Architecting a Secure Wireless Network-on-Chip

Brian Lebiednik*, Sergi Abadal†, Hyoukjun Kwon* and Tushar Krishna*
*Georgia Institute of Technology, †Universitat Politècnica de Catalunya
brian.lebiednik@usma.edu, abadal@ac.upc.edu, hyoukjun@gatech.edu, tushar@ece.gatech.edu

Abstract—**With increasing integration in SoCs, the Network-on-Chip (NoC) connecting cores and accelerators is of paramount importance to provide low-latency and high-throughput communication. Due to limits to scaling of electrical wires in terms of energy and delay, especially for long multi-mm distances on-chip, alternate technologies such as Wireless Network-on-Chip (WNoC) have shown promise. WNoCs can provide low-latency one-hop broadcasts across the entire chip and can augment point-to-point multi-hop signaling over traditional wired NoCs. Thus, there has been a recent surge in research demonstrating the performance and energy benefits of WNoCs. However, little to no work has studied the additional security and fault tolerance challenges that are unique to WNoCs. In this work, we study potential threats related to denial-of-service, spoofing, and eavesdropping attacks in WNoCs, due to malicious hardware trojans or faulty wireless components. We introduce Prometheus[1], a drop-in solution inside the network interface that provides protection from all three attacks, while adhering to the strict area, power and latency constraints of on-chip systems.**

I. Introduction

Network-on-Chip (NoC) is currently the paradigm of choice to interconnect the different components of System-on-Chips (SoCs) or Chip Multiprocessors (CMPs). As the levels of integration continue to grow, however, current NoCs face significant scalability limitations and may become a performance bottleneck in manycore systems. One promising solution to this problem is the introduction of new interconnect technologies as an extension of the NoC paradigm. Among them, wireless on-chip communications have garnered considerable attention due to their low latency, architectural flexibility, and inherent broadcast capabilities [1], [2]. Architecting manycore systems with Wireless Network-on-Chips (WNoCs) is an active area of research [3] since low-latency broadcasts can facilitate scalable coherence and consistency.

The adoption of the WNoC paradigm brings up new challenges, including the implementation and efficient integration of high-performance and low-cost transceivers or the development of appropriate communication protocols. Security is another important aspect to address as the broadcast nature of the wireless transmissions introduces new points of entry for an attacker to compromise the chip. If accesses to the wireless medium are not protected, smart Hardware Trojans (HTs) placed within the network or in third-party components can degrade the system performance, write corrupt data in memory, or steal sensitive information.

This paper focuses on the security aspects of WNoC, and we build on two observations. First, the communication mechanism of a WNoC is essentially different from that of wired on-chip networks. This introduces new threats, such as the possibility of eavesdropping or generating a global Denial-of-Service (DoS) attack from any node's Medium Access Control (MAC) module, and prevents the use of existing NoC protection strategies, e.g. [4], [5], [6], [7]. Second, the performance requirements of a WNoC are radically different from that of conventional wireless networks, driving the need for fast and lightweight solutions and preventing the use of existing strategies for wireless security, e.g. [8], [9].

The main contribution of this paper is Prometheus, a tripartite solution to three threats to the WNoC: DoS, spoofing, and eavesdropping. Prometheus is placed within each node's Network Interface (NIF), as shown in Figure 1, and is capable of deactivating transceivers affected by HT to mitigate their impact. DoS attacks are detected through observation of the medium accesses and the collection of network statistics from the NIF. To combat spoofing attacks, Prometheus incorporates our previous work Veritas [10], which identifies spoofers opportunistically by comparing the reception power profiles of the presumed and actual source of a message. Lastly, eavesdropping is prevented by securing the communications with very low-cost encryption schemes. Through performance and cost analysis, Prometheus can protect a WNoC from advanced HTs with reasonable power, area, and network performance overheads. To the best of the authors' knowledge, this is the first work jointly addressing DoS, spoofing, and eavesdropping in Wireless NoCs. Existing works either address a single threat [11] or incur unacceptable overheads [12].

This paper is set forth as follows. First, we provide some background on WNoC in Section II. Next, we discuss the threat model in Section III and our proposed architecture solution in Section IV. Then, we evaluate the performance and cost of our proposal in Sections V and VI, respectively. Finally, we discuss related work in Section VII, and conclude the paper in Section VIII.

II. Background

Constant downscaling of Radio Frequency (RF) circuits have opened the door to the conception of a wide variety of WNoC architectures. In this approach, wireless interfaces are co-located with cores and generally complement the wired NoC. Due to the relatively large size of the RF passives, most

[1]The Prometheus$_{\text{Spoof}}$ portion of this paper is based on the work accepted in the 3rd International Workshop on Advanced Interconnect Solutions and Technologies for Emerging Computing Systems. The other two components are extensions onto the system. We have also extended the evaluation of Prometheus$_{\text{Spoof}}$.

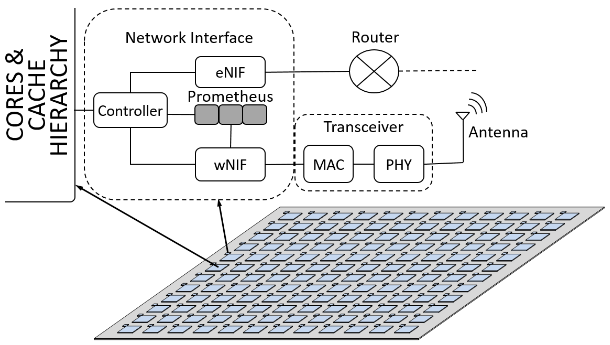


Fig. 1: Schematic representation of Prometheus within a wired-wireless NoC architecture.

WNoC proposals assume that wireless interfaces are shared among several cores [1], whereas more aggressive proposals consider per-core integration as exemplified in Figure 1 [3]. Next, we provide some background on the components of a WNoC and discuss their security implications.

