this thesis, LSDoS attacks are capable of causing the same impact as FDoS attacks with the transmission of less data.

Apart from explaining the above four attacks, this thesis also introduced four approaches for their detection, two others techniques for their avoidance, and a system-level description of how they can be leveraged in a Multi-tenant NoC-based MPSoC to tackle the studied attacks. Furthermore, an algorithm for the generation of all the targeted attacks and a model for assessing the impact they cause on any given NoC was also introduced.

Moreover, it was observed that some authors are lately focusing on the detection of DoS attacks in NoC-based MPSoCs by using machine learning. However, no normal traffic profile can be learned from a vastly dynamic environment such as a multi-tenant execution system with many users. Since this type of environment follows a pay-per-use model, it was proposed in this thesis that from the, currently used, System Level Agreements (SLAs) be derived Bandwidth Control Policies. Additionally, those policies should be enforced to prevent an application execution from disrupting others, and violation attempts monitored for allowing resource-demanding applications to use more resources than those stipulated in the SLA, provided that such resources are indeed available and that an increment to the payment is accepted along with a new SLA.

With the research done during the development of this project, and as explained in this thesis, it was found that despite efforts being made against DoS attacks in NoC-based MPSoCs, some possible DoS attack variations are still being neglected. Additionally, although machine learning approaches for DoS attack detection in NoC-based MPSoCs have been gaining ground, they require the existence of a normal traffic profile to be learned which is not the case for dynamic multi-tenant execution environments. Therefore, it is concluded that for multi-tenant execution environments containing NoC-based MPSoCs following a pay-per-use model, attempts for DoS attack detection become irrelevant. As an alternative, it is proposed that the established SLAs be enforced and violation attempts be monitored. With such an approach, executions requiring more resources will trigger a new SLA negotiation, allowing the user to pay for

more resources, provided that they have not been allocated to other executions, thus shielding NoC-based MPSoCs against DoS attacks which are intrinsic to multi-tenant execution environments.

# List of Figures

# List of Tables

# References


[1] Cesar G. Chaves, Siavoosh P. Azad, Thomas Hollstein, and Johanna Sepúlveda, "A Distributed DoS Detection Scheme for NoC-based MPSoCs," in *Circuits and Systems (NorCAS), 2018 IEEE Nordic Conference on*, IEEE, October 2018.

[2] Cesar G. Chaves, Siavoosh P. Azad, Thomas Hollstein, and Johanna Sepúlveda, "DoS Attack Detection and Path Collision Localization in NoC-Based MPSoC Architectures," *Journal of Low Power Electronics and Applications*, vol. 9, no. 1, 2019.

[3] Cesar G. Chaves, Siavoosh P. Azad, Johanna Sepúlveda, and Thomas Hollstein, "Detecting and Mitigating Low-and-Slow DoS Attacks in NoC-based MPSoCs," in *14th International Workshop on Reconfigurable Communication-Centric Systems-on-Chip (ReCoSoC)*, IEEE, July 2019.

[4] Cesar G. Chaves, Johanna Sepúlveda, and Thomas Hollstein, "Lightweight Monitoring Scheme for Flooding DoS Attack Detection in Multi-Tenant MPSoCs," in *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5, May 2021.

[5] Cesar G. Chaves, Siavoosh P. Azad, Thomas Hollstein, and Johanna Sepúlveda, "Diagnosing DoS Attacks in NoC-based MPSoCs," in *Testmethoden und Zuverlässigkeit von Schaltungen und Systemen (TuZ 2019)*, February 2019.

[6] Cesar G. Chaves, Johanna Sepúlveda, and Thomas Hollstein, "Lightweight Packet Monitoring for Flooding DoS Attack Detection in NoC-based MPSoCs," in *Testmethoden und Zuverlässigkeit von Schaltungen und Systemen (TuZ 2021)*, February 2021.

[7] P. M. Mell and T. Grance, "SP 800145. The NIST Definition of Cloud Computing," tech. rep., National Institute of Standards and Technology, Gaithersburg, MD, United States, 2011.

[8] M. Iorga, L. Feldman, R. Barton, M. J. Martin, N. S. Goren, and C. Mahmoudi, "Fog Computing Conceptual Model," tech. rep., National Institute of Standards and Technology, 2018.

[9] G. P. Fettweis, "5G and the Future of IoT," in *ESSCIRC Conference 2016: 42nd European Solid-State Circuits Conference*, pp. 21–24, Sept 2016.

[10] T. Perez and S. Pagliarini, "A Side-Channel Hardware Trojan in 65nm CMOS with $2\mu W$ precision and Multi-bit Leakage Capability," in *2022 27th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 9–10, IEEE, 2022.

[11] C. Reinbrecht, A. Susin, L. Bossuet, and J. Sepúlveda, "Gossip NoC – Avoiding Timing Side-Channel Attacks Through Traffic Management," in *2016 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pp. 601–606, July 2016.

[12] L. Daoud, "Secure Network-on-chip Architectures for MPSoC: Overview and Challenges," in *2018 IEEE 61st International Midwest Symposium on Circuits and Systems (MWSCAS)*, pp. 542–543, IEEE, 2018.

[13] R. JS, D. M. Ancajas, K. Chakraborty, and S. Roy, "Runtime Detection of a Bandwidth Denial Attack from a Rogue Network-on-chip," in *Proceedings of the 9th International Symposium on Networks-on-Chip*, NOCS '15, (New York, NY, USA), Association for Computing Machinery, September 2015.

[14] J. Sepúlveda, "Secure Cryptography Integration: NoC-Based Microarchitectural Attacks and Countermeasures," in *Network-on-Chip Security and Privacy*, pp. 153–179, Springer, 2021.

[15] S. Evain and J. P. Diguet, "From NoC Security Analysis to Design Solutions," in *IEEE Workshop on Signal Processing Systems Design and Implementation, 2005.*, pp. 166–171, Nov 2005.

[16] K. Madden, J. Harkin, L. McDaid, and C. Nugent, "Adding Security to Networks-on-chip Using Neural Networks," in *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 1299–1306, IEEE, November 2018.

[17] T. H. Boraten and A. K. Kodi, "Securing NoCs Against Timing Attacks with Non-interference Based Adaptive Routing," in *2018 Twelfth IEEE/ACM International Symposium on Networks-on-Chip (NOCS)*, pp. 1–8, IEEE, October 2018.

[18] S. Charles, Y. Lyu, and P. Mishra, "Real-time Detection and Localization of DoS Attacks in NoC Based SoCs," in *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1160–1165, IEEE, March 2019.

[19] S. Charles, Y. Lyu, and P. Mishra, "Real-time Detection and Localization of Distributed DoS Attacks in NoC Based SoCs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, February 2020.

[20] M. Sinha, S. Gupta, S. S. Rout, and S. Deb, "Sniffer: A Machine Learning Approach for DoS Attack Localization in NoC-based SoCs," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, May 2021.

[21] Z. Pan, J. Sheldon, C. Sudusinghe, S. Charles, and P. Mishra, "Hardware-Assisted Malware Detection Using Machine Learning," in *Design Automation and Test in Europe (DATE)*, July 2021.

[22] C. Sudusinghe, S. Charles, and P. Mishra, "Denial-of-Service Attack Detection using Machine Learning in Network-on-Chip Architectures," in *Proceedings of the 15th IEEE/ACM International Symposium on Networks-on-Chip*, pp. 35–40, 2021.

[23] J.-P. Diguet, S. Evain, R. Vaslin, G. Gogniat, and E. Juin, "NOC-centric Security of Reconfigurable SoC," in *First International Symposium on Networks-on-Chip (NOCS'07)*, pp. 223–232, IEEE, May 2007.

[24] L. Fiorin, G. Palermo, and C. Silvano, "A Security Monitoring Service for NoCs," in *Proceedings of the 6th IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis*, CODES+ISSS '08, (New York, NY, USA), pp. 197–202, ACM, October 2008.

[25] D. Fang, H. Li, J. Han, and X. Zeng, "Robustness Analysis of Mesh-based Network-on-chip Architecture Under Flooding-based Denial of Service Attacks," in *2013 IEEE Eighth International Conference on Networking, Architecture and Storage*, pp. 178–186, IEEE, July 2013.

[26] M. J. Sepúlveda, J. P. Diguet, M. Strum, and G. Gogniat, "NoC-Based Protection for SoC Time-Driven Attacks," *IEEE Embedded Systems Letters*, vol. 7, pp. 7–10, March 2015.

[27] A. B. Achballah, S. B. Othman, and S. B. Saoud, "Toward on Hardware Firewalling of Networks-on-chip Based Systems," in *2017 International Conference on Advanced Systems and Electric Technologies (IC_ASET)*, pp. 7–13, IEEE, January 2017.

[28] J. Yao, Y. Zhang, Z. Mao, S. Li, M. Ge, and X. Chen, "On-line Detection and Localization of DoS Attacks in NoC," in *2020 IEEE 9th Joint International Information Technology and Artificial Intelligence Conference (ITAIC)*, vol. 9, pp. 173–178, IEEE, February 2020.

[29] J. Jeffers, J. Reinders, and A. Sodani, *Intel Xeon Phi processor high performance programming: knights landing edition*. Morgan Kaufmann, 2016.

[30] S. K. Sadasivam, B. W. Thompto, R. Kalla, and W. J. Starke, "IBM Power9 Processor Architecture," *IEEE Micro*, vol. 37, no. 2, pp. 40–51, 2017.

[31] Marvell, "Marvell® ThunderX2® CN99XX Product Brief." `https://www.marvell.com/content/dam/marvell/en/public-collateral/server-processors/marvell-server-processors-thunderx-cn99xx-product-brief-2020.pdf`, 2020. Accessed: 2022-02-04.

[32] M. Kovač, P. Notton, D. Hofman, and J. Knezović, "How Europe is Preparing its Core Solution for Exascale Machines and a Global, Sovereign, Advanced Computing Platform," *Mathematical and Computational Applications*, vol. 25, no. 3, p. 46, 2020.

[33] EPI Consortium, "European Processor Initiative." `https://www.european-processor-initiative.eu/`, 2018. Accessed: 2022-02-04.

[34] A. K. Singh, T. Srikanthan, A. Kumar, and W. Jigang, "Communication-aware Heuristics for Run-time Task Mapping on NoC-based MPSoC Platforms," *Journal of Systems Architecture*, vol. 56, no. 7, pp. 242–255, 2010.

[35] M. Fattah, M. Ramirez, M. Daneshtalab, P. Liljeberg, and J. Plosila, "CoNA: Dynamic Application Mapping for Congestion Reduction in Many-core Systems," in *2012 IEEE 30th International Conference on Computer Design (ICCD)*, pp. 364–370, IEEE, 2012.

[36] M. Fattah, M. Daneshtalab, P. Liljeberg, and J. Plosila, "Smart Hill Climbing for Agile Dynamic Mapping in Many-core Systems," in *2013 50th ACM/EDAC/IEEE Design Automation Conference (DAC)*, pp. 1–6, IEEE, 2013.

[37] S. P. Azad, B. Niazmand, P. Ellervee, J. Raik, G. Jervan, and T. Hollstein, "$SoCDep^2$: A Framework for Dependable Task Deployment on Many-core Systems Under Mixed-criticality Constraints," in *2016 11th International Symposium on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC)*, pp. 1–6, June 2016.

[38] C.-H. Huang, C.-Y. Chen, and H.-Y. Huang, "Hierarchical and Dependency-aware Task Mapping for NoC-based Systems," in *2018 11th International Workshop on Network on Chip Architectures (NoCArc)*, pp. 1–6, IEEE, 2018.

[39] W. Amin, F. Hussain, S. Anjum, S. Khan, N. K. Baloch, Z. Nain, and S. W. Kim, "Performance Evaluation of Application Mapping Approaches for Network-on-Chip Designs," *IEEE Access*, vol. 8, pp. 63607–63631, 2020.

[40] K. Shibin, S. Devadze, A. Jutman, M. Grabmann, and R. Pricken, "Health Management for Self-Aware SoCs Based on IEEE 1687 Infrastructure," *IEEE Design & Test*, vol. 34, no. 6, pp. 27–35, 2017.

[41] S. P. Azad, B. Niazmand, K. Janson, N. George, A. S. Oyeniran, T. Putkaradze, A. Kaur, J. Raik, G. Jervan, R. Ubar, and T. Hollstein, "From Online Fault Detection to Fault Management in Network-on-Chips: A ground-up Approach," in *2017 IEEE 20th International Symposium on Design and Diagnostics of Electronic Circuits Systems (DDECS)*, pp. 48–53, April 2017.

[42] C. J. Glass and L. M. Ni, "The Turn Model for Adaptive Routing," *ACM SIGARCH Computer Architecture News*, vol. 20, no. 2, pp. 278–287, 1992.

[43] S. P. Azad, B. Niazmand, K. Janson, T. Kogge, J. Raik, G. Jervan, and T. Hollstein, "Comprehensive Performance and Robustness Analysis of 2D Turn Models for Network-on-chips," in *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–4, May 2017.

[44] J. Flich and J. Duato, "Logic-based Distributed Routing for NoCs," *IEEE computer architecture letters*, vol. 7, no. 1, pp. 13–16, 2008.

[45] "Bonfire Wiki." `https://github.com/Project-Bonfire/Bonfire/wiki`, 2016. Accessed: 2021-11-25.

[46] "Secure Bonfire Wiki." `https://github.com/Project-Bonfire/Secure_Bonfire/wiki`, 2016. Accessed: 2021-11-25.

[47] M. L. Pinedo, *Scheduling: Theory, Algorithms, and Systems*, vol. 29. Springer, 2012.

[48] R. Pop and S. Kumar, "A Survey of Techniques for Mapping and Scheduling Applications to Network on Chip Systems," *School of Engineering, Jonkoping University, Research Report*, vol. 4, no. 4, 2004.

[49] A. K. Singh, M. Shafique, A. Kumar, and J. Henkel, "Mapping on Multi/Many-core Systems: Survey of Current and Emerging Trends," in *2013 50th ACM/EDAC/IEEE Design Automation Conference (DAC)*, pp. 1–10, IEEE, 2013.

[50] P. K. Sahu and S. Chattopadhyay, "A Survey on Application Mapping Strategies for Network-on-chip Design," *Journal of systems architecture*, vol. 59, no. 1, pp. 60–76, 2013.

[51] C.-L. Chou and R. Marculescu, "User-aware Dynamic Task Allocation in Networks-on-chip," in *2008 Design, Automation and Test in Europe*, pp. 1232–1237, IEEE, 2008.

[52] C.-L. Chou, U. Y. Ogras, and R. Marculescu, "Energy- and Performance-aware Incremental Mapping for Networks on Chip with Multiple Voltage Levels," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 10, pp. 1866–1879, 2008.

[53] U. Y. Ogras and R. Marculescu, *Modeling, analysis and optimization of network-on-chip communication architectures*, vol. 184. Springer Science & Business Media, 2013.

[54] Y.-K. Kwok and I. Ahmad, "Static Scheduling Algorithms for Allocating Directed Task Graphs to Multiprocessors," *ACM Computing Surveys (CSUR)*, vol. 31, pp. 406–471, Dec 1999.

[55] I. Foster, "Designing and Building Parallel Programs," 1995.

[56] T. A. Lopes Genez, R. Sakellariou, L. F. Bittencourt, E. R. Mauro Madeira, and T. Braun, "Scheduling Scientific Workflows on Clouds Using a Task Duplication Approach," in *2018 IEEE/ACM 11th International Conference on Utility and Cloud Computing (UCC)*, pp. 83–92, IEEE, 2018.

[57] Z. Ahmad, A. I. Jehangiri, M. A. Ala'anzy, M. Othman, R. Latip, S. K. U. Zaman, and A. I. Umar, "Scientific workflows management and scheduling in cloud computing: taxonomy, prospects, and challenges," *IEEE Access*, vol. 9, pp. 53491–53508, 2021.

[58] M. Hosseini Shirvani and R. Noorian Talouki, "Bi-objective Scheduling Algorithm for Scientific Workflows on Cloud Computing Platform with Makespan and Monetary Cost Minimization Approach," *Complex & Intelligent Systems*, pp. 1–30, 2021.

[59] S. Bharathi, A. Chervenak, E. Deelman, G. Mehta, M.-H. Su, and K. Vahi, "Characterization of Scientific Workflows," in *2008 third workshop on workflows in support of large-scale science*, pp. 1–10, IEEE, 2008.

[60] Q. Tian, J. Li, D. Xue, W. Wu, J. Wang, L. Chen, and J. Wang, "A Hybrid Task Scheduling Algorithm Based on Task Clustering," *Mobile Networks and Applications*, vol. 25, no. 4, pp. 1518–1527, 2020.

[61] M. Masdari and M. Jalali, "A Survey and Taxonomy of DoS Attacks in Cloud Computing," *Security and Communication Networks*, vol. 9, no. 16, pp. 3724–3751, 2016.

[62] T. Boraten and A. K. Kodi, "Mitigation of Denial of Service Attack with Hardware Trojans in NoC Architectures," in *2016 IEEE international parallel and distributed processing symposium (IPDPS)*, pp. 1091–1100, IEEE, May 2016.

[63] S. Charles and P. Mishra, "A Survey of Network-on-Chip Security Attacks and Countermeasures," *ACM Comput. Surv.*, vol. 54, may 2021.

[64] S. Ramanauskaite and A. Cenys, "Taxonomy of DoS Attacks and Their Countermeasures," *Central European Journal of Computer Science*, vol. 1, no. 3, pp. 355–366, 2011.

[65] T. Peng, C. Leckie, and K. Ramamohanarao, "Survey of Network-based Defense Mechanisms Countering the DoS and DDoS Problems," *ACM Computing Surveys (CSUR)*, vol. 39, no. 1, pp. 3–es, 2007.

[66] L. Fiorin, C. Silvano, and M. Sami, "Security Aspects in Networks-on-chips: Overview and Proposals for Secure Implementations," in *Digital System Design Architectures, Methods and Tools, 2007. DSD 2007. 10th Euromicro Conference on*, pp. 539–542, IEEE, August 2007.

[67] R. O. Mueller and G. R. Hancock, *Structural Equation Modeling*. Routledge/Taylor & Francis Group, 2019.

[68] IBM, "IBM SPSS Amos." `https://www.ibm.com/us-en/marketplace/structural-equation-modeling-sem`. Accessed: 2022-01-31.

[69] D. Charântola, A. C. Mestre, R. Zane, and L. F. Bittencourt, "Component-based Scheduling for Fog Computing," in *Proceedings of the 12th IEEE/ACM International Conference on Utility and Cloud Computing Companion*, pp. 3–8, 2019.

[70] J. Sepúlveda, D. Aboul-Hassan, G. Sigl, B. Becker, and M. Sauer, "Towards the formal verification of security properties of a network-on-chip router," in *2018 IEEE 23rd European Test Symposium (ETS)*, pp. 1–6, IEEE, 2018.

[71] "IEEE Xplore Digital Library." https://ieeexplore.ieee.org/. Accessed: 2022-01-22.

[72] "ACM Digital Library." https://dl.acm.org/. Accessed: 2022-01-22.

[73] "Springer Link." https://link.springer.com/. Accessed: 2022-01-22.

[74] "Science Direct." https://www.sciencedirect.com/. Accessed: 2022-01-22.

[75] J. Sepúlveda, M. Strum, and W. Chau, "An Hybrid Switching Approach for NoC-Based Systems to Avoid Denial-of-Service SoC Attacks," *16th Iberchip Wksp (IWS 2010)*, pp. 23–25, 2010.

[76] 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*, vol. 34, pp. 1344–1357, June 2015.

[77] Mentor, "ModelSim ASIC and FPGA Design." https://www.mentor.com/products/fv/modelsim/, 2017. Accessed: 2018-09-26.

[78] J. Duato, S. Yalamanchili, and N. Lionel, *Interconnection Networks: An Engineering Approach*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2002.

[79] A. M. Shafiee, M. Montazeri, and M. Nikdast, "An innovational intermittent algorithm in networks-on-chip (noc)," *International Journal of Computer, Electrical, Automation, Control and Information Engineering*, vol. 2, no. 9, pp. 2907 – 2909, 2008.

[80] C. J. Glass and L. M. Ni, "The turn model for adaptive routing," in *Computer Architecture, 1992. Proceedings., The 19th Annual International Symposium on*, pp. 278–287, 1992.

[81] S. Mubeen and S. Kumar, "Designing efficient source routing for mesh topology network on chip platforms," in *Digital System Design: Architectures, Methods and Tools (DSD), 2010 13th Euromicro Conference on*, pp. 181–188, Sept 2010.

[82] Y. Miura, K. Shimozono, S. Watanabe, and K. Matoyama, "An Adaptive Routing of the 2-D Torus Network Based on Turn Model," in *Computing and Networking (CANDAR), 2013 First International Symposium on*, pp. 587–591, Dec 2013.

[83] "Genus Synthesis Solution." https://www.cadence.com/en_US/home/tools/digital-design-and-signoff/synthesis/genus-synthesis-solution.html, 2020. Accessed: 2020-11-07.

[84] C. S. Yeo, M. D. de Assunção, J. Yu, A. Sulistio, S. Venugopal, M. Placek, and R. Buyya, "Utility Computing and Global Grids," *CoRR*, vol. abs/cs/0605056, 2006.

[85] A. V. Dastjerdi and R. Buyya, "An Autonomous Time-dependent SLA Negotiation Strategy for Cloud Computing," *The Computer Journal*, vol. 58, pp. 3202–3216, 07 2015.

## Acknowledgements

I would like to begin by thanking my supervisors Prof. Dr.-Ing. Thomas Hollstein and Dr. Johanna Sepúlveda for giving me the opportunity and guidance to carry out Ph.D. studies, including numerous meetings and discussions, without them I could not have undertaken this journey.

I would also like to express my deepest gratitude to the German Academic Exchange Service (Deutscher Akademischer Austauschdienst - DAAD) for the support given through their very well-structured exchange program. Special thanks to Maria José Salgado for providing such reliable support and calming advice that allowed me to maintain focus on the Ph.D. regardless of the circumstances.

I am also thankful to Tallinn University of Technology (TalTech) for providing a high-performance server infrastructure and licenses of specialized software, resources used for the implementation, and extensive simulations while validating the presented proposals. I am extremely grateful to Dr. Siavoosh Payandeh Azad for the invaluable support which helped put me on track at the beginning of the Ph.D. Many thanks to Prof. Samuel Pagliarini and Dr. Tiago Diadami Perez for the nice professional and personal conversations, as well as to Dr. Karl Janson for the assistance while preparing Systems-on-Chip Design labs, for teaching me some words in Estonian, and for translating the abstract of this document. I am also grateful to Katri Kadakas and Katrin Tõemets for their kindness and prompt replies when seeking help.

I would also like to thank the Frankfurt University of Applied Sciences (Fra-UAS) for providing an excellent office, the place from where most of the research was done. Special thanks to Zühal Demirezen and Marianne Jäger for their kindness and support, as well as to Dr. Dominik Wolf for facilitating a cooperative Ph.D. between TalTech and Fra-UAS. I also wish to thank Prof. Dr.-Ing. Carolina Tranchita and Prof. Dr.-Ing. Heiko Hinkelmann for the nice conversations during my time at Fra-UAS.

I am also grateful to the friends made during the Ph.D. period and to all of those who kept in touch for making this time much more enjoyable and keeping me sane throughout the whole process.

Last but not least, I would like to thank my family who over the years has been my inspiration and safe harbor due to their encouragement and unfailing support. To them, I dedicate all my effort and with them, I share the satisfaction of accomplishing this endeavor.

# Abstract
# Bandwidth Reduction DoS Attacks in Multi-Tenant NoC-based MPSoCs: Detection and Avoidance Strategies

Security is one of the main concerns when developing systems that can be accessible to many users. Such matter needs to be revisited when combining systems that independently had been considered secure so that the combination is also secure. Moreover, Denial of Service (DoS) attacks are one of the intrinsic vulnerabilities of multi-tenant systems such as Cloud computing since one malicious user can attempt to affect the quality of service experienced by others. Such attacks do not only disrupt the execution time of legitimate applications by extending them but may also force the service providers to pay fines for not being able to honor established service-level agreements.

Furthermore, with the constant performance increase of Multi-processor Systems on Chip (MPSoCs) and their ability to enhance the performance of execution environments, efforts towards tackling DoS attacks in such systems have also seen an increase during the last few years. This thesis summarizes all the concepts required for understanding the addressed topic and its context along with a detailed representation of the targeted threat model. The contribution of this thesis is composed of the answers given to three research questions, namely RQ1, RQ2, and RQ3.

Research question RQ1 focuses on the identification of DoS attacks that can be implemented by malicious users targeting MPSoCs of a multi-tenant execution environment. This thesis details the characteristics of the two types of attacks found in the literature as well as those of two other attacks that have gone unnoticed by other authors.

Research question RQ2 addresses directives that can be adopted in the system for avoiding, detecting, localizing, and/or mitigating the identified DoS attacks. A summary of the approaches introduced by other authors, including their limitations, is presented. Additionally, three complementing and one independent detection mechanism were proposed, to overcome such limitations. Moreover, two DoS attack avoidance approaches were also proposed as well as a system-level description of how service providers can guarantee that established service-level agreements will always be honored while penalizing the malicious users.

Research question RQ3 looks for a way to assess the impact of DoS attacks on any NoC-based MPSoC design. For the first time, an algorithm capable of generating the four identified DoS attacks is presented. Additionally, a complete assessment model is also proposed.

After the development of the project presented in this thesis, it was concluded that despite several authors targeting DoS attacks in NoC-based MPSoCs, there are still some DoS attacks to be studied. Additionally, a trend toward the use of machine learning for DoS attack detection was identified. However, such approaches are not suitable for dynamic multi-tenant execution environments. Finally, it is explained that DoS attack detection becomes irrelevant if service level agreements are enforced, while attempts to exceed established limits generate an overcharge following a pay-per-use model.

# Kokkuvõte

## Ribalaiuse vähendamise DoS-rünnakud mitme rentnikuga NoC-põhiste MPSoC-de puhul: tuvastamise ja vältimise strateegiad

Mitme kasutajaga süsteemide arendamisel on üks peamisi probleeme turvalisus. Eriti oluline on tuvalisust silmas pidada olemasolevate komponentide integreerimisel, sest nende kombineerimisel loodav süsteem ei pruugi olla turvaline, isegi kui komponendid ise seda on. Lisaks on mitme rentnikuga süsteemid (multi-tenant systems), nagu näiteks pilvepõhiseid teenuseid pakkuvad süsteemid, haavatavad teenusetõkesusrünnakute (DoS-rünnakute) poolt, sest üks pahatahtlik kasutaja saab mõjutada teenuse kvaliteeti paljude teiste kasutajate jaoks. Sellised rünnakud mitte ainult ei häiri süsteemi toimimist ja ei pikenda aega, mis kulub legitiimsete rakenduste jooksutamiseks, vaid võivad põhjustada teenusepakkujale ka reaalset rahalist kahju. Seda näiteks seepärast, et rünnakute tõttu ei suuda teenusepakkuja tagada lepingujärgseid kokkuleppeid teenuse kvaliteedi osas.

Kuna mitmeprotsessoriliste kiipsüsteemide jõudlus on pidevalt kasvanud, on viimastel aastatel samuti suurenenud DoS-rünnakute arv selliste süsteemide vastu. See doktoritöö annab ülevaate kõigist kontseptsioonidest, mis on vajalikud selle teema ja konteksti mõistmiseks. Samuti antakse detailne ülevaade töös kasutatud ohumudelist. Selle väitekirja panus tugineb vastustest kolmele uurimisküsimusele, mida tähistatakse lühenditega RQ1, RQ2 ja RQ3.

Uurimisküsimus RQ1 keskendub mitme rentnikuga MPSoC-des pahatahtlike kasutajate poolt sooritatud DoS rünnakute tuvastamisele. Selles väitekirjas kirjeldatakse detailselt nii kaht eelnevalt teiste poolt erialases kirjanduses avaldatud rünnakutüüpi, kui ka kaht täiesti uut rünnakut, mis on teistel autoritel märkamata jäänud.

Uurimisküsimus RQ2 käsitleb direktiive, mida saab süsteemis DoS rünnakute vältimiseks, tuvastamiseks, lokaliseerimiseks ja / või leevendamiseks vastu võtta. Esitatakse kokkuvõte teiste autorite poolt välja pakutud lahendustest ning samuti nende puudustest. Samuti pakutakse välja kolm rünnakute tuvastamise mehhanismi, mis täiendavaid olemasolevaid meetodeid ning lisaks üks täiesti iseseisev DoS rünnakute tuvastamise mehhanism. Lisaks pakutakse välja kaks lahendust DoS rünnakute ärahoidmiseks. Samuti käiakse välja süsteemitaseme kirjeldus selle kohta kuidas teenusepakkujad saaks teenustasemeleppeid täita ka rünnakute korral, samas pahatahtlike kasutajaid karistades.

Uurimisküsimus RQ3 otsib viise, mis võimaldaks hinnata DoS rünnakute mõju igasuguse arhitektuuriga kiipvõrgupõhistel MPSoC-idel. Ühtlasi kirjeldab selles väitekirjas esimsest korda algoritmi, mis suudab genereerida nelja identifitseeritud DoS rünnakut. Samuti tuuakse välja täielik hindamismudel rünnakute analüüsimiseks.

