*Regular Article*

# FPGA-Based Multiple DDoS Countermeasure Mechanisms System Using Partial Dynamic Reconfiguration

**Tran Ngoc Thinh, Cuong Pham-Quoc, Biet Nguyen-Hoang, Thuy-Chau Tran-Thi, Chien Do-Minh, Quoc Nguyen-Bao, Nguyen Quoc Tuan**

Ho Chi Minh City University of Technology, Vietnam National University, Ho Chi Minh city, Vietnam

Correspondence: Cuong Pham-Quoc, cuongpham@hcmut.edu.vn

Communication: received 17 December 2015, revised 15 March 2016, accepted 4 April 2016
Online publication: 10 October 2016, Digital Object Identifier: 10.21553/rev-jec.137
The guest editor coordinating the review of this article and recommending it for publication was Dr. Tran Manh Ha.

*Abstract*– **In this paper, we propose a novel FPGA-based high-speed DDoS countermeasure system that can flexibly adapt to DDoS attacks while still maintaining system performance. The system includes a packet decoder module and multiple DDoS countermeasure mechanisms. We apply dynamic partial reconfiguration technique in this system so that the countermeasure mechanisms can be flexibly changed or updated on-the-fly. The proposed system architecture separates DDoS protection modules (which implement DDoS countermeasure techniques) from the packet decoder module. By using this approach, one DDoS protection module can be reconfigured without interfering with other modules. The proposed system is implemented on a NetFPGA 10G board. The synthesis results show that the system can work at up to** 116.782 **MHz while utilizing up to** 39.9% **Registers and** 49.85% **BlockRAM of the Xilinx Virtex xcv5tx240t FPGA device on the NetFPGA 10G board. The system achieves the detection rate of** 100% **with the false negative rate at 0% and false positive rate closed to** 0.16%**. The prototype system achieves packet decoding throughput at** 9.869 **Gbps in half-duplex mode and** 19.738 **Gbps in full-duplex mode.**

*Keywords*– **Partial Reconfiguration, ICAP, Reconfigurable hardware, Distributed Denial of Service (DDoS), Hop-count, Ingress, Egress.**

## 1 Introduction

Distributed Denial of Service (DDoS) is a network attack method to prevent legitimate users from accessing network resources or services. It consumes network resources or server resources by employing multiple computer zombies to simultaneously send requests to a victim. The victim will be overloaded. It then cannot respond to legitimate requests and causes denial of service. Most of DDoS attacks use Internet Protocol (IP) address spoofing technique [1] that allows attackers to modify source IP address of a packet. Hence, its original address is hidden. Spoofer Project has shown that 13.5% of the overall address space is spoofable [2]. Network router only checks packet destination address to make a routing decision while keeping source address intact. The way a router routes a packet is a vulnerability that attackers can exploit to perform DDoS attacks. There is much research on proposing DDoS countering systems in the literature [1]. However, most of those proposed systems are implemented as software programs. Therefore, they could not quickly react to DDoS attacks in a high-speed network environment. To eliminate the limitation of those systems, we need to develop a platform that not only is programmable but also has a high-performance.

In last decades, computer society has seen the evolution of reconfigurable computing from less-complex prototyping to high density and performance platforms. Field Programmable Gate Array (FPGA) is usually used for implementing reconfigurable computing systems. FPGAs are programmable logic devices that consist of a matrix of Configurable Logic Blocks (CLBs) connected through programmable interconnects that can be re-programmed. FPGA not only has advantages of hardware-based high-speed parallel processing but also takes the flexibility of software-based programmability. In this work, we take these main advantages of FPGA devices into consideration to quickly adapt to various DDoS attack mechanisms and achieves high-speed computation. Dynamic partial reconfiguration technique is applied to quickly react to the changes of vulnerability exploitations.

An FPGA device is configured by loading application-specific configuration data, named bitstream, into internal configuration memory. Partial reconfiguration (PR) is the modification of an operating FPGA configuration memory by loading a partial configuration file. With the rapid development of technology, FPGAs allow dynamic partial reconfiguration (DPR). It means that some parts of an FPGA device can be reconfigured at runtime while other parts are still working. This runtime reconfiguration helps systems be updated while still operating. The design flow of DPR partitions configuration memory into *static logic* and *reconfigurable logic* [3]. In DPR process, the static logic remains functioning while the reconfigurable logic is modified by the partial configuration file.

In this paper, we propose a novel FPGA-based high-speed DDoS countermeasure system using reconfigurable computing platform with taking dynamic partial reconfiguration technique into consideration. The system consists of three main components: Base System, DDoS Filtering, and Dynamic Partial Reconfiguration (DPR). The Base System takes responsibility to extract header and store raw packets while waiting for classifying results from the DDoS Filtering component. The DDoS Filtering component classifies packets based on the header received from Base System. The DDoS Filtering component can include multiple filtering mechanisms and can be updated or changed dynamically. DPR consists of DPR Controller and Internal Configuration Access Port (ICAP) Controller [4] that partially reconfigures DDoS Filtering on-the-fly while the system is still operating. DPR uses ICAP primitive to make reconfiguration.