A. Physical Layer (PHY)

The PHY defines how bits are transmitted over the wireless links and, thus, influences the design of the antenna and the transceiver. In a WNoC, the PHY module basically serializes processor messages, modulating the resulting bits at a given frequency much higher than the processor clock, and deliver the modulated signal to the antenna. The inverse operation is performed at reception.

Current transceiver proposals for chip communication use frequencies around 60 GHz with simple modulations to minimize area and power. As a reference, the 65-nm CMOS design from [13] uses on-off keying and achieves 16 Gbps with a bit error rate of 10^{-15} while taking 31.2 mW of power and 0.25 mm^2 of silicon area. Future trends point to even higher RF frequencies to further reduce area and increase the speed of the WNoC [14]. Coding in WNoC needs to be fast and simple but capable of detecting small bursts of errors [1].

These PHY design decisions have important security implications. First, encryption mechanisms (if any) should also be fast and efficient to avoid becoming a performance bottleneck. Second, it is necessary to achieve a fairly large signal-to-noise ratio to maintain the error rate requirements of the scenario. For instance, the transceiver in [13] is designed assuming a signal strength 18 dB above noise. Due to this, we can assume that thermal noise fluctuations hardly affect the signal and, also, that it is theoretically possible to distinguish the very sparse errors generated by thermal noise and those caused by a wireless collision. As we will see, such distinction is important to provide security at the MAC level.

B. Medium Access Control (MAC)

The MAC[2] layer implements mechanisms to ensure that nodes can access the medium reliably. This is a key determinant of performance in any wireless network as two simultaneous accesses to the same channel will fail. In WNoCs, the MAC mechanism becomes fundamental as the medium will be densely populated and the load will be probably high.

Related works in WNoC consider a variety of MAC designs. We classify them into two kinds, each with its own pros and cons. *Contention-free* schemes [1], [2], [15] avoid collisions via arbitration or the use of different frequency bands or time slots. These techniques can deliver high throughput, but do not work well under variable workloads since bandwidth is statically allocated. A popular implementation of this type of MAC is the token passing protocol, where only the node holding the token can transmit [1].

Contention-based schemes [3], [16], [17] allow all nodes to attempt to transmit on the shared medium at any time instant. This provides flexibility and reduces overall latency, but comes at the cost of limited throughput due to the potential collisions. These schemes use various mechanisms to minimize collisions and to recover from them. In wireless networks in general, Carrier Sensing Multiple Access (CSMA) protocols where nodes listen to the medium before transmitting and perform a random backoff if they collide are widely considered. These type of protocols have been adapted to the WNoC scenario as well [17]. Variations of CSMA, such as MACAW [18], can further improve on the performance of contention-based protocol by introducing explicit fairness measures. We based our CSMA protocol on [17] with the fair backoff of MACAW.

As we will see, both types of protocols have their own security implications. To cover the most representative cases, we will consider both token passing and CSMA protocols in our analysis. Malicious nodes will attempt to cause a denial of service by generating collisions in both schemes.

C. Network Interface (NIF)

The NIF performs address translation and admission control tasks. In our target WNoC, it can implement load balancing and Quality of Service (QoS) functions. To this end, we assume the NIF to be composed of a controller, which determines the path (wired or wireless) to follow by each message, and two specific interfaces connected to their respective network planes. At the interfaces, network performance statistics can be collected to assist congestion avoidance and fairness mechanisms or, in our case, security policies.

III. Threat Model

Insecure wireless NoCs can potentially be vulnerable to various kinds of attacks. Next, we discuss the main assumptions related to security and then describe the threats addressed in this work: DoS, spoofing, and eavesdropping.

A. Assumptions

As a first assumption, we consider a system with a single point of attack. We will see that this assumption is enough to provide significant harm in the WNoC scenario and avoids considering an scenario where an unbounded attacker could disrupt the system. Another common assumption we make is that Prometheus, our solution, cannot be compromised [19]. It

[2]In the security literature, MAC could refer to Message Authentication Code. But to remain consistent with RF networks, we use MAC to mean Medium Access Control.

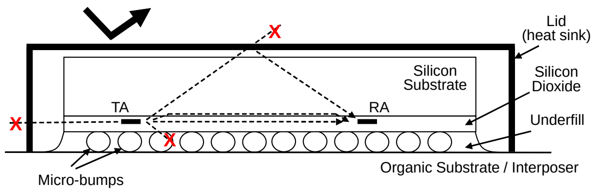


Fig. 2: The wireless intra-chip channel.

is also common practice to assume that the HT is placed in a digital circuit. Therefore, HTs cannot alter the PHY layer.

We further consider the propagation of RF signals within the system, which determines the origin of potential attacks in WNoC. The chip (or system) package can be seen as a metallic box that prevents RF signals from leaking to or coming from outside the package. This is important to security because it is safe to assume that DoS, spoofing, or eavesdropping can only be performed from the *inside*.

To better justify this last assumption, we illustrate the typical structure of a processor within a flip-chip package in Figure 2 (a similar analysis can be performed in a system-in-package due to structural resemblances [20]). Circuits are placed within an insulator on top of a silicon substrate. This structure is flipped, connected to the rest of the system via an array of metallic micro-bumps, and often covered with a metallic lid that acts as a heat sink. In this configuration, signals propagate throughout the chip via multiple paths, but cannot scatter outside the package due to the presence of the heat sink and the micro-bumps [21]. Despite not being a solid chunk of metal, the micro-bumps and subsequent metallizations disrupt propagation because their pitch is generally much lower than the wavelength of the RF waves (i.e., 10–100 μm against ∼1 mm in mmWave bands). A final reasonable consideration is that, in such a controlled and enclosed environment, signals do not suffer from humidity effects.

B. Denial of Service

Any misconfiguration at the MAC layer can cause severe problems inside the WNoC. Two nodes transmitting on the same channel cause a collision, corrupting the message. Continued collisions lead to loss of bandwidth or breakdown of the wireless network. We do not consider the case where the attacker waits until someone transmits to then create a collision on purpose, as this requires the HT to be placed at the PHY layer. Yet, malicious or faulty entities could exploit vulnerabilities at the MAC layer to perform a DoS attack.