Pärast käesolevas doktoritöös esitletud projekti arendamist jõuti järeldusele, et vaatamata sellele, et DoS rünnakuid kiipvõrgupõhistes MPSoC-des on palju uuritud, eksisteerib endiselt ka uudseid, läbiuurimata, DoS rünnakuid. Ühtlasi tuvastati trend masinõppe kasutamiseks DoS rünnakute avastamisel. Selliselt lahendused ei ole aga siiski kasutatavad mitme rentnikuga süsteemides. Lõpetuseks selgitatakse, et DoS-rünnak muutub ebaeffektiivseks kui jõustatakse teenusetaseme lepinguid, sest seatud kasutuslimiitide ületamine DoS rünnakute käigus tekitab pahataktlikele kasutajatele lisatasu.

# Appendix 1

**I**

Cesar G. Chaves, Siavoosh P. Azad, Thomas Hollstein, and Johanna Sepúlveda, "A Distributed DoS Detection Scheme for NoC-based MPSoCs," in *Circuits and Systems (NorCAS), 2018 IEEE Nordic Conference on*, IEEE, October 2018

# A Distributed DoS Detection Scheme for NoC-based MPSoCs

Cesar G. Chaves*†, Siavoosh Payandeh Azad†, Thomas Hollstein*†, Johanna Sepúlveda‡
*Frankfurt University of Applied Sciences, Frankfurt, Germany
†Department of computer systems, Tallinn University of Technology, Tallinn, Estonia
‡Technical University of Munich, Munich, Germany
email: cesar.chaves@stud.fra-uas.de, siavoosh.azad@ttu.ee, hollstein@fb2.fra-uas.de, johanna.sepulveda@tum.de

***Abstract*—The increasing threat of Denial of Service (DoS) attacks targeting Networks-on-Chip (NoC) based Multi-processors System-on-Chip (MPSoCs) imposes unprecedented challenges in terms of communication availability, especially for identifying the source of an attack in the NoC. Previous works show the possibility of executing DoS attacks on NoCs and propose mitigation methods that are deployed in all the NoC routers. However, this protection countermeasures usually impact the communication heavily. In order to achieve a better DoS protection, the point of attack should be identified. Towards this direction, in this paper, we propose a novel distributed DoS detection scheme being able to measure the performance degradation of sensitive flows and to detect the router where the attack enters the sensitive communication path. We perform an exploration regarding the effect of different types of attackers, where experimental work shows the best attack configuration parameters, and that a combination of two latency metrics can be leveraged not only for detecting a DoS attack, but also its interference point.**

***Index Terms*—NoC, MPSoC, DoS attack, Distributed monitoring.**

## I. Introduction

The comprehensive use of Internet-of-Things (IoT) will be the driver of digitization in all domains, e.g. industry automation, automotive, avionics, and health-care. IoT is a powerful technology that captured the attention of researchers, industry and governments. Since by distributing computation and increasing the hyper-connectivity of devices through machine-to-machine communication, tasks can be deployed into a sea of devices, reducing their execution time and enabling smart behaviors. Three characteristics turn IoT relevant: i) Number of connected devices: being estimated that by 2021 an amount of 28 billion of devices will be part of the IoT [1]; ii) Economical impact: by introducing automated and smart processes, the IoT is foreseen to achieve the mark of 11 trillion US Dollars by 2025 [2]; and iii) Integration into a broad spectrum of markets, including different industry and service markets. Increasingly complex and powerful Systems-on-Chips (SoCs) connected through a 5G network, form the basis of the IoT, where Multi-Processors System-on-Chip (MPSoCs) are considered the key enabler technology for IoT. They are composed of two main structures: 1) the computation structure, composed by Processing Elements (PEs) such as: processors, hardware accelerators, memories, peripherals and other Intellectual Property (IP) hardware cores to process and store information, and 2) the communication structure, which perform data exchange among the IP cores. Network-on-Chips (NoCs) are the communication structure choice for MPSoCs that integrate a large amount of IP cores. NoCs integrate routers and links to exchange information wrapped as packets.

978-1-5386-7656-1/18/$31.00 ©2018 IEEE

A major requirement for all IoT domains is the security. The hyper-connectivity of devices also represents a risk. MPSoCs are not isolated anymore, being able to download pieces of software code used for executing tasks and updating firmware. However, this software may be malicious. Currently MPSoCs are now target of attacks whose goal is to retrieve the secret information (secret keys and intellectual property), modify the system operation (sabotage) or deny the system operation.

Denial of Service (DoS) attacks target to degrade the MPSoC operation by turning unavailable or obstructing the efficient computation or communication services. The goal of the attacker is to avoid the completion of a sensitive task (e.g. data encryption, trigger of an alarm), so that users decide to turn off the encryption or to migrate tasks to an alternative infected neighbor device (when tasks are deployed in the IoT infrastructure).

The shared and main role of the NoC in the MPSoC operation has turned NoCs into target of DoS. This kind of attacks are typically executed through infected IP cores, which execute malicious software. The goal of the attacker is to inject traffic such that it degrades the sensitive communication, thus avoiding that sensitive communication flows inside the MPSoC meet the communication time-line constraints. There are three types of DoS attacks on NoCs: i) volumetric, whose goal is to cause congestion into a single point to overwhelm the NoC bandwidth; ii) state-exhaustion, which attempts to use up all the NoC available connections; and iii) connection-based, which exploits the priority-based NoCs to monopolize the NoC communication. All these attacks result in an increase of the latency of the sensitive flows. Previous works show the possibility of executing NoC DoS attacks and propose new router architectures to mitigate these attacks. However, usually this protection countermeasures impact heavily the communication. In order to allow a dynamic and customized DoS protection, the attack point should be identified.

In this work we provide a novel lightweight path-based method for the detection and localization of DoS attacks in NoCs. The presented approach provides

- a NoC architecture able to detect DoS attacks, via end-

to-end latency calculation for packets
- identification of attacker location on the MPSoC, using per-hop latency calculation
- an environment for the exploration of attack scenarios using different attack patterns, attacker's packet size, and different network traffic scenarios

To the best of our knowledge, we are presenting the first mechanism to identify the location of the attacker on the MPSoC when a DoS is performed.

This paper is organized as follows: Section II gives an overview on related work in the field of DoS avoidance and detection in NoC-based platforms. After providing a threat model in Section III, in Section IV a minimalist NoC router and NoC packet structure modification is presented, providing necessary information for the identification of attack points. Section V describe extensive simulation-based experiments with different traffic and packet size scenarios, which approve the feasibility of the concept. In the final conclusions, the presented approach is summarized and open points for future work are outlined.

## II. Related Works

Denial-of-Service (DoS) has been widely explored in the context of extra-chip networks. However, DoS attacks in NoCs have just been recently addressed. The works of [3], [4] introduce for the first time the concept of DoS in the NoC context. Mitigation strategies for NoC DoS attacks have been proposed in [3]–[14]. Moreover, these works can be categorized in two main classes: **DoS Avoidance** and **DoS detection and recovery**. The first focuses on providing infrastructure to avoid such attacks all together (see section II-A). On the other hand, the second tries to identify the occurrence of such attacks and disabling them (see section II-B). The latter can be divided into two sub groups: works which focus on techniques for detecting the occurrence of the attack, but can not detect the location of the node which runs malicious code, and works that can also identify the location of the attacker.

### A. DoS Attack Avoidance

In [4] a mitigation strategy based on the hybrid switching routing mechanism is proposed. Circuit switching is used for sensitive traffic and packet switching is used otherwise. As a result, predictable latency for sensitive traffic is guaranteed. The works of [5], [6], and [7] mitigate the DoS and timing attacks through the generation of security zones. Their proposal ensures that only the secure nodes can communicate into a virtual and physical space, respectively. Security zones are built through dynamic routing. In [8] non-sensitive traffic is deeply inspected in order to access to security zones. The works of [6] and [7] reroute the traffic outside from the security zone. On the other hand, the authors of [9] propose the use of separate virtual channels for secure and non-secure packets as a counter measure for bandwidth denial attacks in NoCs. However, this approach suffers from the hardware overhead of additional buffers used for virtual channels.

### B. DoS Attack Detection and Recovery

In [10], authors consider that DoS attacks are executed by a rogue third-party NoC architecture. When a suspicious communication delay is detected between two nodes, the firmware tries to identify if the attack is taking place at the source or at the destination, this by time-stamping the packets and dividing into two communication paths.

In [11], Fiorin et al. provide an overview of the attacks in network on chips. The authors propose the use of buffer occupancy monitors to detect traffic anomaly in the network but provide no details on the technique. Later, in [3], Fiorin et al. proposed the implementation of denial of service probes in the Network Interface. This approach only covers the DoS attacks from the Processing Elements connected to the network interface by detecting deviations from the average bandwidth usage expected at the design time. This approach does not actively monitor the latency of the sensitive packets in the network.

In [12], Diguet et al. tackle DoS attacks by monitoring live-lock occurrence in the network. However, this approach utilizes source-routing, which imposes considerable overhead to the packet size.

In [13], Grammatikakis et al. target distributed DoS attacks via a firewall in control of configurable access rules in the network interface. The security risk is evaluated by product of frequency and magnitude of losses (by dropping the packets at the Network Interface). Although such firewall does not allow the DoS attack to be effective, it was designed to protect the destination PE (i.e. the on-chip memory), thus it does not detect the source of the attach, nor does it kill the attack related traffic congesting the network.

In [14], Achballah et al. target computation of occupation time of the physical links in number of clock cycles. The occupation time of the physical link is compared to an expected value. In case of the link occupation exceeding this value, the transaction would be stopped and a notification would be sent to system's manager. Here the decision is made locally in the router, which may lead to undesirable packet dropping.

The evaluation of the related works show that the DoS attack avoidance approaches require significant additional hardware effort in the routers and the system. Since scalable NoCs need an overall dependability management ( [15]), the router size should be as minimalist as possible (e.g. no virtual channels). The works focusing on detection and recovery, cannot be effective without identification of location of the attacker. The works that can successfully detect and identify the location of the attacker, do not provide lightweight routers.

In this work we propose a scalable distributed solution for monitoring the traffic over the network, having minimal hardware overhead and allowing to transport router-internal DoS attack relevant information to the endpoint of the communication. This enables the detection and localization of DoS attacks on sensitive routing paths.

## III. Threat Model

MPSoCs with a heterogeneous array of PEs provide programmability and parallelism, yielding flexibility, processing

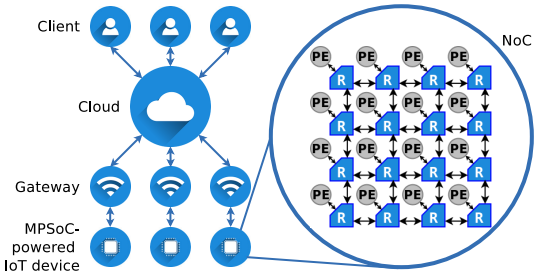



Fig. 1: MPSoC-powered IoT Execution Environment

performance and power efficiency, which can be leveraged for minimizing the latency of communication links of the edge cloud [16]. An example of such scenario is depicted in Fig. 1, where IoT devices, equipped with NoC-based MPSoCs, improve the performance of general-purpose computers-exclusive execution environments by adding application specific circuits. However, this junction brings new security vulnerabilities to the NoC-based MPSoC world, such as susceptibility to DoS attacks, as will be explained in this section.

*Attack Scenario:* During a DoS attack, an attacker seeks to degrade the performance of a processing/communication environment, which in the scope of this work is represented by a NoC-based MPSoCs. In an environment, such as the one illustrated in Fig. 1, a task scheduler in the firmware of the IoT device maps user applications to the PEs of the incorporated MPSoC. However, one of those users can be an attacker with the intention of performing a DoS attack to the NoC. Moreover, the most common DoS attack method is network flooding, which a malicious application running on any PE of the MPSoC is able to do by sending packets to another PE in the NoC. These additional packets compete for crossbars output resources of the routers in their path, and while successful, they increment the forwarding delays of other packets that share at least one segment of the malicious traffic path, resulting in an overall additional communication delay that will be unacceptable for time sensitive applications.

*Attack Steps:* In order to perform a DoS attack in a NoC-based MPSoC, the following two steps have to be carried out:

1) Infect an MPSoC processor; this can be done by any attacker with normal user privileges.
2) Take possession of the crossbar of some NoC routers; This can be done by continuously sending packets to other destinations within the NoC.

*Attack Success Conditions:* In order to impact a NoC-based MPSoC with a DoS attack, the following conditions must be met:

- The malicious packets compete for the use of the crossbar of a router with the sensitive packets (a.k.a attempt to be forward through a same output port); Assuming that the attacker has no knowledge of the network segments being used by the sensitive traffic, setting long paths for the malicious traffic will increase the probability of colliding with the sensitive traffic.
- The attacker can decide the insertion rate of the malicious packets into the NoC; By modifying its insertion rate and monitoring decreases in its own throughput, an attacker can assure that his packets are colliding with other traffic [17]. Additionally, depending on the router arbiter implementation, a burst of packets can secure a crossbar grant, causing the starvation of other packets.
- The attacker can decide the length of the malicious packets; The usage grant of the crossbars of some routers is done in a per packets basis, thus, longer packets occupy the crossbar longer and consequently generate a longer inter-router forwarding delay to sensitive packets.

This work targets the analysis of flooding DoS attacks from the attacker point of view, and also propose a detection mechanism that does not only detect the flooding DoS attacks, but that also identifies the collision point of malicious packets with the secure traffic. Such information can subsequently be used by the system resource and deployment manager in order to stop the execution of the malicious code.

## IV. Secure Architecture

In order to achieve a scalable NoC Architecture that is not only able to detect a DoS attack, but also its source, we propose three small modifications to the Bonfire NoC project [15]. The first two of these modifications are regarding the structure of the communication packets and the third relates to the architecture of the routers.

As shown in Fig. 2, one of the modifications to the structure of the communication packets refers to the definition of the *Last Body Flit*, which will carry a time-stamp of the packet's generation time. The time-stamp is introduced to enable the

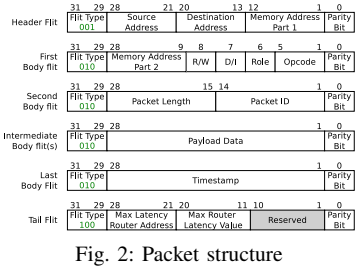



Fig. 2: Packet structure

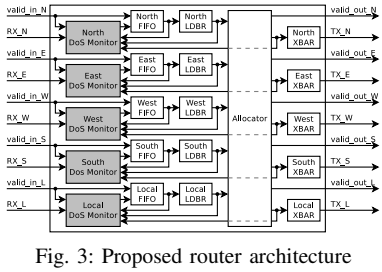



Fig. 3: Proposed router architecture

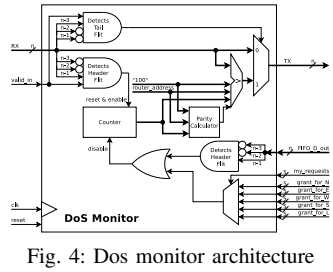
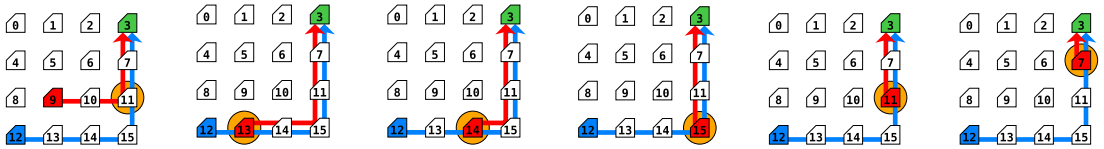



Fig. 4: Dos monitor architecture



(a) Source 9 (b) Source 13 (c) Source 14 (d) Source 15 (e) Source 11 (f) Source 7

Fig. 5: Experiment Scenarios

system to calculate end-to-end latency of each packet, and based on it, detect if a DoS attack is taking place. The second modification is related to the *Tail Flit*, which now carries the address of the router where the packet waited while its required output port was being used by other packets (i.e. *Max Latency Router Address*), and the amount of clock cycles it waited (i.e. *Max Router Latency Value*). This information is evaluated on every router by a proposed DoS Monitor, and updated in case the waiting time in the current router is longer than the one stored in the packet.

As said before, and shown in Fig. 3, the third modification is to the router's architecture, where the proposed *DoS Monitor* is added to the beginning of the router's data path (i.e. before each of the five FIFOs). Such a monitor enables the NoC to distributively accuse the router where the DoS attempting traffic enters the sensitive communication path. The internal architecture of the proposed *DoS Monitor* is depicted in Fig. 4. This monitor starts a 10-bit counter every time a new Header flit arrives to the input of the FIFO, and stops it when the Header flit leaves the FIFO's output. The counter is incremented on each clock cycle while other packets are being forwarded through its required output port. This is done by monitoring the output grants provided by the router's Allocator circuit filtered by the request sent by the LDBR circuit. Furthermore, once the tail flit of the packet arrives, the counter value is compared against the *Max Router Latency Value* stored in the tail flit. If the counter value is smaller or equal, the tail flit will be forwarded as it is, otherwise, the monitor replaces the stored value with the value of the counter. Additionally, the *Max Latency Router Address* is updated with the current router address, and a new parity for the tail flit is calculated and updated.

Once packets reach their destinations, the firmware of the MPSoC-powered IoT device can calculate their end-to-end transmission latency based on the packet's time-stamp. Subsequently that value is compared to that of a maximum expected, and if it is out of the acceptable limit, the *Max Latency Router Address* and the *Max Router Latency Value* are also retrieved from the packet and analyzed together with previous DoS suspicion reports. Finally, when decided that a DoS attack is taking place, the firmware can reset the PE identified as the sourced attack, thus eliminating its malicious traffic generation.

## V. Experimental Work

The DoS Monitor, presented in Section IV, has been implemented on RTL level and integrated to the Bonfire framework [15]; subsequently, the proposed NoC architecture with the distributed DoS detection system has been simulated using *Modelsim* from *Mentor Graphics* [18]. The Bonfire framework provides an environment for dependability research in NoC-based Systems-on-Chip (SoC), hence it is not only modular and easy to scale, but also integrates traffic generators, which can be leveraged for simulating normal traffic and DoS attacks. The considered NoC scenarios are detailed in Subsection V-A, the simulation results are presented and discussed in Subsection V-B, and an evaluation of the overhead of the proposed mechanism in terms of area, critical-path delay, and power is presented in Subsection V-C.

### A. Simulation Scenarios

In order to evaluate the performance of the proposed Distributed DoS Detection system, simulations have been executed on a $4 \times 4$ mesh NoC-based MPSoC architecture, as the one depicted in each of the scenarios in Fig. 5. The network routers (applying an xy-routing algorithm) use a credit base flow control with fair round robin switch matrix output arbitration (on packet level) and utilize wormhole switching with 4-flit deep FIFOs.

The numbered elements in Fig. 5 represent the routers of the NoC. Router 12 (blue) is the source of the sensitive traffic and router 3 (green) its destination; furthermore the blue arrow, connecting these two routers, represents the sensitive path. Additionally, each scenario considers a different attack source (router connected to the infected PE), symbolized with the red router (specified in the label of each sub-figure). The malicious traffic runs from the scenario specific source to the same destination as the sensitive traffic (router 3); the arrow connecting these two routers represents such a path and a circle indicates the collision point, where DoS traffic is being inserted into the path of the sensitive traffic. Finally, as depicted in Fig. 6, depending on the selected attack source, the malicious path length and the collision point to the sensitive path vary, as being presented subsequently in this section. In the cases where the attack source is directly in the sensitive path, the source and collision point routers are the same.

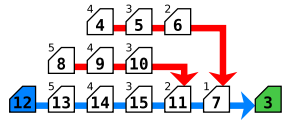


Fig. 6: Attack path lengths and collision points

### B. Results

The experiments presented in this section target the analysis of the DoS attack from two sides: 1) from the point of view of the attacker in order to maximize the effectiveness of the attack, and 2) from the point of view of network protection, testing metrics that can be leveraged for detecting an attack and qualifying the effectiveness of the detection mechanism. Within this analysis, simulations were done considering different packet injection rates (PIR) for sensitive packets (PIR_S), attack packets (PIR_A), and random traffic (PIR_R) (simulation data generated for 20 pseudo-random simulation start seeds). Also, a packet length of 10 flits was adopted for the sensitive (PL_S) and random traffic (PL_R) while in the scenarios where an attack attempt was present, three different packet lengths were considered for the attacker's packets (PL_A) (i.e. 10, 30, and 50).

Considering router 15 as the attack source (S_A)(scenario *d* of Fig. 5), packet lengths for attacker, secure, and random traffic of 30, 10, and 10, respectively, and a random traffic PIR of 0.01 (one packet every 100 clock cycles), Table I details results from the attacker's point of view, while Table III from the attack detection point of view, both for different PIR configurations for the sensitive and attacking traffic.

As mentioned in Section III, an attacker can identify the success/effectiveness of the attack by measuring its own throughput [17]. As it can be seen in the first and third column of Table I, as the intended PIR value is incremented, the gap with the Attacker's Effective PIR also increases. Where the 0.01 attacker's PIR_A value showed to be over the limit of the network's capacity for the adopted configuration, since trying to send one 30-flit-length packet every 100 clock cycles resulted in an insertion rate of one packet every 140 clock cycles, approximately. Such difference, expressed as a growth percentage related to the intended PIR, is presented in the fourth column. For the purpose of this work, we considered an attacker's PIR_A variation of 10% to be considered as an actual attack.

TABLE I: Attack assurance (S_A: 15, PL_A: 30, PL_S: 10, PL_R: 10, PIR_R: 0.01)

| Attacker PIR_A | Sensitive PIR_S | Attacker Effective PIR_A | Attacker PIR_A deviation (%) | Attack effective |
|---|---|---|---|---|
| 0.003 | 0.003 | 0.003 | 0 | False |
| 0.003 | 0.01 | 0.003 | 0 | False |
| 0.003 | 0.03 | 0.003 | 0 | False |
| 0.01 | 0.003 | 0.0071 | 29 | True |
| 0.01 | 0.01 | 0.0071 | 29 | True |
| 0.01 | 0.03 | 0.0070 | 30 | True |
| 0.03 | 0.003 | 0.0086 | 71.33 | True |
| 0.03 | 0.01 | 0.0085 | 71.67 | True |
| 0.03 | 0.03 | 0.0086 | 71.33 | True |

As mentioned in Section IV, each packet carries a time-stamp generated when the packet entered the NoC. Subsequently, that time-stamp is used for calculating the End-to-End communication delay upon arrival. Table II lists the End-to-End delay values of the sensitive packets for different PIR_S values in a scenario with no attackers. This mean value and the sample standard deviation (SSD) are used for detecting the DoS attacks setting a threshold $T$ calculated by $T = mean\_delay + 0.5 \times SSD$.

TABLE II: End-to-End delay under no attack (PL_S: 10, PL_R: 10, PIR_R: 0.01)

| Sensitive PIR_S | Mean End-to-End delay (clock cycles) | Sample Standard deviation (clock cycles) | Detection Threshold (clock cycles) |
|---|---|---|---|
| 0.003 | 88.9056 | 17.3167 | 97.5640 |
| 0.01 | 162.0356 | 43.6219 | 183.8466 |
| 0.03 | 915.5529 | 405.9398 | 1118.5228 |

Furthermore, Table III lists the obtained End-to-End delays of the secure packets in the scenario under attack (scenario *d* of Fig. 5). In some situations, where an attacker started to attempt an attack, but in where from his/her point of view it has not yet been successful (in this case for a 0.003 PIR_A value), the transmission delay starts to grow ultra-passing the threshold, consequently, an under attack signal is triggered, however, the collision point may not yet be determined. On the other hand, for PIR_A configurations, where the attacker noticed success, the proposed DoS detection mechanism managed not only to detect the attack, but also the point where the malicious traffic intercepted the sensitive traffic.

TABLE III: Attack detection (S_A: 15, PL_A: 30, PL_S: 10, PL_R: 10, PIR_R: 0.01)

| Attacker PIR_A | Sensitive PIR_S | End-to-End delay (Clock cycles) | Attack detected | Collision point detection confidence |
|---|---|---|---|---|
| 0.003 | 0.003 | 181.331 | True | 0.35 |
| 0.003 | 0.01 | 765.047 | True | 0 |
| 0.003 | 0.03 | 2679.22 | True | 0 |
| 0.01 | 0.003 | 656.859 | True | 1 |
| 0.01 | 0.01 | 3346.42 | True | 0.8 |
| 0.01 | 0.03 | 4336.19 | True | 0.7 |
| 0.03 | 0.003 | 308.823 | True | 1 |
| 0.03 | 0.01 | 2937.8 | True | 1 |
| 0.03 | 0.03 | 4318.1 | True | 0.95 |

In order to find configurations that would maximize the success of an attack, experiments were done for three different PL_A values (i.e. 10, 30, and 50), as well as, placing the attack source in different routers of the NoC. For such experiments, PL_S and PL_R were set to 10 flits, and all the PIRs were set to 0.01 (i.e. PIR_A = PIR_S = PIR_R = 0.01). End-to-End mean delay values of the sensitive packets for each of those scenarios are listed in Table IV. As expected, results show that the End-to-End delay of the sensitive packets is proportional to the PL_A value, achieving the highest mean delay values for the 50-flit PL_A configurations. This because a longer packet manages to retain the grant of a router output longer, preventing other packets of being forwarded.

Regarding the location of the attacker, as shown in the last row of table Table IV, the highest sensitive End-to-End transmission delays were caused when injecting the malicious traffic into the routers closest to the sensitive destination.

### C. Overhead Evaluation

The proposed architecture was synthesized using the $0.18\mu m$ AMS library and Synopsys design vision at 200 $MHz$. Area overhead and critical path delay of the proposed architecture compared to the baseline architecture are reported

TABLE IV: End-to-End latency vs attacker's packet length and attack source

| Attack Source | Attacker's Packet Length: 10 | 30 | 50 | MAX lat. at |
|---|---|---|---|---|
| **5** | 472.66 | 2756.88 | 5106.99 | 50 |
| **6** | 520.70 | 4286.14 | 6205.65 | 50 |
| **7** | 925.57 | 5759.36 | 7276.82 | 50 |
| **9** | 453.37 | 2244.05 | 4656.09 | 50 |
| **11** | 1170.40 | 5602.28 | 6865.34 | 50 |
| **13** | 314.93 | 2070.89 | 3252.36 | 50 |
| **14** | 830.55 | 3039.49 | 3039.49 | 50 |
| **15** | 658.71 | 3346.42 | 6276.82 | 50 |
| **MAX lat. at** | 11 | 7 | 7 | |

in Table V. The critical path delay overhead of the proposed method is negligible and the proposed monitors only add $17\%$ area overhead to the minimalist router area (each monitor only adds $3.4\%$ overhead to the router's area). The main contribution to area overhead is from the inclusion of the counter registers in the FIFOs.

Power analysis of the proposed method has been performed for random uniform traffic with a packet injection rate of $0.01$ for the baseline architecture and the proposed architecture (without presence of attacker, see Fig. 5e). The results of these experiments are reported in Table VI. The results show that the proposed approach adds $5\%$ power overhead which is for five monitors in the system.

TABLE V: Area and critical path delay overhead

| Router | Area: Sequential ( $\mu m^2$ ) | Combinational ( $\mu m^2$ ) | Total ( $\mu m^2$ ) | overhead (%) | Critical Path Delay (ns) |
|---|---|---|---|---|---|
| Baseline | 48378.7 | 42669.0 | 91047.8 | – | 4.82 |
| Proposed | 52033.7 | 55129.9 | 107163.7 | 17.7% | 4.8 |

TABLE VI: Power overhead

| Router | Switching Power ($mW$) | Internal Power ($mW$) | Leakage Power ($mW$) | Total Power ($mW$) | Total overhead (%) |
|---|---|---|---|---|---|
| Baseline | 0.151 | 3.663 | 0.261 | 3.814 | – |
| Proposed | 0.273 | 3.374 | 0.327 | 4.008 | 5% |

## VI. Conclusions

This paper proposes a distributed DoS detection scheme for measuring the performance degradation of sensitive data transmissions under denial of service attacks and for detecting the injection point of the DoS packets into the sensitive path. We perform an exploration of the effect of different attack configurations targeting the sensitive traffic. Such configurations include different packet lengths, packet injection rates, and attack sources. Our experimental work shows that longer attacker packets, intercepting the sensitive path closer to its destination, cause a better effect of the attack, and that a combination of two delay metrics can be leveraged for not only detecting a DoS attack, but also identifying its entry point into the sensitive path with high accuracy. Future work will include the localization of the source of DoS attack (i.e. the infected PE), which may be injecting malicious data directly into the sensitive path or from a certain distance to it, affecting the sensitive traffic by congesting some router crossbars in the sensitive path.

**Acknowledgments.** This work was partially funded by the German Academic Exchange Service (DAAD) and by the German Federal Ministry of Education and Research (BMBF), grant number 01IS160253 (ARAMiS II) and also supported partly by ETAG IUT19-1 grant.

## References

[1] "Ericsson Mobility Report," https://www.ericsson.com/assets/local/mobility-report/documents/2016/ericsson-mobility-report-june-2016.pdf, accessed: 2018-09-26.

[2] "The internet of things: mapping the value beyond the hype," https://www.mckinsey.de/#energiewende, accessed: 2018-09-26.

