TARN: Trust Aware Routing To Enhance Security In 3D Network-on-Chips

Hasin Ishraq Reefat*, Alec Aversa†, Ioannis Savidis† and Naghmeh Karimi*.
*CSEE Department
University of Maryland Baltimore County
Baltimore, MD, US
†ECE Department
Drexel University
Philadelphia, PA, US

***Abstract*—The growing complexity and performance demands of modern computing systems resulted in a shift from traditional System-on-Chip (SoC) designs to Network-on-Chip (NoC) architectures, and further to three-dimensional Network-on-Chip (3D NoC) solutions. Despite their performance and power efficiency, the increased complexity and inter-layer communication of 3D NoCs can create opportunities for adversaries who opt to prevent reliable communications between embedded nodes by inserting hardware Trojans in such nodes. The hardware Trojans, introduced through untrusted third-party Intellectual Property (IP) blocks, can severely compromise 3D NoCs by tampering with data integrity, misrouting packets, or dropping them; thus triggering denial-of-service attacks. Detecting such behaviors is particularly difficult due to their infrequent activation. Thereby it is of utmost importance to take the trustworthiness of the embedded nodes into account when routing the packets in the NoCs. Accordingly, this paper proposes a trust-aware routing scheme, so-called TARN, to significantly reduce the rate of packet loss that can occur due to malicious behaviors of one or more nodes (or interconnects). Our distributed trust-aware path selection protocol bypasses malicious IPs and securely routes packets to their destination. Furthermore, we introduce a low-overhead mechanism for delegating trust scores to neighboring routers, thereby enhancing network efficiency. Experimental results demonstrate significant improvements in packet loss while imposing low performance and energy overhead.**

I. Introduction

With the increasing complexity of System-on-Chips (SoCs), efficient on-chip communication has become crucial. Network-on-Chips (NoCs) offer a scalable, high-performance solution [1]–[3], with three-dimensional NoCs (3D NoCs) providing enhanced performance and energy efficiency [4]. However, the complexity and shared resources in NoCs have introduced security vulnerabilities [5], particularly due to the widespread use of third-party hardware Intellectual Property (IP) to meet time-to-market demands. This reliance on untrusted IP raises concerns about the trustworthiness and security of the chip [6], and in particular hardware Trojans inserted by third-party vendors [7]–[10].

Hardware Trojans in NoC nodes or communication lines can disrupt the network by dropping or misrouting packets, extracting data, tampering with transmissions, or launching distributed Denial-of-Service (DoS) attacks [11]. This issue is exacerbated in 3D NoCs, where attackers can exploit vulnerabilities in one layer to manipulate inter-layer communications [12]. Hardware Trojans are subtle additions to the network architecture and are rarely activated, making traditional testing methods ineffective. Their minimal impact on circuit characteristics like power and delay also makes them difficult to detect via side-channel analysis. As a result, runtime monitoring countermeasures have been proposed. However, these often rely on a central control unit with dedicated connections to each node, which itself can be a Trojan target. One such method is [13] which deploys an insitu hardware Trojan detection module that relies on verification units embedded in NoC components to detect such Trojans. Also [14] detects malicious activities at runtime by broadcasting predefined test packets. In addition to requiring a central control unit, this method suffers from high latency due to the increased traffic [15].

Traditionally, secure data transmission in networks like wireless sensor networks, IoT, and mobile ad-hoc networks has relied on trust scoring to identify trusted nodes or paths [16], [17]. Trust scoring reflects a subjective assessment of an entity's behavior. Extending this concept, trust-aware routing protocols have been developed for NoC architectures to counter hardware Trojan attacks [18]–[20]. These protocols use various mechanisms to update and manage trust scores, with one common method relying on acknowledgment-based feedback. In this method, when a sender transmits a packet, the destination node sends an acknowledgment back, and routers update trust scores of neighboring routers based on the feedback. However, this approach requires storing the full routing path in the packet header, leading to significant overhead.

Another method for updating trust scores is through packet retransmission [20]. If an acknowledgment is not received, the sender retransmits the packet. Routers detect the retransmission via the packet ID and adjust trust scores of neighboring nodes accordingly. This method requires the packet to follow the same route for accurate updates and routers to store the packet ID and previous routing direction. Beyond the storage requirements, both aforementioned trust-scoring approaches also increase communication overhead due to periodic exchanges of trust scores with neighboring routers.

The third approach for updating trust scores employs a central controller to oversee both routing and trust management across the entire network [19]. In this method, routers periodically report their status to the central system, which then updates trust scores and determines routing paths. However, this approach demands dedicated communication channels with each router, leading to considerable overhead.

To fill the gap and address the above challenges, this paper introduces a low-cost distributed trust-aware routing protocol that leverages extended trust information gathered from not only the neighbor nodes but also from two hops away. Unlike previous methods, the proposed approach, so-called TARN, eliminates the need to store the routing path, packet ID, or other packet-specific data, requiring only the storage of trust scores for nodes one and two hops away. This significantly

978-3-9826741-0-0/DATE25/© 2025 EDAA

reduces overhead while maintaining security.

As will be discussed in more detail in Sec. III, *our proposed scheme offers significant benefits for 3D NoCs. It excels in selecting secure paths across different planes and multiple layers while minimizing the need for frequent layer switching, particularly when some routers lack vertical links. Additionally, this method provides multiple alternative routes for acknowledgments to return to the sender node in 3D NoCs, reducing the risk of single points of failure.* In sum, our major contributions are as follows:

- Proposing a trust-aware routing protocol for 3D NoCs that considers the trust scores of nodes within 2 hops when deciding the routing path, thereby enhancing security;
- Achieving a low-cost solution by avoiding the storage of the entire routing path despite using an acknowledgment scheme to assist in updating trust scores;
- Delegating trust to neighboring nodes by utilizing specific bytes in the packet header during following transmissions; eliminating the communication overhead for trust delegation while keeping the overall packet size overhead low;
- Achieving a significant improvement in packet loss rates, overall performance, and energy efficiency in 3D NoCs;
- Extensive high-level simulations while resembling Trojan activation in different timestamps in one (or more) NoC nodes and demonstrating the high efficiency of the proposed scheme in transferring the packets.