We quantify the potential harm by modeling a game [22] with every node playing one of six possible configurations (Strategy × Traffic) shown in Fig. 3(a) for a CSMA system. "Selfish" nodes try to maximize their utility by playing unfairly. In the context of WNoCs, an unfair play could be (a) sending out of turn in case of a token-based arbitration scheme or (b) lowering the backoff delay in case of a CSMA scheme. In either scenario, this can result in bandwidth stealing from normal ("healthy") nodes. It can be quite hard to detect this behavior since the selfish nodes can hide behind the protocol and pretend to be "hotspot" nodes with more traffic to send. This is especially so in the CSMA protocol, which by itself cannot distinguish between genuine contention and DoS.

Fig. 3(b) sweeps the design space modeled above by varying the injection rate x in a 16-core contention-based system, and plots the relative bandwidth of each node as a function of the total injected load. With healthy nodes, hotspot nodes utilize higher bandwidth than normal nodes, as expected. With selfish nodes, at normal loads, the relative bandwidth occupancy is almost the same as that of healthy nodes, which shows that the CSMA protocol itself is robust. However, once the selfish nodes start injecting at moderate to heavy loads, their relative bandwidth occupancy becomes significantly larger than that of the healthy nodes (both normal and hotspot). Fig. 3(c) plots the latency versus throughput of all nodes across the same set of configurations. Here, we can see that the latency of healthy nodes is quadrupled and the throughput drops by over 70% in the presence of selfish hotspot nodes, thereby slowing down forward progress of any parallel application significantly. Note that this attack is unique to a WNoC domain and no known solution exists to the best of our knowledge.

C. Spoofing

Since the WNoC naturally acts as a shared medium [23], any node can broadcast information. This can be leveraged by malicious cores to cause system-level problems by manipulating the source address of flits.Spoofing could be employed to bypass memory access protection by impersonating a core that has permission to write, eventually writing in prohibited regions of memory to steal sensitive information or disrupt execution. Spoofing may also be leveraged to respond to legitimate requests originally intended for a given node n. Before n can answer with the requested information, another rogue node r responds with false information, causing the application to crash. A more complex r might respond with incorrect data that does not cause the application to crash, but rather provide incorrect outputs or loss in performance. We fully explore spoofing and provide solutions with results in [10] and expand upon our results in VI.

D. Eavesdropping

Broadcast messages are inherently vulnerable to eavesdropping attacks because all nodes are always listening. Different processes or Virtual Machines (VMs) may need to keep the passed information secret from other processes because they are running at different permission levels or different users. Also, large cloud processing NoCs can have proprietary information running for clients that they want kept secret. In traditional NoCs, the traffic is kept separate between applications using QoS, but this is not suitable in a broadcast environment [24]. Also, if the admission control mechanism of the NIF is compromised, such secret information will be exposed anyway. Thus, there is a need for messages to be sent securely through the WNoC.

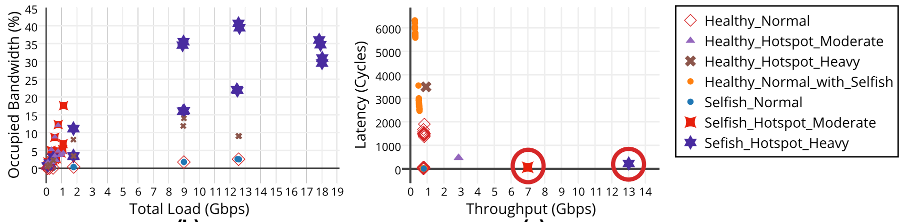
IV. Secure WNoC Microarchitecture: Prometheus

We design Prometheus, a set of hardware solutions, to address the three threat models described earlier in Section III.

Strategy	Description
Healthy	Plays according to protocol
Selfish	Attempts to get higher utility by not backing off on a collision

Traffic	Description
Normal	Injects x packets/cycle
Hotspot_Moderate	Injects 10x packets/cycle
Hotspot_Heavy	Injects 100x packets/cycle

(a)



(b) (c)

Fig. 3: Potential DoS Attack. (a) Types of wireless nodes in the system. Each node plays a particular Strategy × Traffic in each game. (b) Channel bandwidth occupied by healthy and selfish nodes. (c) Throughput and latency characteristics for healthy and selfish nodes (Hotspot nodes shown in circles).

Each of these can be added as a module into the NIF, as shown in Fig. 4, to protect the system from the particular threat model at hand –be it due to a HT, or a faulty MAC hardware. Upon detection of a potential threat, a flag is raised and the ID of the malicious node is sent to the OS which disables (or resets) its WNIF, forcing it to only use its wired NoC either temporarily or indefinitely. Although Prometheus works well independently of the number and location of transceivers, we henceforth assume a single transceiver per core in a homogeneous processor.

A. *DoS protection via unfairness detection*

We consider DoS attacks in both contention-free and contention-based schemes as described earlier in Section II-B.

1) Contention-free MAC: In contention-free MAC protocols (see Section II), there should never be a collision since only the node possessing a token [1] is allowed to transmit. In this scenario, a selfish node (either malicious or faulty) would transmit out of turn and potentially cause a collision leading to data corruption. Prometheus$_{\text{DoS}}$ proposes to have a node monitor collisions, and using the wired NoC, prompt the owner of the token to suppress its own transmissions for a fixed period of time. Transmissions by the selfish node can now be identified by its ID, and turned off by the OS. If the selfish node spoofs its ID, Prometheus$_{\text{Spoof}}$ (Section IV-B) kicks in.

2) Contention-based MAC: Detecting a DoS attack is more challenging in contention-based protocols such as CSMA because collisions in the channel are an inherent part of the protocol. A selfish node (Figure 3(a)) can cause repeated collisions but is indistinguishable from a healthy node with hotspot traffic. We identify that the key difference between the two cases is *unfairness*.