[3] L. Fiorin, G. Palermo, and C. Silvano, "A security monitoring service for nocs," in *Proceedings of the 6th IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis*, ser. CODES+ISSS '08. New York, NY, USA: ACM, 2008, pp. 197–202. [Online]. Available: http://doi.acm.org/10.1145/1450135.1450180

[4] J. Sepúlveda, M. Strum, and W. Chau, "An hybrid switching approach for noc-based systems to avoid denial-of-service soc attacks," *16th Iberchip Wksp (IWS 2010)*, pp. 23–25, 2010.

[5] J. Sepúlveda, D. Flórez, and G. Gogniat, "Efficient and flexible noc-based group communication for secure mpsocs," in *2015 International Conference on ReConFigurable Computing and FPGAs (ReConFig)*, Dec 2015, pp. 1–6.

[6] M. J. Sepulveda, J. P. Diguet, M. Strum, and G. Gogniat, "Noc-based protection for soc time-driven attacks," *IEEE Embedded Systems Letters*, vol. 7, no. 1, pp. 7–10, March 2015.

[7] L. L. Caimi, V. Fochi, E. Wachter, D. Munhoz, and F. G. Moraes, "Activation of secure zones in many-core systems with dynamic rerouting," in *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2017, pp. 1–4.

[8] J. Sepúlveda, D. Flórez, and G. Gogniat, "Reconfigurable security architecture for disrupted protection zones in noc-based mpsocs," in *2015 10th International Symposium on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC)*, June 2015, pp. 1–8.

[9] S. Evain and J. P. Diguet, "From noc security analysis to design solutions," in *IEEE Workshop on Signal Processing Systems Design and Implementation, 2005.*, Nov 2005, pp. 166–171.

[10] R. JS, D. M. Ancajas, K. Chakraborty, and S. Roy, "Runtime detection of a bandwidth denial attack from a rogue network-on-chip," in *Proceedings of the 9th International Symposium on Networks-on-Chip*. ACM, 2015, p. 8.

[11] L. Fiorin, C. Silvano, and M. Sami, "Security aspects in networks-on-chips: Overview and proposals for secure implementations," in *Digital System Design Architectures, Methods and Tools, 2007. DSD 2007. 10th Euromicro Conference on*. IEEE, 2007, pp. 539–542.

[12] J.-P. Diguet, S. Evain, R. Vaslin, G. Gogniat, and E. Juin, "Noc-centric security of reconfigurable soc," in *Proceedings of the First International Symposium on Networks-on-Chip*, ser. NOCS '07, 2007, pp. 223–232.

[13] 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*, vol. 34, no. 8, pp. 1344–1357, Aug 2015.

[14] A. B. Achballah, S. B. Othman, and S. B. Saoud, "Toward on hardware firewalling of networks-on-chip based systems," in *2017 International Conference on Advanced Systems and Electric Technologies (IC_ASET)*, Jan 2017, pp. 7–13.

[15] S. P. Azad, B. Niazmand, K. Janson, N. George, A. S. Oyeniran, T. Putkaradze, A. Kaur, J. Raik, G. Jervan, R. Ubar, and T. Hollstein, "From online fault detection to fault management in network-on-chips: A ground-up approach," in *2017 IEEE 20th International Symposium on Design and Diagnostics of Electronic Circuits Systems (DDECS)*, April 2017, pp. 48–53.

[16] G. P. Fettweis, "5g and the future of iot," in *ESSCIRC Conference 2016: 42nd European Solid-State Circuits Conference*, Sept 2016, pp. 21–24.

[17] J. Sepulveda, D. Aboul-Hassan, G. Sigl, B. Becker, and M. Sauer, "Towards the formal verification of security properties of a network-on-chip router," 2018, pp. 1–6.

[18] "ModelSim ASIC and FPGA Design," https://www.mentor.com/products/fv/modelsim/, accessed: 2018-09-26.

# Appendix 2

**V**

Cesar G. Chaves, Siavoosh P. Azad, Thomas Hollstein, and Johanna Sepúlveda, "Diagnosing DoS Attacks in NoC-based MPSoCs," in *Test-methoden und Zuverlässigkeit von Schaltungen und Systemen (TuZ 2019)*, February 2019

# Diagnosing DoS Attacks in NoC-based MPSoCs

Cesar G. Chaves*†, Siavoosh Payandeh Azad†, Thomas Hollstein*†, Johanna Sepúlveda‡
*Frankfurt University of Applied Sciences, †Tallinn University of Technology, ‡Technical University of Munich
email: cesar.chaves@stud.fra-uas.de, siavoosh.azad@ttu.ee, hollstein@fb2.fra-uas.de, johanna.sepulveda@tum.de

***Abstract*—Communication availability in Networks-on-Chip (NoC) based Multi-processors System-on-Chip (MPSoCs) is being threatened by Denial of Service (DoS) attacks. We propose a novel distributed DoS diagnosis scheme which detects DoS attacks based on the performance degradation of sensitive flows. We explore the configuration parameters that assure a successful attack and demonstrate that a combination of two latency metrics can be leveraged not only for detecting a DoS attack, but also, aiming to locate the attack source, it identifies the router where the malicious traffic enters the sensitive communication path.**

## I. Introduction

5G communication technologies allow the connection of a vast amount of devices, enabling the creation of a massive Internet of Things (IoT). Moreover, devices equipped with Multi-Processor Systems-on-Chip (MPSoCs) with a heterogeneous array of Processing Elements (PEs) provide programmability and parallelism, yielding flexibility, processing performance and power efficiency, which can be leveraged for minimizing the latency of communication links of the edge cloud [1]. However, when downloading pieces of software code for executing tasks, malicious tasks can also be downloaded.

MPSoCs are composed of two main structures: 1) PEs such as: processors, hardware accelerators, memories, peripherals and other Intellectual Property hardware cores; and 2) the communication structure, such as: Networks-on-Chip (NoCs), composed of routers and links that exchange information among PEs wrapped as packets (Fig. 1). In NoC-based MPSoCs, DoS attacks are typically executed through infected PEs, which execute malicious software that inject traffic into the NoC, preventing sensitive packets from meeting communication time-line constraints by increasing their latency (Fig. 2).

Previous works regarding NoC DoS attacks propose new router architectures to mitigate these attacks, however, the presented protection countermeasures usually impact the communication heavily [2]–[4]. In this work we present a novel lightweight path-based approach that provides: 1) a NoC architecture able to detect single-source DoS attacks based on the end-to-end latency calculation of packets; 2) the identification of the DoS attack collision point, using per-hop latency calculation; and 3) an exploration of attack scenarios using different attack patterns, packet sizes, and network traffic.

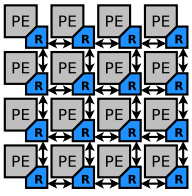



Fig. 1: NoC-Based MPSoC

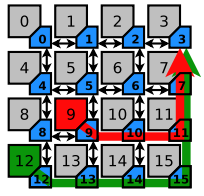



Fig. 2: DoS Attack Scenario

The remainder of the paper is organized as follows: Section II presents a minimalist NoC router and a NoC communication packet structure, providing necessary information for the detection of DoS attacks and the identification of the router where the malicious and sensitive traffic collide. Section III describes the different traffic and packet size scenarios considered for the experiments, which approve the feasibility of the concept. Finally, Section IV outlines the conclusions of the paper and open points for future work.

## II. Diagnosis Method

In order to diagnose a single-source DoS attack and to provide information for detecting its source in a NoC-based MPSoC, we propose three small modifications to the Bonfire NoC project [5]:

*1)* The *Last Body Flit* now carries a time-stamp of the packet's generation time (Fig. 3). Such time-stamp enables the system to calculate the end-to-end latency of each packet, and based on it, determine if a DoS attack is taking place.

*2)* The *Tail Flit* now carries the address of the router where the packet waited the longest while other packets were being forwarded, and the amount of clock cycles it waited (Fig. 3).

*3)* The router's architecture (Fig. 4), by adding the proposed *DoS Monitor* to the beginning of the router's data path (i.e. before each of the five FIFOs). The monitors enable the NoC to distributively report the router where the DoS attempting traffic enters the sensitive communication path. The internal architecture of the proposed *DoS Monitor* is depicted in Fig. 5.

For the diagnosis, DoS monitors have 10-bit counters that measure the local forwarding delay of each received packet. Such counters are reset every time a new header flit enters the FIFO and stopped when the header flit is forwarded or when they reach their maximum value. The counters are incremented on each clock cycle while other packets are being forwarded through their required output port. This is done by monitoring the output grants provided by the router's allocator circuit filtered by the request sent by the Logic-Based Distributed Routing (LBDR) circuit. Furthermore, once the tail flit of the packet arrives, the counter value is compared against the one stored in the tail flit (i.e. *Max Router Latency Value*); If the counter value is greater, the DoS monitor updates the tail flit with the values of the current router and calculates a new parity bit.

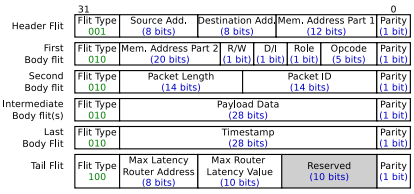



Fig. 3: Packet structure

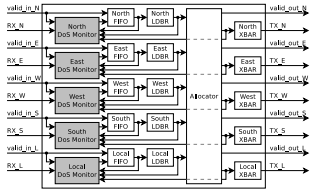



Fig. 4: Proposed router architecture

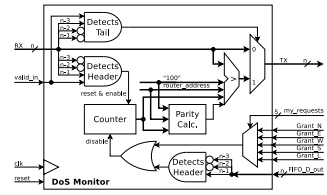



Fig. 5: Dos monitor architecture

Once packets reach their destinations, the system manager of the MPSoC calculates their end-to-end transmission latency based on their time-stamp. Then the latency value is compared to the maximum expected to diagnose the DoS attack. If the latency value exceeds the threshold, the *Max Latency Router Address* and the *Max Router Latency Value* are also retrieved. Such values are analyzed with previous DoS suspicion reports to determine if a DoS attack is taking place and the router where the malicious packets enter the sensitive path (a.k.a. the router that reported the greatest forwarding delay). Finally, when a DoS attack has been diagnosed and the location of the collision point has been identified, the system manager can send control packets from other sources that pass through the collision point to narrow down the source of the attack and issue a reset command, thus eliminating the malicious traffic generation.

## III. Experimental Work

The DoS Monitor, presented in Section II, was implemented on RTL level and integrated to the Bonfire framework [5]. Subsequently, the modified NoC architecture was simulated using *Modelsim* from *Mentor Graphics* [6]. The following Subsections detail the simulation scenarios and the experimental results, III-A and III-B, respectively.

### A. Simulation Scenarios

Simulations were executed adopting a $4\times4$ mesh NoC-based MPSoC architecture (Fig. 2). The network routers (applying an xy-routing algorithm) use a credit-base flow control with fair Round Robin switch matrix output arbitration (on packet level) and utilize wormhole switching with 4-flit deep FIFOs.

In all scenarios, PE3 was the destination of all traffic, PE12 the source of the sensitive traffic (i.e. sensitive path: PE12→PE3), and for each scenario, malicious traffic was generated from a different PE, thus, depending on the selected attack source, the malicious path length and the router where paths intersect vary. In the case of the scenario depicted in Fig. 2, the malicious path is represented by: PE9→PE3, its length is 4 hops and the intersection router is R11. In the cases where the attack source is directly in the sensitive path, the source and collision point routers are the same.

Simulations were done considering different packet injection rates (PIR) for sensitive packets (PIR_S), attack packets (PIR_A), while the random traffic (PIR_R) was set to $0.01$ (one packet every $100$ clock cycles), and data generated for $20$ pseudo-random seeds. Also, a packet length of $10$ flits was adopted for the sensitive (PL_S) and random traffic (PL_R) while in the scenarios where an attack attempt was present, three different packet lengths were considered for the attacker's packets (PL_A) (i.e. 10, 30, and 50).

### B. Experimental Results

The experiments targeted the analysis of the DoS attack from two points of view: 1) attacker side: in order to maximize the effectiveness of the attack, and 2) network protection side: testing metrics that can be leveraged for detecting an attack and qualifying the effectiveness of the detection mechanism.

Since an attacker can identify the success/effectiveness of the attack by measuring its own throughput [7], we measured the effective PIR_A by considering the configurations listed in Section III-A, and setting PE15 as the attack source (S_A), attacking with 30-flit-length packets. Results showed that as the PIR_A value increments, the deviation regarding the Attacker's Effective PIR also increases. A PIR_A value of $0.01$ showed to be over the limit of the network's capacity for the adopted configuration (i.e. the attack was successful). For the purpose of this work, we considered an attack effective when the PIR_A deviation percentage was over $10\%$.

As mentioned in Section II, the End-to-End communication delay of each packet is calculated using its time-stamp, which is generated when the packet enters the NoC. End-to-End delay values of the sensitive packets for different PIR_S values were calculated in a scenario with no attackers, subsequently, their mean values and the sample standard deviations (SSD) were used for detecting the DoS attacks, setting a threshold $T$ calculated by $T = mean\_delay + 0.5 \times SSD$.

Table I lists the reported end-to-end delays of the secure packets. In configurations where the attacker was unable to identify the attack's effectiveness, delay values exceeded the threshold, signaling the attack, but the collision point was not determined. On the other hand, for PIR_A configurations, where the attacker noticed success (i.e. when PIR_A = $0.01$ or PIR_A = $0.03$), the proposed DoS detection mechanism managed not only to detect the attack, but also the point where the malicious traffic intercepted the sensitive path.

In order to find configurations that would maximize the success of an attack, experiments were done for three different PL_A values (i.e. 10, 30, and 50), as well as, placing the attack source in different routers of the NoC. For such experiments, PL_S and PL_R were set to $10$ flits, and all the PIRs were set to $0.01$ (i.e. PIR_A = PIR_S = PIR_R = $0.01$). As expected,

TABLE I: Attack detection (S_A: 15, PL_A: 30, PL_S: 10, PL_R: 10, PIR_R: 0.01)

| Attacker PIR_A | Sensitive PIR_S | End-to-End delay (Clock cycles) | Attack detected | Collision point detection confidence |
|---|---|---|---|---|
| 0.003 | 0.003 | 181.331 | ✓ | ✗ |
| 0.003 | 0.01 | 765.047 | ✓ | ✗ |
| 0.003 | 0.03 | 2679.22 | ✓ | ✗ |
| 0.01 | 0.003 | 656.859 | ✓ | 1 |
| 0.01 | 0.01 | 3346.42 | ✓ | 0.8 |
| 0.01 | 0.03 | 4336.19 | ✓ | 0.7 |
| 0.03 | 0.003 | 308.823 | ✓ | 1 |
| 0.03 | 0.01 | 2937.8 | ✓ | 1 |
| 0.03 | 0.03 | 4318.1 | ✓ | 0.95 |

results show that the End-to-End delay of the sensitive packets is proportional to the PL_A value, achieving the highest mean delay values for the 50-flit PL_A configurations. This because a longer packet manages to retain the grant of a router output longer, preventing other packets from being forwarded.

Results also showed that the highest sensitive end-to-end transmission delays were achieved when injecting the malicious traffic into the routers closest to the sensitive destination.

### C. Overhead Evaluation

The proposed architecture was synthesized using the $0.18\mu m$ AMS library and Synopsys design vision at 200 $MHz$. Area overhead and critical path delay of the proposed architecture compared to the baseline architecture are reported in Table II. The critical path delay overhead of the proposed method is negligible and the proposed monitors only add $17\%$ area overhead to the minimalist router area (each monitor only adds $3.4\%$ overhead to the router's area). The main contribution to area overhead is from the inclusion of the counter registers in the FIFOs.

Power analysis of the proposed method has been performed for random uniform traffic with a packet injection rate of $0.01$ for the baseline architecture and the proposed architecture (without the presence of an attacker). The results of these experiments are reported in Table III. The results show that the proposed approach adds $5\%$ power overhead which is for five monitors in the system.

TABLE II: Area and critical path delay overhead

| Router | Area | | | | Critical Path Delay (ns) |
|---|---|---|---|---|---|
| | Sequential ( $\mu m^2$ ) | Combinational ( $\mu m^2$ ) | Total ( $\mu m^2$ ) | overhead (%) | |
| Baseline | 48378.7 | 42669.0 | 91047.8 | – | 4.82 |
| Proposed | 52033.7 | 55129.9 | 107163.7 | 17.7% | 4.8 |

TABLE III: Power overhead

| Router | Switching Power ($mW$) | Internal Power ($mW$) | Leakage Power ($mW$) | Total Power ($mW$) | Total overhead (%) |
|---|---|---|---|---|---|
| Baseline | 0.151 | 3.663 | 0.261 | 3.814 | – |
| Proposed | 0.273 | 3.374 | 0.327 | 4.008 | 5% |

## IV. Conclusions

This paper proposes a distributed monitoring scheme that, by measuring the performance degradation of sensitive data transmissions, is able to diagnose denial of service attacks in NoC-based MPSoCs and to identify the injection point of the DoS packets into the sensitive path. We perform an exploration of the effect of different attack configurations targeting the sensitive traffic. Such configurations include different packet lengths, packet injection rates, and attack sources. Our experimental work shows that longer attacker packets, intercepting the sensitive path closer to its destination, cause a better effect of the attack, and that a combination of two delay metrics can be leveraged for not only detecting a DoS attack, but also identifying its entry point into the sensitive path with high accuracy.

Future work will aim to enhance the efficiency of the diagnosis method, this by filtering the DoS attack source suspects when detecting the collision point, hence reducing the amount of control packets to be sent, and consequently, the source detection time. Also, multi-source DoS attack detection and localization will be addressed.

**Acknowledgments.** This work was partially funded by the German Academic Exchange Service (DAAD) and by the German Federal Ministry of Education and Research (BMBF), grant number 01IS160253 (ARAMiS II) and also supported partly by the ETAG IUT19-1 grant.

## References

[1] G. P. Fettweis, "5G and the Future of IoT," in *ESSCIRC Conference 2016: 42nd European Solid-State Circuits Conference*, Sept 2016, pp. 21–24.

[2] L. Fiorin, C. Silvano, and M. Sami, "Security Aspects in Networks-on-Chips: Overview and Proposals for Secure Implementations," in *Digital System Design Architectures, Methods and Tools, 2007. DSD 2007. 10th Euromicro Conference on*. IEEE, 2007, pp. 539–542.

[3] J. Sepúlveda, M. Strum, and W. Chau, "An Hybrid Switching Approach for NoC-Based Systems to Avoid Denial-of-Service SoC Attacks," *16th Iberchip Wksp (IWS 2010)*, pp. 23–25, 2010.

[4] L. L. Caimi, V. Fochi, E. Wachter, D. Munhoz, and F. G. Moraes, "Activation of Secure Zones in Many-Core Systems with Dynamic Rerouting," in *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2017, pp. 1–4.

[5] S. P. Azad, B. Niazmand, K. Janson, N. George, A. S. Oyeniran, T. Putkaradze, A. Kaur, J. Raik, G. Jervan, R. Ubar *et al.*, "From Online Fault Detection to Fault Management in Network-on-Chips: A Ground-Up Approach," in *2017 IEEE 20th International Symposium on Design and Diagnostics of Electronic Circuits Systems (DDECS)*, April 2017, pp. 48–53.

[6] "ModelSim ASIC and FPGA Design," https://www.mentor.com/products/fv/modelsim/, accessed: 2018-09-26.

[7] J. Sepulveda, D. Aboul-Hassan, G. Sigl, B. Becker, and M. Sauer, "Towards the formal verification of security properties of a network-on-chip router," in *2018 IEEE 23rd European Test Symposium (ETS)*. IEEE, 2018, pp. 1–6.

# Appendix 3

**II**

Cesar G. Chaves, Siavoosh P. Azad, Thomas Hollstein, and Johanna Sepúlveda, "DoS Attack Detection and Path Collision Localization in NoC-Based MPSoC Architectures," *Journal of Low Power Electronics and Applications*, vol. 9, no. 1, 2019

Journal of
*Low Power Electronics and Applications*

MDPI

*Article*

# DoS Attack Detection and Path Collision Localization in NoC-Based MPSoC Architectures

**Cesar Giovanni Chaves [1,2,†], Siavoosh Payandeh Azad [2,†], Thomas Hollstein [1,2] and Johanna Sepúlveda [3,*]**

[1] Faculty 2: Computer Science and Engineering, Frankfurt University of Applied Sciences, 60318 Frankfurt am Main, Germany; cesar.chaves@stud.fra-uas.de (C.G.C.); hollstein@fb2.fra-uas.de (T.H.)
[2] Department of Computer Engineering, Tallinn University of Technology, 12618 Tallinn, Estonia; siavoosh.azad@taltech.ee
[3] Department of Electrical and Computer Engineering, Technical University of Munich, 80333 Munich, Germany
* Correspondence: johanna.sepulveda@tum.de; Tel.: +49-(89)-289-28256
† These authors contributed equally to this work.

Received: 27 November 2018; Accepted: 1 February 2019; Published: 5 February 2019

**Abstract:** Denial of Service (DoS) attacks are an increasing threat for Multiprocessor System-on-Chip (MPSoC) architectures. By exploiting the shared resources on the chip, an attacker is able to prevent completion or degrade the performance of a task. This is extremely dangerous for MPSoCs used in critical applications. The Network-on-Chip (NoC), as a central MPSoC infrastructure, is exposed to this attack. In order to maintain communication availability, NoCs should be enhanced with an effective and precise attack detection mechanism that allows the triggering of effective attack mitigation mechanisms. Previous research works demonstrate DoS attacks on NoCs and propose detection methods being implemented in NoC routers. These countermeasures typically led to a significantly increased router complexity and to a high degradation of the MPSoC's performance. To this end, we present two contributions. First, we provide an analysis of information that helps to narrow down the location of the attacker in the MPSoC, achieving up to a 69% search space reduction for locating the attacker. Second, we propose a low cost mechanism for detecting the location and direction of the interference, by enhancing the communication packet structure and placing communication degradation monitors in the NoC routers. Our experiments show that our NoC router architecture detects single-source DoS attacks and determines, with high precision, the location and direction of the collision, while incurring a low area and power overhead.

**Keywords:** denial of service attacks; network-on-chip; distributed online monitoring; multi-processor system-on-chip; security

## 1. Introduction

The extensive use of Internet-of-Things (IoT) will be the driver of the ongoing digitization in many application domains, as in smart living and working environments, smart cities, health care industry automation, automotive, and avionics. IoT is a pervasive technology that increasingly captures the attention of researchers, industry, and governments. Computational tasks are mapped on a larger number of distributed IoT nodes, having increased connectivity through device-to-device communication. Operations can be deployed into a sea of devices, their execution time can be reduced and smart behaviors can be established in such distributed systems. Three characteristics turn the IoT paradigm relevant in modern systems:

(i) Number of connected devices: it is estimated that by 2021, 28 billion devices will be part of IoT [1];

(ii) Economical impact: by introducing automated and smart processes, the IoT is foreseen to achieve the mark of 11 trillion US Dollars by 2025 [2]; and
(iii) Integration: IoT devices will integrate into a broad spectrum of markets, including different industry and service markets.

Increasingly complex and powerful Systems-on-Chips (SoCs), connected via wireless communication technologies as WLAN, WPAN, Bluetooth, or 5G networks, form the basis of the IoT. Multi-Processors System-on-Chips (MPSoCs) are considered as an increasingly important key enabler technology for the implementation of IoT nodes. They are composed of two main structural types of components:

(i) the computation structure, composed of Processing Elements (PEs) such as: processors, hardware accelerators, memories, peripherals, and other Intellectual Property (IP) hardware cores to process and store information; and
(ii) the MPSoC internal communication infrastructure, which performs the data exchange among the IP cores.

Networks-on-Chip (NoCs) are the communication structure choice for MPSoCs that integrate a large amount of IP cores. NoCs integrate routers and links to exchange information being wrapped as packets.

NoC-based SoCs allow a very flexible task deployment onto the MPSoC's computational nodes. This flexibility also enables enhanced dependability concepts [3,4], providing fault-tolerance and, with this, a high-level of reliability in case of wear-out of some of the chip's nano-scaled structures. In order to detect deviations in the system health state, critical resources (as routers) have to be equipped with local health monitoring technology (hardware fault checkers). Since these put a certain linear overhead on the hardware size, the routers should be designed as minimalist as possible. This goes well along with the requirement of energy-efficient and sustainable operation of the MPSoC-based IoT component.

Security is a major requirement for all IoT domains. The configurability and hyper-connectivity of these devices represent a risk to the system. Since these systems are connected to the internet, it is more convenient for the developers to enable them to download software code and firmware updates remotely. However, the downloaded software may be malicious. This can lead to retrieving secret information (secret keys and intellectual property), modifying the system's operation or disrupting its services. One of the approaches for disrupting the services in NoC-based MPSoCs is by means of Denial of Service (DoS) attacks. Since the NoC is a shared communication medium, any disruption of or obstruction to the efficient computation or communication services will affect other system services. The goal of the attacker is to prevent the completion of a sensitive task (e.g., data encryption, trigger of an alarm), so that users decide to turn off the encryption or to migrate tasks to an alternative infected neighbor device (when tasks are deployed in the IoT infrastructure).

In NoC-based MPSoCs, Denial of Service (DoS) attacks typically originate from infected IP cores, which execute malicious software. The goal of the attacks is to interfere with and degrade the sensitive data communication by injection of malicious traffic, thus communication time-line constraints are missed. The Worldwide Infrastructure Security Report classifies DoS attacks into three categories [5]:

(i) Volumetric: the objective is to cause congestion at a single injection point in order to overwhelm the NoC's bandwidth;
(ii) State-exhaustion: the goal is to try to use up all available NoC connections; and
(iii) Connection-based (Application-Layer Attacks): exploitation of priority-based NoCs in order to monopolize the NoC communication.

These three classes can be directly applied in NoC infrastructures, which result in an increase in the end-to-end communication latency of the sensitive packet flows [6]. Previous published research outlines the possibility of executing DoS attacks in NoCs and propose new router architectures to mitigate them. However, this protection countermeasures usually have a heavy impact on network

communication and/or router complexity. Moreover, in order to create effective countermeasures that neutralize threats, as well as to enable dynamic and customized DoS protection methods, the attacker's location needs to be identified. However, identifying the attacker's location is a complex task, especially when considering minimalist hardware requirements for NoC routers design.

In this work we analyze the size of the attack suspect search space (i.e., the number of nodes to be checked based on the detection information) using the information about the collision location and the direction of the attacker packets. We show that by knowing not only the collision location, but also the input direction of the malicious packets, the search space can be reduced a maximum of 69%. Later we propose two online monitoring mechanisms able to provide the above mentioned information to the NoC's system manager. Subsequently, we present an analysis of different attack configurations, showing their effectiveness. The experimental results show the proposed approaches' efficacy and efficiency to detect the DoS attacks, identify the collision point of malicious and sensitive data, and find the direction from which the malicious packets enter the sensitive path.

The rest of this paper is organized as follows: Section 2 describes the state of the art in defense against DoS attacks in NoCs. Sections 3 and 4 describe the baseline framework used for analysis and the threat model considered for this work. Section 5 analyses the benefits of acquiring detailed information about the point of collision during a DoS attack. Section 6 presents the proposed approaches to acquire such information. Section 7 presents the experimental results and finally Section 8 concludes the paper.

## 2. Literature Review

The concept of DoS in the NoC context was introduced in [7]. Subsequently, approaches with different mitigation strategies for these kinds of attacks have been proposed [7–18]. Such approaches can be categorized in two main classes: *DoS Avoidance* and *DoS detection and recovery*. The first focuses on providing an infrastructure to avoid such attacks altogether (see Section 2.1). On the other hand, the second tries to identify the occurrence of such attacks and disabling them (see Section 2.2). The latter can be divided into two sub groups: works which focus on techniques for detecting the occurrence of the attack, but cannot detect the location of the node which runs malicious code, and works that can also identify the location of the attacker.

### 2.1. DoS Attack Avoidance

In [8] a mitigation strategy based on the hybrid switching routing mechanism is proposed. Circuit switching is used for sensitive traffic and packet switching is used otherwise. As a result, predictable latency for sensitive traffic is guaranteed. The works of [9,11,12] mitigate DoS and timing attacks by generation of security zones. Their proposal ensures that only the secure nodes can communicate into a virtual and physical space, respectively. Security zones are built through dynamic routing. In [13] non-sensitive traffic is deeply inspected in order to access to security zones. The works of [11,12] reroute the traffic outside from the security zone. On the other hand, the authors of [7] propose the use of separate virtual channels for secure and non-secure packets as a countermeasure for bandwidth denial attacks in NoCs. However, this approach suffers from the hardware overhead of additional buffers used for virtual channels.