II. Research Background

A. Related Works

Various routing algorithms have been developed for NoC architectures to address latency challenges, including adaptive congestion-aware routing [21], fault-tolerant routing [22], [23], thermal-aware routing [24], and aging-aware routing [25]. However, the focus of these schemes is only the network latency, and they do not address the transmission of the packets in the presence of malicious intruders.

Several schemes have been proposed to enhance security against DoS attacks (e.g., [14], [26]), yet they often involve broadcasting diagnostic messages to detect and localize the malicious nodes, leading to high communication overhead. SECTAR [27] addresses hardware Trojans causing packet misrouting by bypassing affected nodes in future transmissions. While cost-effective, SECTAR is limited to XY routing mechanisms [4] and may struggle to manage multiple malicious nodes effectively [19].

Trust-aware routing schemes ensure that data packets follow secure and reliable paths; thus reducing the risk of breaches or corruption. In [18], direct and indirect trust scoring mechanisms were introduced to route packets through trusted routers, adapting from the congestion-aware DyXY protocol [28] by replacing congestion values with trust values. However, this method needs the sender to send a second request only after the first one is served; thus requiring the sender to maintain pending requests and adding a flag to the header, thereby increasing memory overhead. TROP, an opportunistic trust-aware routing strategy [20], also adapts the DyXY routing protocol to circumvent hardware Trojans by using a retransmission table to manage trust values, which increases memory usage during runtime. MARTR [19], an adaptive trust-based approach learned from [29], [30], avoids malicious routers by assessing the trustworthiness of each node based on the ratio of flits correctly processed by that node to the total number of flits processed in the network at each timestamp. This assessment is performed by a central controller based on reports received from trusted network interfaces, which gather information from the embedded nodes. Although secure, this approach introduces significant system overhead.

In 3D NoCs, addressing TSV vulnerabilities like crosstalk and signal integrity issues [31] is critical, as adversaries can exploit malicious workloads to accelerate aging and disrupt signals [32]. The 3D-LeukoNoC architecture [33] mitigates TSV-related threats with a biologically-inspired packet management system but struggles with scalability. Similarly, 3D-SECTSV [34] improves scalability using a status broadcast system but remains limited to TSV island-based designs.

Despite growing interest in trust-aware routing and TSV vulnerabilities in NoC research, there's a clear gap in addressing trust-aware routing for 3D NoC architectures. This underscores the need for scalable and low-overhead trust-based routing protocols that can accommodate the complexities of 3D NoCs. Thus, this paper focuses on developing a trust-aware routing scheme that highly benefits 3D NoCs.

B. System Model

We consider a 3D NoC with Through-Silicon Vias (TSVs) for vertical connections across the 3D stack (Fig. 1), reducing data transfer latency and improving communication efficiency. Each Processing Element (PE) is connected to a router, which handles data transfer to neighboring nodes via bidirectional links in 6 directions: East, West, North, South, Up, and Down. The router also manages data routing, virtual channel (VC) allocation to reduce congestion, and switch allocation for optimized packet paths via dynamic crossbar assignment.

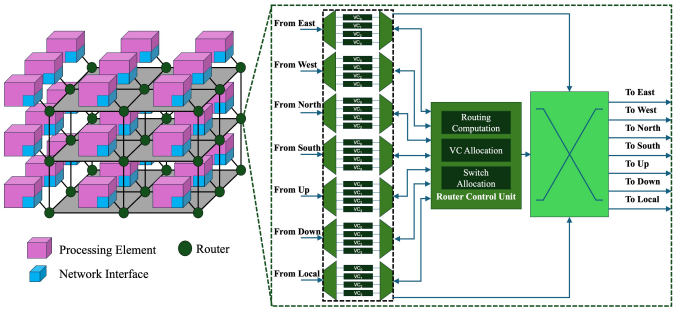


Fig. 1: System model.

Each Processing Element (PE) connects to its router via a dedicated Network Interface (NI), which generates packets from PE data and forwards them to the router for transmission through the NoC. The NI also receives incoming packets from the NoC and delivers them to the corresponding PE.

C. Threat Model

Our threat model (Fig. 2) assumes that some NoC nodes are compromised by hardware Trojans where such a node, upon receiving a packet, drops it instead of forwarding it, leading to packet loss and potential Denial of Service if critical packets

are consistently lost. Accordingly, to address such a threat we propose a trust-aware routing to mitigate the packet loss rate. Note that addressing other malicious behaviors such as data corruption is out of scope of this paper.

Note that this paper does not address the circuit-level details of Trojan implementation but focuses on mitigating packet loss at the system level in the presence of such malicious behaviors. In our threat model, Trojans remain stealthy to avoid detection, becoming active in specific nodes at particular times (potentially overlapping). However, *as a baseline for comparison, we also consider the case where a node is permanently malicious.*

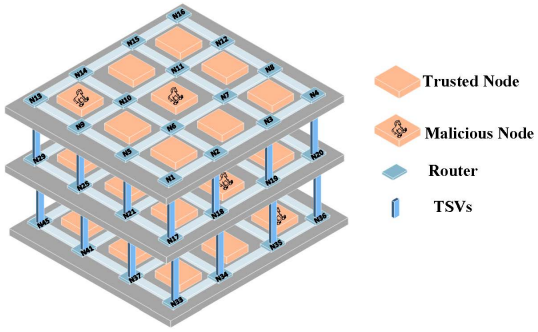


Fig. 2: Network architecture where some nodes are malicious.