Healthy nodes should exhibit the following properties: at low-loads, they should experience a high injection throughput (T_{TX}), a low reception throughput (T_{RX}), and low backoff delay (B), translating to low latency and high bandwidth from the channel. At high-loads, they should experience moderate T_{TX}, high T_{RX}, and a high value of B, translating to moderate latency and bandwidth. In case of a DoS attack, as Figure 3(b) and (c) demonstrated, a selfish node will always experience a high T_{TX}, low T_{RX}, and low B.

Unfairness Ratio Γ. We propose to use T_{TX}, T_{RX}, B, and the wireless channel capacity C (in Gbps) to determine whether the node is healthy or selfish in a distributed way. This is in contrast to typical fairness ratios, which gener-

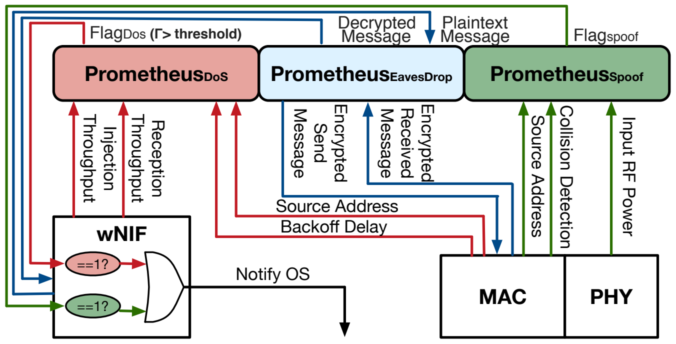


Fig. 4: Prometheus Microarchitecture

ally use a centralized approach to compare a single metric as they assume that the protocol is not compromised [25]. Prometheus$_{\text{DoS}}$ periodically probes the WNIF and MAC for these statistics as Fig. 4 shows. Over three thousand samples across several hundred simulations of different node behavior combinations, as described in Fig. 3(a), we derive a metric called the *unfairness ratio* Γ

$$\Gamma = \frac{T_{TX}}{C + T_{RX}} \times \frac{B_{min}^2}{1 + B^2} \qquad (1)$$

where B_{min} is the minimum backoff (1 cycle in our case).

Intuitively, Γ will increase for nodes that are experiencing disproportionately high amounts of injection, indicating a potential DoS attack. As T_{TX} increases and T_{RX} decreases, Γ increases. A selfish node will knowingly reduce B to more successfully obtain the channel. Nodes with high T_{TX}, that are healthy, should naturally experience a higher B, lowering the overall Γ. Healthy nodes will have a Γ near zero (as T_{TX} is close to T_{RX}) relative to selfish nodes that have a higher Γ. If Γ is greater than a configurable threshold, it signals the OS about the unfairness.

False Positives and Negatives. As Γ is a heuristic, naturally it can have both false positives (a healthy hotspot node being tagged by its NIF as selfish) and false negatives (selfish nodes go undetected). Also, Γ may vary across different WNoC configurations. Section V evaluates the robustness of Γ through an example.

B. *Spoofing protection via RF power analysis*

Since the WNoC naturally acts as a shared medium, any node can broadcast information. Leveraging this, a HT node may choose to masquerade as another node to make unauthorized memory access or to cause performance loss.

In off-chip wireless networks, authenticity can be guaranteed with asymmetric keys. Asymmetric key encryption, however, is orders of magnitude slower than symmetric encryption [26], which already takes several clock cycles per byte of information [27]. Thus, authenticity via asymmetric encryption becomes impractical in NoC and WNoC environments. Fast and lightweight alternatives are required instead.

The WNoC paradigm offers a unique possibility of using the received RF power levels to determine the identity of the source of a given packet. In conventional wireless communications, propagation is modeled as a stochastic process as it depends on many random factors such as the environment, mobility, or blocking, among others. On the contrary, the WNoC scenario is static, highly controlled, and confined. As a result, the wireless channel becomes time-invariant [28] and quasi-deterministic –aspects such as humidity effects can be neglected, multipath and path loss are static and can be known beforehand. Moreover, since we expect a high signal-to-noise ratio (see Section II), we can consider path loss measurements to be temperature-invariant as well.

Building on these observations, Prometheus$_{\text{Spoof}}$ (formerly Veritas [10]) converts the received power into an effective source address and compares it with the ID contained in the packet header. A mismatch raises a spoofing alert. To prevent from repeating the same research, we leave the full implementation details of Prometheus$_{\text{Spoof}}$ to [10].

C. *Eavesdropping protection via low-cost encryption*

Encryption keeps messages secret between sender and receiver.We consider symmetric key encryption because of the large overhead of asymmetric keys and hashing. The National Institute for Standards and Technology (NIST) prescribes encryption based on the use and the encryption strength needed. For network communications, specifically IPSec, NIST recommends the Advanced Encryption Standard (AES)-128 in Cipher Block Chaining (CBC) mode [29]. The cryptographic community considers AES, the standard for network communication, as a fast encryption scheme and the only known attacks are side channel attacks [26], such as observing CMOS gates to decipher the encryption key [30]. Such an attack would be near impossible in a WNoC.

Challenges with AES-128 in WNoCs. Though AES is fast by modern network standards for off-chip wireless networks, using AES for on-chip communication comes with delay, area, and power penalties, as we show in this work. Instead, we believe that a secure WNoC can use other faster but less secure encryption algorithms such as stream ciphers to encrypt data.

Proposed Solution: Stream Ciphers. Stream ciphers perform an operation on each bit (flip or not flip) using synced timing and symmetric keys. We consider two stream ciphers, RC4A and Py [31] (pronounced "roo"). Py improves on RC4A security and speed by using rolling arrays that rotate every rotation step by one unit. RC4A and Py are only vulnerable to *linear distinguishing attacks*, meaning that given a certain number of bytes an attacker can distinguish between a random stream of bytes and a stream encrypted with RC4A. Many consider this an academic break of the encryption because an attacker only knows that the sender used RC4A to encrypt the message [32]. Table I provides processing time and known attacks for each algorithm with the associated number of flits needed for each attack.