### 2.2. DoS Attack Detection and Recovery

In [14], authors consider that DoS attacks are executed by a rogue third-party NoC architecture. When a suspicious communication delay is detected between two nodes, the firmware tries to identify if the attack is taking place at the source or at the destination by time-stamping the packets and dividing into two communication paths. In [15], Fiorin et al. provide an overview of the attacks in network on chips. The authors propose the use of buffer occupancy monitors to detect traffic anomaly in the network but provide no details on the technique. Later, in [10], Fiorin et al. proposed the implementation of denial of service probes in the Network Interface. This approach only covers the DoS attacks from the Processing Elements connected to the network interface by detecting deviations

from the average bandwidth usage expected at the design time. This approach does not actively monitor the latency of the sensitive packets in the network. In [16], Diguet et al. tackle DoS attacks by monitoring live-lock occurrence in the network. However, this approach utilizes source-routing, which imposes considerable overhead to the packet size. In [17], Grammatikakis et al. target distributed DoS attacks via a firewall in control of configurable access rules in the network interface. The security risk is evaluated by the product of frequency and magnitude of losses (by dropping the packets at the Network Interface). Although such a firewall does not allow the DoS attack to be effective, it was designed to protect the destination PE (i.e., the on-chip memory), thus it does not detect the source of the attach, nor does it kill the attack-related traffic congesting the network. In [18], Achballah et al. target computation of occupation time of the physical links in number of clock cycles. The occupation time of the physical link is compared to an expected value. In case of the link occupation exceeding this value, the transaction will be stopped and a notification will be sent to the system's manager. Here the decision is made locally in the router, which may lead to undesirable packet dropping.

The evaluation of the related works shows that the DoS attack avoidance approaches require significant additional hardware effort in the routers and the system. Since scalable NoCs need an overall dependability management [19], the router size should be as minimal as possible (e.g., no virtual channels).

Our proposed approaches can be classified under *DoS Attack Detection and Recovery*, since they analyze communication degradation in order to detect DoS attacks. Additionally, once an attack is detected, they proceed to reduce the attack suspects so that the attacker can be located and neutralized.

Even though the works listed in Section 2.2 seek the same objective as our approaches, they either are limited to parameters defined at design time or to the source routing algorithm. In contrast, the decision parameters of our approaches are controlled by the system manager, and, as explained along the paper, our proposed diagnosis mechanisms can be of use for different routing algorithms, making them more flexible when compared to the other approaches.

Furthermore, we propose an architecture of scalable distributed monitoring solutions for NoC traffic, in order to diagnose DoS attacks and identify attack characteristics, that help to find the source of the attack. These monitors transmit relevant performance information that is analyzed at the endpoint of the communication. We considered two diagnosis mechanisms based on the approaches for search space reduction mentioned above. The first, called Collision Point Router Detection (CPRD), identifies the router where the attacker's malicious packets collide with the sensitive packets. The second approach, called Collision Point Direction Detection (CPDD), extends the CPRD by not only identifying the point of collision, but also the input port through which the attacker's flow intercepted the sensitive path. The importance of such an extension is explained in Section 5, where the two approaches are compared.

## 3. Router Architecture

Bonfire is an open-source framework for developing dependability mechanisms for NoC-based Systems-on-Chip (SoCs). The target NoC architecture is a 2D mesh topology. Each network tile consists of a wormhole switching router equipped with fault tolerance mechanisms and a Network Interface (NI), which is connected to the local resource. The routers use credit-based flow-control and support any turn-model-based minimal path adaptive routing algorithm [19,20]. Figure 1a shows a block diagram of the credit-based router used in this work. This router contains input buffers and routing units for each input port, one switch allocator that provides support for any turn-model based adaptive, a minimal path routing algorithm, and a crossbar switch for each output port. In credit-based flow-control, each upstream router keeps track of empty buffers in the downstream router and sends the flits with the assumption that the downstream router can receive them. The FIFO unit utilizes a 4-flit deep circular buffer and is in charge of issuing a credit signal to the upstream router. The routing logic is implemented using a Logic Based Distributed Routing (LBDR) mechanism [21], which is a

lightweight distributed routing mechanism that supports any turn-model based routing algorithm and provides the possibility of an in-system reconfiguration of the routing algorithm.

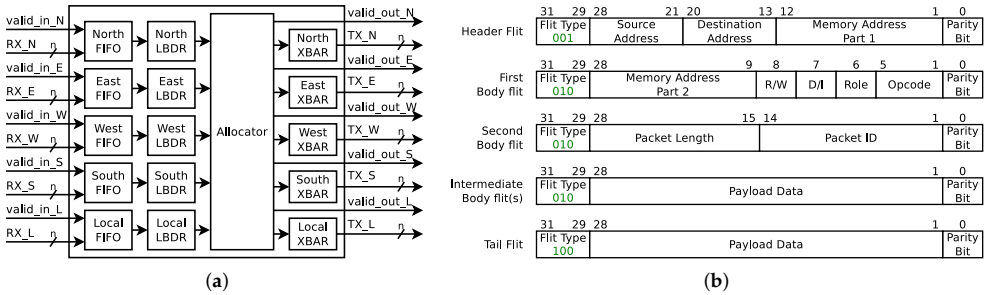



**Figure 1.** Bonfire baseline frameworks: (**a**) router architecture and (**b**) packet structure [20].

The allocator unit utilizes a Round Robin arbitration, where each input direction can hold access to an output port for the length of a single packet. This arbitration mechanism has an important impact on the resilience of the network to DoS attacks, since it prevents the starvation of packets entering the router through other input ports.

Figure 1b shows the packet format used by the Secure Bonfire variant of the Bonfire framework. This packet format is designed to be compatible with the Open Core Protocol (OCP) and each body flit carries a payload of 28 bits.

## 4. Threat Model

MPSoCs with a heterogeneous array of PEs provide programmability and parallelism, yielding flexibility, processing performance and power efficiency, which can be leveraged for minimizing the latency of communication links of the edge cloud [22]. An example of such a scenario is depicted in Figure 2, where IoT devices, equipped with NoC-based MPSoCs, improve the performance of conventional general-purpose computer execution environments by adding application specific circuits. However, this junction brings new security vulnerabilities to the NoC-based MPSoC world, such as susceptibility to DoS attacks, as will be explained in this section.

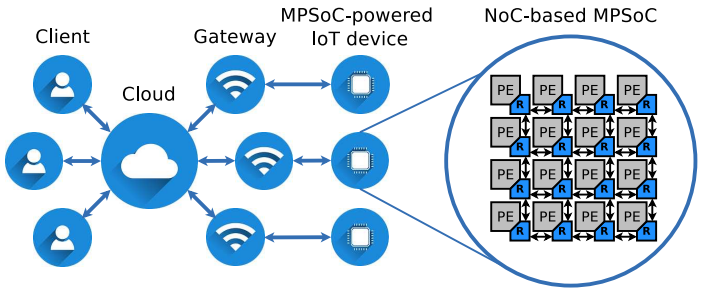



**Figure 2.** Multiprocessor System-on-Chip (MPSoC)-powered Internet-of-Things (IoT) execution environment.

*Attack Scenario:* During a DoS attack, an attacker seeks to degrade the performance of a processing/communication environment, which in the scope of this work is represented by a NoC-based MPSoC. In an environment, such as the one illustrated in Figure 2, users submit their applications for execution on MPSoC-powered IoT devices. Subsequently, a task scheduler in the firmware of the IoT device, during its normal operation, maps the applications to the PEs of the incorporated MPSoC. However, one or more of those applications could have been submitted by a

user with the intention of performing a DoS attack to the NoC. Moreover, the most common DoS attack method is network flooding, where a malicious application running on any PE of the MPSoC is able to do by sending packets to another PE in the NoC. These additional packets compete for the crossbars of the routers in their path, and, while successful, block the transmission of other packets. Blocked packets experience an increment of their forwarding delays, which results in an overall additional communication delay that will be unacceptable by time sensitive applications.

*Attack Steps:* A malicious user (a.k.a. an attacker), submits his/her application to the execution platform as any other user. Subsequently, a task scheduler maps the application to the PEs of the MPSoC in order to start its execution. A basic procedure being followed by a simple malicious application, whose goal is to attempt a flooding DoS attack on a NoC, is depicted in Figure 3. First, the malicious application sets its Packet Injection Rate (PIR) to an initial value. Then, packets are transmitted to another PE in the NoC at the defined PIR. As the packets are transmitted, the application calculates the Effective PIR (EPIR) in order to assess the congestion of the NoC, which reflects the effectiveness of the attack [23]. If the NoC is still not congested (i.e., PIR = EPIR), the application continues to send packets increasing the PIR value. Finally, once the application establishes that its attack is being effective, it continues to send packets maintaining the current PIR until the end of the attack.

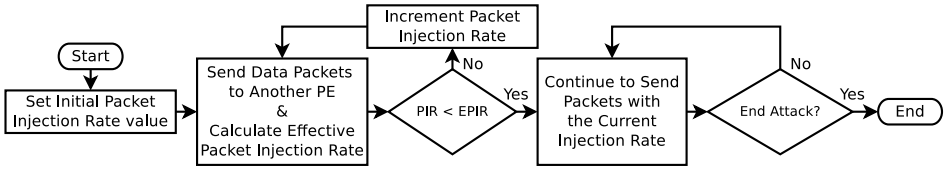



**Figure 3.** Basic procedure of a malicious application.

*Attack Success Conditions:* For a successful attack to be launched on a NoC-based MPSoC, the following conditions must be met:

- The attacker packets should be able to interfere with the flow of sensitive packets in at least one location. This means that both traffic flows should compete for at least a single router output. Assuming that the attacker might not have prior knowledge of the network segments used by the sensitive traffic, setting long paths for the malicious traffic will increase the probability of colliding with the sensitive traffic.
- The attacker can control its own packet injection rate into the network. The attacker then can increase the injection rate and monitor its own throughput. Based on its throughput information, an attacker can infer if its traffic has interfered with other traffic in the network [23]. Additionally, depending on the router's arbitration, a burst of packets can hold a crossbar grant, causing the starvation of other packets.
- The attacker node should be able to control the length of the packets that it injects into the network. In practice, a single input direction cannot hold an output longer than the duration of a single packet (e.g., Bonfire that uses wormhole switching with round robin arbitration in a per packet basis). Thus, longer packets occupy the crossbar longer and consequently generate a longer inter-router forwarding delay to sensitive packets, which is reflected in the end-to-end communication delay.

## 5. Analysis of Attack Suspects

In order to mitigate the DoS attacks, it is important that the system manager identifies the physical location of the attacker. However, there is no straightforward approach for acquiring such information. The system manager can only collect indirect or partial information regarding the incident in order to infer the location of the attacker. Without any additional information, excluding the source and destination of the sensitive traffic, the system will have N-2 suspects, where N is the number of

nodes in the network. In this section, we assume that the location of the collision and direction of the incoming attacking flow can be recovered during the transmission of a packet. Later we analyze how this information, combined with details of the routing algorithm, can be leveraged towards narrowing down the search space for finding the source of the attack.

This part of the analysis is performed by the use of Routing Graphs [3]. A Routing Graph, $RG(V, E)$, is a directed acyclic graph where $V$ is the set of vertices that represent all network ports (input and output ports of the routers including the local ports) and $E$ is the set of all edges which represent connections between the input and output ports. The connections in the set $E$ in $RG$ can be of two types: (i) internal; inside a router, between the input and output ports (including turns and other paths inside the router) and (ii) external; the physical link between neighbor routers. By setting the internal connections, it is possible to model any turn-model-based routing algorithm. Moreover, using routing graphs enables the tool to use graph algorithms for finding paths between any source and destination.

Algorithm 1 provides python pseudo code for the steps required for finding all possible attacker suspect nodes, by only considering the location of the router where the malicious and sensitive packets collide. The main idea is to find all the output ports on every path between source and destination, and subsequently to find out how many nodes can send packets to these output ports under minimal path routing.

**Algorithm 1:** Attack suspects considering collision location.

```
   # Dictionary of all attackers
1  all_attackers = { };
2  for all paths P in {all minimal paths between S_s and S_d in RG} do
       # list of all output ports on the path
3      O = [ ];
       # for each path find all unchecked output ports
4      for all outports in P do
5          if outport not in all_attackers.keys() then
6              O.append(outport);
       # for each outport find all attacker candidates
7      if len(O) != 0 then
8          for all outports in O do
9              A = [ ];
10             for local node L in RG do
11                 if L has minimal path to outport in RG then
12                     A.append(L);
13             all_attackers[outport] = A;
14 return all_attackers
```

Algorithm 2 provides python pseudo code for the steps required to find all possible attacker suspect nodes by considering the collision router location and the input direction of the malicious packets. The main idea is to find all the output ports on every minimal route path from the source to destination. Subsequently, for each output port, all the possible input ports that can forward packets to this port are checked. Finally, all the local nodes that can send packets to that port via minimal path routing are found, under the current routing algorithm.

In the next subsection, we will describe the process via three examples using the XY routing algorithm for the sake of simplicity. Later we will show the results for adaptive routing as well.

**Algorithm 2:** Attack suspects considering collision location and input direction of interference.

```
    # Dictionary of all attackers
1   all_attackers = { };
2   for all paths P in {all minimal paths between S_s and S_d in RG} do
        # list of all input that can reach output ports on the path
3       I = [ ];
        # list of all nodes that can reach output ports on the path
4       A = [ ];
5       for all outports in P do
            # for each outport find all attacker candidates and all connected inports
6           I = RG.predecessors(outport);
7           for all local nodes L in RG do
8               if L has minimal path to outport in RG then
9                   A.append(L);
        # Checking if the nodes in A can reach inports
10      if len(I) != 0 then
11          for all inports in I do
12              for all local nodes L in A do
13                  if (L has minimal path to inport in RG) and (L not in all_attackers[inport]) then
14                      all_attackers[inport].append(L);
15  return all_attackers
```

### 5.1. Example Using XY Routing Algorithm

In this section we explore a simple example based on the XY routing algorithm [24]. This example is taken due to the simplicity of the routing algorithm. In the next section adaptive routing algorithms will be considered as well.

Figure 4 shows the first scenario considered for this exercise. The sensitive path is represented by the green arrow, which begins in router 12 and ends in router 3, going through routers 13, 14, 15, 11, and 7. In case the system manager labels node 13 as the collision point, it has only a single candidate for the attacker, namely node 13, since no other node's traffic (excluding node 12 as the sensitive source) can request node 13's east output. However, in more sophisticated cases, the set of candidates is enlarged (as listed in Table 1). It is important to note that the manager excludes the nodes that have higher chance of collision in their own tile. For example, if the reports label node 14 as the collision point, the manager will not include node 13 as a suspect since it has a better chance of interference on node 13's east output. This assumption might not hold for coordinated distributed denial of service attacks.

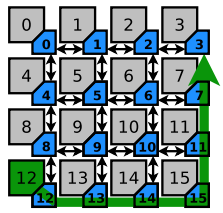



**Figure 4.** Sensitive path 1.

**Table 1.** Attack suspects for Sensitive path 1.

| Collision Point | Suspects with Collision Point Router Detection | Detected Direction | Suspects with Collision Point Direction Detection |
|---|---|---|---|
| 13 | 13 | L | 13 |
| 14 | 14 | L | 14 |
| 15 | 15 | L | 15 |
| 11 | 8, 9, 10, 11 | W | 8, 9, 10 |
| | | L | 11 |
| 7 | 4, 5, 6, 7 | W | 4, 5, 6 |
| | | L | 7 |
| 3 | 0, 1, 2 | W | 0, 1, 2 |

Figure 5 describes a scenario where the source and destinations of the sensitive path are nodes 8 and 2, respectively. This example is different from the previous example in one important aspect: the sensitive path can be interfered with from both sides. This in turn will increase the search space for the possible suspects dramatically. As it can be seen in Table 2, node 10 is a descriptive example of such a situation. However, knowing the direction of the interference will further decrease the search space (from 6 nodes to worst case number of 4 nodes, and in best case scenario down to a single node).

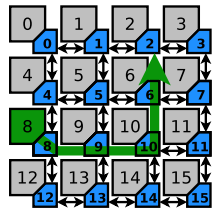



**Figure 5.** Sensitive path 2.

**Table 2.** Attack suspects for Sensitive path 2.

| Collision Point | Suspects with Collision Point Router Detection | Detected Direction | Suspects with Collision Point Direction Detection |
|---|---|---|---|
| 9 | 9 | L | 9 |
| 10 | 10, 11, 12, 13, 14, 15 | E | 11 |
| | | S | 12, 13, 14, 15 |
| | | L | 10 |
| 6 | 4, 5, 6, 7 | E | 7 |
| | | W | 4, 5 |
| | | L | 6 |
| 2 | 0, 1, 3 | E | 3 |
| | | W | 0, 1 |

The scenario depicted in Figure 6 shows a more extreme case than Scenario 2, where a single node (node 5) has a search space of 11 suspect nodes (refer to Table 3). Further investigation shows that knowing the direction of interference reduces the number of suspects, in the worst case to 8 nodes and in the best case to 1 node (3.6 nodes on average).

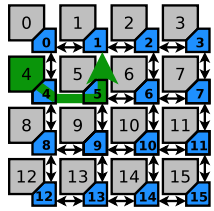



**Figure 6.** Sensitive path 3.

**Table 3.** Attack suspects for Sensitive path 3.

| Collision Point | Suspects with Collision Point Router Detection | Detected Direction | Suspects with Collision Point Direction Detection |
|---|---|---|---|
| 5 | 5, 6, 7 , 8, 9, 10, 11, 12, 13, 14, 15 | E | 6, 7 |
| | | S | 8, 9, 10, 11, 12, 13, 14, 15 |
| | | L | 5 |
| 1 | 0, 2, 3 | E | 2, 3 |
| | | W | 0 |

## 5.2. Discussion

When applying the same method on different routing algorithms, the gain of the proposed mechanisms becomes more clear. Figures 7–9 depict the minimum, maximum and average number of attack source suspects to be checked for the three scenarios (i.e., Sensitive paths 1 to 3), for different minimal-path turn-model based routing algorithms. In this analysis the following routing algorithms have been considered: XY [24], YX [25], West-First [26], East-First [27], North-First [28], North-Last [26], Negative First [26], and South-First [28]. In this work, the *SocDep*$^2$ [3] framework has been used for modeling the routing graphs for the above mentioned turn models and for evaluating the presented Algorithms 1 and 2 for different scenarios.

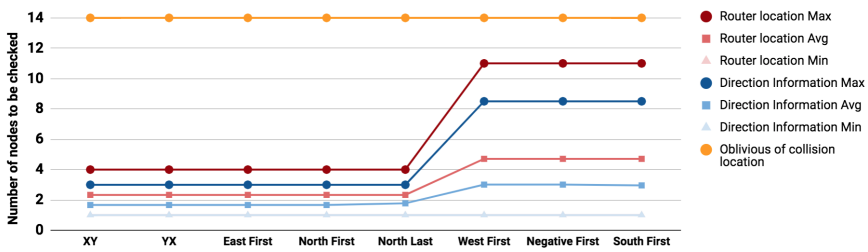



**Figure 7.** Average number of nodes to be checked under different routing algorithms (Sensitive path 1).

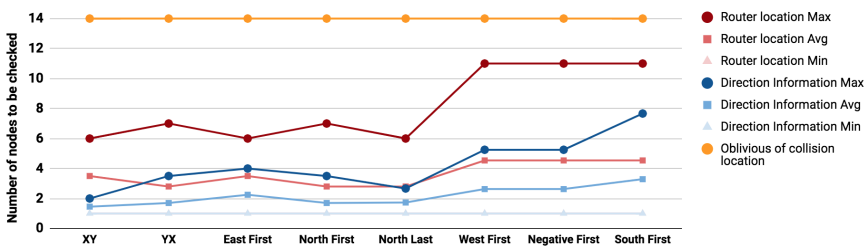



**Figure 8.** Average number of nodes to be checked under different routing algorithms (Sensitive path 2).

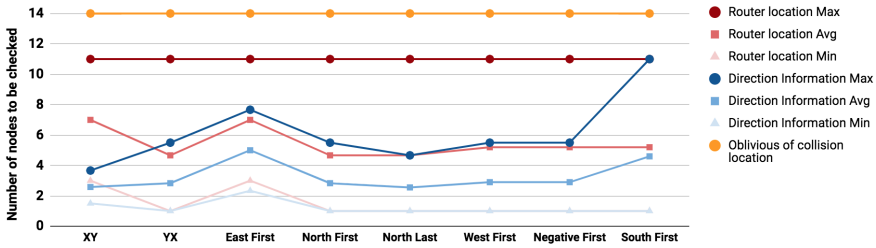



**Figure 9.** Average number of nodes to be checked under different routing algorithms (Sensitive path 3).

Compared to oblivious search (marked in orange in Figures 7–9), the use of router locations will reduce the search space in the worst case effort by 52%, 42%, and 21% for the previously illustrated sensitive paths 1, 2, and 3. Similarly the use of directions will further reduce the worst case effort by 63%, 69%, and 56%. The use of collision direction in locating the attacker source reduces the maximum effort of the location based solution by 24%, 48%, and 44% for paths 1, 2, and 3 respectively and the average effort by 30.6%, 40%, and 39%.

## 6. Proposed Router Architecture

As explained in the previous section, knowing the location of the router where the malicious packets collide with the sensitive packets and the direction from where they entered the sensitive path drastically narrows down the attacker's source search space. In this section we introduce the two proposed approaches: One that targets the detection of the collision point router, and another which also targets the detection of the direction from which the colliding packets enter the sensitive path. Furthermore, both approaches are compatible with the same router architecture. The proposed architecture extends the original (presented in Project Bonfire, Figure 1a), by the addition of *DoS monitors* between the input ports and their corresponding FIFOs, as depicted in Figure 10. Apart from the packets, the DoS monitors also receive information regarding all the output requests and grants, which they process according to the chosen approach. The details of these methods are explained in the following subsections.

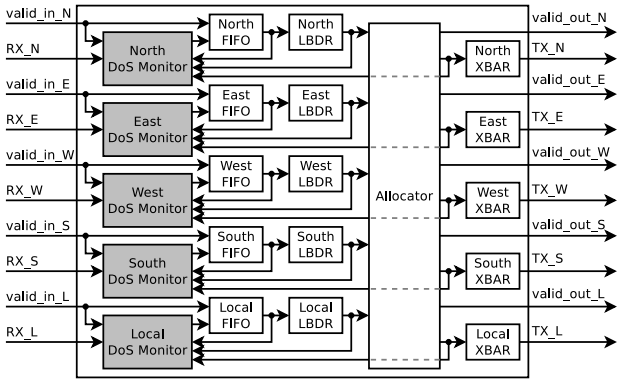
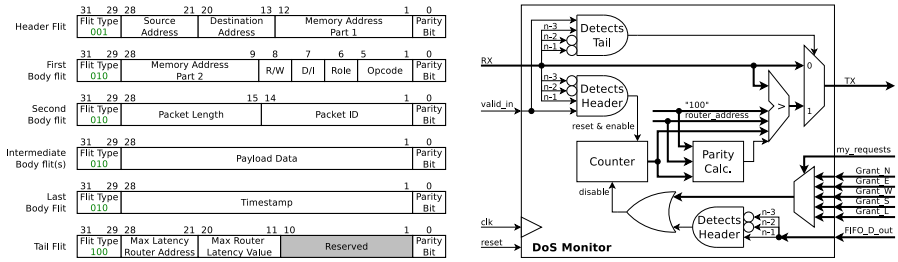



**Figure 10.** Proposed router architecture with Denial of Service (DoS) detection.

### 6.1. Collision Point Router Detection (CPRD) Architecture

The goal of this approach is not only to detect a DoS attack, but also the router where the attacker's packets collide with the sensitive traffic. Hence, this architecture is called Collision Point Router Detection (CPRD) (The code is available for download at [29]). We propose two main modifications to the Bonfire NoC project [19]: (i) addition of the local and end-to-end delay and router information to the packet's tail, and (ii) introduction of the monitors in the routers.

As shown in Figure 11a (in comparison to Figure 1b), the structure of the communication packets has been modified by adding the packet generation time stamp to the *Last Body Flit*. This time-stamp is introduced to enable the system to calculate end-to-end latency of each packet, in order to detect the existence of a DoS attack. Additionally, the last flit of the packet (i.e., the *Tail Flit*) now carries the address of the router where the packet waited the most, while its required output was busy (i.e., *Max Latency Router Address*), and the amount of clock cycles it waited (i.e., *Max Router Latency Value*). This information is evaluated on every router by a proposed DoS Monitor, and updated in case the waiting time in the current router is longer than the one stored in the packet.




(**a**) CPRD packet structure (**b**) CPRD DoS monitor architecture

**Figure 11.** DoS detection using Collision Point Router Detection (CPRD).

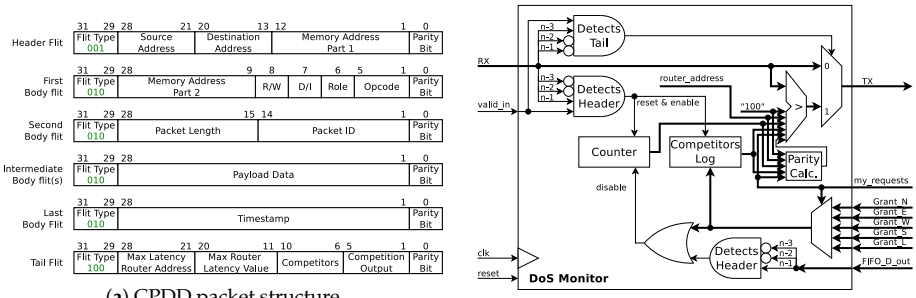
Figure 10 depicts the addition of the proposed *CPRD DoS Monitor* to the router's data path. By adding the CPRD DoS monitor before each of the five FIFOs of each router, a NoC is able to distributively monitor DoS attacks and to locate the router where the attacker's traffic enters the sensitive communication path. Figure 11b depicts the internal architecture of the proposed *CPRD DoS Monitor*. Once a new Header flit arrives at the input of the FIFO, the CPRD monitor starts a 10-bit counter, which is stopped when the Header flit leaves the FIFO's output. The counter is incremented on each clock cycle while a packet is waiting to be forwarded, however, only while other packets are being forwarded through its required output port. This is achieved by monitoring the grants issued by the router's allocator circuit and filtered by the request sent by the LBDR circuit. Furthermore, once the tail flit of the packet arrives, the counter value is compared against the *Max Router Latency Value* stored in the tail flit. If the counter's value is larger than the packet's *Max Router Latency Value*, the monitor replaces *Max Router Latency Value* and *Max Latency Router Address* with the new values and updates the parity bit for the tail flit.

The firmware of the MPSoC-powered IoT device will calculate the packets' end-to-end latency upon their arrival using the packets' generation time-stamp. Later, this value is compared with the maximum expected network latency. If the calculated packet latency is out of the acceptable latency range, the *Max Latency Router Address* and the *Max Router Latency Value* are also retrieved from the packet and analyzed together with previous DoS suspicion reports. Finally, when decided that a DoS attack is taking place, the firmware can localize and reset the PE identified as the source of the attack, thus eliminating its malicious traffic generation.

### *6.2. Collision Point Direction Detection (CPDD) Architecture*

The second monitoring architecture works similar to the CPRD monitor, but also extends it to report the input direction of the malicious traffic through which it enters the collision point router. The packet format has been slightly changed in order to include the inputs competing to enter the sensitive path and the output for which they competed (see Figure 12a). Figure 12b shows the internal structure of the CPDD DoS monitor. The router uses the same structure as the one shown in Figure 10. The most important task is to filter the requests sent to the Arbiter based on the issued requests from the input routing logic, once the requests from the other ports are identified. In case the latency value

stored in the packet's tail flit is less than the value of the local latency counter, the monitor updates the tail flit not only with the router address and the waiting time, but also the competitors and the output for which they compete. With the purpose of registering grants, given for the required output, while a packet is waiting to be forward, the CPDD monitor is equipped with a 5-bit register (*a.k.a.* the Competitors log).




(**a**) CPDD packet structure

(**b**) CPDD DoS monitor architecture

**Figure 12.** DoS detection using Collision Point Direction Detection (CPDD).

Once a sensitive packet reaches its destination, equal to the CPRD approach, the firmware of the MPSoC-powered IoT device calculates the end-to-end delay based on the timestamp contained in the packet. Provided that the calculated value exceeds the threshold, the collision point is determined. However, for the CPDD approach, the firmware will also extract the direction information to narrow down the suspects list and localize the compromised PE.

## 7. Experimental Results

In order to evaluate the performance of the proposed Distributed DoS Detection systems presented in Section 6 (i.e., CPRD and CPDD), they have been implemented on the RTL level and integrated to the Bonfire framework [19]. Traffic generators included in the Bonfire platform were leveraged for simulating normal traffic and DoS attacks on $4 \times 4$ mesh NoC-based MPSoC scenarios. Moreover, the network routers (applying either XY- or WestFisrt-routing algorithm) use a credit base flow control with fair Round-Robin arbitration (on packet level) and utilize Wormhole switching with 4-flit deep FIFOs. Furthermore, random traffic packets were transmitted with a length of 10 flits and a PIR of 0.01 (i.e., one packet every 100 clock cycles). The length and PIR of the sensitive and attacker packets, as well as their source and destination, vary according to the purpose of each scenario. Details of the scenario setup of the experiments and the results of simulations for CPRD are provided in Section 7.1 and for CPDD in Section 7.2. Additionally, an evaluation of the overhead of the proposed mechanisms in terms of area, critical-path delay, and power is presented in Section 7.3.

### *7.1. Collision Point Router Detection (CPRD)*

The experiments presented in this section target the analysis of the DoS attack from two sides: (i) from the point of view of the attacker in order to maximize the effectiveness of the attack, and (ii) from the point of view of the network protection, evaluating different metrics that can be leveraged for detecting an attack and qualifying the effectiveness of the CPRD mechanism. Section 7.1.1 presents both sides of the DoS attack, considering a scenario that combines different PIR values for the attacker as well as for the sensitive traffic. Section 7.1.2 goes further and evaluates the effects caused by attacks with different packet lengths and from different locations within the NoC.