III. TARN Methodology

Here we present our proposed trust-aware routing scheme. TARN relies on a distributed control mechanism where each node determines the routing direction by taking the trust scores of its immediate neighbors as well as the nodes two hops away into account. This decentralized approach eliminates the need for a central control unit, ensuring secure transmission between nodes. Note that this method can be leveraged for a broad range of NoC architectures with different topologies.

As mentioned in Sec. II-C, our threat model considers packet-dropping attacks by malicious nodes, potentially leading to denial of service. Detecting and bypassing compromised routers in runtime is challenging due to unpredictable malicious activity. To mitigate packet loss, TARN employs a trust-scoring mechanism where nodes build trust based on packet-handling history. *TARN can also be applied to 2D NoCs, but it suits 3D NoCs better due to the larger number of routing paths, offering more options for trusted paths.* TARN details and its integration into routing are discussed below.

A. TARN Trust Scoring

TARN establishes trust by leveraging both direct and indirect (i.e., delegated) trust assessments where the former is derived from a node's firsthand interactions with its neighbors, while the latter involves a neighbor node recommending the trustworthiness of another node; thereby delegating trust. To enhance security while not imposing communication overhead for exchanging trust scoring, TARN considers nodes 2 hops away for indirect trust calculation, yet instead of broadcasting the trust scores, it embeds trust scores in packet headers, allowing the receiver to use them for routing decisions when needed. This will be discussed in Sec. III-D.

Fig. 2 shows an example of a $m \times n \times l$ NoC, used in this section to discuss the trust assessment process in TARN. The distance between each two nodes N_a and N_b, located at (x_1, y_1, z_1) and (x_2, y_2, z_2) coordinates, which refers to the number of hops N_a is away from N_b can be found by Eq. 1.

$$D_{N_a,N_b} = |x_1 - x_2| + |y_1 - y_2| + |z_1 - z_2| \quad (1)$$

In a 3D NoC, each node N_a can have up to 6 neighbors, i.e., there are up to 6 nodes with a distance of 1 from such a node. On the other hand, each of those neighboring nodes can have up to 5 additional neighbors. Thus, each node can have a maximum of 30 nodes in a distance of 1 or 2 hops away. Indeed, when a packet is sent out from an embedded node, TARN considers the trust scores of all nodes in the distance of 1 and 2 to determine the next node the packet should be routed to. Thereby path selection to route a packet from node N_S to N_D is performed in a distributed manner, i.e., each node receiving a packet makes a local decision about the next hop for that packet. In other words, N_S selects a neighboring node, say N_F where $D_{N_S,N_F} = 1$, through which the packet should be transferred. In turn once N_F receives the packet, it then determines the next hop from its neighbors (excluding N_S).

The trust score (TS) for each node in TARN is a fractional value between 0 and 1 where the higher the value the more trustworthy the node is. In this scheme, the router embedded in each node keeps the trust scores for the nodes with a distance of 1 and 2, in a lookup table called the Trust Table (TT), and uses those values upon receiving a packet to decide where the packet should be routed to (i.e., selecting the next node among the neighboring ones), i.e., the TT of each node N_f only stores the values of TS_{N_f,N_q} that satisfies $1 \leq D_{N_f,N_q} \leq 2$.

Initially, all nodes populate their TT with a score of 1 for all entries (i.e., nodes within 2 hops). These scores are then dynamically updated at runtime based on the history of packet transfer status, including successful deliveries or losses, and are used to guide future packet routing (details in Sec. III-C).

In TARN, the packet transmission process begins with a sender node N_S sending a packet P to a destination node N_D. As P moves through the network, intermediate nodes are chosen based on the trust-aware routing process in Sec. III-B. Once P reaches N_D, an acknowledgment A is generated by N_D and sent back to N_S (lines 1-4 of Alg. 1). A key feature of this protocol is that the acknowledgment A doesn't necessarily need to follow the same route as P traveled; thus significantly reducing the memory overhead by eliminating the need to store the original path for each packet. Indeed, that route that A traverses to reach N_S, itself depends on the network conditions and routing decisions. In a 3D NoC, the availability of multiple routing options for A mitigates risk of single points of failure.

B. Trust-Aware Routing

As mentioned earlier, the routing decision at each node is taken locally by considering the Trust Score (referred to as TS hereafter) of the nodes within 2 hops (lines 10-23 of Alg. 1). Thereby in TARN each router gains a more comprehensive view of the trustworthiness of the routing path compared to the cases where only the direct trust is taken into account. In general, for the packet P to be sent from the sender N_S to the final destination (say N_D), N_S is to determine the next node to route P to among its neighboring nodes, i.e., among all nodes that satisfy $D_{N_S,N_F} = 1$. To make such a selection,

N_S calculates the overall trust shown as OTS_F, in Eq. 2. In other words, TS is assessed by averaging the delegated trust scores of two-hops-away routers and adding this to the direct trust scores of the corresponding immediate neighbors. The node with the highest cumulative trust score (here N_F^*) is then selected among all candidates for packet transmission. Note that OTS_F is not necessarily $\in [0, 1]$ and may exceed.

$$\forall N_F : D_{N_S,N_F} = 1$$
$$N_F^* = \arg\max_{N_F} \left(\overbrace{TS_{N_S,N_F} + \frac{1}{|N_M|} \sum_{N_M : D_{N_F,N_M}=1} TS_{N_F,N_M}}^{OTS_F} \right) \quad (2)$$

To explain why TARN considers both immediate neighbors and nodes two hops away when determining the routing path, let's consider an example. If N_S were to rely solely on direct trust scores, it might choose one of its immediate neighbors, say N_H (where $D_{N_S,N_H} = 1$) based on its highest trust score. However, if the other neighbors of N_H are untrustworthy, there is a risk that N_H could end up routing the packet back to N_S, leading to inefficiencies and potential security risks. By factoring in two-hop trust scores, TARN ensures more reliable and secure routing by evaluating the broader network.