The main contributions of the paper are as follows:

- High-speed packet decoder: the packet decoder module is implemented in an FPGA device. It takes advantages of hardware-based parallel processing, which is faster than software-based implementation. Experimental results show that the packet decoder in our proposed system reaches the line rate of 10Gbps in a high-speed network.
- A novel system architecture for DDoS countermeasure mechanisms: the proposed architecture separates the packet decoder module from the DDoS Filtering component implementing different DDoS filtering mechanisms. This architecture helps developers to implement filtering modules independently using output information from the packet decoder. Therefore, one filtering module can be reconfigured and updated dynamically without any interference from other filtering modules.
- Online reconfiguration system: the architecture allows DDoS filtering modules to be reconfigured while operating, without affecting other modules or changing the system architecture. Therefore, system performance is still maintained while being reconfigured.

The rest of the paper is organized as follows. Section 2 analyzes background and related work. Section 3 discusses our proposed DDoS countermeasure system architecture. Section 4 introduces our system implementation using a NetFPGA-10G board. Experimental results are presented in Section 5. Finally, Section 6 concludes the paper and introduces the future work.

## 2 Background and Related Work

In this section, we present background on DDoS attacks and DDoS countermeasures. Several proposed systems in the literature to defend against the DDoS attacks also discussed in this section.

### 2.1 Background

Attackers often employ computers or zombies controlled by malicious software to create a botnet to perform DDoS attacks. They are usually motivated by incentives such as financial/economical gains, revenge, ideological belief, intellectual challenge or cyberwarfare [1]. Attacks that are for financial gains are dangerous and hard to mitigate.

Zargar *et al.* [1] classified DoS/DDoS attacks into two categories: network/transport-level and application-level flooding attacks. Network/Transport-level based flooding attacks are performed by exploiting vulnerabilities of layer 2 to layer 4 in the Open Systems Interconnection (OSI) network model to exhaust victim network resources. This category includes flooding attacks, protocols exploitation flooding attacks, reflection-based flooding attacks, and amplification-based flooding attacks. Flooding attacks often exhaust network resources by consuming bandwidth or overburdening network devices. In protocol-exploitation attacks, an attacker sends malformed packets, such as TCP SYN flood [5, 6] and TCP SYN/ACK flood [7], to confuse a victim. In reflection and amplification based attacks, an attacker broadcasts spoofed packets whose source addresses are the IP address of a specific victim to make reflectors/amplifiers. Consequently, responses are sent back to the victim and cause flooding (i.e., Smurf attacks, Fraggle attacks).

Application-level based flooding attacks exploit application-level vulnerabilities, including protocols and application code, to exhaust victim server resources. Attackers often exploit stateless protocols for this kind of attack, such as DNS and NTP. Research in [8] and [9] recorded DNS amplification DDoS attacks with 300 Gbps. NTP amplification DDoS set a new record with 400 Gbps in 2014 [10, 11].

Based on attack classifications, DDoS defense mechanisms are also classified into network-level and application-level [1] defense mechanisms. Network-level based defense mechanisms are deployed to mitigate DDoS attacks under network layers. They are categorized into source-based, network-based, destination-based, and hybrid mechanisms based on deployment locations. Port Ingress/Egress Filtering (PIEF) method [12] can be deployed as a source-based or destination-based mechanism. In the destination-based mechanism, Management Information Base (MIB) [13] is used to monitoring network traffic to detect DDoS attacks. Hop-Count Filtering (HCF) method [14] can filter out spoofed packets based on the number of routers these packets traversed. Research in [1] introduces hybrid methods, such as Stop-It and Active Internet Traffic Filtering (AITF), which incorporate multiple components across network systems to counter DDoS attacks.

Application-level based defense mechanisms are deployed to detect application vulnerabilities attacks. CAPTCHA [15] is an well-known application-based method to differentiate DDoS flooding bots from human. This method helps servers to classify and filter bot-based packets.

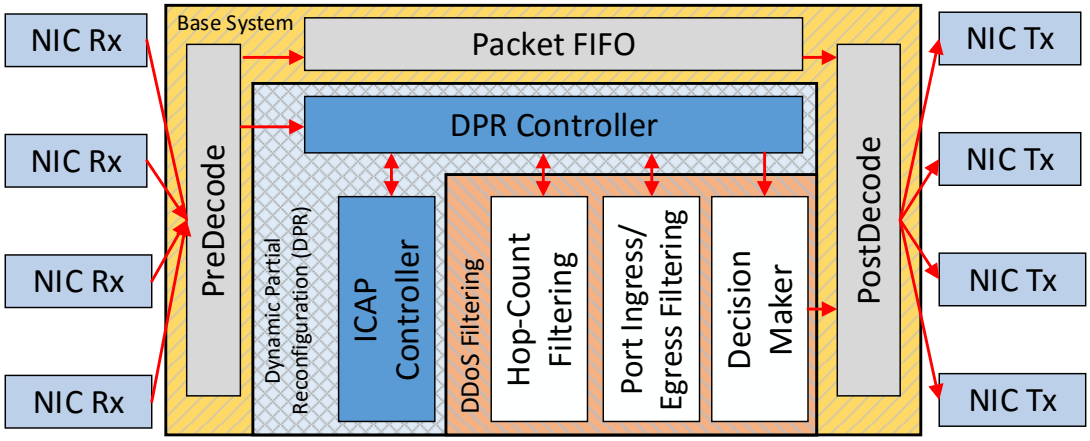



Figure 1. The proposed system architecture.