#### 7.1.1. Effect of the Attacker's PIR

In order to evaluate the effect of the attacker's PIR value on the effectiveness of DoS attacks, the scenario described in Figure 13 has been adopted. The sensitive packets go from router 12 (marked in blue) to the router 3 (marked in green) directed by the XY routing algorithm. This flow is marked with the blue arrow in the figure. The attacker, located on router 15 (marked in red), attempts to send packets to the same destination (router 3). The attacker's packets' flow has been marked by a red arrow on the figure. Moreover, the attacker's and sensitive packets collide in router 15 (marked with an orange circle), while attempting to be forwarded to their destinations through the north port of the router.

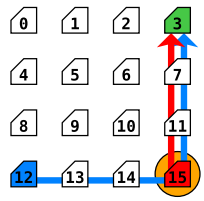



**Figure 13.** Experiment scenario for attacker's Packet Injection Rates (PIR) efficiency.

Within this analysis, simulations were done considering different packet injection rates (PIR) for sensitive packets (PIR_S) and attack packets (PIR_A). The network's random traffic was generated with a PIR_R of 0.01 (one packet every 100 clock cycles). Each experiment was performed for 20 pseudo-random simulation seeds to provide uniform results. The network's random traffic and the sensitive node's traffic have been generated with a packet length of 10 flits (named (PL_R) and PL_S respectively). In the scenarios where an attack attempt was present, three different packet lengths were considered for the attacker's packets (PL_A): 10, 30, and 50 flits.

As mentioned in Section 4, an attacker can identify the success/effectiveness of the attack by measuring its own throughput [23]. Table 4 lists results for different combinations of PIR_S and PIR_A from the attacker's point of view. The first and third column of Table 4, show the attacker's intended and effective PIR, respectively. It can be seen that as the attacker's intended PIR value is increased, the gap between the attacker's intended and Effective PIRs widens. For example, in the case where the attacker's intended PIR is 0.01, the attacker is trying to send a 30-flit-length packet every 100 clock cycles. However, the effective PIR shows an insertion rate of approximately one packet every 140 clock cycles. The deviation of the attacker's effective PIR from the intended value is expressed as a growth percentage in the fourth column of Table 4. For the purpose of this work, we established that an attacker's PIR_A variation (fourth column of Table 4) of 10% is needed to consider the attack effective.

**Table 4.** Attack effectiveness (S_A: 15, PL_A: 30, PL_S: 10, PL_R: 10, PIR_R: 0.01).

| Attacker PIR_A | Sensitive PIR_S | Attacker Effective PIR_A | Attacker PIR_A Deviation (%) | Attack Effective |
|---|---|---|---|---|
| 0.003 | 0.003 | 0.003 | 0 | ✗ |
| 0.003 | 0.01 | 0.003 | 0 | ✗ |
| 0.003 | 0.03 | 0.003 | 0 | ✗ |
| 0.01 | 0.003 | 0.0071 | 29 | ✓ |
| 0.01 | 0.01 | 0.0071 | 29 | ✓ |
| 0.01 | 0.03 | 0.0070 | 30 | ✓ |
| 0.03 | 0.003 | 0.0086 | 71.33 | ✓ |
| 0.03 | 0.01 | 0.0085 | 71.67 | ✓ |
| 0.03 | 0.03 | 0.0086 | 71.33 | ✓ |

As mentioned in Section 6, each packet carries (in the last body flit) a time-stamp of the instant when the packet is generated. This time-stamp is used for calculating the End-to-End communication

latency upon arrival to the destination node. Table 5 lists the mean End-to-End delay values of the sensitive packets for different PIR_S values in absence of an attacker. This mean value and the sample standard deviation (SSD) are used for calculating a threshold $T$ as shown in (1), and such a threshold for diagnosing DoS attacks.

$$T = mean\_delay + 0.5 \times SSD \quad (1)$$

**Table 5.** End-to-End delay under no attack (PL_S: 10, PL_R: 10, PIR_R: 0.01).

| Sensitive PIR_S | Mean End-to-End Delay (Clock Cycles) | Sample Standard Deviation (Clock Cycles) | Detection Threshold (Clock Cycles) |
|---|---|---|---|
| 0.003 | 88.9056 | 17.3167 | 97.5640 |
| 0.01 | 162.0356 | 43.6219 | 183.8466 |
| 0.03 | 915.5529 | 405.9398 | 1118.5228 |

Furthermore, Table 6 lists the results of the experiments from the network protection's point of view for the same scenario where the attacker is located at router 15 (Figure 13). Since a DoS attack was present, the end-to-end delay reached values greater than the threshold $T$ (see Equation (1) and Table 5), enabling the diagnosis mechanism to detect the attack. However, for a PIR_A of 0.01, the collision point was not determined. This because the traffic entering from other inputs and requiring the same output was more or less equal to the one from the attacker. On the other hand, for PIR_A configurations, where the attacker noticed success by monitoring its own throughput (i.e., PIR_A $\geq$ 0.01), the proposed DoS CPRD mechanism managed not only to detect the attack, but also the point where the malicious traffic intercepted the sensitive path. For the scenarios where the collision point router was found, the last column of the table lists the detection confidence. In some cases the confidence is not complete due to the randomness of the additional traffic.

**Table 6.** Attack detection (S_A: 15, PL_A: 30, PL_S: 10, PL_R: 10, PIR_R: 0.01).

| Attacker PIR_A | Sensitive PIR_S | End-to-End Delay (Clock cycles) | Attack Detected by System Manager | Attack Detected by Attacker | Collision Point Detection Confidence |
|---|---|---|---|---|---|
| 0.003 | 0.003 | 181.331 | ✓ | ✗ | ✗ |
| 0.003 | 0.01 | 765.047 | ✓ | ✗ | ✗ |
| 0.003 | 0.03 | 2679.22 | ✓ | ✗ | ✗ |
| 0.01 | 0.003 | 656.859 | ✓ | ✓ | 1 |
| 0.01 | 0.01 | 3346.42 | ✓ | ✓ | 0.8 |
| 0.01 | 0.03 | 4336.19 | ✓ | ✓ | 0.7 |
| 0.03 | 0.003 | 308.823 | ✓ | ✓ | 1 |
| 0.03 | 0.01 | 2937.8 | ✓ | ✓ | 1 |
| 0.03 | 0.03 | 4318.1 | ✓ | ✓ | 0.95 |

#### 7.1.2. Effect of the Attacker's Packet Length and the Location

In order to find configurations that would maximize the success of an attack, experiments were done for four different PL_A values (i.e., 10, 20, 30, and 50 flits), as well as placing the attack source in different routers of the NoC (router connected to the infected PE) and for the three scenarios presented in Section 5.1; Figure 14a–d shows some of the scenarios with different sources of attacks (symbolized with the red router, and specified in the label of each sub-figure). The malicious traffic runs from the scenario specific source to the same destination as the sensitive traffic (router 3); as in the scenario from the previous section, the red arrow connecting these two routers represents such a path and the orange circle indicates the collision point, where DoS traffic collides with the path of the sensitive traffic. Finally, as depicted in Figure 14e, depending on the selected attack source, the malicious path length and the collision point to the sensitive path vary. In the cases where the attack source is directly in the sensitive path, the source and collision point routers are the same. For such experiments, PL_S and PL_R were set to 10 flits, and all the PIRs were set to 0.01 (i.e., PIR_A = PIR_S = PIR_R = 0.01).

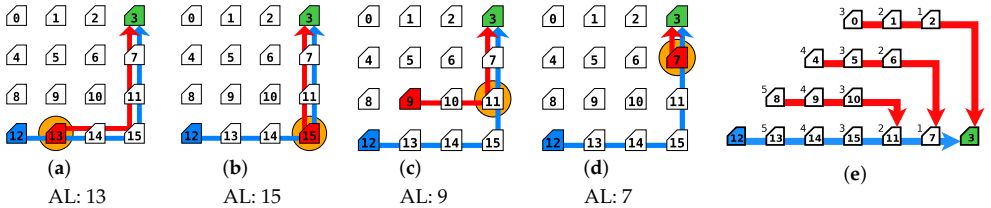



**Figure 14.** Experimental examples (**a**–**d**) for attack scenario 1 and (**e**) illustration of different malicious path length and the collision point based on the location of the attacker.

End-to-End mean delay values of the sensitive packets for each of the scenarios are depicted in Figure 14 and more, are presented in Figure 15. As expected, the results show that the trend of End-to-End delay of the sensitive packets rises proportionally to the PL_A value, achieving in most cases the highest mean delay values for the 50-flit PL_A configurations. This is due to the fact that longer packets manage to retain the grant of a router output longer, preventing other packets of being forwarded. Regarding the location of the attacker, as shown in Figure 15, the highest sensitive End-to-End transmission delays were caused when injecting the malicious traffic into the routers closest to the sensitive destination. This is due to the propagation of arbitration fairness of each router. Consider that the attacker can block 50% of the traffic on the point of insertion (in case its only competing with one other input direction). If this point of insertion is close to the source of the sensitive data, the attacker can successfully block 50% of the sensitive packets. However, if the attacker is close to the destination node, it can block 50% of the destination stream where the sensitive data flow is a part of it. Hence, the effect of the attack gets amplified by the distance of the attacker from the source.

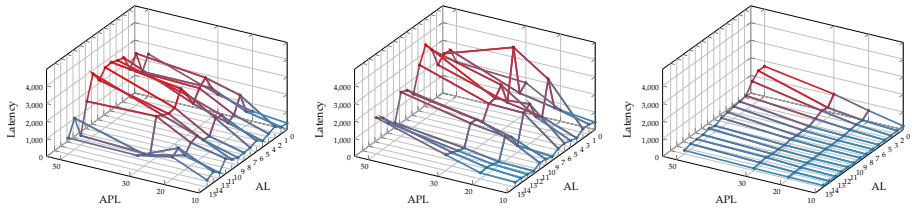



**Figure 15.** End-to-End latency vs attacker's packet length for different attack sources (XY Routing) for (**a**) Scenario 1, (**b**) Scenario 2, and (**c**) Scenario 3.

### *7.2. Collision Point Direction Detection (CPDD)*

Section 5 explained the benefit of identifying the port through which malicious traffic intercepts the sensitive path. This section presents the direction detection success and misses of the CPDD mechanism for the scenarios described in Section 5.1. Simulations were executed considering a PIR_A value of 0.03 and PIR_S = PIR_R = 0.01. Each experiment was performed for 20 pseudo-random simulation seeds to provide uniform results. Additionally, four attacker packet lengths were adopted (i.e., 10, 20, 30, and 50 flits), while the network's random traffic and the sensitive node's traffic generated packets with the length of 10 (i.e., PL_R = PL_S = 10). Moreover, experiments were executed considering not only a deterministic routing algorithm, but also an adaptive routing algorithm. Results for each routing algorithm are presented separately in Sections 7.2.1 and 7.2.2, respectively.

### 7.2.1. Analysis of the Proposed Method under Deterministic Routing

Figure 16 summarizes the obtained detection results for scenarios depicted in Figures 4–6 under XY routing. Each row lists the detection effectiveness of the collision point router and input direction of the malicious packets for every possible attack source. Also, each pair of columns group the router and direction detection results for each of the attacker packet length values.

| | APL = 10 | | APL = 20 | | APL = 30 | | APL = 50 | |
|---|---|---|---|---|---|---|---|---|
| AS | CPRD | CPDD | CPRD | CPDD | CPRD | CPDD | CPRD | CPDD |
| 0 | ✗ | – | ✗ | – | ✓ | ✓ | ✓ | ✓ |
| 1 | ✗ | – | ✗ | – | ✓ | ✓ | ✓ | ✓ |
| 2 | ✗ | – | ✗ | – | ✓ | ✓ | ✓ | ✓ |
| 3 | – | – | – | – | – | – | – | – |
| 4 | ✗ | – | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 5 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 6 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 7 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 8 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 9 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 10 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 11 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 12 | – | – | – | – | – | – | – | – |
| 13 | ✗ | – | ✗ | – | ✓ | ✓ | ✓ | ✓ |
| 14 | ✗ | – | ✗ | – | ✓ | ✓ | ✓ | ✓ |
| 15 | ✗ | – | ✗ | – | ✓ | ✓ | ✓ | ✓ |

(**a**)

| | APL = 10 | | APL = 20 | | APL = 30 | | APL = 50 | |
|---|---|---|---|---|---|---|---|---|
| AS | CPRD | CPDD | CPRD | CPDD | CPRD | CPDD | CPRD | CPDD |
| 0 | ✗ | – | ✗ | – | ✓ | ✓ | ✓ | ✓ |
| 1 | ✗ | – | ✗ | – | ✓ | ✓ | ✓ | ✓ |
| 2 | – | – | – | – | – | – | – | – |
| 3 | ✗ | – | ✗ | – | ✓ | ✓ | ✓ | ✓ |
| 4 | ✗ | – | ✗ | – | ✓ | ✓ | ✓ | ✓ |
| 5 | ✗ | – | ✗ | – | ✓ | ✓ | ✓ | ✓ |
| 6 | ✗ | – | ✗ | – | ✓ | ✓ | ✓ | ✓ |
| 7 | ✗ | – | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 8 | – | – | – | – | – | – | – | – |
| 9 | ✗ | – | ✗ | – | ✗ | – | ✓ | ✓ |
| 10 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 11 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 12 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 13 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 14 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 15 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |

(**b**)

| | APL = 10 | | APL = 20 | | APL = 30 | | APL = 50 | |
|---|---|---|---|---|---|---|---|---|
| AS | CPRD | CPDD | CPRD | CPDD | CPRD | CPDD | CPRD | CPDD |
| 0 | ✗ | – | ✗ | – | ✓ | ✓ | ✓ | ✓ |
| 1 | – | – | – | – | – | – | – | – |
| 2 | ✗ | – | ✗ | – | ✓ | ✓ | ✓ | ✓ |
| 3 | ✗ | – | ✗ | – | ✓ | ✓ | ✓ | ✓ |
| 4 | – | – | – | – | – | – | – | – |
| 5 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ |
| 6 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 7 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 8 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 9 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 10 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 11 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 12 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 13 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 14 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 15 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |

(**c**)

**Figure 16.** Effectiveness of the proposed architecture in establishing the attacker search space for XY routing of (**a**) Scenario 1, (**b**) Scenario 2, and (**c**) Scenario 3.

Results showed that the CPDD managed to detect the input direction for almost all the scenarios where the collision point router was found. Also, once again, due to the wormhole switching, that as the packet length of the attacker was increased, it became easier to detect the collision point router.

### 7.2.2. Analysis of the Proposed Method under Adaptive Routing

A similar analysis to the one shown in the previous section has been performed on the system adopting the West-First adaptive routing algorithm. Figure 17 summarizes the obtained detection results when adopting the West-First routing in the scenarios depicted in Figures 4–6. Each row lists the detection effectiveness of the collision point router and input direction of the malicious packets for every possible attack source. Also, each pair of columns group the router and direction detection results for each of the attacker packet length values.

| | APL = 10 | | APL = 20 | | APL = 30 | | APL = 50 | |
|---|---|---|---|---|---|---|---|---|
| AS | CPRD | CPDD | CPRD | CPDD | CPRD | CPDD | CPRD | CPDD |
| 0 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 1 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 2 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 3 | – | – | – | – | – | – | – | – |
| 4 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 5 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 6 | ✓ | ✗ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 7 | ✓ | ✗ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 8 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 9 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 10 | ✓ | ✗ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 11 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 12 | – | – | – | – | – | – | – | – |
| 13 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 14 | ✓ | ✗ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 15 | ✓ | ✗ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |

(**a**)

| | APL = 10 | | APL = 20 | | APL = 30 | | APL = 50 | |
|---|---|---|---|---|---|---|---|---|
| AS | CPRD | CPDD | CPRD | CPDD | CPRD | CPDD | CPRD | CPDD |
| 0 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 1 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 2 | – | – | – | – | – | – | – | – |
| 3 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 4 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 5 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 6 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 7 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 8 | – | – | – | – | – | – | – | – |
| 9 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 10 | ✓ | ✗ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 11 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 12 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 13 | ✓ | ✓ | ✓ | ✗ | ✓ | ✓ | ✓ | ✓ |
| 14 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 15 | ✓ | ✗ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |

(**b**)

| | APL = 10 | | APL = 20 | | APL = 30 | | APL = 50 | |
|---|---|---|---|---|---|---|---|---|
| AS | CPRD | CPDD | CPRD | CPDD | CPRD | CPDD | CPRD | CPDD |
| 0 | ✓ | ✗ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 1 | – | – | – | – | – | – | – | – |
| 2 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 3 | ✓ | ✓ | ✗ | – | ✓ | ✓ | ✓ | ✓ |
| 4 | – | – | – | – | – | – | – | – |
| 5 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 6 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 7 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 8 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 9 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 10 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 11 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 12 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 13 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 14 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 15 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |

(**c**)

**Figure 17.** Effectiveness of the proposed architecture in establishing the attacker search space for West-First routing of (**a**) Scenario 1, (**b**) Scenario 2, and (**c**) Scenario 3.

The results for this set of experiments show a better detection efficiency of the attacker's location compared to a system using XY routing, however, since traffic between two routers may have more than one minimal path, the size of the search space for the attacker is much larger (please refer to Figures 7–9).

### 7.3. Overhead Evaluation

The proposed architectures (CPRD and CPDD Routers) were synthesized using the 0.18 µm AMS library and Synopsys design vision at 200 MHz. Area overhead and critical path delay of the proposed architecture compared to the baseline architecture are reported in Table 7. The critical path delay overhead of the CPRD method is negligible and the proposed CPRD monitors only add 17% area overhead to the minimalist router area (each CPRD monitor only adds 3.4% overhead to the router's area). Its important to note that the main contribution to area overhead is from the inclusion of the counter register. The width of the counters can be adjusted based on the application. The CPDD router applies 23.2% area overhead to the baseline router. In order to put the area overheads into perspective, we will consider the overheads on a 4 $mm^2$ chip. For an SoC using a $4 \times 4$ mesh NoC, the CPRD and CPDD would impose 0.4% and 0.5% overhead to the system respectively.

**Table 7.** Area and critical path delay overhead.

| Router | Area | | | | Critical Path Delay (ns) |
|---|---|---|---|---|---|
| | Sequential ($\mu m^2$) | Combinational ($\mu m^2$) | Total ($\mu m^2$) | Overhead (%) | |
| Baseline | 48378.7 | 42669.0 | 91047.8 | – | 4.82 |
| CPRD Router | 52033.7 | 55129.9 | 107163.7 | 17.7% | 4.8 |
| CPDD Router | 58313.6 | 53873.9 | 112187.5 | 23.2% | 4.79 |

*Power analysis:* the power consumption of the proposed methods (CPRD and CPDD routers) and the baseline architecture have been evaluated for random uniform traffic with a packet injection rate of 0.01 (without presence of attacker). The results of these experiments are reported in Table 8. The results show that the CPRD approach induces 5% power overhead and the CPDD approach adds 9.4% to the baseline router.

**Table 8.** Power overhead.

| Router | Switching Power (mW) | Internal Power (mW) | Leakage Power (mW) | Total Power (mW) | Total Overhead (%) |
|---|---|---|---|---|---|
| Baseline | 0.151 | 3.663 | 0.261 | 3.814 | – |
| CPRD Router | 0.273 | 3.374 | 0.327 | 4.008 | 5% |
| CPDD Router | 0.269 | 3.905 | 0.346 | 4.174 | 9.4% |

## 8. Conclusions

Network-on-Chip solutions have become the central communication infrastructure of the modern MPSoCs. However, Denial of Service (DoS) attacks have been shown as an important threat to NoC integrity. Hence, it is of utmost importance to detect the occurrence of such attacks in the system, and also to localize the attacker in order to neutralize its effects. To this end, this paper provides two main contributions. First we analyze the effect of obtaining additional information about the collision location and the incoming direction of the attacker's flow in the collision point with the sensitive data on localizing the attacker. Second we propose two distributed DoS detection schemes for measuring the performance degradation of sensitive data transmissions under Denial of Service attacks and for detecting the collision point and direction of the collision of the DoS packets into the sensitive path. We perform an exploration of the effect of different attack configurations targeting the sensitive traffic including different packet lengths, packet injection rates, and attack sources. Our experimental results show that longer attacker packets, intercepting the sensitive path closer to its destination, cause a greater effect of the attack. We have also shown that we can almost in all cases identify the direction of incoming attack flows.

**Author Contributions:** Conceptualization, C.G.C., S.P.A., T.H. and J.S.; Data curation, C.G.C. and S.P.A.; Formal analysis, C.G.C., S.P.A., T.H. and J.S.; Funding acquisition, T.H. and J.S.; Investigation, C.G.C., S.P.A., T.H. and J.S.; Methodology, C.G.C., S.P.A., T.H. and J.S.; Project administration, T.H.; Resources, S.P.A.; Software, C.G.C. and S.P.A.; Supervision, T.H. and J.S.; Validation, C.G.C. and S.P.A.; Visualization, C.G.C. and S.P.A.; Writing—original draft, C.G.C., S.P.A., T.H. and J.S.; Writing—review & editing, T.H. and J.S.

**Funding:** This work was partially funded by the German Academic Exchange Service (DAAD) and by the German Federal Ministry of Education and Research (BMBF), grant number 01IS160253 (ARAMiS II) and also supported partly by the ETAG IUT19-1 grant.

**Acknowledgments:** The authors thank TalTech (Tallinn University of Technology), Department of Computer Engineering, for providing a high performance server infrastructure as resource for the extensive simulations needed to prove the evidence of the presented research.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Ericsson Mobility Report. Available online: https://www.ericsson.com/assets/local/mobility-report/documents/2016/ericsson-mobility-report-june-2016.pdf (accessed on 16 January 2019).
2. The Internet of Things: Mapping the Value Beyond the Hype, 2015. Available online: https://www.mckinsey.de/#energiewende (accessed on 16 January 2019).
3. Azad, S.P.; Niazmand, B.; Ellervee, P.; Raik, J.; Jervan, G.; Hollstein, T. *SoCDep*$^2$: A framework for dependable task deployment on many-core systems under mixed-criticality constraints. In Proceedings of the 2016 11th International Symposium on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC), Tallinn, Estonia, 27–29 June 2016; pp. 1–6.
4. Hollstein, T.; Azad, S.P.; Kogge, T.; Ying, H.; Hofmann, K. NoCDepend: A Flexible and Scalable Dependability Technique for 3D Networks-on-Chip. In Proceedings of the 2015 IEEE 18th International Symposium on Design and Diagnostics of Electronic Circuits Systems, Belgrade, Serbia, 22–24 April 2015; pp. 75–78.
5. Worldwide Infrastructure Security Report. Available online: https://pages.arbornetworks.com/rs/082-kna-087/images/12th_worldwide_infrastructure_security_report.pdf (accessed on 16 January 2019).
6. Fang, D.; Li, H.; Han, J.; Zeng, X. Robustness analysis of mesh-based network-on-chip architecture under flooding-based denial of service attacks. In Proceedings of the 2013 IEEE Eighth International Conference on IEEE Networking, Architecture and Storage (NAS), Xi'an, China, 17–19 July 2013; pp. 178–186.
7. Evain, S.; Diguet, J.P. From NoC security analysis to design solutions. In Proceedings of the IEEE Workshop on Signal Processing Systems Design and Implementation, Athens, Greece, 2–4 November 2005; pp. 166–171.
8. Sepúlveda, J.; Strum, M.; Chau, W. An Hybrid Switching Approach for NoC-Based Systems to avoid Denial-of-Service SoC Attacks. In Proceedings of the 16th Iberchip Wksp (IWS 2010), Iguaczú Falls, Brazil, 23–25 February 2010; pp. 23–25.
9. Sepúlveda, J.; Flórez, D.; Gogniat, G. Efficient and flexible NoC-based group communication for secure MPSoCs. In Proceedings of the 2015 International Conference on ReConFigurable Computing and FPGAs (ReConFig), Mexico City, Mexico, 7–9 December 2015; pp. 1–6.
10. Fiorin, L.; Palermo, G.; Silvano, C. A Security Monitoring Service for NoCs. In *Proceedings of the 6th IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis*; ACM: New York, NY, USA, 2008; pp. 197–202.
11. Sepulveda, M.J.; Diguet, J.P.; Strum, M.; Gogniat, G. NoC-Based Protection for SoC Time-Driven Attacks. *IEEE Embedded Syst. Lett.* **2015**, *7*, 7–10. [CrossRef]
12. Caimi, L.L.; Fochi, V.; Wachter, E.; Munhoz, D.; Moraes, F.G. Activation of secure zones in many-core systems with dynamic rerouting. In Proceedings of the 2017 IEEE International Symposium on Circuits and Systems (ISCAS), Baltimore, MD, USA, 28–31 May 2017; pp. 1–4.
13. Sepúlveda, J.; Flórez, D.; Gogniat, G. Reconfigurable security architecture for disrupted protection zones in NoC-based MPSoCs. In Proceedings of the 2015 10th International Symposium on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC), Bremen, Germany, 29 June–1 July 2015; pp. 1–8.
14. JS, R.; Ancajas, D.M.; Chakraborty, K.; Roy, S. Runtime detection of a bandwidth denial attack from a rogue network-on-chip. In Proceedings of the 9th International Symposium on Networks-on-Chip, Vancouver, BC, Canada, 28–30 September 2015; p. 8.

15. Fiorin, L.; Silvano, C.; Sami, M. Security aspects in networks-on-chips: Overview and proposals for secure implementations. In Proceedings of the 2007 10th Euromicro Conference on Digital System Design Architectures, Methods and Tools, Lubeck, Germany, 29–31 August 2007; pp. 539–542.
16. Diguet, J.P.; Evain, S.; Vaslin, R.; Gogniat, G.; Juin, E. NOC-centric Security of Reconfigurable SoC. In Proceedings of the First International Symposium on Networks-on-Chip, Princeton, NJ, USA, 7–9 May 2007; pp. 223–232.
17. Grammatikakis, M.D.; Papadimitriou, K.; Petrakis, P.; Papagrigoriou, A.; Kornaros, G.; Christoforakis, I.; Tomoutzoglou, O.; Tsamis, G.; Coppola, M. Security in MPSoCs: A NoC Firewall and an Evaluation Framework. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* **2015**, *34*, 1344–1357. [CrossRef]
18. Achballah, A.B.; Othman, S.B.; Saoud, S.B. Toward on hardware firewalling of networks-on-chip based systems. In Proceedings of the 2017 International Conference on Advanced Systems and Electric Technologies (IC_ASET), Hammamet, Tunisia, 14–17 January 2017; pp. 7–13.
19. Azad, S.P.; Niazmand, B.; Janson, K.; George, N.; Oyeniran, A.S.; Putkaradze, T.; Kaur, A.; Raik, J.; Jervan, G.; Ubar, R.; Hollstein, T. From online fault detection to fault management in Network-on-Chips: A ground-up approach. In Proceedings of the 2017 IEEE 20th International Symposium on Design and Diagnostics of Electronic Circuits Systems (DDECS), Dresden, Germany, 19–21 April 2017; pp. 48–53.
20. Bonfire Wiki. Available online: https://github.com/Project-Bonfire/Bonfire/wiki (accessed on 16 January 2019).
21. Flich, J.; Duato, J. Logic-Based Distributed Routing for NoCs. *IEEE Comput. Archit. Lett.* **2008**, *7*, 13–16. [CrossRef]
22. Fettweis, G.P. 5G and the future of IoT. In Proceedings of the ESSCIRC Conference 2016 42nd European Solid-State Circuits Conference, Lausanne, Switzerland, 12–15 September 2016; pp. 21–24.
23. Sepulveda, J.; Aboul-Hassan, D.; Sigl, G.; Becker, B.; Sauer, M. Towards the formal verification of security properties of a Network-on-Chip router. In Proceedings of the 2018 IEEE 23rd European Test Symposium (ETS), Bremen, Germany, 28 May–1 June 2018; pp. 1–6.
24. Duato, J.; Yalamanchili, S.; Lionel, N. *Interconnection Networks: An Engineering Approach*; Morgan Kaufmann Publishers Inc.: San Francisco, CA, USA, 2002.
25. Shafiee, A.M.; Montazeri, M.; Nikdast, M. An Innovational Intermittent Algorithm in Networks-On-Chip (NOC). *Int. J. Comput. Electr. Autom. Control Inf. Eng.* **2008**, *2*, 2907–2909.
26. Glass, C.J.; Ni, L.M. The Turn Model for Adaptive Routing. In Proceedings of the 19th Annual International Symposium on Computer Architecture, Queensland, Australia, 19–21 May 1992; pp. 278–287.
27. Mubeen, S.; Kumar, S. Designing Efficient Source Routing for Mesh Topology Network on Chip Platforms. In Proceedings of the 13th Euromicro Conference on Digital System Design: Architectures, Methods and Tools (DSD), Lille, France, 1–3 September 2010; pp. 181–188.
28. Miura, Y.; Shimozono, K.; Watanabe, S.; Matoyama, K. An Adaptive Routing of the 2-D Torus Network Based on Turn Model. In Proceedings of the 2013 First International Symposium on Computing and Networking (CANDAR), Matsuyama, Japan, 4–6 December 2013; pp. 587–591.
29. Secure Bonfire. Available online: https://github.com/Project-Bonfire/Secure_Bonfire/releases/tag/CPRD (accessed on 3 February 2019).