C. Updating Trust Scores

Upon successfully receiving acknowledgment A from the destination N_D, the sender N_S increases the trust score of its neighbor node N_F^* through which the packet routed to N_D (recall Eq. 2) by a constant value α (In our experiment, we considered $\alpha = 0.1$) to boost confidence in N_F^*. If the trust score is already at its maximum (i.e., 1), it remains unchanged. Conversely, if A does not arrive within a specified timeout, N_S decreases TS_{N_S,N_F^*} by α, indicating potential packet loss during the last transmission (see lines 5-9 and 24-33 in Alg. 1).

TARN avoids updating the TS of entire routing path after each packet transmission or loss to minimize communication overhead, as it would otherwise require tracking and broadcasting updates for all routers along the path. While updating the TS of a neighbor node (i.e., TS_{N_S,N_F^*}) in its Trust Table, N_S also puts update flags corresponding to that neighbor (N_F^*), signaling that the new score should be shared in the next transmission to any other nodes N_q that satisfies $D_{N_S,N_q} = 1$ to be used in their TS assessment for future data transmissions.

D. Delegation of Trust Scores

For subsequent transmissions, when N_S sends another packet—whether to the same or another destination— it includes the updated trust score and the corresponding router ID in the packet header (lines 34-40 of Alg. 1). For example, as mentioned earlier, upon updating TS_{N_S,N_F^*} in N_S based on the status of packet P sent earlier from N_S to N_D through N_F^*, node N_S postpones delegating the updated TS until it sends another packet (say P') to N_D or another destination via a different neighbor N_q (where $D_{N_S,N_q} = 1$). The updated TS_{N_S,N_F^*} is then included in the header of P'.

In the packet header, specific bytes are allocated for transferring the updated TS_{N_S,N_F^*} and the router ID of N_F^*. By including this in the header, N_S delegates its trust for N_F^* to the neighboring router N_q. Once completed, N_S resets the update flag for N_F^*, indicating the TS has been successfully delegated.

Algorithm 1 TARN Methodology

```
/* Distance between 2 nodes N_a & N_b can be calculated as D_{N_a,N_b} (Eq. 1) */

Initialization: Set all TS values 1 initially

Input: Packet: P; Source: N_S; Destination: N_D
/* Packet P travels from N_S to N_D through N_F (D_{N_S,N_F} = 1) and Acknowledgment A comes back if N_D receives P */
1: P.trustRouting(N_S, N_D)
2: if N_D.receive(P) then
3:     N_D.sendAck(A)
4: end if
5: if N_S.receiveAck(A) then
6:     N_S.updateTrustscore(N_F, up)
7: else if ackTimout() then
8:     N_S.updateTrustscore(N_F, down)
9: end if

/* N_S sends a packet P to N_D, intermediate router N_F^* gets selected by trustRouting() function */
10: function trustRouting(N_S, N_D)
11:     if N_S == N_D then
12:         P reached at the destination N_D
13:     else if D_{N_S,N_D} == 1 then
14:         Route P towards N_D
15:     else
16:         for N_F do
17:             if D_{N_S,N_F} = 1 then
18:                 N_F^* = arg max_{N_F} (TS_{N_S,N_F} + 1/|N_M| Σ_{N_M:D_{N_F,N_M}=1} TS_{N_F,N_M})
19:             end if
20:         end for
21:         Route P towards N_F^*
22:     end if
23: end function

/* N_S updates TS of N_F^* based on updateStatus */
24: function updateTrustscore(N_F^*,updateStatus)
25:     if updateStatus == up then
26:         if TS_{N_S,N_F^*} ≠ 1 then
27:             TS_{N_S,N_F^*} ← TS_{N_S,N_F^*} + α
28:         end if
29:     else
30:         TS_{N_S,N_F^*} ← TS_{N_S,N_F^*} − α
31:     end if
32:     TS_{N_S,N_F^*}.updateFlag ← 1
33: end function

/* N_S delegates updated TS of N_F^* through new packet P''s header */
34: function delegateTrustscore(N_F^*)
35:     if TS_{N_S,N_F^*}.updateFlag == 1 then
36:         P'.header.nodeID ← N_F^*
37:         P'.header.trustScore ← TS_{N_S,N_F^*}
38:         TS_{N_S,N_F^*}.updateFlag ← 0
39:     end if
40: end function

/* N_q gets packet P', updates delegated TS of N_F^* came from N_S and adds the updated TS of N_S with P''s header */
41: function trustPropagation( )
42:     if P'.header.nodeID.available() then
43:         TS_{N_q,N_F^*} ← TS_{N_q,N_S} × P'.header.trustScore
44:         P'.header.nodeID ← NULL
45:         P'.header.trustScore ← NULL
46:     end if
47:     if TS_{N_q,N_S}.updateFlag == 1 then
48:         delegateTrustscore(N_S)
49:     end if
50: end function
```

E. Trust Score Propagation Across the Network

As packets with updated trust scores in their headers traverse the network, intermediate routers read and propagate these scores to neighboring routers (lines 41-50 of Alg. 1). For example, when packet P' reaches N_q (where $D_{N_S,N_q} = 1$), this node identifies trust score TS_{N_S,N_F^*} and router ID of N_F^* in the trust delegation field (where $D_{N_q,N_F^*} = 2$). N_q updates its local trust score TS_{N_q,N_F^*} by multiplying TS_{N_q,N_S} with TS_{N_S,N_F^*}, then clears the trust delegation field in the packet

header to avoid redundancy. If N_q has its own updated trust score for another neighboring router, say N_S, it can insert router ID of N_S and trust score TS_{N_q,N_S} into the now-empty trust delegation field in the P' header before forwarding the packet. This reuses packet header space, reducing overhead and enabling efficient trust score dissemination across the network, without separate broadcasts.