### 2.2 Related Work

This section introduces several proposed systems in the literature for detecting spoofed packets. We list those systems based on their published years.

Ferguson *et al.* [12] proposed Port Ingress/Egress Filtering method to filter spoofed packets. The Ingress or Egress name depends on its deployment position. Ingress filter is deployed to filter inbound traffic. If an incoming packet is spoofed, it is blocked. Egress filter monitors outbound traffic to ensure that malicious packets cannot leave internal networks.

Research in [16] implemented a DDoS countermeasure system applied neural network and Bloom Filter. The neural network is trained so that it can recognize abnormal incoming packets. Those packets are then removed from networks by Bloom Filter.

Katashita *et al.* [17] introduced an intrusion detection system (IDS) on FPGA by porting Snort rules into FPGA devices using the NFA-based method. The system is implemented on a Virtex-II Pro board and supports PR to update Snort rules, but the PR process is offline.

Wang *et al.* [14] proposed a method named Hop-Count Filtering (HCF) to filter spoofed packets based on the number of hops packets traversed before arriving their destinations. Each packet traveling on the network has its own Time-To-Live (TTL) value. When a packet traverses a router (hop), its TTL value is decreased by one before forwarding to the next hop. Hop-count is calculated by comparing initial TTL value to final TTL value when the packet arrives at its destination. A packet is dropped in two difference cases. The first one is when its TTL value is equal to zero while the second one is when its Hop-count is not identical with Hop-counts of other packets coming from the same source. The paper claimed that HCF can identify 90% of spoofed packets.

Wang *et al.* [18] proposed a distributed HCF (DHCF) model, which was implemented in intermediate routers. This method protects not only hosts but also intermediate networks from malicious packets and traffic congestion. Experimental results showed that DHCF achieved better performance than conventional HCF but maintained user access.

Ayman *et al.* [19] proposed an upgraded version of HCF, by storing multiple Hop-count values according to multiple routes. This approach modified HCF method can increase true positive rate because Hop-count may vary if packets travel through multiple routes. However, this method suddenly increases false negative rate, because it increases the chance to attackers to bypass the detector.

Maheshwari *et al.* [20] combined probabilistic and round trip time in Distributed Probabilistic HCF-Round trip time (DPHCF-RTT). Packets are checked once by intermediate DPHCF-RTT routers (nodes) and then they are forwarded to destinations. The larger number of intermediate routers implemented, the higher detection rate of malicious packets is. The paper claimed that detection rate is up to 99.33%.

## 3 System Architecture

In this section, we describe our proposed FPGA-based DDoS countermeasure system architecture taking the dynamic partial reconfiguration technique into consideration. The proposed system architecture includes three main components: Base System, DDoS Filtering, and Dynamic Partial Reconfiguration (DPR) components as shown in Figure 1.

### 3.1 Base System

The Base System component is responsible for receiving packets from networks and extracting header fields from these incoming packets. These header fields are sent to the DDoS Filtering component to determine whether a packet should be dropped or bypassed. Base System consists of three modules: PreDecode, Packet FIFO, and PostDecode.

*3.1.1 PreDecode:* this module decodes and extracts headers from incoming packets. These headers are then transferred to the filtering modules in the DDoS Filtering component for classifying. These headers includes source IP, destination IP, source port, destination port and TTL value. The raw packets are stored in the Packet

FIFO module while waiting for classifying results from the filtering modules in the DDoS Filtering component.

*3.1.2 Packet FIFO:* incoming packets can be processed by two different approaches. In the first approach, a packet is de-encapsulated into header and payload fields. The header field is then sent to the filtering modules to classify as described above. Finally, if the packet is legitimate, the header and payload fields are encapsulated into the packet and sent out to networks. In contrast, if the packet is classified as a DDoS attack packet, the whole packet is removed. However, this approach is time-consuming because encapsulating process takes time. The second approach is to use a buffer to store full raw packets. This approach helps to reduce system latency. In this work, we implement the second approach and name the buffer module as Packet FIFO.

*3.1.3 PostDecode:* the PostDecode module selects packets from the Packet FIFO module and waits for decisions from the Decision Maker module in the DDoS Filtering component. The PostDecode module determines whether the packets should be forwarded or dropped based on the feedback of the DDoS Filtering component. If packets are legitimate, they are sent out to networks. Otherwise, they are removed.

### 3.2 DDoS Filtering

In Section 2.2, we present several proposed DDoS countermeasure systems in the literature. However, each of those approaches only counters a specific DDoS attack. Therefore, those systems cannot completely recognize DDoS attack packets when working alone. In this section, we introduce a combination of the Port Ingress/Egress Filtering (PIEF) and Hop-count Filtering (HCF) methods in the DDoS Filtering component. This combination lets our DDoS Filtering system classify packets more efficient than systems using only one DDoS defense method.

