Poster: More is Less

Denial-of-Service Attacks and Solutions in Many-Core On-Chip Networks

Xin Zhang (student), Yanlin Li (student), and Onur Mutlu (faculty)
Carnegie Mellon University

1 Introduction

A many-core system is expected to outperform a traditional single-core system by enabling multiple applications to be executed on separate cores in parallel, *given the generous assumption that the execution of an application on one core does not interfere with an application on other cores*. Regrettably, this optimistic belief about independence of different applications turns out widely untrue in practice. Interdependencies and contention between cores abound in many-core systems, most prominently caused by a set of *shared resources*, ranging from on-chip networks, shared memories/disks, to controller buffers etc. The existence of such shared resources lends a great help to Denial-of-Service (DoS) attacks: a greedy application can immoderately engulf certain share resource, depriving other applications of gaining fair access to that shared resource.

On-chip networks are one of the critical shared resources in future many-core systems. An on-chip network connects tens to hundreds of components (CPUs, caches, memory controllers, accelerators, etc) via on-chip routers, executing hundreds of tasks concurrently. For example, Tilera already has a 100-node on-chip network organized as a 10x10 mesh [1]. Figure 1 depicts an example 3x3 mesh on-chip network. Although DoS attacks and countermeasures have been extensively studied in the context of computer networks, DoS attacks on emerging on-chip networks are barely recognized by the security community. As one of very few exceptions, Moscibroda and Mutlu propose, implement, and prevent an on-chip DoS attack against shared memories [7]. However, their work does not consider on-chip networks. Due to substantial differences between computer networks and on-chip networks, solutions for computer networks cannot be transplanted to on-chip networks either. Because on-chip networks are implemented in hardware, any on-chip network mechanism needs to be simple, easy-to-implement, and energy-efficient.

This poster intends to raise the awareness of DoS attacks for on-chip networks among the security community. More specifically, our contribution is three-fold. (i) First, we bring forth this important problem from a security point of view, and formalize the attack model and security definition. (ii) Second, we simulate effective DoS attacks which successfully overload a node of an on-chip network. (iii) Third, we review the state-of-art mechanisms and propose possible solutions.

2 Problem Statement

We formalize the attack model and security problem below.

Definition 1. An **on-chip network DoS attacker** is a set of malicious applications controlling a set of on-chip network routers or input ports. The attacker aims to overload certain bottleneck component(s) of the network by generating large traffic toward the victim(s), in order to prevent a benign application from gaining fair access to the memory system.

Definition 2. $(\alpha, \beta)-$ **Security** is achieved under an on-chip network DoS attacker, iff (i) an application can obtain at least α fraction of the resource of any shared on-chip network component, and (ii) the malicious application(s) can be detected after launching attack for at most β time.

In the security definition above, the α parameter characterizes the fairness property. When $\alpha = \frac{1}{n}$ where n is the number of applications currently running, we achieve perfect fairness for individual applications and prevent any *single* malicious application from exhausting the resource of a target component. The β parameter represents a complementary security property to the α parameter: when the on-chip network DoS attacker controls a sufficiently large number of malicious applications, even when the resource usage quota is bounded by α for each individual application, the attacker can still overload a victim by aggregating traffic from all malicious applications. By achieving a small β value, we can guarantee that such attack can be detected timely, thus limiting the damage the attacker can infuse.

3 Design and Simulate Attacks

Based on our attack model, we design a malicious application to flood service requests toward a network node (the target shared component). However, the flooding rate of service requests is limited by the network accesses (e.g., cache misses) that the malicious application can generate, otherwise the request will not be transmitted over the on-chip network. To circumvent this limitation, we simulate an attacker controlling multiple malicious applications, each taking up a network

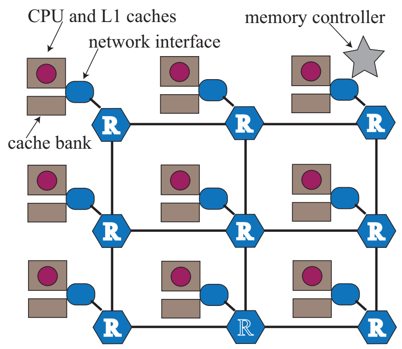

Figure 1: An example 3x3 mesh on-chip network, connecting CPUs, caches, and memory controllers.

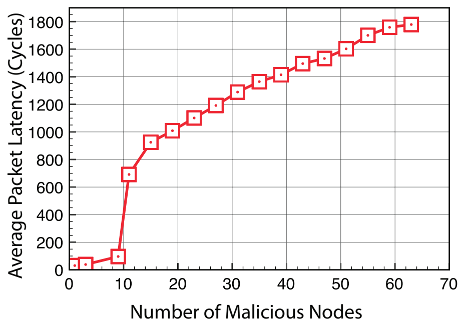

Figure 2: DDoS attack by flooding the same destination output port.

node (or a core) of the on-chip network; and these attacker nodes collectively aggregate traffic toward a victim node.