TABLE I: Known attacks, length of the attack (128-bit flits), and processing delay of the proposed encryption schemes.

Encryption	**Attack**	**Length**	**Delay per Byte**
AES	Side Channel	N/A	20 cycles [27]
RC4A	Distinguishing	2^{51} flits	7 cycles [33]
Py	Distinguishing	2^{65} flits	2.85 cycles [34]

Prometheus$_{\text{Eavesdrop}}$. We implement Py in Prometheus$_{\text{Eavesdrop}}$ since it has the lowest performance, power, and area overhead, as we show in our evaluations. It uses the following key distribution scheme. We add a key distribution center (KDC) where nodes request keys if the information needs confidentiality from other nodes in the system. When the system starts, each node negotiates a new key with the KDC using a preshared potion of memory only know to the KDC and the respective node. The communication between the KDC and each node is done with this unique key. This requires only n additional keys where the node would share that key with the key distribution scheduler only. That ensures that the system does not need to use asymmetric encryption. Using this key, each node can negotiate a key with the KDC everytime it wants to created an encrypted session with another node. The other node does likewise so only the KDC, node $n1$, and node $n2$ know the key. The key request could delay the communication slightly, but once established the key can be used for up to 2^{65} flits before requesting another key to prevent linear distinguishing attacks. Lastly, we also encrypt the Cyclic Redundancy Check (CRC) along with the message. This makes the encryption scheme resistant to plaintext and ciphertext attacks. Like messages will have the same CRC, so sending a message without encrypting the CRC will reveal the underlying message.

V. Evaluations

A. *Prometheus$_{DoS}$*

Simulation Methodology. We modeled and simulated Prometheus$_{\text{DoS}}$ in the PhoenixSim framework [35]. PhoenixSim is an event-driven NoC simulator that, while oriented to photonic NoCs, incorporates a complete set of accurate models for the evaluation of electrical NoC designs. Those models include several parameterized NIF and router designs, as well as a remarkable amount of routing protocols and topologies. On top of this, Abadal *et al.* implemented the necessary modules for the simulation of wireless on-chip communication, including different PHY, MAC, and wireless NIF designs [36], [37].

Relevant to the evaluation of Prometheus$_{\text{DoS}}$, each node in our simulator is configured to be a Healthy/Selfish node sending Normal/Hotspot traffic (Fig. 3(a)) at varying injection rates and burst rates to sweep the design space. Our experiments simulate 16 antennas, a number consistent with current

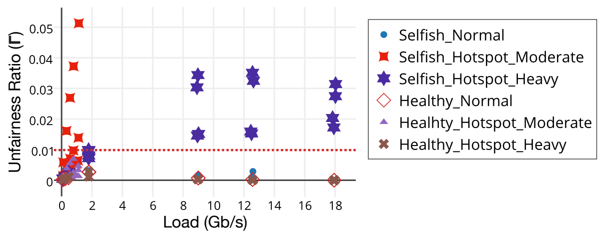


Fig. 5: Unfairness ratio Γ of healthy/selfish nodes across normal/hotspot traffic at varying loads in a 16-core system.

WNoC designs [37], [1]. In the interest of space, we do not present results for contention-free protocols which are easier to safeguard. Instead, we implement the contention-based protocol from [17] augmented with the well-known explicit fairness mechanisms of MACAW [18], and the contention-free token-passing variant from [36].

Results. We evaluated the validity of Prometheus$_{\text{DoS}}$ with the unfairness ratio Γ described in Sec. IV-A. Figure 5 plots Γ as a function of network load across a suite of traffic patterns. All points with the same color and symbol represent different injection and burst rates at that particular configuration. We make three key observations from our results:

- Selfish nodes with moderate to heavy hotspot traffic have a $\Gamma > 0.01$, which can be set as the threshold to raise a flag to the OS. Note that the exact value of the threshold may vary as the size of the WNoC changes and can be configured after stress testing as our simulations have.
- Healthy nodes with moderate to heavy hotspot traffic have a low Γ, all of them below 0.01 in this case, demonstrating that we completely avoided false positives.
- Selfish nodes with normal traffic have a low Γ below the threshold, showing that there could be false negatives. However, these false negatives are harmless since these selfish nodes do not actually steal bandwidth from other nodes (as selfish hotspot nodes do) due to the inherent robustness of the protocol.

Thus, we find that a selfish node must have heavy traffic to unfairly use the channel, at which point Γ can detect it.

B. *Prometheus$_{Spoof}$*

Impact of chip package. To extend on the work in [10], further investigation was required in the package design. The chip package has a strong impact on the channel response. Therefore, it is pertinent to evaluate Prometheus$_{\text{Spoof}}$ in different package configurations. For instance, since low-resistivity silicon introduces substantial losses, some works have proposed to thin the silicon die [38]. To illustrate the potential impact of such decisions on Prometheus$_{\text{Spoof}}$, we evaluate the dynamic range and resolution requirements as a function of the thickness of the silicon die. We refer the reader to [10] for details on the simulation methodology.

Figure 6 shows the results of the analysis. It is observed that reducing the silicon die indeed brings the dynamic range requirements down because losses are minimized, whereas the resolution requirements oscillate around acceptable levels (5 dB). These results demonstrate that Prometheus$_{\text{Spoof}}$ would benefit from *a priori* co-design efforts to (i) avoid bad design points, like the unacceptable dynamic range over 100 dB for a 0.45-mm silicon die; and (ii) point towards designs with relaxed requirements, i.e. at 0.15 mm the dynamic range is 36.57 dB and the resolution requirement is 4.11 dB.