*3.2.1 Port Ingress/Egress Filtering:* in computer network, Ingress filtering is a technique used to guarantee that incoming packets are actually coming from their original networks. Routers integrated the Ingress filtering method check source IP addresses of traversing packets. A router drops a packet if its source IP address does not belong to the range of addresses to which the router is connected. Meanwhile, Egress filtering technique monitors outbound traffic to ensure that spoofed or malicious packets are not allowed to leave internal networks. There is a special-purpose address registry [21] which defines IP address blocks that do not either appear or exist on the Internet as usual. Therefore, they should be blocked in PIEF module.

*3.2.2 Hop-Count Filtering:* although DDoS attackers can forge any data in the header field of a packet, they cannot falsify the number of hops that a packet traverses to reach its destination. The number of traversed hops of a packet, named hop-count, is calculated by subtracting the final Time-to-Live (TTL) value from the initial TTL value. TTL is an 8-bit field [22] in the header field that is originally introduced to define the maximum lifetime of a packet on the Internet. The final TTL value is the value when the packet reaches the destination. The initial TTL values are set to 30, 32, 60, 64, 128, or 255 according to Operating System (OS) where the packet is packed. Listing 1 shows the complete HCF algorithm we use to detect an illegitimate packet in this work. The algorithm first calculates Hop-count value of an incoming packet (line 2-6) using the TTL values. This value is then compared to the stored Hop-count (already extracted from previous packets coming from the same source) (line 7-11). If these values are not equal, the coming packet is spoofed.

Listing 1. The algorithm of HCF module

```
1  for each packet begin
2    Tf = extract_TTL(packet);
3    S = extract_IP(packet);
4    Ti = infer_initial_TTL(S);
5    //compute hop count
6    Hc = Ti - Tf;
7    Hs = get_stored_HC(S);
8    if (Hc <> Hs)
9      packet is spoofed;
10   else
11     packet is legitimate;
12 end
```

*3.2.3 Decision Maker:* bBased on the output information of the PIEF and HCF modules, Decision Maker module issues a decision to the PostDecode module. A drop signal alerts the PostDecode module if either PIEF or HCF realizes a sign of DDoS attack. Otherwise, a bypass signal is sent to the PostDecode module to allow the corresponding packet to go through the countering system.

### 3.3 Dynamic Partial Reconfiguration

Dynamic Partial Reconfiguration (DPR) is a specific technique in reconfigurable technology. The technique allows a reconfigurable device to be re-programmed some areas while keeping other areas unchanged. The DPR process can be done even when the system implemented on the device is running. Currently, only Xilinx FPGAs supporting the DPR technique are available as commercial devices. Therefore, we focus on Xilinx FPGA devices and design tools in our discussion from this section. Many interfaces can be used to configure FPGA devices such as serial configuration interface, Join Test Access Group (JTAG)/Boundary-Scan port, SelectMAP, and ICAP. However, only the ICAP interface supports reconfiguration from internal FPGA device while others are external interfaces connected through connection pins. Xilinx supports an ICAP primitive interface [23] to enable read and write instructions to access the configuration memories of FPGA devices. The ICAP interface separates read and write data buses, which can be configured to be 8, 16 or 32 bit-width. The ICAP interface should be operated at a frequency of 100MHz. Table I shows the specifications of those configuration interfaces.

In this proposed system architecture, a DPR Controller and an ICAP Controller are used to control the

Table I
CONFIGURATION INTERFACES

| Configuration Interface | Type | Bit Width | Freq MHz | Bandwidth MBytes/s |
|---|---|---|---|---|
| Serial Configuration Port | External | 1 | 100 | 12.5 |
| JTAG/Boundary Scan Port | External | 1 | 66 | 8.25 |
| SelectMap Port | External | 8 | 100 | 100 |
| | | 16 | 100 | 200 |
| | | 32 | 100 | 400 |
| ICAP | Internal | 8 | 100 | 100 |
| | | 16 | 100 | 200 |
| | | 32 | 100 | 400 |

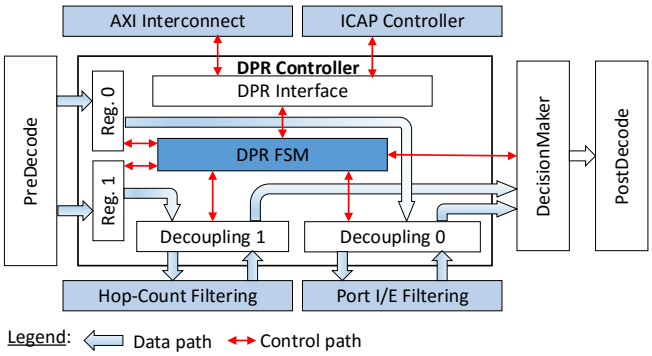



Figure 2. The DPR Controller.