IV. Experimental Setup and Results

A. Experimental Setup

We used the Ratatoskr framework [35] to model a $5 \times 5 \times 3$ Mesh NoC system and evaluate TARN with synthetic traffic. Ratatoskr is a fast, cycle-accurate NoC simulator [36], featuring routers connected via bidirectional links with 4 virtual channels per port. Simulations ran for 10,000 ns with 8-flit packets, where packet size is 128 bytes. We compared TARN against XYZ, ZXYZ [37], HeteroXYZ [37], and the trust-based routing scheme in [18].

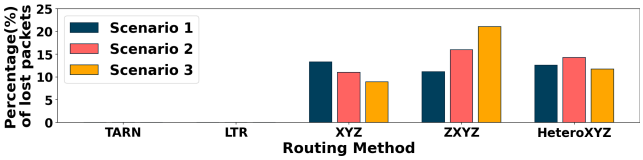
In our threat model, Trojans get activated in specific nodes over randomly selected time slots, though we also tested a permanently malicious node for baseline comparison. We consider two types of malicious activity: 1) a node N_a is always malicious and drops all received packets (Always-ON), and 2) N_a is intermittently malicious, dropping packets only when its Trojan is activated (Sometimes-ON). Our experiments also consider scenarios with multiple simultaneously malicious nodes.

Without loss of generality, for the Always-ON case, we randomly selected three scenarios in each 2 out of the 3 nodes N_{10}, N_{31}, and N_{67} (selected randomly among all nodes) were permanently malicious. For the Sometimes-ON case, we considered a dynamic situation and introduced up to M randomly timed malicious nodes, where $M \in \{7, 9, 11\}$, allowing for overlapping malicious activity in time. For each routing algorithm (TARN and previous work) and threat scenario (Always-ON vs. Sometimes-ON), we collected key performance and security metrics, including flit and packet latency, overall network latency, normalized energy consumption, and packet loss.

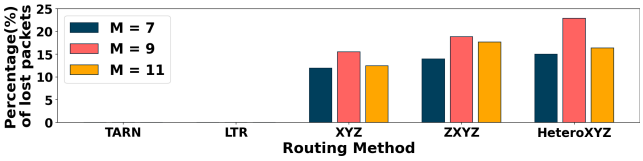
B. Experimental Results

*1) **Lost Packet Rate**:* Figure 3 shows the percentage of lost packets under various malicious conditions. In particular, Fig. 3a compares packet loss for scenarios with two consistently malicious nodes in each scenario, i.e., in each of Scenarios 1-3, two of three randomly selected nodes (N_{10}, N_{31}, and N_{67}) are consistently malicious, dropping all packets they receive. Note that we show the results for 3 cases to depict the stability of TARN in different cases. On the other hand, Fig. 3b shows loss rates during intermittent (Sometimes-ON) malicious activity when up to M nodes are malicious intermittently. In all cases of malicious activities, TARN achieves zero packet loss, even with up to 11 intermittently malicious nodes. In contrast, traditional algorithms (XYZ, ZXYZ, HeteroXYZ) suffer from significantly higher loss rate due to their lack of trust-aware mechanisms. Though the trust-based LTR algorithm from [18] performs similarly well in terms of packet loss, later we will show that TARN highly outperforms LTR in terms of energy and latency.

Figure 4 shows packet loss rates in real time over the 10,000ns simulation. The simulator tracks lost packets by



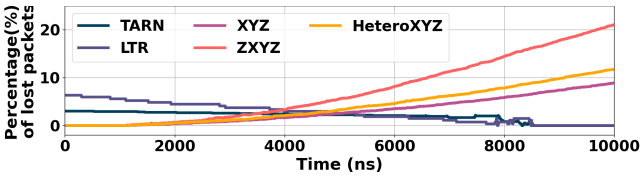
(a) When 2 nodes are malicious permanently. Each bar relates to the malicious activity of 2 nodes selected randomly.



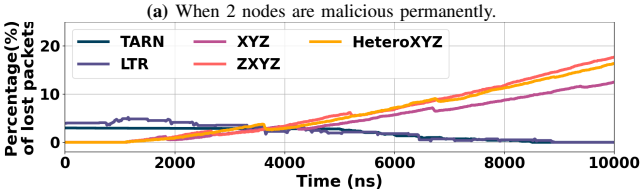
(b) When up to M nodes are intermittently malicious.

Fig. 3: Percentage of lost packets under different malicious conditions.

adjusting the count as packets are initiated and successfully delivered. TARN consistently achieves the lowest packet loss, stabilizing much earlier than other methods. Its flat curve after 8000 ns indicates strong resilience to prolonged attacks. In contrast, traditional algorithms show steadily increasing losses, highlighting their vulnerability. The LTR algorithm performs similarly to TARN but with slightly higher early losses before stabilizing. The takeaway is that TARN demonstrates superior resilience, quickly stabilizing and minimizing packet loss.



(a) When 2 nodes are malicious permanently.

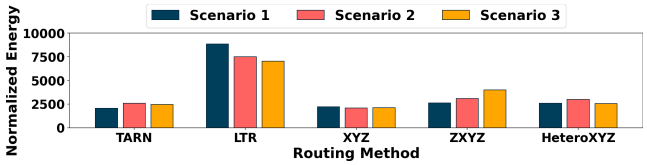


(b) When up to 11 nodes are intermittently malicious (M=11).