We simulate our attack using BookSim [2], a cycle-accurate on-chip network simulator. We consider a typical topology, a 4-radix 3-stage butterfly topology [5], where 64 inputs and 64 outputs are connected by 48 routers. Our simulation results show that, with more colluding malicious nodes, the latency of benign applications' packets can increase dramatically, as Figure 2 shows. Hence, benign applications can slow down significantly because they are denied service in the network for long time periods.

4 Possible Solutions

4.1 For Achieving α (Fairness)

Several mechanisms have been recently proposed for achieving fairness for on-chip networks [3,4,6]. Though the authors do not frame the problem as a security problem, the solutions can be used to achieve α.

4.2 For Achieving β (Detection)

We present the following possible solutions and discuss the limitations and trade-offs below.

Correlation-based detection. When the traffic each individual malicious application can generate is bounded by α, the colluding malicious applications must aggregate their traffic by flooding the same victim (spatial correlation) at the same time (temporal correlation), in order to maximize the *distributed* DoS (DDoS) effectiveness. Hence there must exist a strong *temporal and spatial correlation* between the traffic sent by the colluding malicious applications.

We can leverage a graph data structure to capture such a correlation, where each node corresponds to an application and the edge weight indicates the correlation degree of the two applications connected by that edge. If two requests from two applications A_i and A_j are issued within a time window T and go through the same component (e.g., a router or an output port), the weight of the edge connecting A_i and A_j is increased. We discuss how to obtain the correlation data and build the graph shortly. After some observation time, colluding malicious applications will exhibit high weights between each other in the graph. Then the scheme will report suspicious applications (with edge weights more than a certain threshold between each other) to the operating system for future investigation. However, this scheme cannot directly distinguish between colluding malicious applications and benign applications which naturally share the same resource and thus have access patterns similar to malicious applications (also exhibiting high correlation on that shared resource). Therefore, further investigation from the operating system will be needed.

Router-OS Interface buffer. On-chip routers usually have limited computational and storage capacity, and are thus incapable of maintaining the data structure and performing the analysis presented above. However, we can build a Router-OS Interface buffer, where the router can populate hardware logs, and the operating system can periodically fetch the logs, based on which it can maintain the data structure and perform detection analysis. The log simply keeps track of the number of accesses received from each other core or each application. In this way, we devolve the task of difficult analysis from the routers to the operating system, at the price of delay for fetching the logs from the on-chip routers to the operating system. However, this approach keeps the hardware simple, which is necessary in future many-core hardware.

For the purpose of detecting correlated flooding requests, for each request, the router logs the timestamp of the request to capture temporal correlation, and the application from which the traffic is sent. Note that the router may not be able to log each request to the interface buffer. Instead, we can employ a sampling rate ρ ($\rho \in (0,1)$), so the router *randomly* selects only one of every $\frac{1}{\rho}$ requests to write to the log. Choosing different ρ values enables a tradeoff between logging overhead and detection accuracy.

5 Future Work

After simulating DoS attacks for on-chip networks and envisioning possible solutions, we plan as a future work to implement real colluding malicious applications [1], and implement our solutions to study their effectiveness. We also note that the proposed router-OS interface buffer can essentially overcome the limited capacity of on-chip routers, and enable us to use software (e.g., the operating system) to address hardware problems. We believe that this is a promising framework and would like to explore the role that it may play for other hardware-based denial-of-service scenarios.

References

[1] http://www.tilera.com.

[2] B.Towels and W.J.Dally. Booksim 1.0. http://cva.stanford.edu/books/ppin.

[3] R. Das, O. Mutlu, T. Moscibroda, and C. R.Das. Application-aware prioritization mechanisms for on-chip networks. In *MICRO*, 2009.

[4] B. Grot, S. Keckler, and O. Mutlu. Preemptive virtual clock: A flexible, efficient, and cost-effective qos scheme for networks-on-chip. In *MICRO*, 2009.

[5] J. Kim, J. Balfour, and W. Dally. Flattened butterfly topology for on-chip networks. In *MICRO*, 2007.

[6] J. W. Lee, M. C. Ng, and K. Asanovic. Globally-synchronized frames for guaranteed quality-of-service in on-chip networks. In *ISCA*, 2008.

[7] T. Moscibroda and O. Mutlu. Memory performance attacks: Denial of memory service in multi-core systems. In *Usenix Security 07*.

[1]Though an on-chip DoS attack has recently been simulated [4], a distributed DoS attack by real colluding nodes belonging to a malicious application has not been implemented.