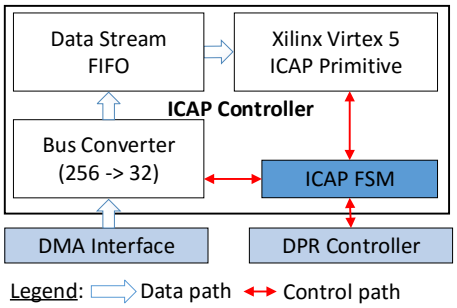



Figure 3. The ICAP Controller.

partial reconfiguration (PR) process. When receiving a PR signal from the host, DPR Controller sends a signal to the PreDecode module to stop sending data to the reconfigured DDoS filter module. DPR Controller then deactivates the filtering module before sending a signal to ICAP Controller to accept the partial bitstream. When the PR process is finished, ICAP Controller informs DPR Controller to reset the upgraded module. DPR Controller sends a signal to PreDecode to accept data to the upgraded module. The DPR and ICAP architectures are shown in Figure 2 and Figure 3.

## 4 System Implementation

As stated above, currently only Xilinx FPGA devices supporting dynamic partial reconfiguration are available in commercial. Therefore, we choose a Xilinx FPGA board, the NetFPGA-10G board [24] to develop our prototype system. In this section, we present in detail our implementation to develop the prototype system.

The NetFPGA-10G board includes four SFP+ ports and one Xilinx Virtex-5 TX240T FPGA device. Four SFP+ ports are suitable to build high-speed network applications. Besides, Xilinx Virtex-5 TX240T provides powerful hardware resources to handle huge traffic on a network. We use Hardware Description Language (HDL) to develop all modules in the three main components.

### 4.1 The PreDecode Module

This module receives raw packets from the network interfaces through an AXI interface [25]. The AXI interconnect is an Xilinx IP core used to connect modules together in a bus protocol. The raw incoming packets are parsed to extract header fields. These raw packets are then sent to the Packet FIFO module while the header fields are sent to the DDoS filtering component for further processing.

### 4.2 The Packet FIFO Module

This module functions as a First In First Out buffer. In this work, we use the FIFO IP core provided by Xilinx to implement this buffer. The size of Packet FIFO is 256-bits in width and 1024 entries in depth. With this configuration, the Packet FIFO module can store 21 packets in 1500 bytes of size at minimum and 512 packets in 64 bytes of size at maximum.

### 4.3 The PostDecode Module

The functionality of this module is to send packets to the 10G NIC TX output ports of the board through the AXI interface. The PostDecode module receives control signals from the Decision Maker module (the DDoS Filtering component). Based on these signals, the PostDecode module decides whether the packets are dropped out from the system or forwarded to networks. In both cases, when packets are dropped or forwarded, they are removed from the Packet FIFO.

### 4.4 The Port Ingress/Egress Filtering Module

This module consists of two blocks: Content Addressable Memory (CAM) and Comparator. Figure 4 describes a diagram of Port Ingress/Egress Filtering (PIEF) module. CAM stores special IP address blocks [21]. The Comparator compares the IP addresses of incoming packets with stored IP addresses in CAM. When the packet source IP address is sent to PIEF, PIEF searches the address in CAM. If a MISS signal is returned (i.e., no record was found in CAM), the packet is legitimate. Otherwise, the packet is illegitimate (i.e., a HIT signal is returned). the PIEF module forwards MISS or HIT signal to Decision Maker for further processing.

### 4.5 The Hop-Count Filtering Module

This module consists of three blocks: CAM, Comparator, and Register Array as described in Figure 5. Due to the limitation of NetFPGA 10G board hardware

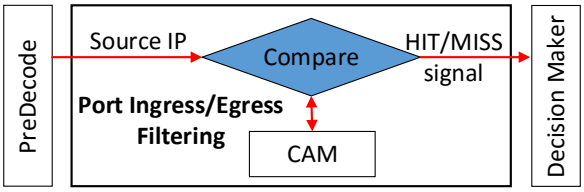



Figure 4. The architecture of the Port Ingress/Egress module.

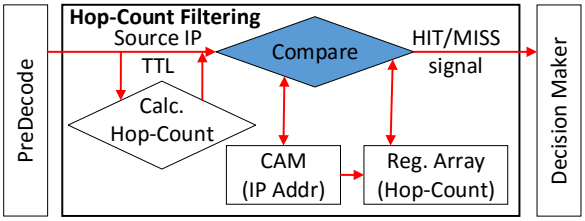



Figure 5. The architecture of HCF module.

resources, we build two versions of CAM, 128 and 256 entries. The module receives a packet source IP address and its TTL value from the PreDecode module. This module calculates the actual Hop-count of the incoming packet using TTL value. The module, then, looks up this IP address in CAM and gets the stored Hop-count in the Register Array. If the stored Hop-count is equal to calculated Hop-count, the packet is legitimate. Otherwise, the packet is spoofed. When this is the first time the packet has come to the system (i.e., whose source IP does not exist in CAM), CAM returns a MISS signal and stores both the packet source IP address and calculated Hop-count into Register Array. However, the initial TTL value can be forged. It means that the packet is spoofed, but HCF could not recognize that because of the lack of information. That is the main disadvantage and limitation of the HCF mechanism. Therefore, we consider combining PIEF and HCF.