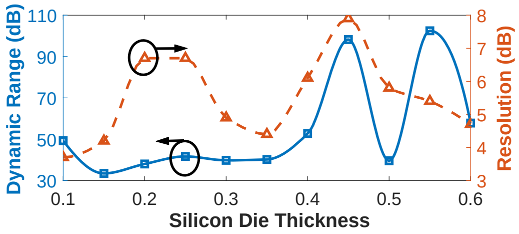


Fig. 6: Dynamic range and resolution requirements of Prometheus$_{\text{Spoof}}$ as a function of the silicon thickness.

C. *Prometheus$_{Eavesdrop}$*

Simulation Methodology. We use PhoenixSim [35] to evaluate the performance overheads of various encryption schemes using 16 antennas distributed over the chip with uniform random broadcast traffic. We model the encryption process as a pipelined delay at both the transmitter (encrypting) and receiver (decrypting). The delay of each pipeline stage is the number of cycles required in hardware to produce a block or bit of the encrypted message. The rest of the network remains consistent with a standard WNoC: see Section V-A and [36], [37] for more details on the PhoenixSim and the modifications made to accommodate a WNoC.

Results. Figure 7 plots the average message latency as a function of injection rate across the encryption schemes and assuming different wireless transmission speeds. On average across channel bandwidths, we observed that adding Py encryption saturates the network at 90% of the unencrypted traffic saturation rate, whereas the RC4A and AES implementations saturate much earlier, at 50% and 24% respectively. The secure-hash implementation proposed in [12] is reportedly even slower than AES, making it a non-starter.

Current technologies allow for a 16 Gbps channel for wireless transmissions [13]. With this bandwidth, the network with encryption saturates at the same injection rate as the standard network and, therefore, it does not suppose a bottleneck at the moment. With respect to latency, Py adds less than 5 cycles of overhead. Such delay is tolerable in manycores, where long-range transfers take several tens of cycles, as well as in broadcast-oriented architectures [37]. RC4A and AES increase latency by 80% and 260% compared to unencrypted traffic, respectively, becoming a heavy burden.

As we increase the channel bandwidth to 32 Gbps, which would be possible in the future by using more complex modulations or frequencies in the 90 GHz band or beyond, Py becomes the only option that does not reduce throughput. Py starts to affect throughput as the bandwidth is further pushed up to 64 Gbps, currently unfeasible. Specifically, throughput is reduced by 30% and the 5-cycle added delay supposes a 46% increase over the unencrypted delay.

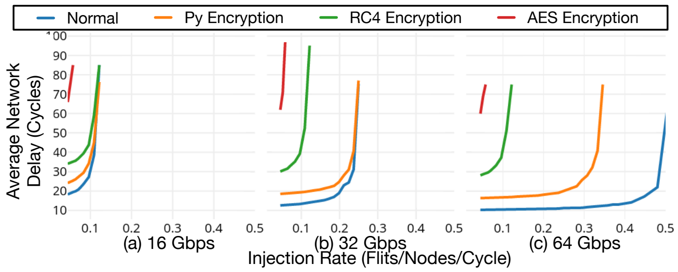


Fig. 7: Average message latency (in cycles) using proposed encryption schemes in (a) 16, (b) 32 and (c) 64 Gbps channel.

The main conclusion of this scalability analysis is that lightweight encryption mechanisms like Py are a reasonable option in the mid term for WNoC, but that faster alternatives will be required further down the road. A workaround to this problem would be to not encrypt non-sensitive messages that need to traverse the network, leaving the sensitivity decision to Prometheus and the NIF. Finally, secure-hash and AES result in unacceptable performance penalties in any case.

VI. Implementation Cost

Prometheus incurs power and area overheads stemming from the implementation of its three mechanisms. To evaluate the overheads of the digital part of Prometheus, we synthesized RTL implementations of each module and a state-of-the-art mesh router targeting a clock frequency of 1 GHz and using Synopsys Design Compiler with the Nangate 15nm FreePDK library [39]. For the analog parts, area and power have been estimated using designs from the literature that meet the requirements of Prometheus. Conservatively, we do not scale down the cost of analog components to 15 nm.

The results are summarized in Table II. Prometheus, when containing the three protection modules, occupies 0.025 mm^2 of silicon area and consumes less than 8 mW at 15 nm. However, this is only required in a few cores; whereas the rest can be safe by incorporating Prometheus$_{\text{Eavesdrop}}$ only, which results in an overhead of 4.67 mW and 0.008 mm^2. In comparison, a 5-port 128-bit 3-VC router synthesized at 15 nm would consume 20.96 mW and 0.023 mm^2, whereas recent 22-nm estimations of a wireless transceiver point towards a cost of more than 16 mW and 0.1 mm^2 [3], [14]. Prometheus thus incurs a reasonable overhead, especially taking into consideration that several cores share a single transceiver in typical WNoC designs. The overhead decreases proportionately to the sharing degree and can be further cut down if any of the submodules is not really needed.

The cost of Prometheus$_{\text{DoS}}$ is affordable as the module only consists of a logic circuit that evaluates Eq. 1 and a comparator. The circuit can be broken down into several floating point registers, adders and multipliers. Since the operation of Prometheus$_{\text{DoS}}$ is not time-critical, we apply optimization techniques to minimize the cost. Our 15-nm implementation yields a power of 1.46 mW and an area of 0.002 mm^2.