# Appendix 4

**III**

Cesar G. Chaves, Siavoosh P. Azad, Johanna Sepúlveda, and Thomas Hollstein, "Detecting and Mitigating Low-and-Slow DoS Attacks in NoC-based MPSoCs," in *14th International Workshop on Reconfigurable Communication-Centric Systems-on-Chip (ReCoSoC)*, IEEE, July 2019

# Detecting and Mitigating Low-and-Slow DoS Attacks in NoC-based MPSoCs

Cesar G. Chaves*†, Siavoosh Payandeh Azad†, Johanna Sepúlveda‡, Thomas Hollstein*†
*Frankfurt University of Applied Sciences, Frankfurt, Germany
†Tallinn University of Technology, Tallinn, Estonia
‡Technical University of Munich, Munich, Germany
email: cesar.chaves@stud.fra-uas.de, siavoosh.azad@taltech.ee, johanna.sepulveda@tum.de, hollstein@fb2.fra-uas.de

***Abstract*—As Multi-Processor Systems-on-Chip (MPSoCs) permeate the Internet by powering IoT devices, they are exposed to new threats. One major threat is Denial-of-Service (DoS) attacks, which make communication services slow or even unavailable. While mainly studied on desktop and server systems, some DoS attacks on mobile devices and Network-on-Chip (NoC) platforms have also been considered. In the context of NoC-based MPSoC architectures, previous works have explored flooding DoS attacks and their countermeasures, however, these protection techniques are ineffective to mitigate new DoS attacks. Recently, a shift of the network attack paradigm from flooding DoS to Low-and-Slow DoS has been observed. To this end, we present two contributions. First, we demonstrate, for the first time, the impact of Low-and-Slow DoS attacks in NoC environments. Second, we propose a lightweight online monitor able to detect and mitigate these attacks. Results show that our countermeasure is feasible and that it effectively mitigates this new attack. Moreover, since the monitors are placed at the entry points of the network, both, single- and multi-source attacks can be neutralized.**

***Index Terms*—Network on Chip (NoC), Muti-Processor System on Chip (MPSoC), Denial of Service (DoS) attack, Low-and-Slow DoS attack, Distributed monitoring**

## I. Introduction

The ever growing use of MPSoCs by introducing them into the IoT edge cloud and fog computational nodes, increment the security threats targeting these type of architectures. Networks-on-Chip (NoCs) have shown to be a scalable communication infrastructure for the growing number of connected Processing Elements (PEs) in an MPSoC. Such communication structure transports data encapsulated in packets which in turn are divided in to flits (flow control units).

In order for a NoC to be scalable, the incorporated routers must meet special requirements such as: limited hardware resources and high throughput/low latency communications. To this end, designers focus on: i) area optimization by reducing the buffer size of routers and ii) to use of control flow schemes that attempt to forward the packets as soon as they are received. The Wormhole switching mechanism, being widely used in NoCs, addresses both above mentioned objectives. With Wormhole switching, small buffers are used in each router, hence, at a given time, a packet in the network can occupy multiple router buffers.

A whole class of attacks, named Denial of Service (DoS) attacks, attempt to diminish the quality of service of on-chip network communications. Classical attacks target network traffic either: i) directly, where rouge NoCs increase the forwarding delay of packets; or ii) indirectly, where applications running on compromised PEs exploit Wormhole switching by generating traffic that occupies the NoC, preventing legitimate traffic to flow through the network or to forcing it to take longer to reach its destination.

When focusing on the development of a secure network infrastructure, direct DoS attacks to the network traffic become irrelevant, whereas indirect DoS attacks become one of the major concerns. Attacks that flood the network with a large amount of small packets (high packet injection rate) or with packets of larger-than-normal size, called Flooding DoS (FDoS) attacks, have been investigated in NoCs. As shown in Section II-B, countermeasures to such attacks have been proposed, targeting either attack prevention, or attack detection and source localization.

However, NoCs using wormhole switching are also vulnerable to a specific type of attacks referred as Low-rate DoS, Slow DoS, or Low-and-Slow DoS (LSDoS) attacks. Since the minimalist wormhole switching allocates a path for the transmission of a packet until its last flit is transmitted, and typically does not allow interleaving of flits from different packets (unless specific interleaving techniques are employed, as in [1]), this type of attacks can effectively degrade the network performance by injecting short packets with a low flit injection rate or just by the injecting packet fragments. As shown in this paper, the effectiveness on LSDoS attacks is as high as that of FDoS attacks. In this paper we will also provide a novel method for mitigation of LSDoS attacks in NoCs. To the best of our knowledge this is the first analysis of this attack method in the context of NoCs.

Moreover, MPSoCs in the silicon nano-scale era are facing an ongoing wear-out of structures during life-time, therefore, dependable and mission-critical systems must be designed adopting a fault management architecture [2]. Such an architecture includes additional hardware like monitors, sensors, and checkers that allow a detailed assessment of the system's performance. A central system manager (SM), running on one processor core of the MPSoC, provides essential services such as Task Scheduling and Distribution, a system health management service (SHM), and a Security Surveillance and Management (SSM) service. The SHM gathers information and mitigates the effects of faults. The communication between the SHM and the distributed hardware elements is done in accordance to *standard 1687* of the Institute of Electrical and Electronics Engineers (IEEE), also known as IJTAG. Related to security, the SSM collects information about unexpected anomalous behaviour in the network, analyses the situation and decides to employ security countermeasures.

The contributions of this work are as follows: In Section II the state-of-the art for DoS attacks in MPSoCs and Low-

This work was partially funded by the German Academic Exchange Service (DAAD) and by the German Federal Ministry of Education and Research (BMBF), grant number 01IS160253 (ARAMiS II) and also supported partly by the ETAG IUT19-1 grant.

978-1-7281-4770-3/19/$31.00 ©2019 IEEE

and-Slow attacks in normal computer networks will be outlined, followed by a definition and detailed description of the adopted DoS threat model (Section III). An illustration of the practicality of LSDoS attacks in wormhole switching NoCs is outlined in Section IV, including a comparison of FDoS and LSDoS attacks. In Section V an exploration of possible countermeasures is performed combined with the proposal of a novel online monitoring mechanism for detection and mitigation of such type of attacks. The proposed LSDoS countermeasure is deployed at the entry points of the network, dealing with each attack attempt separately at its origin, hence, it can equally mitigate single and multiple source LSDoS attacks. An experimental evaluation of the impact of the proposed LSDoS attacks compared to FDoS attacks, as well as the effectiveness of the proposed countermeasure and its synthesis results are presented in Section V. The experimental results show that LSDoS attacks can cause the same impact of long-size-packet FDoS attacks and that the proposed countermeasure can neutralize these new type of NoC attack with a small area overhead.

## II. Related Work

DoS attacks have been widely researched in computer networks and somewhat in MPSoCs, however, most publications related to MPSoCs focus on rouge NoCs and Flooding DoS (FDoS) attacks. Low-and-Slow DoS (LSDoS) attacks, on the other hand, have been only researched, until now, in the context of computer networks. Section II-B summarizes some FDoS in NoC, and Section II-C LSDoS attacks in computer networks.

### A. DoS attacks caused by rouge NoCs

The authors of [3] consider DoS attacks being executed by a rogue third-party NoC architecture. On detection of a suspicious communication delay between two nodes, the firmware tries to identify if the attack is taking place at the source or at the destination by time-stamping the packets and dividing them into two communication paths. A NoC containing a hardware trojan (HT) that compromises data integrity is presented in [4]. Upon arrival to their destination, packets that do not pass an error detection mechanism are drooped and there retransmission requested, flooding the network on demand. The proposed countermeasure is obfuscating the packet data so that it is not recognizable by the HTs, thus not triggering it.

### B. Flooding DoS attacks in NoCs

Approaches against Flooding DoS (FDoS) attacks in NoCs can be divided into two categories: 1) DoS attack avoidance, and 2) DoS Attack Detection and Recovery.

*1) DoS attack avoidance:* The authors of [5] propose a mitigation strategy based on a hybrid switching routing mechanism. Here for sensitive packets circuit switching is used and packet switching is used otherwise. This approach guarantees predictable latency for sensitive traffic. In [6] DoS and timing attacks are mitigated by the creation of security zones. This proposal ensures that only secure nodes can communicate into a virtual and physical space, re-routed other traffic outside of security zones. An alternative approach is followed by the authors of [7], who propose the use of separate virtual channels for secure and non-secure packets as a countermeasure for bandwidth denial attacks in NoCs. However, this approach suffers from the hardware overhead of additional buffers used for virtual channels.

*2) DoS Attack Detection and Recovery:* Fiorin et al. [8], present an overview of different attacks in Networks-on-Chip. The authors propose the use of buffer occupancy monitors for detection of traffic anomalies in the network without providing details of the technique. In a subsequent publication [9], Fiorin et al. propose the implementation of denial of service probes in the network Interfaces. This approach detects DoS attacks outgoing from a processing elements being connected to a network interface by monitoring deviations of the average bandwidth from expected values determined during design time. This method does not actively monitor the latency of the sensitive packets in the network. In [10], Diguet et al. cope with DoS attacks by monitoring live-lock occurrence in the network. This approach is based on source-routing, which imposes considerable overhead to the packet size and doesn't allow adaptive routing with respect to current traffic volume. Grammatikakis et al. [11] tackle distributed DoS attacks by application of a firewall enforcing configurable access rules in the network interface. The security risk is defined by the product of frequency and magnitude of losses (by dropping the packets at the Network Interface). Although such a firewall prevents the DoS attack from being effective since it has been designed to protect the destination PE (i.e., the on-chip memory), it does neither detect the source of the attack nor does it kill the attack-related traffic congesting the network. In [12], Achballah et al. focus on the computation of occupation time of physical links in terms of clock cycles. The occupancy of the physical link is compared to an expected value. In case of the link occupation time exceeding this expectation, the transaction will be flushed and a notification will be sent to the system manager. In this approach the decision is made locally in the router, which may lead to undesirable packet dropping. The authors in [13] study a DoS attack caused by an infected processing element that floods the network with useless packets at a greater than normal packet injection rate. Distributed monitors detect the attack by comparing the amount of received packets through each router input port in a given time interval. Once the attack has been detected, additional packets are sent for localizing the source of the attack. The authors of [14] propose *DoS Attack Detection and Recovery* approaches, which analyze communication degradation in order to reveal DoS attacks. After detection of an attack, the described scalable distributed monitoring architectures identify attack suspects such that attackers can be located and countermeasures can be applied.

### C. Low-and-slow DoS attacks in computer networks

As mentioned before, LSDoS attacks (also called Low-rate DoS and Slow DoS attacks) had not been studied in the context of NoC until now, however, they have been analyzed in different stages of computer networks. In [15], the authors present Slow DoS attacks that target HTTP servers, e.g. the Slowloris and the Slow Read DoS attacks. A Slowloris attack is implemented by sending legitimate but incomplete HTTP requests to the server. Once a connection has been established, it will be locked until a server timeout is triggered. After the connection has been released, the attacker will repeat the attack. Additionally, since this attack requires little bandwidth, a same attacker can execute many simultaneous Slowloris attacks. In contrast, when attempting a Slow Read DoS attack, the attacker sends a complete legitimate HTTP requests, but

specifies a small client side reception buffer, forcing the server to respond in a low-rate manner, thus locking the connection longer.

The Jellyfish delay variance (JFDV) attack is presented in [16]. It adds a random delay between packets, which leads to a longer allocation time of resources and subsequently, it can be confused as the normal behavior of a congested network. In such scheme, the transmission and reception time of each packet are used for calculating its end-to-end delay, which is compared to a normal behavior time threshold to identify JFDV attacks. At the same time, the congestion of the network is monitored so that a JFDV attack detection may only be signaled when the end-to-end delay is greater then the threshold in a non-congested network situation.

## III. Threat Model

MPSoCs with a heterogeneous array of Processing Elements (PEs) provide programmability and parallelism, yielding flexibility, processing performance and power efficiency, which can be leveraged for minimizing the latency of communication links of the edge cloud [17]. Fig. 1 illustrates such scenario, in which the performance of Cloud computing is enhanced by attaching IoT devices that are powered by NoC-based MPSoCs.

As depicted in Fig. 1, MPSoCs are composed of two main structural types of components:

(i) the computational structure, consisting of Processing Elements (PEs) such as: processors, hardware accelerators, memories, peripherals, and other Intellectual Property (IP) hardware cores to process and store information; and

(ii) the MPSoC internal communication structure, performing data exchange between PEs. NoCs are used as communication structure of MPSoCs, integrating a high number of PEs. A NoC transmits data via routers and communication links. In order to achieve an optimized pipelined data transmission characteristics, typically applied routing topologies are mesh-based NoCs (as shown in Fig. 1) or FAT tree-based architectures. Mesh topologies are composed by a set of five-port routers (Local, North, East, South, West).

Computational components are attached to the MPSoC communication architecture via Network Interfaces (NIs). In a typical mesh-based NoC, NIs are connected to the fifth router port being called *local*. Data is transmitted via the NoC in the form of data packets. At the packet injection locations, NIs receive data coming from the PEs, encapsulate it as packets and send it to the router (source router). Subsequently, the router hands it over to one of its neighbor routers according to the defined routing algorithm. All routers will forward the packet applying this routing scheme until the packet reaches its destination. The last router (destination router) directs it to the local port, thus ejecting the packet from the NoC. NIs depacketize the data and forward the information to the connected destination PE.

Even though the attachment of NoC-based MPSoCs to the cloud can be beneficial for the system's overall performance, it exposes them to new security threats, such as DoS attacks.

The characteristics of such a scenario and its vulnerability to DoS attacks will be analyzed in this section, as well as the steps followed by an attacker during execution of a DoS attack and related success conditions.

### A. Attack Scenario

The attack scenario depicted in Fig. 1 was used in [18] and [14] for evaluating FDoS attacks in NoCs. In this paper it is used for comparing FDoS and LSDoS attacks. Such scenario corresponds to a typical IoT environment is shown, where gateways (fog-level concentrators) connect a network of MPSoC-powered IoT devices (edge devices) to the cloud in order to create an enhanced execution environment. In traditional cloud computing execution environments, a user/client submits an application to a cloud service; subsequently, it is executed by a server of the shared infrastructure; finally, the results are sent back to the user. In the proposed scenario, a cloud scheduler or a local domain-internal scheduler can delegate the execution of specific tasks to MPSoC-powered IoT devices. Such tasks will subsequently be locally mapped by an MPSoC task scheduler firmware to a specific PE. However, ungovernable by the schedulers, the execution of one or more of the tasks can impact the NoC throughput, either intentionally (a DoS attack) or unintentionally (a faulty task).

Fig. 2a depicts a DoS attack scenario on a NoC in which sensitive traffic is being transmitted from PE12 to PE3 (a.k.a. the sensitive path, composed by the set of routers and links used to forward the sensitive traffic, which is represented by a green arrow). Additionally, a compromised PE, i.e. PE0, attempts a DoS attack either by transmitting incomplete packets to PE3 or by using a low flit rate (a.k.a. malicious traffic, which is represented by a red arrow). Both traffic flows, sensitive and malicious, meet in Router 3 and compete for the same output port (i.e. the local output). Fig. 2b depicts the arising competition situation at the crossbar of the local output of R3. While the use of a required output is granted to one of its competitors, a sensitive packet will experience a forwarding delay longer than normal, resulting in an overall additional communication delay which is not acceptable for time sensitive applications. It is worth to notice that even

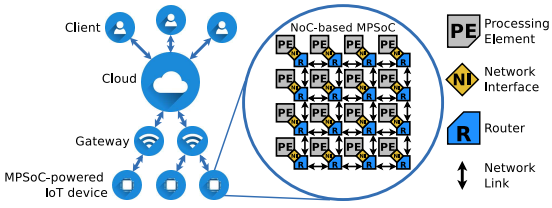
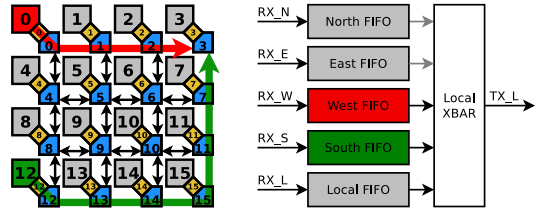



Fig. 1: MPSoC-powered IoT Execution Environment




(a) Sensitive path 12→3, attack path 0→3

(b) Local crossbar router 3

Fig. 2: Attack scenario example considering XY routing

though in the given example in Fig. 2a, the sensitive traffic has only one potential competitor for the local output (i.e. malicious traffic entering via the west input), in other situations it can have at most three competitors (no u-turns are allowed within the router).

### B. Attack Steps

During a DoS attack, an attacker seeks to disrupt the services of a system by degrading a distinct communication channel's performance. In this work we focus on DoS attack methods which are new to the context of NoCs. In low-and-slow DoS attacks, a malicious application, running on any processing element of the MPSoC, tries to disrupt the communication of other system components by sending complete packets slowly or incomplete packets to another PE in the NoC, blocking the transmission of other packets on the occupied communication resources.

In order to perform a Low-and-Slow DoS (LSDoS) attack on a a NoC-based MPSoC, such as the one illustrated by Fig. 1, the attacker, as any other client/user, submits an application to the execution environment. Such an injected application contains at least one task that, according to its characteristics/requirements, will be scheduled to run in a NoC-based MPSoC.

Once the malicious task starts to be executed in a PE, it selects a destination PE to which its packets will be sent (in many cases a memory IP). At this point, the actual attack starts. The application running in the infected PE would classically execute here a FDoS attack, but for this paper, it will execute any of the newly proposed attacks: i) the Incomplete packet transmission (IPT) DoS, in which the PE sends information used for generating the header and packet info flits (see Fig. 3), but provides less flits than specified, thus, blocking the acquired path indefinitely; or ii) the Jellyfish Inter-flit Delay Variance (JFIDV) DoS, in which the PE provides all the flits of the packet, however, adding unnecessary delays between the transmission of each flit.

### C. Attack Success Conditions:

In order to impact a NoC-based MPSoC with a DoS attack, the following conditions must be met:

- A malicious process is able to be executed in a PE inside the MPSoC; As any other user, applications with MPSoC execution requirements submitted by an attacker will be mapped to a PE of an available MPSoC.
- Malicious packets compete with sensitive packets for one and the same output of the crossbar of a router. Assuming that the attacker has no knowledge of the network segments being used by the sensitive traffic, setting long paths for the malicious traffic will increase the probability of colliding with the sensitive traffic.

In this work, we describe a specific exemplary scenario with distinct PEs acting as transmitter or receiver nodes of sensitive, malicious or random traffic. This assures that the malicious traffic will flow in at least one NoC link required by the sensitive communication, thus, forcing a competition for a router crossbar output between the two traffic flows.

## IV. Low-and-Slow DoS attacks in NoCs

In general terms, the objective of any DoS attack is to prevent legitimate users from using normally available resources, or to obstruct such resources in order to degrade the performance of the system [19] (i.e. increasing processing and/or communication times). In the context of NoCs, disregarding physical tampering, a DoS attack is commonly performed by transmitting malicious packets that take possession of dedicated input to output paths in one or more routers for extended periods of time [14]. Flooding DoS (FDoS) attacks in NoCs and their countermeasures have been analyzed in recent research (as detailed in Section II-B).

In order to perform a FDoS attack in a NoC-based MPSoC, an infected PE should either: i) send a greater than normal amount of packets (high packet injection rate (PIR)) [13]; or ii) send abnormally long packets in order to generate congestion on a longer path in the network [20]. The former FDoS attack class can be detected by monitoring the amount of packets sent by a single PE [13], or even being attenuated by implementing a fair Round-Robin arbitration on a packet level, which avoids a PE from monopolizing a router's cross-bar [21].

Even though such countermeasures prove to be effective against the former FDoS type, they are circumvented by long-packet FDoS attacks. Long-packet FDoS can be executed with a smaller amount of packets that block a router's crossbar connection longer than packets sent from legitimate sources. However, the Long-packet FDoS type can be detected if each router compares the size of the packets arriving through their local interface to the size of other packets.

In this work we present for the first time in the NoC context a DoS attack that is immune to the aforementioned FDoS countermeasures. Such an attack, called low-and-slow DoS (LSDoS) attack, takes advantage of the switching method (virtual cut-through or wormhole) used by the NoC, and is capable of achieving the same impact as a long-packet FDoS attack, with packet sizes equal or even smaller than those sent by legitimate PEs.

Generally, the switching methods used in NoCs can be divided into 2 classes: i) approaches that forward packets only after all their flits have been collected (namely store-and-forward switching), and ii) approaches where packets are forwarded as soon as they are received (e.g. virtual-cut-through and wormhole switching) [21]. The store-and-forward switching method ensures that no incomplete packets are forwarded, however, depending on the maximum packet length allowed in the NoC, its implementation will require a large memory area for storage and handling of whole packets (Virtual-cut-through requires memory as well). Another drawback of store-and-forward switching is that it produces a greater end-to-end communication latency, thus packets have to arrive completely to one router before being forwarded to the next router, making the delay proportional to the amount of hops required to reach the destination. In contrast, the wormhole switching method requires less buffer area. Here only a couple of flits are stored while the routing decision is made. However, since a path is exclusively allocated to a single packet, it will only be available to other sources again, when the tail flit has passed. Thus, an attacker can send an incomplete packet (i.e. without a tail flit), performing an IPT DoS attack or stretch the transmission time of small packets by intentionally introducing delays between the transmission of their flits, performing a JFIDV DoS attack.

For a better understanding of the proposed JFIDV LSDoS attack, we adopted the simple packet structure depicted in Fig. 3. The NI implements the protocol, by wrapping the

information generated by the PE into packets, which are then injected to the NoC routers. However, it is the application running in the PE that decides the rate at which the information is sent from the PE to the NI. Therefore, as shown in Fig. 4, with the addition of inter-flit delays ($D_{if}$), the transmission of a normal-sized packet of length $PL_2$ can be stretched to take control of the router cross-bars along a path in the same way as a longer packet of length $PL_1$ would do. Moreover, any of the three variables can be calculated based on the other two following one of the equations (1), (2), or (3). Finally, the equivalence between the FDoS and the JFIDV LSDoS attack is proven by the experimental results presented in Section VI-B.

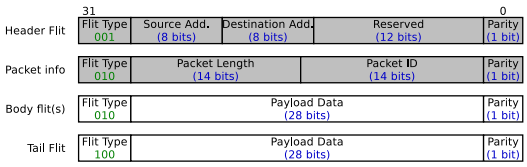



Fig. 3: Packet structure

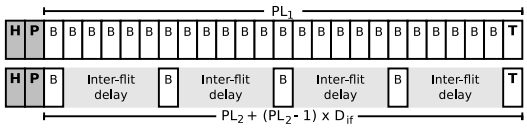



Fig. 4: Transmission equivalence among DoS attacks

$$PL_1 = PL_2 + (PL_2 - 1) \times D_{if} \quad (1)$$

$$PL_2 = \frac{PL_1 + D_{if}}{1 + D_{if}} \quad (2)$$

$$D_{if} = \frac{PL_1 - PL_2}{PL_2 - 1} \quad (3)$$

## V. Architecture of the Proposed LSDoS Monitor

Aiming to achieve LSDoS robustness in NoCs, to IPT and JFIDV DoS attacks, we propose the use of specialized hardware monitors. The proposed functionality can be implemented in the network interfaces, between the network interfaces and the routers, or at the router's local port. Since the NI is traditionally made as an in-house IP while the Network is bought from a 3rd party, we opted for the last option. However, our choice of monitor location has no effect on the functionality and efficacy of the approach.

Based on the router architecture of the Bonfire open-source framework, found in [22], we present a new architecture that can avoid LSDoS attacks in NoCs (depicted in Fig. 5). As in the base architecture, the router contains: i) input buffers (FIFOs) that store flits received through each input port until they can be forwarded; ii) routing units that implement a Logic Based Distributed Routing (LBDR) mechanism [23], which supports any turn-model based routing algorithm and provide the possibility of an in-system reconfiguration of the routing algorithm; iii) one switch allocator that arbiters data transmission from the FIFOS to the required output port; and iv) a crossbar that provides the connection between the FIFOS and the required output port. In the proposed architecture, the Low-and-Slow DoS monitor (depicted in Fig. 6) sits between the local *FIFO* and the subsequent blocks. Such location allows the monitor to truncate malicious LSDoS packets and drop their remaining flits from the *FIFO* (if any).

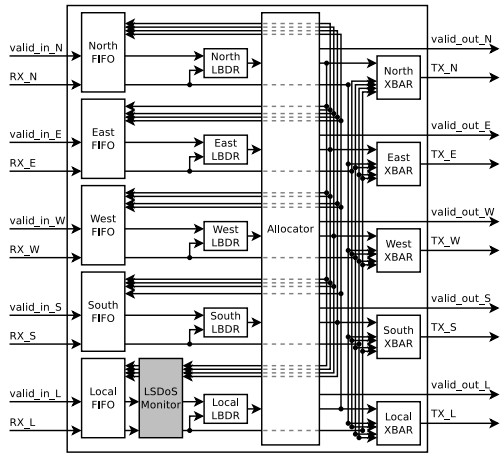



Fig. 5: LSDoS monitor - Router architecture

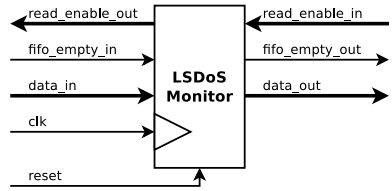



Fig. 6: LSDoS monitor - Black box diagram

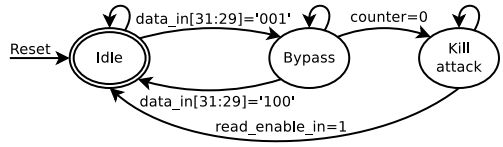



Fig. 7: LSDoS monitor - Finite State Machine (FSM)

TABLE I: LSDoS monitor - FSM assignments

| | |
|---|---|
| Idle | counter = MAX_VALUE - 1;<br>read_en_out = '1';<br>data_out = 0;<br>fifo_empty_out = '1'; |
| Bypass | read_en_out = read_enable_in;<br>data_out = data_in;<br>fifo_empty_out = fifo_empty_in;<br><br>if ( fifo_empty_in = '0' ) then<br>  counter = MAX_VALUE - 1;<br>elsif ( read_enable_in = '1' ) then<br>  counter = counter - 1;<br>end if; |
| Kill attack | counter = MAX_VALUE-1;<br>read_en_out = 1;<br>data_out = "1000 0000 0000 0000 0000 0000 0000 0001";<br>fifo_empty_out = '1'; |

Fig. 7 depicts the Finite State Machine (FSM) that describes the functionality of the LSDoS monitor and Table I details the main assignments done during each state.

On reset, the LSDoS monitor goes to *Idle*, where the counter is set to the maximum acceptable inter-flit delay (which can be set any time by the SSM), from where it will transition to the *Bypass* state as soon as a header flit is outputted by the local *FIFO*. During the *Bypass* state, data will flow as if no LSDoS monitor were present. However, the counter will be reset every time a new flit is received, or decreased every clock cycle in which the FIFO remains empty and data is being required. If a tail flit is received, the monitor will go back to the *Idle* state and wait for a new packet. Otherwise, if the counter times out, the presence of an LSDoS attack is assumed. Consequently, the monitor will transmit a predefined tail flit and go back to *Idle*. Thanks to this tail flit, incomplete packets generated by an IPT DoS attack attempt are terminated while packets generated by a JFIDV DoS attack attempt are cropped, releasing the communication path. Consequently, at the destination, the received packet will be dropped, since the declared packet size will not match the actual size. Moreover, by returning to *Idle*, all the remaining flits from the malicious packet, in case of a JFIDV DoS attack attempt, will be dropped, thus neutralizing the attack. This because the proposed LSDoS monitor only forwards flits while being at the *Bypass* state, to which it will only transition when detecting a header flit.

## VI. Experimental Work

In order to validate the impact to a NoC caused by the introduced Low-and-slow DoS attack and the efficacy of the proposed LSDoS monitor, the implementation of the attack behaviour and the detection mechanism has been done based on a VHDL RTL description, integrated into the Bonfire NoC platform [22]. Moreover, such an integration allowed the use of the platform's traffic generators to create sensitive, normal and DoS attack traffic with different PIR values and packet lengths. The traffic generators, however, were modified so that transmission delays could be added between the flits of the attack packets. Section VI-A details the scenario configurations set for the experiments. Section VI-B summarizes the obtained results regarding the comparison of FDoS and LSDoS attacks. Finally, Section VI-C presents the results that prove the efficacy of the proposed LSDoS monitors.

### A. Simulation Scenarios

For the experiments detailed in this paper, we adopted the simulation scenarios used for FDoS attacks in [14]. Such scenarios are depicted in Fig. 8, where, in $4 \times 4$ mesh NoC-based MPSoCs, network routers apply the XY-routing algorithm and a credit-based flow control with fair Round-Robin arbitration (on packet level), as well as the Wormhole switching with 4-flit deep FIFOs.

Moreover, as shown in Figure 8, each scenario has its own distinct source and destination of sensitive communication. For each scenario, 14 sets of experiments were performed. Each set of experiments uses a unique attacker location, covering all the nodes in the network (except the sensitive origin and destinations of the evaluated scenario). The attacker nodes, in all the scenarios, send their traffic to the sensitive nodes destination, ensuring a collision with the sensitive traffic. It is important to note that each experiment was performed for 20 pseudo-random traffic simulation seeds to provide uniform results.