The method storing and looking up Hop-count of an incoming packet as described above is named as IP2HC method. CAM is usually used for storing IP address in most FPGA-based networking system because of its fast query response. However, CAM can return index and a HIT or MISS signal only. Therefore, we store hop-count in Register Array instead of CAM. Furthermore, the incorporation of CAM and Register Array improves query time. The index of CAM and Register Array is a direct one-to-one mapping. If source IP address of a packet is stored in CAM at index k, the returned value of Register Array at index k is the hop-count value of that packet. Table II and III show an example of the IP2HC method that we develop in this work. When looking for IP address 134.170.188.221, for example, CAM returns an index of 1. Based on this index, Register Array returns 10 as the hop-count value of this IP address.

### 4.6 The Decision Maker module

This module issues final decisions to the Base System component based on the decision signals from the PIEF and HCF modules. Either PIEF or HCF sends a drop signal, Decision Maker instantly turns on a drop signal to the PostDecode module. Otherwise, Decision Maker sends a bypass signal to the PostDecode module.

Table II
CAM

| Index | IP blocks |
|---|---|
| 1 | 134.170.188.0/24 |
| ... | ... |
| N -1 | 98.138.253.0/24 |
| N | 216.58.221.0/24 |

Table III
Register Array

| Index | Hop-count value |
|---|---|
| 1 | 10 |
| ... | ... |
| N - 1 | 25 |
| N | 8 |

Table IV
The Device Utilization Summary of the System

| Modules | Max Frequency (Mhz) | Registers | BlockRAM (KByte) |
|---|---|---|---|
| System | 116.782 | 59,784 | 5,814 |
| Base System | 262.522 | 95 | 468 |
| HCF_128 | 120.043 | 29 | 4,608 |
| HCF_256 | 118.876 | 54 | 6,300 |
| PIEF | 247.452 | 111 | 108 |
| DPR & ICAP | 116.875 | 4,766 | 0 |

### 4.7 The DPR Controller

This module receives PR signal from the host through the AXI-Lite interface, an another bus-like protocol interconnect. This module has a control signal register to receive PR command from the host and get feedback status. Each reconfigurable module has a decoupling gate to eliminate pulse noise while reconfiguring.

### 4.8 The ICAP Controller

This module receives a partial bitstream from the host DMA through an AXI-Stream interconnect. AXI-Stream provides 256-bit data bus while the ICAP primitive supports a 32 bit-width data interface only. Therefore, we have to segment data into 32-bit blocks to feed into the Xilinx Virtex 5 ICAP Primitive, an Xilinx ICAP IP core. A data buffer is used to sequentially transfer partial bitstream to ICAP.

The system is synthesized by ISE 13.4 without any manual optimization. Table IV shows hardware resources usage as well as maximum possible frequency of each module.

## 5 Experiments

In this section, we present our experiments with the system implemented in the previous section. We also analyze the throughput and packet classification results of the system in this section.

### 5.1 Experimental Setup

To measure the accuracy and throughput of the system, we deploy a testing model as shown in Figure 6. NetFPGA 10G boards and Open Source Network Tester (OSNT) [26] are used for both throughput and accuracy

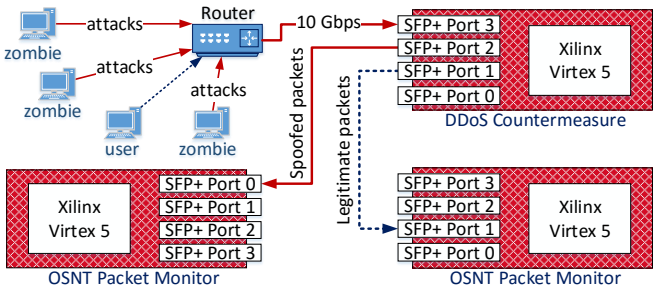



Figure 6. The accuracy testing model of the system.

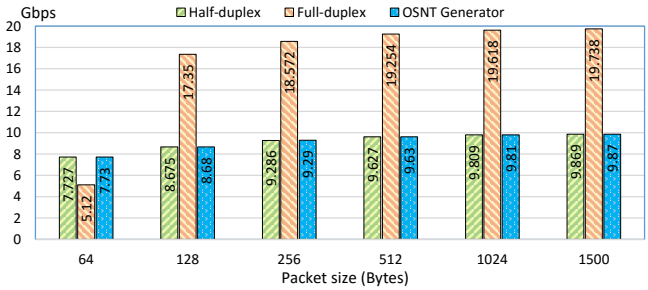



Figure 7. The throughput of the experimental system

testing model. OSNT is a flexible tool; it can generate and capture packets of any size at the line-rate speed of 10 Gbps. The throughput testing model is used to test decoding speed. We prepare TCP/UDP packets with various sizes from 64 to 1500 bytes. These packets are sent out at the maximum speed of OSNT Generator. We use OSNT Monitor to measure the throughput. In the accuracy testing model, we test the combination of two filters: PIEF and HCF. Three computers function as zombies to attack the system, another one is a normal user. We prepare TCP/UDP packets with various sizes. Some of those packets are real and collected from the Internet to test HCF. Some packets are synthetic generation to test PIEF. Classified packets are captured to evaluate and figure out the detection rate (DR), false positive rates (FPR), and false negative rates (FNR).