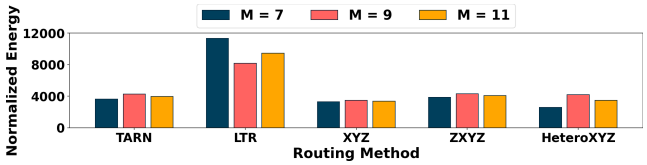
Fig. 4: Percentage of lost packets over time in different malicious conditions.

*2) **Normalized Energy**:* Figure 5 depicts the normalized energy consumption of routers under different malicious conditions. As shown TARN consumes significantly less energy than LTR, with 67.37% and 57.84% lower energy consumption during permanent (Fig. 5a) and intermittent (Fig. 5b) activities in the network. While TARN energy consumption is comparable to XYZ, ZXYZ, and HeteroXYZ, those schemes suffer from high packet loss. The key takeaway is that TARN dynamically adjusts its routing securely and efficiently without much energy overhead, even with unpredictable malicious nodes.

*3) **Average Flit and Packet Latency**:* Figure 6 shows average flit and packet latencies across routing algorithms. In scenarios with two permanently malicious nodes (Fig. 6a), TARN shows slightly lower latencies compared to others. The minimal difference between flit and packet latencies indicates efficient transmission. In cases with random malicious nodes (Fig. 6b), TARN maintains stable latencies, avoiding the significant increases seen in other algorithms as malicious

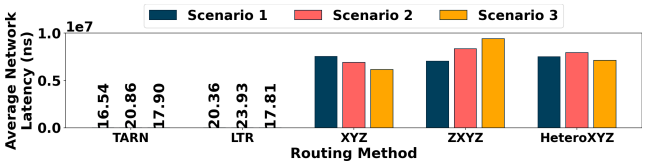


(a) When 2 nodes are malicious permanently. Each bar relates to the malicious activity of 2 nodes selected randomly.

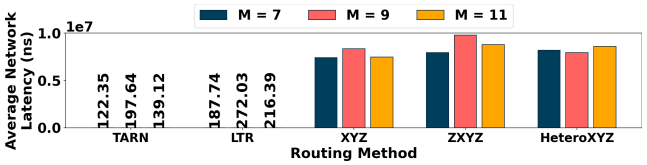


(b) When up to M nodes are intermittently malicious.

Fig. 5: Normalized Energy under different malicious conditions.



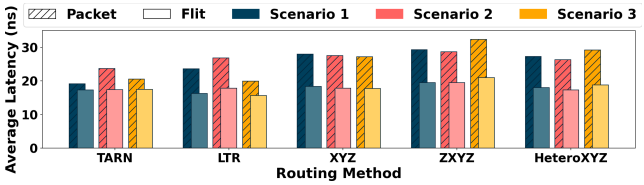
(a) When 2 nodes are malicious permanently. Each bar relates to the malicious activity of 2 nodes selected randomly.



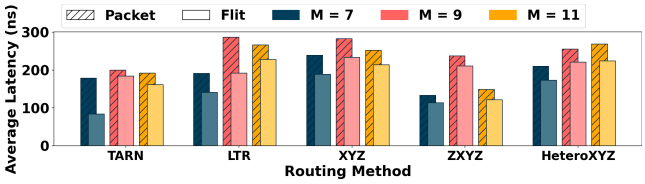
(b) When up to M nodes are intermittently malicious.

Fig. 7: Average network latency under different malicious conditions.

activity grows. TARN also outperforms the LTR algorithm by 24.65% lower in flit latency and 21.5% lower in packet latency under intermittent malicious behaviors. The takeaway is that TARN ensures low latency and secure timely packet delivery, which is crucial for high-performance NoC systems.



(a) When 2 nodes are malicious permanently. Each bar relates to the malicious activity of 2 nodes selected randomly.



(b) When up to M nodes are intermittently malicious.

Fig. 6: Average flit and packet latency under different malicious conditions. Hatched and solid bars represent average packet and flit latencies, respectively.

*4) **Average Network Latency**:* Figure 7 shows average network latency across routing algorithms under constant and random malicious conditions. In Fig. 7a, with two permanently malicious nodes, TARN Network latency is on average 377,000, 451,000, and 411,000 times lower than XYZ, ZXYZ, and HeteroXYZ, respectively. It also outperforms LTR with 10.34% lower latency on average. This occurs because these algorithms use a timeout for receiving packets, so any packet loss results in a delay equal to the timeout. With a high loss rate, network latency increases significantly. TARN maintains low network latency even as the number and timing of malicious nodes vary (Fig. 7b) with around 52,000, 58,000, 60,000 times lower latency compared to XYZ, ZXYZ, and HeteroXYZ respectively and 36.6% lower than LTR. This clearly confirms the superiority of TARN over other schemes.

*5) **Memory Overhead**:* TARN minimizes memory overhead by requiring each router to store only the trust scores of nodes within 2 hops, with a maximum of 145 bytes regardless of the NOC size (i.e., 4 bytes per score for each node within 2 hops since each trust score is stored in floating point, and 6 bits total—rounded to 1 byte—for update flags). Edge routers require even less memory. Trust score delegation adds just 5 bytes to the packet header—4 for the score and 1 for the router ID. In contrast, LTR requires significantly more resources, including 144 bytes for trust scores and additional memory to track packet IDs—e.g., 400 extra bytes to track 200 packet IDs. LTR's memory needs grow over time. LTR also imposes significant communication overhead by broadcasting trust scores, unlike TARN, which avoids broadcasting.