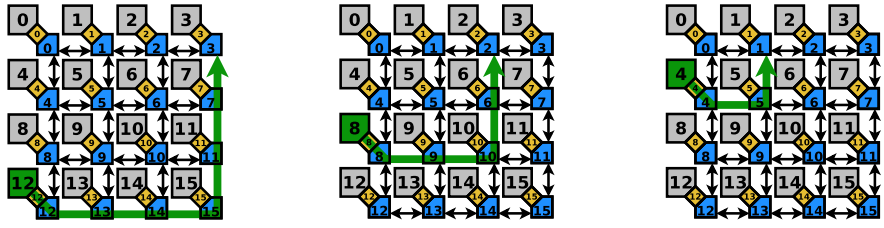
### B. Flooding DoS vs Low-and-Slow DoS attacks

To evaluate the impact of low-and-slow DoS (LSDoS) attacks in NoCs, experiments were done using Scenario 1 (Fig. 8a), where the sensitive traffic flows from PE12 to PE3. In each experiment, all traffic was transmitted with the same PIR value, either 0.003, 0.01 (one packet every 100 clock cycles), or 0.017. For both attacks, FDoS and LSDoS, the source is PE0, which sends the malicious packets also to PE3, as shown in Fig. 2a.

Fig. 9 summarizes the results obtained with different configurations of LSDoS attacks and an their equivalent configuration for FDoS, as well as configurations where no attack was present (Horizontal dashed lines). Results are grouped by the inter-flit delay value of the LSDoS attack, together with its equivalent FDoS packet length. Moreover, each group is composed of three pairs of bars, one for each PIR value, and the bars of each pair correspond to a different attack type, LSDoS and FDoS, respectively from left to right. Additionally, the Y axis corresponds to the mean end-to-end communication delay of sensitive packets in clock cycles (transmitted from PE12 to PE3). As shown in Fig. 9, the effect caused by each LSDoS attack has a correspondent FDoS configuration, regardless of the packet injection rate.

### C. Efficacy of the LSDoS monitor

The results of the experiments done to test the LSDoS Monitor for each of the three scenarios presented in Section VI-A,



(a) Scenario 1: Sensitive path 12→3 (b) Scenario 2: Sensitive path 8→2 (c) Scenario 3: Sensitive path 4→1

Fig. 8: Scenarios considering XY routing

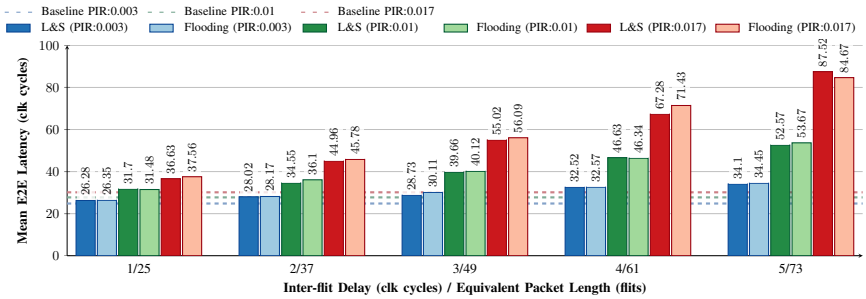



Fig. 9: Comparison of mean end-to-end latency of sensitive traffic under flooding and equivalent low-and-slow DoS attacks for different PIR values

| AL | Detected |
|---|---|
| 0 | ✓ |
| 1 | ✓ |
| 2 | ✓ |
| 3 | S. dest |
| 4 | ✓ |
| 5 | ✓ |
| 6 | ✓ |
| 7 | ✓ |
| 8 | ✓ |
| 9 | ✓ |
| 10 | ✓ |
| 11 | ✓ |
| 12 | S. src |
| 13 | ✓ |
| 14 | ✓ |
| 15 | ✓ |

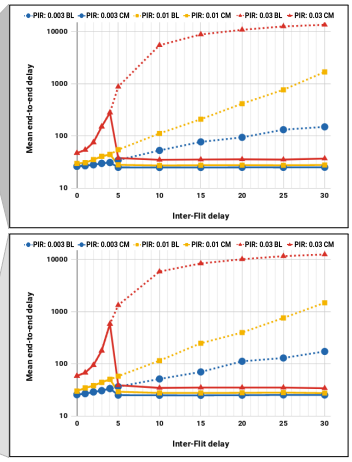



Fig. 10: Detection and counter-measure results for Scenario 1

| AL | Detected |
|---|---|
| 0 | ✓ |
| 1 | ✓ |
| 2 | S. dest |
| 3 | ✓ |
| 4 | ✓ |
| 5 | ✓ |
| 6 | ✓ |
| 7 | ✓ |
| 8 | S. src |
| 9 | ✓ |
| 10 | ✓ |
| 11 | ✓ |
| 12 | ✓ |
| 13 | ✓ |
| 14 | ✓ |
| 15 | ✓ |

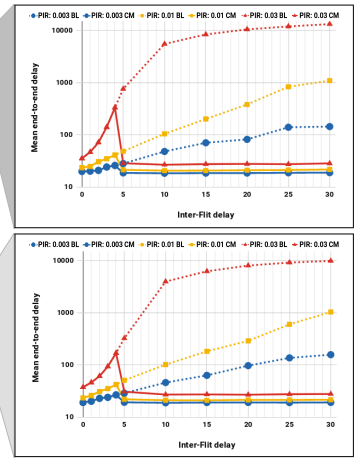



Fig. 11: Detection and counter-measure results for Scenario 2

are summarized in Fig. 10, Fig. 11, and Fig. 12, respectively. For all the figures, the X axis corresponds to the amount of clock cycles that the attacking PE (or NI) waits between the transmission of consecutive flits of a same packet. On the other hand, the Y axis corresponds to the mean end-to-end delay values of the sensitive packets (also in clock cycles). Additionally, the blue circular markers identify the results when considering a PIR of 0.003 for all the packets flowing through the NoC (i.e. sensitive, malicious and random). The yellow square markers correspond to a PIR of 0.01 and the red triangles to a PIR of 0.03. Furthermore, dotted lines show the proportionality between the inter-flit delay and the mean end-to-end latency of the sensitive packets when the proposed countermeasure is disabled. In contrast, the continuous lines show the impact of the Low-and-slow DoS attack to the sensitive packets in a NoC where the proposed countermeasure is implemented. As shown by each pair of lines (i.e. lines with the same color/marker), experiments with and without countermeasure achieve the same results until an inter-flit delay threshold is reached, this because the countermeasure is triggered by exceeding the threshold. (for our experiments, we adopted an inter-flit delay threshold of 5 clock cycles, nevertheless, dynamic configuration of this value can be done by the SSM).

Additionally, the resulting latency after exceeding the adopted inter-flit delay threshold is smaller than that of normal conditions (when no attack had happened) because, once the attack is detected, the proposed countermeasure truncates the malicious packet at the source, hence, the channel is freed earlier than if a normal-sized packet were transmitted.

Even though experiments were done considering the attack from each of the 14 available locations (every PE, except the source and destination of the sensitive traffic in each scenario), figures show results of only two attacker locations for each scenario, hence, as expected, all locations presented similar behavior.

### *D. Overhead Evaluation*

In this section we will evaluate the overheads of the proposed mechanism on the router and the system. The proposed architecture were synthesized using the $0.18\ \mu$m AMS library and Synopsys Design Vision at 200 MHz. Table

| AL | Detected |
|---|---|
| 0 | ✓ |
| 1 | S. dest |
| 2 | ✓ |
| 3 | ✓ |
| 4 | S. src |
| 5 | ✓ |
| 6 | ✓ |
| 7 | ✓ |
| 8 | ✓ |
| 9 | ✓ |
| 10 | ✓ |
| 11 | ✓ |
| 12 | ✓ |
| 13 | ✓ |
| 14 | ✓ |
| 15 | ✓ |

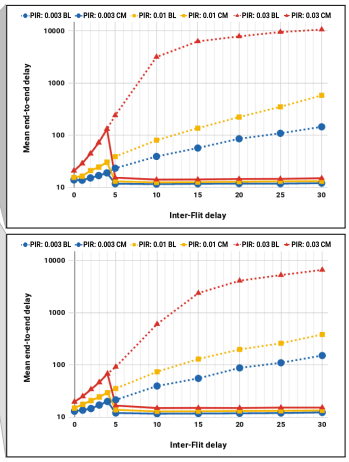



Fig. 12: Detection and counter-measure results for Scenario 3

II shows the area overhead of the proposed mechanism on the baseline router. However, even though the overhead of the proposed solution is just 2%, it is important to note that the baseline architecture is very minimalist. To put the results in perspective, for a 4 $mm^2$ chip with 16 cores For an SoC using a $4 \times 4$ mesh network, the area overhead would be around 0,06 % which is completely negligible. Table II also shows that the overhead of the proposed monitor on the critical path delay of the system is negligible.

TABLE II: Area comparison of the proposed architecture and the baseline router

| | Area | | | | Timing | |
|---|---|---|---|---|---|---|
| | Sequential ($\mu m^2$) | Combinatorial ($\mu m^2$) | Total ($\mu m^2$) | Overhead (%) | Critical-Path Delay ($ns$) | Overhead (%) |
| Baseline | 48562.21 | 48336.42 | 96898.63 | – | 4.98 | – |
| Proposed | 49411.75 | 49832.29 | 99244.05 | 2.4% | 4.97 | $\approx 0$ |

## VII. Conclusion

In this paper the first analysis of Low-and-Slow DoS attacks on NoC platforms and an effective mitigation concept have been presented. Wormhole routing makes NoCs prone to LSDoS attacks. A comparative analysis with Flooding DoS attacks has been carried out, proving that LSDoS attacks can be as effective as Flooding DoS attacks at a fraction of attacker effort. The mitigation of single source and multi-source attacks can be effectively handled at the connection point between the network interface and the local port of the NoC router with neglectable overhead compared to a minimalist router.

## References


[1] F. A. Samman, T. Hollstein, and M. Glesner, "Multicast parallel pipeline router architecture for network-on-chip," in *2008 Design, Automation and Test in Europe*, March 2008, pp. 1396–1401.

[2] K. Shibin, S. Devadze, A. Jutman, M. Grabmann, and R. Pricken, "Health management for self-aware socs based on ieee 1687 infrastructure," *IEEE Design & Test*, vol. 34, no. 6, pp. 27–35, 2017.

[3] R. JS, D. M. Ancajas, K. Chakraborty, and S. Roy, "Runtime detection of a bandwidth denial attack from a rogue network-on-chip," in *Proceedings of the 9th International Symposium on Networks-on-Chip*. ACM, 2015, p. 8.

[4] T. Boraten and A. K. Kodi, "Mitigation of denial of service attack with hardware trojans in noc architectures," in *2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, May 2016, pp. 1091–1100.

[5] J. Sepúlveda, M. Strum, and W. Chau, "An hybrid switching approach for noc-based systems to avoid denial-of-service soc attacks," *16th Iberchip Wksp (IWS 2010)*, pp. 23–25, 2010.

[6] M. J. Sepúlveda, J. P. Diguet, M. Strum, and G. Gogniat, "Noc-based protection for soc time-driven attacks," *IEEE Embedded Systems Letters*, vol. 7, no. 1, pp. 7–10, March 2015.

[7] S. Evain and J. P. Diguet, "From noc security analysis to design solutions," in *IEEE Workshop on Signal Processing Systems Design and Implementation, 2005.*, Nov 2005, pp. 166–171.

[8] L. Fiorin, C. Silvano, and M. Sami, "Security aspects in networks-on-chips: Overview and proposals for secure implementations," in *Digital System Design Architectures, Methods and Tools, 2007. DSD 2007. 10th Euromicro Conference on*. IEEE, 2007, pp. 539–542.

[9] L. Fiorin, G. Palermo, and C. Silvano, "A security monitoring service for nocs," in *Proceedings of the 6th IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis*, ser. CODES+ISSS '08. New York, NY, USA: ACM, 2008, pp. 197–202. [Online]. Available: http://doi.acm.org/10.1145/1450135.1450180

[10] J.-P. Diguet, S. Evain, R. Vaslin, G. Gogniat, and E. Juin, "Noc-centric security of reconfigurable soc," in *Proceedings of the First International Symposium on Networks-on-Chip*, ser. NOCS '07. Washington, DC, USA: IEEE Computer Society, 2007, pp. 223–232. [Online]. Available: http://dx.doi.org/10.1109/NOCS.2007.32

[11] 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*, vol. 34, no. 8, pp. 1344–1357, Aug 2015.

[12] A. B. Achballah, S. B. Othman, and S. B. Saoud, "Toward on hardware firewalling of networks-on-chip based systems," in *2017 International Conference on Advanced Systems and Electric Technologies (IC_ASET)*, Jan 2017, pp. 7–13.

[13] S. Charles, Y. Lyu, and P. Mishra, "Real-time detection and localization of dos attacks in noc based socs," in *Design, Automation and Test in Europe (DATE)*, 2019, pp. 1–6.

[14] C. G. Chaves, S. P. Azad, T. Hollstein, and J. Sepúlveda, "Dos attack detection and path collision localization in noc-based mpsoc architectures," *Journal of Low Power Electronics and Applications*, vol. 9, no. 1, 2019. [Online]. Available: http://www.mdpi.com/2079-9268/9/1/7

[15] M. Aiello, E. Cambiaso, M. Mongelli, and G. Papaleo, "An on-line intrusion detection approach to identify low-rate dos attacks," in *2014 International Carnahan Conference on Security Technology (ICCST)*. IEEE, 2014, pp. 1–6.

[16] B. P. Pooja, M. P. Manish, and B. P. Megha, "Jellyfish attack detection and prevention in manet," in *2017 Third International Conference on Sensing, Signal Processing and Security (ICSSS)*. IEEE, 2017, pp. 54–60.

[17] G. P. Fettweis, "5g and the future of iot," in *ESSCIRC Conference 2016: 42nd European Solid-State Circuits Conference*, Sept 2016, pp. 21–24.

[18] C. G. Chaves, S. P. Azad, T. Hollstein, and J. Sepúlveda, "A distributed dos detection scheme for noc-based mpsocs," in *2018 IEEE Nordic Circuits and Systems Conference (NORCAS): NORCHIP and International Symposium of System-on-Chip (SoC)*. IEEE, 2018, pp. 1–6.

[19] X.-M. Liu, G. Cheng, L. Qi, and M. Zhang, "A comparative study on flood dos and low-rate dos attacks," *The Journal of China Universities of Posts and Telecommunications*, vol. 19, pp. 116–121, 2012.

[20] D. Fang, H. Li, J. Han, and X. Zeng, "Robustness analysis of mesh-based network-on-chip architecture under flooding-based denial of service attacks," in *Networking, Architecture and Storage (NAS), 2013 IEEE Eighth International Conference on*. IEEE, 2013, pp. 178–186.

[21] W. J. Dally and B. P. Towles, *Principles and practices of interconnection networks*. Elsevier, 2004.

[22] "Bonfire Wiki," https://github.com/Project-Bonfire/Bonfire/wiki, accessed: 2019-02-03.

[23] S. P. Azad, B. Niazmand, K. Janson, N. George, A. S. Oyeniran, T. Putkaradze, A. Kaur, J. Raik, G. Jervan, R. Ubar, and T. Hollstein, "From online fault detection to fault management in network-on-chips: A ground-up approach," in *2017 IEEE 20th International Symposium on Design and Diagnostics of Electronic Circuits Systems (DDECS)*, April 2017, pp. 48–53.

# Appendix 5

**VI**
Cesar G. Chaves, Johanna Sepúlveda, and Thomas Hollstein, "Lightweight Packet Monitoring for Flooding DoS Attack Detection in NoC-based MPSoCs," in *Testmethoden und Zuverlässigkeit von Schaltungen und Systemen (TuZ 2021)*, February 2021

# Lightweight Packet Monitoring for Flooding DoS Attack Detection in NoC-based MPSoCs

Cesar G. Chaves*†, Johanna Sepúlveda‡§, Thomas Hollstein*†
*Frankfurt University of Applied Sciences, Frankfurt, Germany - †Tallinn University of Technology, Tallinn, Estonia
‡Airbus Defence and Space, Ottobrunn, Germany - §Technical University of Munich, Munich, Germany
email: cesar.chaves@stud.fra-uas.de, johanna.sepulveda@airbus.com, hollstein@fb2.fra-uas.de

***Abstract*—The use of Multiprocessor Systems-on-Chip (MPSoCs) within scalable fog and cloud computing systems is continuously increasing, facing the challenge of potential attacks from malicious tasks executing on such multi-tenant systems. Flooding Denial-of-Service (FDoS) attacks are one of the most common and powerful threats for Network-on-Chip (NoC)-based MPSoCs. By overwhelming the NoC, the system is unable to respond to normal traffic. However, the effectiveness of the FDoS attack depends on the NoC configuration. In addition, designing a secure MPSoC capable of detecting such attacks while avoiding excessive power/energy and area costs is challenging. To this end, we present two contributions. First, we demonstrate two FDoS attacks based on: packet injection rate (PIR-based FDoS) and packet's payload length (PPL-based FDoS), showing that fair round-robin NoCs are intrinsically protected against PIR-based FDoS while the PPL-based FDoS represents a real threat to MPSoCs. Second, we propose a novel lightweight monitoring method for detecting communication disruptions. Simulation and synthesis results show the feasibility and efficiency of our approach.**

## I. Introduction

Multi-Tenant Computation-as-a-Service (MTCaaS) provides the basis for integrated cloud/fog systems with a high potential of flexibility, performance and efficiency [1]. Multiprocessor Systems-on-Chip (MPSoCs) are increasingly used as computational nodes for these scalable platforms. They are composed of an array of Processing Elements (PEs) (processors, memories, ASIC cores), which are connected through a communication structure, such as a mesh Network-on-chip (NoC). In such structure, Network interfaces (NIs) encapsulate data into packets which are forwarded by routers. Fig. 1 depicts an MPSoC that connects 16 PEs through a 4×4 mesh-base NoC, including a central Global Manager (GM) mapped to run on one of the PEs for providing functions such as task scheduling and fault detection, as well as, for maintaining health and resource maps [2]. However, introducing MPSoCs to bigger and open execution environments exposes them to malicious users. One common technique for attacking MPSoCs is through Flooding DoS attacks (FDoS) targeting the NoC, where packets are sent aiming to clog router output ports, increasing the end-to-end delay of other packets.

Previous works have addressed FDoS attacks mainly considering them as an increase of the packet injection rate (PIR-based FDoS attack). Some of them manage to avoid the attack, however adding a significant hardware overhead and/or affecting legitimate traffic not marked as sensitive [3]. Others focus on detecting the attack and even try to locate the source of the attack [4], [5]. However, as shown in this paper, this attack is not effective when using fairly arbitrated NoCs. In contrast a FDoS attack by increasing the packet payload length (PPL-based FDoS attack) represents a real threat to NoC-based MPSoCs. Therefore, we present a distributed scheme of lightweight monitors for tracking communication behavior of users in multi-tenant MPSoCs, which we called *Communication Disruption Tattletaling (CDT)*. Such mechanism enables monitoring information to piggyback on data packets to be analyzed at the destination, and reported to the GM in the suspicion of an attack.

This work was funded by the German Academic Exchange Service (DAAD), Research Grants - Doctoral Programmes in Germany, 2018/19.

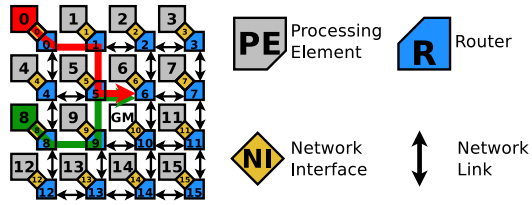



Fig. 1: Components of a NoC-based MPSoC

## II. Threat Model

In a large-scale MTCaaS scenario, as the Fog/Cloud execution environments, a high-level scheduler maps tasks to shared resources, which will include MPSoCs. Within the MPSoC, the GM maps the tasks on the PEs according to their requirements and constraints, taking into account the internal availability and characteristics of the PEs.

In order to perform a FDoS attack in a NoC-based MPSoC, the attacker, as any other client/user, submits a malicious application to the execution environment. Such application contains at least one task that, according to its characteristics/requirements, will suggest a Fog/Cloud scheduler to map its execution in an MPSoC-powered device. Such tasks will then be locally mapped by the *GM* of the MPSoC to a specific PE. However, without either of the schedulers knowing, one or more of the mapped tasks may attempt to flood the NoC; either intentionally (a FDoS attack) or unintentionally (a faulty task). Once a malicious task begins its execution in the PE, it starts sending data to other PEs inside the MPSoC (e.g., memory, a hardware accelerator, a peripheral, or other Intellectual Property component). Such communications may even try to mimic legitimate operations to avoid their detection.

## III. Proposed Architecture

The proposed packet structure for this work is depicted in Fig. 2. In it, the *Tail Flit* is used for enabling FDoS attack detection. By embedding the length of the longest packet found in the path, as well as its source, communication behavior can be monitored regardless of the NoC's topology. Such information is evaluated in every router of the path and updated if necessary. At the destination of each packet, the packet length value within the tail flit is evaluated. Suspicious data is reported to the GM, which, based on reports from the entire NoC, may decide to reset the identified attack source.

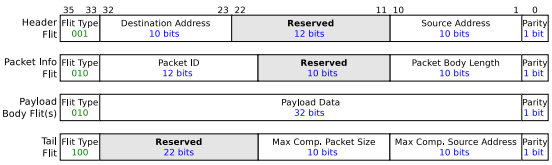



Fig. 2: Proposed Packet Structure

Based on the *Baseline router* architecture of the Bonfire platform [6], we propose the architecture presented in Fig. 3, where *Packet Size Monitors* register the source address and size of the last packet transmitted from each input port. *Greatest Competitor Updater* blocks evaluate/update the tail flits to ensure that the longest packets in paths are reported.

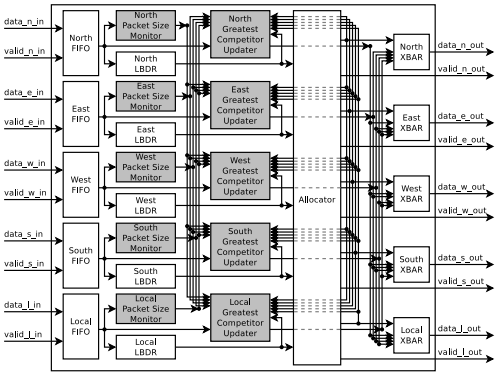



Fig. 3: Proposed Router Architecture

## IV. Experimental Work

### A. Simulation Scenarios

We adjusted the Bonfire platform [6] to be compatible with the packet structure depicted in Fig. 2. Subsequently, the scenario illustrated by Fig. 1 was implemented. In such scenario, PE0 and PE8 send data exclusively to PE6, ensuring the communication collision of both flows. Furthermore, all the attacks executed in the experiments were originated at PE0. To create a more realistic scenario, additional traffic was randomly generated from all the other PEs (through the traffic generators). Normal traffic was configured with a PPL

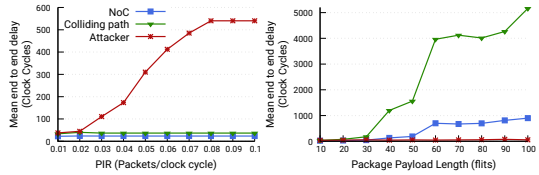



Fig. 4: Comparison of Flooding Dos Attacks

of 10 flits and a PIR of 0.01 (i.e., a rate of one packet every 100 clock cycles). Malicious traffic, on the other hand, was configured differently for the two sets of experiments: i) for the PIR-based FDoS attack, the attacker's PPL was fixed to 10 flits and its PIR varied from 0.01 to full burst transmission (i.e., 10 flits of payload every 10 clock cycles); and ii) for the PPL-based FDos attack, the attacker's PIR was fixed to 0.01 and its PPL varied from 10 to 100 (i.e., maximum of 100 flits every 100 clock cycles). Furthermore, the transmission buffer of NIs was extended so that attacks could completely maximize their parameters.

### B. Effect Comparison of Both Flooding DoS Attacks

As shown in Fig. 4 (left), where only the mean end-to-end delay of the attacker showed a significant increase, a PIR-based FDoS attack is ineffective in a NoC using a fair arbitration mechanism such as Round-robin, which forces malicious packets to remain buffered in the NI. In contrast, Fig. 4 (right) shows the effective communication disruption caused by the PPL-based FDoS attack. This since the mean end-to-end delay of packets colliding with the malicious flow increased during the attack. Additionally, an overall impact to the entire NoC was also reported. Therefore, FDoS attack detection and mitigation efforts should be focused to this scenario.

### C. Overhead Evaluation

Synthesis results were obtained through the Cadence's *Genus Synthesis Solution*, for $40nm$ CMOS technology and a clock frequency of $200MHz$. Table I shows results related to area and power, both for the worst case corner (i.e., slow-slow, $0.99v$, and $125°C$). Note that corner, edge and middle NoC mesh routers have different amount of ports. Results are presented for the 5-port router (R10). Despite the area overheads are considerable (greater than 30%), the baseline router is minimalist (i.e., worst-case overhead). Baseline router does not offer any service (e.g., Quality of communication, priority, fault-resilience) which in practice will incur in a bigger baseline router.

TABLE I: Overhead ($40nm$, $200MHz$, $0.99v$, $125°C$)

| **Area** | | | | | |
|---|---|---|---|---|---|
| Router (R10) | Cell Count | Sequential ($\mu m^2$) | Combinational ($\mu m^2$) | Total ($\mu m^2$) | Overhead (%) |
| Baseline | 2377 | 6114.259 | 3190.724 | 9304.982 | – |
| Proposed | 3913 | 7613.659 | 5337.629 | 12951.228 | 39.19% |
| **Power** | | | | | |
| Router (R10) | Leakage Power ($mW$) | Internal Power ($mW$) | Switching Power ($mW$) | Total Power ($mW$) | Total Overhead (%) |
| Baseline | 0.0094 | 0.6175 | 0.1214 | 0.7482 | – |
| Proposed | 0.0134 | 0.9602 | 0.3159 | 1.2896 | 72.36% |

## V. Conclusion and Future Work

It was proven in this paper that, in contrast to what has been said in related papers, the efficiency of FDoS attacks is related to the payload length of packets rather than the injection rate. This because fair arbitration mechanisms control the NoC's injection rate depending on the amount of traffic generated by other sources, forcing packets to be buffered in the NIs. Additionally, a novel lightweight mechanism for detecting the real FDoS attack threat targeting NoC-based MPSoCs was introduced. Future work will be focused on processing the aggregation of the generated information within a general manager on system-level, allowing the recognition of high-level attack patterns and efficient mitigation of the attacks.

## References


[1] M. Iorga *et al.*, "Fog Computing Conceptual Model," National Institute of Standards and Technology, Tech. Rep., 2018.

[2] S. P. Azad, B. Niazmand, K. Janson, N. George, A. S. Oyeniran, T. Putkaradze, A. Kaur, J. Raik, G. Jervan, R. Ubar, and T. Hollstein, "From Online Fault Detection to Fault Management in Network-on-Chips: A ground-up Approach," in *2017 IEEE 20th International Symposium on Design and Diagnostics of Electronic Circuits Systems (DDECS)*, April 2017, pp. 48–53.

[3] J. Sepúlveda *et al.*, "Reconfigurable Security Architecture for Disrupted Protection Zones in NoC-based MPSoCs," in *ReCoSoC*, June 2015.

[4] K. Madden *et al.*, "Adding Security to Networks-on-chip Using Neural Networks," in *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2018.

[5] S. Charles *et al.*, "Real-time Detection and Localization of DoS Attacks in NoC Based SoCs," in *IEEE DATE*, 2019.

[6] "Bonfire Wiki," https://github.com/Project-Bonfire/Bonfire/wiki, 2016, accessed: 2020-11-05.

# Appendix 6

**IV**
Cesar G. Chaves, Johanna Sepúlveda, and Thomas Hollstein, "Lightweight Monitoring Scheme for Flooding DoS Attack Detection in Multi-Tenant MPSoCs," in *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5, May 2021

# Lightweight Monitoring Scheme for Flooding DoS Attack Detection in Multi-Tenant MPSoCs

Cesar G. Chaves*†, Johanna Sepúlveda‡, Thomas Hollstein*†
*Frankfurt University of Applied Sciences, Frankfurt, Germany - †Tallinn University of Technology, Tallinn, Estonia
‡Technical University of Munich, Munich, Germany
email: cesar.chaves@stud.fra-uas.de, johanna.sepulveda@tum.de, hollstein@fb2.fra-uas.de

2021 IEEE International Symposium on Circuits and Systems (ISCAS) | 978-1-7281-9201-7/20/$31.00 ©2021 IEEE | DOI: 10.1109/ISCAS51556.2021.9401153

***Abstract*—The increasing use of Multiprocessor Systems-on-Chip (MPSoCs) within scalable multi-tenant systems, such as fog/cloud computing, faces the challenge of potential attacks originated by the execution of malicious tasks. Flooding Denial-of-Service (FDoS) attacks are one of the most common and powerful threats for Network-on-Chip (NoC)-based MPSoCs. Since, by overwhelming the NoC, the system is unable to forward legitimate traffic. However, the effectiveness of FDoS attacks depend on the NoC configuration. Moreover, designing a secure MPSoC capable of detecting such attacks while avoiding excessive power/energy and area costs is challenging. To this end, we present two contributions. First, we demonstrate two types of FDoS attacks: based on the packet injection rate (PIR-based FDoS) and based on the packet's payload length (PPL-based FDoS). We show that fair round-robin NoCs are intrinsically protected against PIR-based FDoS. Instead, PPL-based FDoS attacks represent a real threat to MPSoCs. Second, we propose a novel lightweight monitoring method for detecting communication disruptions. Simulation and synthesis results show the feasibility and efficiency of the presented approach.**

## I. Introduction

Multi-Tenant Computation-as-a-Service (MTCaaS) provides the basis for integrated cloud/fog systems [1], [2] with a high potential of flexibility, performance and efficiency [3]. Multi-processor Systems-on-Chip (MPSoCs) are increasingly used as computational nodes for these scalable platforms. However, introducing MPSoCs to bigger and open execution environments exposes them to malicious users. Those who might attempt to disrupt the execution of processes being run by concurrent tenants. One common technique for attacking MPSoCs is through Flooding DoS attacks (FDoS) targeting the NoC, where malicious packets overwhelm a targeted communication path, compromising the operation of the system.