### 5.2 Experimental Results

Figure 7 presents throughput of the experimental system. The vertical axis shows throughput values in Gbps. The horizontal axis shows packet size of test cases. For each test case of packet size, the left, middle, and right columns show packet decoding speed in half-duplex mode, full-duplex mode, and the packet generating speed of OSNT, respectively. Experimental results in Figure 7 show that throughput of the system nearly achieves the maximum speed of packet generator in half-duplex mode at 9.869 Gbps. In full-duplex mode, the system stably achieves twice the throughput of half-duplex mode and up to 19.738 Gbps, except test case of 64-Byte packets.

Table V shows the statistical values of accuracy test of the system. The results are collected by evaluating captured classified packets. We repeat test cases twice in each version of CAM. In all test cases, the DR is up to 100% while the FNRs are 0%. Our achieved DR is better

Table V
The Packet Classification Statistic

| CAM size | Test case | Expected result | | Classified result | |
|---|---|---|---|---|---|
| | | | | Legitimate | Spoofed |
| 128 entries | 1 | Legitimate | 286162 | 286162 | 0 |
| | | Spoofed | 13838 | 0 | 13838 |
| | 2 | Legitimate | 286162 | 286162 | 0 |
| | | Spoofed | 13838 | 0 | 13838 |
| 256 entries | 1 | Legitimate | 266757 | 266319 | 438 |
| | | Spoofed | 33243 | 0 | 33243 |
| | 2 | Legitimate | 266757 | 266285 | 472 |
| | | Spoofed | 33243 | 0 | 33243 |

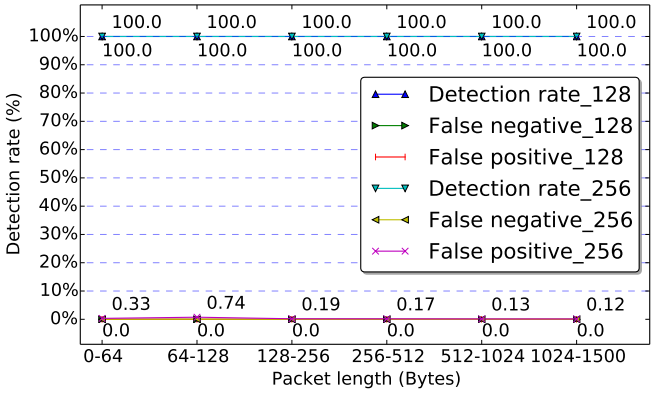



Figure 8. The packet classification statistic ratio.

than results claimed in [14] and [20]. In the 128 entries CAM test cases, FPRs are 0%. In 256 entries CAM test cases, FPRs are closed to 0.16%. We also do statistic based on packet sizes. The packets are classified into 6 groups: less than 64 Bytes, 64-128 Bytes, 128-256 Bytes, 256-512 Bytes, 512-1024 Bytes, and 1024-1500 Bytes. Figure 8 shows the DR, FPR, and FNR of the experimental system based on packet sizes and CAM version. In all test cases, DRs are 100% while FNRs are 0%. In the 128 entries CAM version, the FPRs are 0%. In the 256 entries CAM version, FPRs vary, but they are downtrend on uptrend according to the increasing of packet sizes. FPRs are 0.33%, 0.74%, 0.19%, 0.17%, 0.13%, and 0.12% for packet sizes at less than 64 Bytes, 64-128 Bytes, 128-256 Bytes, 256-512 Bytes, 512-1024 Bytes, and 1024-1500 Bytes, respectively.

We do partial reconfiguration (PR) experiments with two reconfigurable logic regions for the PIEF and HCF modules. We develop a software program to communicate with DPR through DMA to check and send bitstream files to ICAP Controller. The software program is responsible for calculating the PR time. We also do PR experiments through JTAG and DMA AXI-Lite in this work. Table VI shows the PR throughput of the experimental system. We are going to optimize DPR by applying pipelining. As the pipelining simulation, we may achieve throughput up to 370 Mbps at the frequency of 100 MHz.

Table VI
The Partial Reconfiguration Experiment

| Method | Bitstream size (KB) | PR time (second) | Throughput (KByte/s) |
|---|---|---|---|
| JTAG | 256 | 2.577 | 99.340 |
| | 822 | 3.807 | 215.918 |
| DMA (AXI-Lite) | 256 | 0.127 | 2,015.748 |
| | 822 | 0.378 | 2,174.603 |
| DMA (AXI-Stream) | 256 | 0.026 | 9,846.154 |
| | 822 | 0.082 | 9,903.614 |

## 6 Conclusion

In this paper, we propose a novel FPGA-based system to defend against DDoS attacks using reconfigurable hardware. Our DDoS countermeasure system combines Port Ingress/Egress Filtering and Hop-Count Filtering techniques. The Base System component provides high-speed packet decoder and helps reducing system latency. With partial reconfiguration modules, our system can be flexibly adapted to the change of DDoS attacks just in several milliseconds. With our approach, the packet decoding speed of the system reaches the line-rate speed of 10 Gbps network, achieves twice in full-duplex mode. Moreover, the combination of PIEF and HCF improves the packet detection rate up to 100% with the false negative rate closed to 0% and false positive rate closed to 0.16%.

## Acknowledgment

This research is funded by Vietnam National University, Ho Chi Minh City, under grant number C2015-20-09.