*6) **Tolerance**:* Figure 8 shows TARN's tolerance to malicious activities as the percentage of malicious nodes increases. As shown, even when 26% of nodes are malicious there is ≈0% packet loss for the intermittent case (which is the more realistic case considering the stealthiness of Trojans). Even with 40% intermittently malicious nodes, packet loss remains low at 13.14%. Permanent malicious activities result in higher packet loss, but overall if a node is permanently malicious it can be detected easily. The takeaway is that TARN effectively tolerates malicious activities to a high degree.

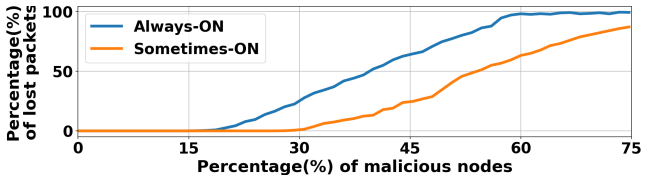


Fig. 8: Lost packet rate in TARN with increasing number of malicious nodes.

V. Conclusion and Future Directions

The increasing complexity of 3D NoCs makes them vulnerable to hardware Trojans embedded in untrusted IP blocks, which disrupt communication by dropping packets. Such malicious activities are hard to detect due to their stealthiness, emphasizing the need for trust-aware routing. Our low-cost TARN algorithm mitigates the impact of such malicious activities by incorporating trust-aware routing. TARN significantly reduces packet loss while ensuring reliable communication. With a distributed path selection and low-overhead trust delegation, TARN enhances security without significant memory, performance or energy costs. In the future, we will implement the TARN algorithm in hardware.

REFERENCES

[1] G. Dessouky, M. Isakov, M. A. Kinsy, P. Mahmoody, M. Mark, A.-R. Sadeghi, E. Stapf, and S. Zeitouni, "Distributed Memory Guard: Enabling Secure Enclave Computing in NoC-based Architectures," in *ACM/IEEE Design Automation Conf. (DAC)*, 2021, pp. 985–990.

[2] V. F. Pavlidis, I. Savidis, and E. Friedman, in *Three-dimensional Integrated Circuit Design (Second Edition)*. Morgan Kaufmann, 2017, pp. 1–768.

[3] D. Petrisko, C. Zhao, S. Davidson, P. Gao, D. Richmond, and M. B. Taylor, "Noc symbiosis (special session paper)," in *IEEE/ACM Int'l Symp. on Networks-on-Chip (NOCS)*, 2020, pp. 1–8.

[4] A. Sarihi, A. Patooghy, A. Khalid, M. Hasanzadeh, M. Said, and A.-H. A. Badawy, "A Survey on the Security of Wired, Wireless, and 3D Network-on-Chips," *IEEE Access*, vol. 9, pp. 107 625–107 656, 2021.

[5] S. Charles and P. Mishra, "A Survey of Network-on-Chip Security Attacks and Countermeasures," *ACM Computing Surveys (CSUR)*, vol. 54, no. 5, pp. 1–36, 2021.

[6] P. Mishra, S. Bhunia, and M. Tehranipoor, *Hardware IP security and trust*. Springer, 2017.

[7] A. Palumbo, L. Cassano, P. Reviriego, and M. Ottavi, "Improving the Detection of Hardware Trojan Horses in Microprocessors via Hamming Codes," in *IEEE Int'l Symp. on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, 2023, pp. 1–6.

[8] J. Francq and F. Frick, "Introduction to hardware Trojan detection methods," in *Design, Automation & Test in Europe Conf. & Exhibition (DATE)*, 2015, pp. 770–775.

[9] L. Wu, X. Zhang, S. Wang, and W. Hu, "Hardware Trojan Detection at LUT: Where Structural Features Meet Behavioral Characteristics," in *IEEE int'l symp. on hardware oriented security and trust (HOST)*, 2022, pp. 121–124.

[10] D. G. Mahmoud, W. Hu, and M. Stojilovic, "X-Attack: Remote activation of satisfiability don't-care hardware Trojans on shared FPGAs," in *Int'l Conf. on Field-Programmable Logic and Applications (FPL)*, 2020, pp. 185–192.

[11] 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 *IEEE Int'l Conf. on Networking, Architecture and Storage*, 2013, pp. 178–186.

[12] V. J. Kulkarni, R. Manju, R. Gupta, J. Jose, and S. Nandi, "Packet header attack by hardware trojan in NoC based TCMP and its impact analysis," in *IEEE/ACM Int'l Symp. on Networks-on-Chip*, 2021, pp. 21–28.

[13] M. Hussain, A. Malekpour, H. Guo, and S. Parameswaran, "EETD: An Energy Efficient Design for Runtime Hardware Trojan Detection in Untrusted Network-on-Chip," in *IEEE Computer Society Annual Symp. on VLSI (ISVLSI)*, 2018, pp. 345–350.

[14] R. Fernandes, C. Marcon, R. Cataldo, and J. Sepúlveda, "Using smart routing for secure and dependable NoC-based MPSoCs," *IEEE/ACM Transactions on Networking*, vol. 28, no. 3, pp. 1158–1171, 2020.

[15] J. Haase, N. Volkens, and D. Goehringer, "Embedded Security Accelerators within Network-on-Chip Environments," in *Int'l Symp. on Highly Efficient Accelerators and Reconfigurable Technologies*, 2024, pp. 37–43.

[16] A. A. Adewuyi, H. Cheng, Q. Shi, J. Cao, Á. MacDermott, and X. Wang, "CTRUST: A dynamic trust model for collaborative applications in the Internet of Things," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 5432–5445, 2019.

[17] S. S. Mehjabin, M. Younis, A. Tekeoglu, M. Ebrahimabadi, T. Sookoor, and N. Karimi, "PETIT: PUF-enabled trust evaluation framework for IoT networks," *Computer Networks*, p. 110772, 2024.