Prometheus$_{\text{Spoof}}$ also incurs minor area and power overheads. Since the anti-spoof mechanism needs to be placed only in a few locations independently of the network size, the cost

TABLE II: Breakdown of the area and power overheads of Prometheus at 15 nm

Module		**Power**	**Area**
Prometheus$_{\text{DoS}}$	Digital logic	1.46 mW	0.002 mm^2
Prometheus$_{\text{Spoof}}$	Power detector	0.8 mW	0.006 mm^2
	Data converter	0.67 mW	0.004 mm^2
	Digital logic	0.37 mW	0.004 mm^2
Prometheus$_{\text{Eavesdrop}}$	Py implementation	4.67 mW	0.007 mm^2
Total (at the corners)		7.97 mW	0.023 mm^2
Total (other wireless interfaces)		6.13 mW	0.009 mm^2
Router		20.96 mW	0.023 mm^2
Wireless Transceiver (estimated)		16 mW	0.1 mm^2

TABLE III: Power and Area for proposed encryption modules

Encryption Scheme	**Power**	**Area**
AES	91.76 mW	0.085 mm^2
RC4A	4.81 mW	0.008 mm^2
Py	4.67 mW	0.007 mm^2

is scalable. Individually, we estimate that each Prometheus$_{\text{Spoof}}$ module will consume less than 0.02 mm^2 of area and 2 mW of power. The main contributions to this cost are from the Power Detector (PD) and the Analog-to-Digital Converter (ADC). Regarding the PD, designs as small as 0.006 mm^2 consuming less than 1 mW are capable of meeting the dynamic range requirements set in Section V [40]. Regarding the ADC, prototypes in 40-nm CMOS operating at $\sim$1 GS/s with 6-bit resolution have been reported to occupy 0.004 mm^2 and consume 5.3 mW. Since spoofing protection is performed on a per-packet basis, the frequency requirements can be relaxed, thereby reducing the power consumption. At 16 Gbps, 128-bit flits are wirelessly transferred in 8 nanoseconds, which yields a sampling requirement of 125 MS/s. We therefore assume a power consumption of 0.67 mW.

The overheads of Prometheus$_{\text{Eavesdrop}}$ come from the implementation of the encryption/decryption modules. Our evaluation results with FreePDK are summarized in Table III. On the one hand, AES increases power by 90 mW and area by 0.085 mm^2 per wireless interface, which represents an overhead of 538% and 464% with respect a NoC router. Thus, AES and similar solutions presented in related work [12] are unacceptable. On the other hand, the power and area overheads of RC4A and Py are 4.67 mW and 0.007 mm^2, which are modest numbers considering the size and power of cores in a manycore processor. We obtained such similar results because RC4A and Py are stream cyphers (Py is built from RC4A).

VII. Related Work

Secure Wireless Networks-on-Chip. Ganguly et al. [11] leverage 24 wireless channels to create small-world topologies capable of mitigating, but not preventing, a DoS attack from a single node. Besides 24 channels being unrealistic, this scheme cannot handle a distributed DoS attack or an attack on a single shared wireless channel, both of which Prometheus$_{\text{DoS}}$ handles. To prevent eavesdropping, a hash-based authentication [12] has been proposed which incurs unacceptable latency overheads as we demonstrated in Section V-C. There has been no work on countering spoof attacks in a WNoC to the best of our knowledge. Prometheus is the first work in WNoCs on a comprehensive solution to provide security.

Secure Wired Networks-on-Chip. In Wired NoCs, researchers have looked into mitigating HTs DoS attacks via performing deep packet inspection [4] and injecting faults in an attempt to activate HTs in order to detect them and prevent them from creating DoS packets [19]. prevent HT DoS attacks [41]. Spoofs can be handled by standards for securing NoCs and the access rights to memory units [5]. Anti-eavesdropping using an AES-like symmetric key encryption combined with an asymmetric key encryption has also been explored [7].

Secure Wireless Sensor Networks. Most wireless networks use heavyweight software-based solutions that are unsuitable in a WNoC environment. Lightweight solutions are used in wireless sensor networks. One proposal combats spoofing, secrecy, and DoS attacks by means of neighbor specific keys, node specific sink keys, and data delivery techniques [9]. Another proposal use the RC6 stream cipher [8].

VIII. Conclusion

In this paper, we proposed a new microarchitecture to secure WNoCs from vulnerabilities associated with wireless communications. Our scheme, called Prometheus, is a low-cost drop-in hardware-only solution that detects DoS and spoof attacks by leveraging network statistics and RF power profiles, and protects against eavesdropping via low-cost encryption. Results show that with small increases in power, area, and latency, Prometheus can detect and reconcile HT or faulty hardware in the WNoC. Prometheus$_{\text{DoS}}$ detects *selfish* nodes in the WNoC using network broadcast predictions for token-based arbitration and network statistics for CSMA networks. Using RF power profiles, Prometheus$_{\text{Spoof}}$ detects a node attempting to spoof the source address of another node in the system using its received power profile. Lastly, Prometheus$_{\text{Eavesdrop}}$ secures communication between nodes in the system by properly encrypting messages sent between the two nodes. The three components of Prometheus protect and defend the WNoC from attacks that could lead to performance loss and eventual breakdown of the network.

References

[1] S. Deb *et al.*, “Wireless NoC as Interconnection Backbone for Multicore Chips: Promises and Challenges,” *IEEE Trans. Emerg. Sel. Topics Circuits Syst.*, vol. 2, no. 2, 2012.

[2] D. Matolak *et al.*, “Wireless networks-on-chips: architecture, wireless channel, and devices,” *IEEE Wireless Commun.*, vol. 19, no. 5, 2012.

[3] S. Abadal *et al.*, “WiSync: An Architecture for Fast Synchronization through On-Chip Wireless Communication,” in *ASPLOS ’16*, 2016.

[4] T. Boraten and A. K. Kodi, “Packet security with path sensitization for nocs,” in *DATE ’16*, 2016.

[5] L. Fiorin *et al.*, “Secure Memory Accesses on Networks-on-Chip,” in *IEEE Trans. Comput.*, vol. 57, 2008.

[6] A. Waksman and S. Sethumadhavan, “Tamper Evident Microprocessors,” in *IEEE SP ’10*, IEEE, 2010.

[7] C. Gebotys and R. Gebotys, “A Framework for Security on NoC Technologies,” in *IEEE ISVLSI ’03*, 2003.