Previous works have addressed FDoS attacks mainly considering them as an increase of the packet injection rate (PIR-based FDoS attack). Some of them manage to avoid the attack, however adding a significant hardware overhead and/or affecting legitimate traffic not marked as sensitive [4]–[8]. Others focus on detecting the attack and even try to locate its source [9]–[14]. However, as shown in this paper, this attack is not effective when using fair-arbitrated NoCs. In contrast a FDoS attack by increasing the packet payload length (PPL-based FDoS attack) represents a real threat to NoC-based MPSoCs. Therefore, we present a distributed scheme of lightweight monitors for tracking communication behavior of users in multi-tenant MPSoCs, which we called *Communication Disruption Tattletaling (CDT)*. Such mechanism enables monitoring information to piggyback on data packets to be analyzed at the destination, and reported to the MPSoC's Global Manager (GM) in the suspicion of an attack.

In summary, the contributions of this work are: i) Demonstration of two FDoS attacks (PIR- and PPL-based DoS) targeting communication disruption of NoC-based MPSoCs, ii) A novel lightweight monitoring scheme for FDoS attack detection, iii) Evaluation of area and power overhead figures based on synthesis results using $40nm$ node technology.

The remainder of the paper is organized as follows: Section II defines important concepts for better understanding the paper; Section III summarizes related work and the motivation for our proposal; Based on a thread-model definition (Section IV), Section V explains the proposed method; Section VI presents the setup of the executed experiments and discuses the obtained results; and finally, Section VII concludes the paper and outlines future work.

## II. Background

### A. NoC-based MPSoCs

NoC-based MPSoCs are composed of an array of Processing Elements (PEs) (processors, memories, ASIC cores), which are connected to NoC routers via Network interfaces (NIs) within a mesh-based NoC architecture, realizing best-effort packet-based communication. Fig. 1 shows an MPSoC that integrates 16 PEs through a $4 \times 4$ mesh-base NoC. The NIs provide packetization of data to be sent over the NoC and depacketization upon arrival at the destination.

MPSoCs used in dependable and mission-critical systems must be designed adopting a fault management architecture [15]. Such an architecture includes additional hardware like monitors, sensors, and checkers that allow a detailed assessment of the system's performance. Usually such a structure

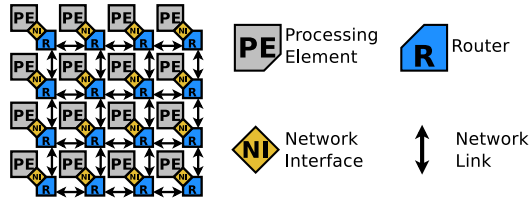
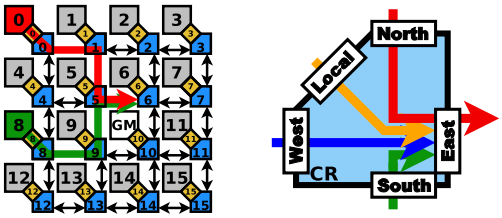



Fig. 1: Components of a NoC-based MPSoC

This work was funded by the German Academic Exchange Service (DAAD), Research Grants - Doctoral Programmes in Germany, 2018/19.




(a) Attacked MPSoC (b) Collision Router

Fig. 2: Example of DoS Attack in MPSoCs

is controlled by a central Global Manager (GM), which is mapped to run on one of the MPSoC's PEs (Fig. 2a) [16]. Such manager provides essential services as task scheduling [17], fault detection and instrument management, as well as, maintains health and resource maps (as established in [15], [16]).

### B. Flooding DoS Attacks in NoCs

In Flooding DoS (FDoS) Attacks, a malicious application running on any PE of the MPSoC tries to disrupt the communication of other system components. By sending packets to another PE in the NoC, the attacker creates additional communication delays to other packets that wish to be forwarded through at least one segment of the path being congested by the attack. Fig. 2a illustrates a scenario in which sensitive packets flow from PE8 to PE6 while malicious traffic from PE0 also to PE6. In such scenario, both traffic flows collide inside router R5 and compete for its East output port. Fig. 2b depicts a worst case scenario in which, apart from the two mentioned traffic flows, two additional flows also enter the competition. Since only one flow is granted access to the output at a time, it is the goal of the malicious traffic that it will be it, making the others wait for their turn, hence increasing their end-to-end communication delay.

Two types of FDoS attacks attempt to gain control of output ports, preventing legitimate traffic from being forwarded. Fig. 3 shows three types of traffic flows: the upper one, considered as normal traffic, is composed of packets with an expected packet injection rate and within an acceptable size range. The first attack (middle flow) attempts to flood the network by increasing its packet injection rate (PIR-based FDoS Attack), while the other by increasing the packet's payload length (PPL-based FDoS Attack).

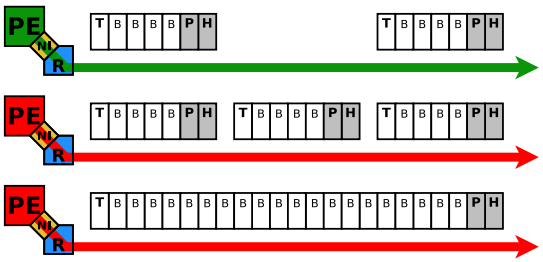



Fig. 3: Flooding DoS Attacks

### C. Wormhole Flow Control

An adopted flow control establishes how the allocation of resources to packets is done as they go along their route [18]. Wormhole flow control is used for targeting two major scalability concerns of NoC designers: i) low latency intra-chip communication, by taking routing decisions upon the reception of only the packet's header, hence reducing the amount of clock cycles needed for forwarding a packet to its destination; and ii) low area routers, by using smaller buffers to store data while the routing decision is being made. However, once an output port of a small-area router is assigned for the transmission of a packet, it will not serve other packets until this transmission is finished, making NoCs vulnerable to PPL-based FDoS attacks.

### D. Fair Round-Robin (RR) Arbitration

Attempting to prevent the disruption of packet transmissions by bursts of packets coming from other inputs, arbitration techniques such as the Fair Round-Robin have been proposed [19] [20]. With this technique, output access is sequentially granted to a single packet from each competing input. As proved in Section VI, fair Round-Robin NoCs reduce the effectiveness of PIR-based FDoS. We also show that PPL-based FDoS attacks are feasible and are still successful with the fair Round-robin arbitration NoCs.

## III. Related Work

### A. DoS attack avoidance

Some of the previous works attempt to avoid DoS attacks in NoC-Based MPSoCs [4]–[8]. The work of [4] implements virtual channels in the NoC to isolate sensitive traffic, resulting in the undesirable effect of significantly increasing area and power consumption.

In [5], circuit switching is used for transmitting sensitive traffic and packet switching for other type of packets. Further approaches presented in [6]–[8] define secure zones. Despite these solutions effectively isolating sensitive traffic from attackers, they might affect the performance of legitimate traffic not labeled as sensitive.

### B. DoS Attack Detection

Additional works have focused on developing techniques for the detection of DoS attacks on NoC-based MPSoCs. However, most of the approaches target PIR-based FDoS [9]–[14]. In [9], the authors propose the implementation of probes in the NI for detecting deviations from expected bandwidth usage at design time. In [10], multi-source attacks are considered with a mitigation concept based on firewalls in the NIs. Authors in [11] propose firewalls between routers, monitoring occupation time of links. The approach presented in [12] applies a spiking neural network for detecting an abnormal amount of communication requests. The work in [13] uses distributed monitors that threshold the amount of received packets in NIs. Once the attack has been detected, additional packets are sent to attempt the localization of the attack's source. Such work was later enhanced considering multi-source scenarios in [14].

Our previews work targeted both types of FDoS attacks (PIR- and PPL-based) by monitoring communication degradation based on end-to-end communication and inter-router forwarding delays [21], [22]. Even though both approaches provide an idea of the attack source, they do not identify the exact attack source location.

Furthermore, as explained in Section II-D and proven in Section VI-B, PIR-based FDoS attacks can be avoided by the use of a fair forwarding arbitration technique. Therefore, our current approach targets PPL-based attacks by implementing a tattletale scheme in which each packet directly informs the source of the packet that disrupted its transmission the most. In that way, FDoS attacks can be detected with no additional packet transmission for locating the source of the attack.

## IV. Threat Model

Adding MPSoC-powered devices to execution environments, such as Fog and Cloud, can exceed the performance of conventional general-purpose computer execution environments, thus it adds application specific circuits to distributed context-related processing nodes. The characteristics of such scenario and its vulnerability to DoS attacks will be explained in this section, as well as, the steps followed by an attacker to execute a FDoS attack.

In a large-scale MTCaaS scenario, as Fog/Cloud execution environments, a high-level scheduler maps tasks to shared resources [23], which will include MPSoCs. Within the MPSoC, the GM maps tasks to PEs according to their requirements and constraints, taking into account the availability and characteristics of the PEs. Moreover, as soon as the execution of the tasks mapped to the MPSoC is finished, the results are returned following the Fog + Cloud schedule. Finally, when the execution of the entire application is over, results are sent back to the user.

In order to perform a FDoS attack in a NoC-based MPSoC, an attacker submits a malicious application to the execution environment, as any other client/user will do with a legitimate application. Such application contains at least one task that, according to its characteristics/requirements, will suggest a Fog/Cloud scheduler mapping its execution to an MPSoC-powered device. Such tasks will then be locally mapped by the *GM* of the MPSoC to a specific PE. However, without either of the schedulers knowing, one or more of the mapped tasks may attempt to flood the NoC; either intentionally (a FDoS attack) or unintentionally (a faulty task). Finally, once a malicious task begins its execution in a PE, it starts sending data to other PEs within the MPSoC that will also be required by legitimate tasks (e.g. memories, hardware accelerators, peripherals, or other Intellectual Property components).

## V. Proposed Architecture

NIs are in charge of encapsulating data into packets at the source of a transmission, as well as, of decapsulating it at the destination. Such packets are composed of flow control units (Flits) following a structure understandable by the routers in the NoC. The proposed structure for this work is depicted in Fig. 4, in which the three most significant bits of each flit

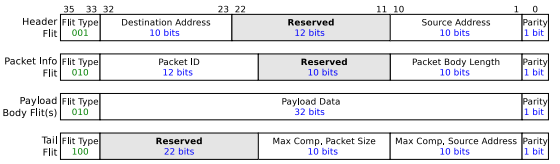



Fig. 4: Proposed Packet Structure

inform its type, while the less significant bit is a parity bit used for single error detection. A *Header Flit* signals the beginning of a transmission and contains the information used for taking routing decisions. The *Packet Info Flit* contains information regarding the packet, i.e. an ID for allowing packet reordering at the destination and the length of the packet for checking its completeness. One or more *Payload Body Flits* follow, carrying the data exchanged between PEs. Subsequently, a *Tail Flit* signals the end of the packet, and when transmitted through a router's output port, the port is released for the transmission of a new packet. In many designs, the *Tail Flit* carries the last part of the payload, however, for the purpose of our work, all the payload is carried by *Payload Body Flits* while the *Tail Flit* is used for DoS attack detection. This by transporting the length of the longest packet found in the path, as well as its source. Such information is evaluated in every router of the path and updated if necessary.

Based on the *Baseline router* architecture of the Bonfire platform [24], we propose the architecture presented in Fig. 5, where a *Packet Size Monitor* at each of the router's input ports extracts the source address and size of the last packet transmitted from such port (respectively the less significant bits of the Header and Packet Info flits, without considering the parity bit). Furthermore, each input port is also equipped with a *Greatest Competitor Updater*, which gathers information from other blocks as follows: i) requests from the *LBDR* (Logic Based Distributed Routing) unit for knowing the output ports that can be used to forward the packet in its monitored port; ii) the grants given by the *Allocator* for using the output ports, which by being matched with the requests, tell the input ports that won the competition for the require output ports; and iii)

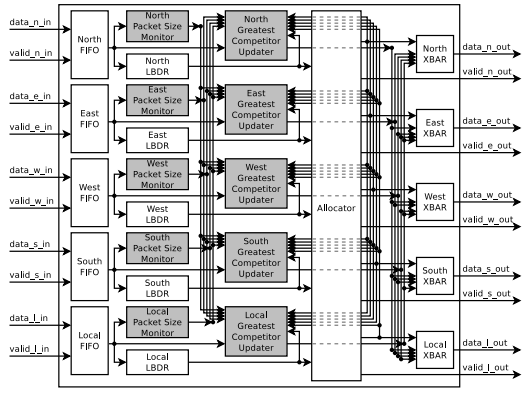



Fig. 5: Proposed Router Architecture

information from the *Packet Size Monitors* for identifying the size and source of the packets that prevented the local packet from being forwarded. Finally, by updating the *Tail flit* with the information of the greatest competitor, each packet is able to tattle tale on the longest packet found in its path, together with its source. Consequently, the destination NI, using a predefined threshold, can decide if a report should be sent to the Global Manager for deciding if the found behavior was allowed.

## VI. Experimental Work

In this work two communication parameters are exploited for attempting Flooding DoS (FDoS) attacks on NoC-based MPSoCs, the packet injection rate (PIR) (used in most state-of-the-art approaches) and the packet's payload length (PPL) (being used in the presented approach). In this section we present an effectiveness comparison of both attacks, as well as synthesis results of the proposed monitoring scheme.

### A. Simulation Scenarios

In order to compare both FDoS attacks, we adjusted the Bonfire platform [25] so that it be compatible with the packet structure depicted in Fig. 4. Subsequently, we implemented the scenario illustrated by Fig. 2a. In such scenario, PE0 and PE8 send data exclusively to PE6, this for ensuring that the path of both sources collide. Furthermore, all the attacks executed in the experiments were originated at PE0 and additional traffic was randomly generated from all the other PEs (i.e. traffic generators included in Bonfire). Normal traffic was configured with a PPL of 10 flits and a PIR of 0.01 (i.e. a rate of one packet every 100 clock cycles). Malicious traffic, on the other hand, was configured differently for two sets of experiments: i) PIR-based FDoS attack: the attacker's PPL was fixed to 10 flits and its PIR varied from 0.01 to full burst transmission (i.e. 10 flits of payload every 10 clock cycles); and ii) PPL-based FDos Attack: the attacker's PIR fixed to 0.01 and its PPL varied from 10 to 100 (i.e. a maximum of 100 flits every 100 clock cycles). Furthermore, the transmission buffer of NIs was extended so that attacks could completely maximize their parameters.

### B. Effect Comparison of Both Flooding DoS Attacks

The effect of the PIR- and PPL-based FDoS attacks is presented in Fig. 6, on the left and right graphs respectively. Both graphs show the effect caused to a single colliding path (i.e. PE8 $\rightarrow$ PE6), as well as, to the entire NoC. Such effect is observed by the increase in the mean end-to-end delay of packets, since they are transmitted by the PE and until they reach their destination, while the attack is intensified.

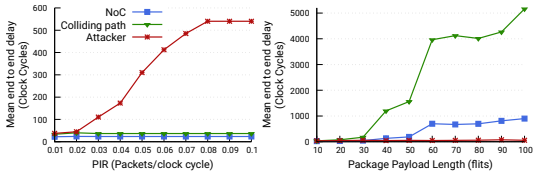



Fig. 6: Comparison of Flooding Dos Attacks

As shown in Fig. 6 (left), a PIR-based FDoS attack is ineffective in a NoC using a fair arbitration mechanism such as Round-robin, which segments bursts while other packets are transmitted, consequently, increasing only the end-to-end delay of the malicious packets buffered in the NI. In contrast, Fig. 6 (right) shows the effective communication disruption caused by the PPL-based FDoS attack to the NoC and even more to the colliding path. Hence FDoS attack detection and mitigation efforts should be focused to this scenario.

### C. Overhead Evaluation

This section presents synthesis results of the baseline router and the overhead when adding the proposed Communication Disruption Tattletale. Synthesis was done with the *Genus Synthesis Solution* provided by Cadence [26], using a $40nm$ commercial CMOS technology and a clock frequency of $200MHz$. Table I shows results related to area and Table II those related to power (using a value change dump), both for the worst case corner (i.e. slow-slow, $0.99v$, and $125°C$). Furthermore, considering that corner, edge and middle routers have different amount of ports in a NoC following a mesh topology, results for each router type are presented. It is worth noticing that even though the reported overheads are greater than 30%, the baseline router is already a minimalist design.

## VII. Conclusion and Future Work

In this contribution, a novel DoS detection approach for NoC-based MPSoCs has been presented. In contrast to the state-of-the-art, as shown by simulation results, an attacker's efficiency is rather related to the attack packet length than the packet injection rate. A lightweight mechanism for the monitoring of communication behaviour and related anomalies has been introduced, such mechanism can be used for the detection of anomalies such as DoS attacks, as well as, for the evaluation of the appropriateness of an actual system task schedule. Future work will be focused on processing the aggregation of the generated information within a general manager on system-level, allowing the recognition of high-level attack patterns and efficient mitigation of the attacks.

TABLE I: Area overhead ($40nm$, $200MHz$, $0.99v$, $125°C$)

| Router | Cell Count | Area: Sequential ($\mu m^2$) | Area: Combinational ($\mu m^2$) | Area: Total ($\mu m^2$) | Overhead (%) |
|---|---|---|---|---|---|
| Router with 3 ports (corner router - R15) | | | | | |
| Baseline | 1104 | 3636.192 | 1313.827 | 4950.019 | – |
| Proposed | 1741 | 4495.848 | 2065.291 | 6561.139 | 32.55% |
| Router with 4 ports (edge router - R14) | | | | | |
| Baseline | 1874 | 4875.696 | 2102.923 | 6978.619 | – |
| Proposed | 2971 | 6051.696 | 3411.106 | 9462.802 | 35.60% |
| Router with 5 ports (middle router - R10) | | | | | |
| Baseline | 2377 | 6114.259 | 3190.724 | 9304.982 | – |
| Proposed | 3913 | 7613.659 | 5337.629 | 12951.228 | 39.19% |

TABLE II: Power overhead ($40nm$, $200MHz$, $0.99v$, $125°C$)

| Router | Leakage Power ($mW$) | Internal Power ($mW$) | Switching Power ($mW$) | Total Power ($mW$) | Total Overhead (%) |
|---|---|---|---|---|---|
| Router with 3 ports (corner router - R15) | | | | | |
| Baseline | 0.0047 | 0.3243 | 0.0280 | 0.3571 | – |
| Proposed | 0.0063 | 0.4513 | 0.0735 | 0.5311 | 48.73% |
| Router with 4 ports (edge router - R14) | | | | | |
| Baseline | 0.0069 | 0.4478 | 0.0627 | 0.5174 | – |
| Proposed | 0.0095 | 0.6453 | 0.1653 | 0.8200 | 58.48% |
| Router with 5 ports (middle router - R10) | | | | | |
| Baseline | 0.0094 | 0.6175 | 0.1214 | 0.7482 | – |
| Proposed | 0.0134 | 0.9602 | 0.3159 | 1.2896 | 72.36% |

## REFERENCES


[1] P. M. Mell and T. Grance, "SP 800145. The NIST Definition of Cloud Computing," National Institute of Standards and Technology, Gaithersburg, MD, United States, Tech. Rep., 2011.

[2] M. Iorga, L. Feldman, R. Barton, M. J. Martin, N. S. Goren, and C. Mahmoudi, "Fog Computing Conceptual Model," National Institute of Standards and Technology, Tech. Rep., 2018.

[3] G. P. Fettweis, "5G and the Future of IoT," in *ESSCIRC Conference 2016: 42nd European Solid-State Circuits Conference*, Sept 2016, pp. 21–24.

[4] S. Evain and J. P. Diguet, "From NoC Security Analysis to Design Solutions," in *IEEE Workshop on Signal Processing Systems Design and Implementation, 2005.*, Nov 2005, pp. 166–171.

[5] J. Sepúlveda, M. Strum, and W. Chau, "An Hybrid Switching Approach for NoC-Based Systems to Avoid Denial-of-Service SoC Attacks," *16th Iberchip Wksp (IWS 2010)*, pp. 23–25, 2010.

[6] J. Sepúlveda, D. Flórez, and G. Gogniat, "Efficient and Flexible NoC-based Group Communication for Secure MPSoCs," in *2015 International Conference on ReConFigurable Computing and FPGAs (ReConFig)*, Dec 2015, pp. 1–6.

[7] M. J. Sepúlveda, J. P. Diguet, M. Strum, and G. Gogniat, "NoC-Based Protection for SoC Time-Driven Attacks," *IEEE Embedded Systems Letters*, vol. 7, no. 1, pp. 7–10, March 2015.

[8] J. Sepúlveda, D. Flórez, and G. Gogniat, "Reconfigurable Security Architecture for Disrupted Protection Zones in NoC-based MPSoCs," in *2015 10th International Symposium on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC)*, June 2015, pp. 1–8.

[9] L. Fiorin, G. Palermo, and C. Silvano, "A Security Monitoring Service for NoCs," in *Proceedings of the 6th IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis*, ser. CODES+ISSS '08. New York, NY, USA: ACM, 2008, pp. 197–202. [Online]. Available: http://doi.acm.org/10.1145/1450135.1450180

[10] 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*, vol. 34, no. 8, pp. 1344–1357, 2015.

[11] A. B. Achballah, S. B. Othman, and S. B. Saoud, "Toward on Hardware Firewalling of Networks-on-chip Based Systems," in *2017 International Conference on Advanced Systems and Electric Technologies (IC_ASET)*, Jan 2017, pp. 7–13.

[12] K. Madden, J. Harkin, L. McDaid, and C. Nugent, "Adding Security to Networks-on-chip Using Neural Networks," in *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2018, pp. 1299–1306.

[13] S. Charles, Y. Lyu, and P. Mishra, "Real-time Detection and Localization of DoS Attacks in NoC Based SoCs," in *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2019, pp. 1160–1165.

[14] ——, "Real-time Detection and Localization of Distributed DoS Attacks in NoC Based SoCs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2020.

[15] K. Shibin, S. Devadze, A. Jutman, M. Grabmann, and R. Pricken, "Health Management for Self-Aware SoCs Based on IEEE 1687 Infrastructure," *IEEE Design & Test*, vol. 34, no. 6, pp. 27–35, 2017.

[16] S. P. Azad, B. Niazmand, K. Janson, N. George, A. S. Oyeniran, T. Putkaradze, A. Kaur, J. Raik, G. Jervan, R. Ubar, and T. Hollstein, "From Online Fault Detection to Fault Management in Network-on-Chips: A ground-up Approach," in *2017 IEEE 20th International Symposium on Design and Diagnostics of Electronic Circuits Systems (DDECS)*, April 2017, pp. 48–53.

[17] W. Amin, F. Hussain, S. Anjum, S. Khan, N. K. Baloch, Z. Nain, and S. W. Kim, "Performance Evaluation of Application Mapping Approaches for Network-on-Chip Designs," *IEEE Access*, vol. 8, pp. 63 607–63 631, 2020.

[18] W. J. Dally and B. P. Towles, *Principles and Practices of Interconnection Networks*. Elsevier, 2004.

[19] L. Fiorin, C. Silvano, and M. Sami, "Security Aspects in Networks-on-chips: Overview and Proposals for Secure Implementations," in *Digital System Design Architectures, Methods and Tools, 2007. DSD 2007. 10th Euromicro Conference on*. IEEE, 2007, pp. 539–542.

[20] D. Fang, H. Li, J. Han, and X. Zeng, "Robustness Analysis of Mesh-based Network-on-chip Architecture Under Flooding-based Denial of Service Attacks," in *2013 IEEE Eighth International Conference on Networking, Architecture and Storage*. IEEE, 2013, pp. 178–186.

[21] Cesar G. Chaves, Siavoosh P. Azad, Thomas Hollstein, and Johanna Sepúlveda, "A Distributed DoS Detection Scheme for NoC-based MPSoCs," in *Circuits and Systems (NorCAS), 2018 IEEE Nordic Conference on*. IEEE, October 2018.

[22] ——, "DoS Attack Detection and Path Collision Localization in NoC-Based MPSoC Architectures," *Journal of Low Power Electronics and Applications*, vol. 9, no. 1, 2019.

[23] D. Charântola, A. C. Mestre, R. Zane, and L. F. Bittencourt, "Component-based Scheduling for Fog Computing," in *Proceedings of the 12th IEEE/ACM International Conference on Utility and Cloud Computing Companion*, 2019, pp. 3–8.

[24] "Baseline router," https://github.com/Project-Bonfire/Bonfire/wiki/Baseline-router, 2016, accessed: 2020-11-05.

[25] "Bonfire Wiki," https://github.com/Project-Bonfire/Bonfire/wiki, 2016, accessed: 2020-11-05.

[26] "Genus Synthesis Solution," https://www.cadence.com/en_US/home/tools/digital-design-and-signoff/synthesis/genus-synthesis-solution.html, 2020, accessed: 2020-11-07.

# Curriculum Vitae

## 1. Personal data

| | |
|---|---|
| Name | Cesar Giovanni Chaves Arroyave |
| Date and place of birth | 30 November 1982 Popayán, Colombia |
| Nationality | Colombian |

## 2. Contact information


| | |
|---|---|
| Address | Tallinn University of Technology, School of Information Technologies,<br>Department of Computer Systems, ICT-509<br>Akadeemia Tee 15A, 12618 Tallinn, Estonia |
| Phone | +372 620 2267 |
| E-mail | cesar.chaves@taltech.ee, cesarchaves1@gmail.com |


## 3. Education

| | |
|---|---|
| 2018–2022 | Tallinn University of Technology, School of Information Technologies,<br>Computer and Systems Engineering, Ph.D. studies |
| 2008–2010 | University of Campinas, Institute of Computing,<br>Computer Science, MSc (Defense 2016) |
| 1998–2004 | Universidad Cooperativa de Colombia, Faculty of Engineering,<br>Systems Engineering, BSc |

## 4. Language competence

| | |
|---|---|
| Spanish | native |
| English | fluent |
| Portuguese | fluent |
| German | basic |

## 5. Professional employment

| | |
|---|---|
| 2022– ... | SiPearl GmbH, Digital Design & Verification Engineer |
| 2012–2018 | Eldorado Research Institute, Development Engineer |
| 2011–2011 | CTI Renato Archer, Trainee in Digital IC Design |
| 2009–2009 | University of Campinas, Intern in Teaching |
| 2005–2008 | Newtech Systems LTDA, Software Engineer |
| 1999–2000 | Universidad Cooperativa de Colombia, Computer Lab Consultant |

# Elulookirjeldus

## 1. Isikuandmed

| | |
|---|---|
| Nimi | Cesar Giovanni Chaves Arroyave |
| Sünniaeg ja -koht | 30.11.1982, Popayán, Kolumbia |
| Kodakondsus | Kolumbialane |

## 2. Kontaktandmed


| | |
|---|---|
| Aadress | Tallinna Tehnikaülikool, Usaldusväärsete arvutisüsteemide keskus,<br>Arvutisüsteemide Instituut,<br>Ehitajate tee 5, 19086 Tallinn, Eesti |
| Telefon | +372 620 2267 |
| E-post | cesar.chaves@taltech.ee, cesarchaves1@gmail.com |


## 3. Haridus

| | |
|---|---|
| 2018–2022 | Tallina Tehnikaülikool, Informaatikateaduskond,<br>Arvutisüsteemide instituut, PhD õpingud |
| 2008–2010 | University of Campinas, Institute of Computing,<br>Computer Science, MSc (Defense 2016) |
| 1998–2004 | Universidad Cooperativa de Colombia, Faculty of Engineering,<br>Systems Engineering, BSc |

## 4. Keelteoskus

| | |
|---|---|
| hispaania keel | emakeel |
| inglise keel | kõrgtase |
| portugali keel | kõrgtase |
| saksa keel | algtase |

## 5. Teenistuskäik

| | |
|---|---|
| 2022– . . . | SiPearl GmbH, Digital Design & Verification Engineer |
| 2012–2018 | Eldorado Research Institute, Development Engineer |
| 2011–2011 | CTI Renato Archer, Trainee in Digital IC Design |
| 2009–2009 | University of Campinas, Intern in Teaching |
| 2005–2008 | Newtech Systems LTDA, Software Engineer |
| 1999–2000 | Universidad Cooperativa de Colombia, Computer Lab Consultant |

ISSN 2585-6901 (PDF)
ISBN 978-9949-83-961-2 (PDF)