[18] S. Charles and P. Mishra, "Lightweight and trust-aware routing in NoC-based SoCs," in *Computer society annual Symp. on VLSI (ISVLSI)*, 2020, pp. 160–167.

[19] E. Franz and A. Grützner, "Trust-Based Adaptive Routing for NoCs," in *Int'l Conf. on Embedded Computer Systems*, 2023, pp. 296–310.

[20] S. Sankar, R. Gupta, J. Jose, and S. Nandi, "TROP: TRust-aware OPportunistic routing in NoC with hardware trojans," *ACM Trans. on Design Automation of Electronic Systems*, vol. 29, no. 2, pp. 1–25, 2024.

[21] S. Ramesh, K. Manna, V. C. Gogineni, S. Chattopadhyay, and S. Mahapatra, "Congestion-Aware Vertical Link Placement and Application Mapping Onto Three-Dimensional Network-On-Chip Architectures," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 2024.

[22] R. Xie, J. Cai, X. Xin, and B. Yang, "LBFT: a fault-tolerant routing algorithm for load-balancing network-on-chip based on odd–even turn model," *The Journal of Supercomputing*, vol. 74, pp. 3726–3747, 2018.

[23] S. Maabi, F. Safaei, A. Rezaei, M. Daneshtalab, and D. Zhao, "ERFAN: Efficient reconfigurable fault-tolerant deflection routing algorithm for 3-D Network-on-Chip," in *IEEE Int'l System-on-Chip Conf. (SOCC)*, 2016, pp. 306–311.

[24] R. Dash, A. Majumdar, V. Pangracious, A. K. Turuk, and J. L. Risco-Martín, "ATAR: An Adaptive Thermal-Aware Routing Algorithm for 3-D Network-on-Chip Systems," *IEEE Trans. on Components, Packaging and Manufacturing Technology*, vol. 8, no. 12, pp. 2122–2129, 2018.

[25] Z. Ghaderi, A. Alqahtani, and N. Bagherzadeh, "AROMa: aging-aware deadlock-free adaptive routing algorithm and online monitoring in 3D NoCs," *IEEE Trans. on Parallel and Distributed Systems*, vol. 29, no. 4, pp. 772–788, 2017.

[26] S. Charles, Y. Lyu, and P. Mishra, "Real-time detection and localization of distributed DoS attacks in NoC-based SoCs," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 39, no. 12, pp. 4510–4523, 2020.

[27] R. Manju, A. Das, J. Jose, and P. Mishra, "SECTAR: Secure NoC using Trojan aware routing," in *Int'l Symp. on Networks-on-Chip (NOCS)*, 2020, pp. 1–8.

[28] M. Li, Q.-A. Zeng, and W.-B. Jone, "DyXY - A Proximity Congestion-Aware Deadlock-Free Dynamic Routing Method for Network on Chip," in *Design Automation Conf. (DAC)*, 2006, pp. 849–852.

[29] T. Li, C. Hofmann, and E. Franz, "Secure and reliable data transmission in SDN-based backend networks of industrial IoT," in *IEEE Conf. on Local Computer Networks (LCN)*, 2020, pp. 365–368.

[30] M. Fattah, A. Airola, R. Ausavarungnirun, N. Mirzaei, P. Liljeberg, J. Plosila, S. Mohammadi, T. Pahikkala, O. Mutlu, and H. Tenhunen, "A Low-Overhead, Fully-Distributed, Guaranteed-Delivery Routing Algorithm for Faulty Network-on-Chips," in *Int'l Symp. on Networks-on-Chip*, 2015, pp. 1–8.

[31] T. Song, C. Liu, Y. Peng, and S. Lim, "Full-chip multiple TSV-to-TSV coupling extraction and optimization in 3D ICs," in *Annual Design Automation Conference*, May 2013, pp. 1–7.

[32] S. Das, K. Basu, J. R. Doppa, P. P. Pande, R. Karri, and K. Chakrabarty, "Abetting Planned Obsolescence by Aging 3D Networks-on-Chip," in *IEEE/ACM Int'l Symp. on Networks-on-Chip (NOCS)*, Oct. 2018, pp. 1–8.

[33] J. Sepulveda, G. Gogniat, D. Flórez, J. Diguet, C. Pedraza, and M. Strum, "3D-LeukoNoC: A dynamic NoC protection," in *Int'l Conf. on ReConFigurable Computing and FPGAs (ReConFig14)*, Dec. 2014, pp. 1–6.

[34] J. Sepúlveda, G. Gogniat, D. Flórez, J. Diguet, R. Pires, and M. Strum, "TSV protection: Towards secure 3D-MPSoC," in *IEEE Latin American Symp. on Circuits & Systems (LASCAS)*, Feb. 2015, pp. 1–4.

[35] J. M. Joseph, L. Bamberg, I. Hajjar, B. R. Perjikolaei, A. García-Ortiz, and T. Pionteck, "Ratatoskr: An Open-Source Framework for In-Depth Power, Performance, and Area Analysis and Optimization in 3D NoCs," *ACM Trans. on Modeling and Computer Simulation (TOMACS)*, vol. 32, no. 1, pp. 1–21, 2021.

[36] V. Catania, A. Mineo, S. Monteleone, M. Palesi, and D. Patti, "Cycle-Accurate Network on Chip Simulation with Noxim," *ACM Trans. on Modeling and Computer Simulation (TOMACS)*, vol. 27, no. 1, pp. 1–25, 2016.

[37] J. M. Joseph, L. Bamberg, D. Ermel, B. R. Perjikolaei, A. Drewes, A. García-Ortiz, and T. Pionteck, "Nocs in heterogeneous 3d socs: Co-design of routing strategies and microarchitectures," *IEEE Access*, vol. 7, pp. 135 145–135 163, 2019.