[8] S. Slijepcevic *et al.*, “On communications security in wireless ad-hoc sensor networks,” in *IEEE WET ICE ’02*, 2002.

[9] K. Ren *et al.*, “Leds: Providing location-aware end-to-end data security in wireless sensor networks,” in *IEEE Trans. Mobile Comput.*, vol. 7, 2008.

[10] B. Lebiednik *et al.*, “Spoofing prevention via rf power profiling in wireless network-on-chip,” in *ACM AISTECS ’18*, 2018.

[11] A. Ganguly *et al.*, “A Denial-of-Service Resilient Wireless NoC Architecture,” in *ACM GLSVLSI ’12*, 2012.

[12] F. Pereñíguez-García *et al.*, “Secure Communications in Wireless Network-on-Chips,” in *AISTECS ’17*, 2017.

[13] X. Yu *et al.*, “Architecture and Design of Multi-Channel Millimeter-Wave Wireless Network-on-Chip,” *IEEE Design & Test*, vol. 31, 2014.

[14] S. Laha *et al.*, “A New Frontier in Ultralow Power Wireless Links: Network-on-Chip and Chip-to-Chip Interconnects,” *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 34, pp. 186–198, 2015.

[15] D. DiTomaso *et al.*, “A-WiNoC: Adaptive Wireless Network-on-Chip Architecture for Chip Multiprocessors,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 12, 2015.

[16] N. Mansoor and A. Ganguly, “Reconfigurable Wireless Network-on-Chip with a Dynamic Medium Access Mechanism,” in *NoCS ’15*, p. Article 13, 2015.

[17] A. Mestres *et al.*, “A MAC protocol for Reliable Broadcast Communications in Wireless Network-on-Chip,” in *NoCArc ’16*, 2016.

[18] V. Bharghavan *et al.*, “MACAW: a media access protocol for wireless LAN’s,” in *ACM SIGCOMM ’94*, 1994.

[19] T. Boraten and A. K. Kodi, “Mitigation of Denial of Service Attack with Hardware Trojans in NoC Architectures,” in *IPDPS ’16*, 2016.

[20] H. Wu *et al.*, “Bond wire antenna/feed for operation near 60 GHz,” *IEEE Trans. Microw. Theory Tech.*, vol. 57, no. 12, 2009.

[21] J. Branch *et al.*, “Wireless communication in a flip-chip package using integrated antennas on silicon substrates,” *IEEE Electron Device Lett.*, vol. 26, no. 2, 2005.

[22] Y. Xiao *et al.*, “Game Theory Models for IEEE 802.11 DCF in Wireless Ad Hoc Networks,” *IEEE Commun. Mag.*, vol. 43, 2005.

[23] S. Abadal *et al.*, “Broadcast-Enabled Massive Multicore Architectures: A Wireless RF Approach,” *IEEE MICRO*, vol. 35, no. 5, 2015.

[24] F. Triviño *et al.*, “Virtualizing network-on-chip resources in chip-multiprocessors,” *Microprocessors and Microsystems*, vol. 35, 2011.

[25] H. Shi *et al.*, “Fairness in wireless networks: Issues, measures and challenges,” *IEEE Commun. Surveys Tuts.*, vol. 16, 2014.

[26] M. Agrawal and P. Mishra, “A Comparative Survey on Symmetric Key Encryption Techniques,” *IJCSE*, vol. 4, no. 5, 2012.

[27] J. Rott, “Intel Advanced Encryption Standard (AES-NI),” in *Intel Developer Zone*, 2012.

[28] D. Matolak *et al.*, “Channel modeling for wireless networks-on-chips,” *IEEE Commun. Mag.*, vol. 51, no. 6, 2013.

[29] M. Bellare and P. Rogaway, “Introduction to modern cryptography,” in *U.S. Department of Commerce*, vol. 207, 2005.

[30] D. Hwang *et al.*, “AES-Based Security Coprocessor IC in 0.18-um CMOS with Resistance to Differential Power Analysis Side-Channel Attacks,” *IEEE J. Solid-State Circuits*, vol. 41, no. 4, 2006.

[31] P. Crowley, “Improved cryptanalysis of Py,” in *SASC 2006 Stream Ciphers Revisited*, 2006.

[32] A. Maximov, “Two Linear Distinguishing Attacks on VMPC and RC4A and Weakness of RC4 Family of Stream Ciphers,” *Fast Software Encryption*, vol. 3557, 2005.

[33] P. Prasithsagaree and P. Krishnamurthy, “Analysis of Energy Consumption of RC4 and AES Algorithms in Wireless LANs,” in *IEEE GLOBECOM ’03*, vol. 3, 2003.

[34] E. Biham and J. Seberry, “Py (Roo): A Fast and Secure Stream Cipher using Rolling Arrays,” in *The ENCRYPT eSTREAM project*, 2005.

[35] J. Chan *et al.*, “PhoenixSim: A Simulator for Physical-Layer Analysis of Chip-Scale Photonic Interconnection Networks,” in *DATE ’10*, 2010.

[36] S. Abadal *et al.*, “Scalability of Broadcast Performance in Wireless Network-on-Chip,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, 2016.

[37] S. Abadal *et al.*, “OrthoNoC: A Broadcast-Oriented Dual-Plane Wireless Network-on-Chip Architecture,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, no. 3, 2018.

[38] L. Yan and G. W. Hanson, “Wave propagation mechanisms for intra-chip communications,” *IEEE Trans. Antennas Propag.*, vol. 57, 2009.

[39] M. Martins *et al.*, “Open cell library in 15nm freepdk technology,” in *ISPD*, ACM, 2015.

[40] A. Serhan *et al.*, “Common-Base/Common-Gate Millimeter-Wave Power Detectors,” *IEEE Trans. Microw. Theory Tech.*, vol. 63, no. 12, 2015.

[41] A. Kulkarni *et al.*, “SVM-based real-time hardware Trojan detection for many-core platform,” in *IEEE ISQED ’16*, 2016.