## References

[1] S. Zargar, J. Joshi, and D. Tipper, "A survey of defense mechanisms against distributed denial of service (DDoS) flooding attacks," *Communications Surveys Tutorials, IEEE*, vol. 15, no. 4, pp. 2046–2069, Apr. 2013.

[2] R. Beverly, M. Luckie, and R. Koga, "Spoofer Project: State of IP Spoofing," Mar. 21, 2016. [Online]. Available: http://spoofer.caida.org/summary.php

[3] Xilinx, "UG702: Partial Reconfiguration Guide," Xilinx, Apr. 24, 2012.

[4] S. Hansen, D. Koch, and J. Torresen, "High Speed Partial Run-Time Reconfiguration Using Enhanced ICAP Hard Macro," in *IEEE International Symposium on Parallel and Distributed Processing Workshops and PhD Forum (IPDPSW)*, May 2011, pp. 174–180.

[5] H. Wang, D. Zhang, and K. Shin, "Detecting SYN flooding attacks," in *Proceedings of the Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies ( INFOCOM 2002)*, vol. 3, Jun. 2002, pp. 1530–1539.

[6] W. Chen and D.-Y. Yeung, "Defending against TCP SYN flooding attacks under different types of IP spoofing," in *International Conference on Networking, International Conference on Systems and International Conference on Mobile Communications and Learning Technologies (ICNICONSMCL'06)*, Apr. 2006, pp. 38–38.

[7] J. Mirkovic and P. Reiher, "A Taxonomy of DDoS Attack and DDoS Defense Mechanisms," *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 2, pp. 39–53, Apr. 2004.

[8] P. Bright, "When spammers go to war: Behind the Spamhaus DDoS," Mar. 29, 2013. [Online]. Available: http://arstechnica.com/security/2013/03/when-spammers-go-to-war-behind-the-spamhaus-ddos/

[9] C. Fachkha, E. Bou-Harb, and M. Debbabi, "Fingerprinting Internet DNS Amplification DDoS Activities," in *6th International Conference on New Technologies, Mobility and Security (NTMS)*, Mar. 2014, pp. 1–5.

[10] A. Network, "Worldwide infrastructure security report," *Technical Report*, vol. 10, 2015.

[11] M. Prince, "Technical Details Behind a 400Gbps NTP Amplification DDoS Attack," Feb. 13, 2014. [Online]. Available: https:// blog. cloudflare. com/ technical- details- behind-a- 400gbps- ntp-amplification-ddos-attack/

[12] F. P and S. D, "Network ingress filtering: Defeating denial of service attacks which employ ip source address spoofing," *Internet RFC2827*, May 2000.

[13] J. Cabrera, L. Lewis, X. Qin, W. Lee, R. Prasanth, B. Ravichandran, and R. Mehra, "Proactive detection of distributed denial of service attacks using mib traffic variables-a feasibility study," in *Proceedings of the IEEE/IFIP International Symposium on Integrated Network Management*, 2001, pp. 609–622.

[14] H. Wang, C. Jin, and K. G. Shin, "Defense Against Spoofed IP Traffic Using Hop-Count Filtering," *IEEE/ACM Transactions on Networking (ToN)*, vol. 15, no. 1, pp. 40–53, Feb. 2007.

[15] L. V. Ahn, M. Blum, N. J. Hopper, and J. Langford, "CAPTCHA: Using Hard AI Problems for Security," in *Proceedings of the 22nd International Conference on Theory and Applications of Cryptographic Techniques*, ser. EUROCRYPT'03. Berlin, Heidelberg: Springer-Verlag, 2003, pp. 294–311.

[16] Y. Xiang and W. Zhou, "Classifying DDoS packets in high-speed networks," *International journal of computer science and network security*, vol. 6, no. 2B, pp. 107–115, 2006.

[17] T. Katashita, Y. Yamaguchi, A. Maeda, and T. Kenji, "FPGA-based intrusion detection system for 10 gigabit ethernet," *IEICE transactions on information and systems*, vol. 90, no. 12, pp. 1923–1931, 2007.

[18] X. Wang, M. Li, and M. Li, "A scheme of distributed hop-count filtering of traffic," in *IET International Communication Conference on Wireless Mobile and Computing (CCWMC 2009)*, Dec. 2009, pp. 516–521.

[19] M. Ayman, E. Imad, K. Ayman, and C. Ali, "IP Spoofing Detection Using Modified Hop Count," *28th International Conference on Advaned Information Networking and Applications, IEEE*, pp. 512–516, May 2014.

[20] R. Maheshwari, C. Krishna, and M. Brahma, "Defending network system against IP spoofing based distributed DoS attacks using DPHCF-RTT packet filtering technique," in *International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT)*, Feb. 2014, pp. 206–209.

[21] M. Cotton and L. Vegoda, "Special Use IPv4 Addresses."

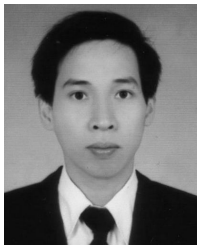
[22] J. Postel, "Internet protocol," 2012. [Online]. Available: https://tools.ietf.org/pdf/rfc791.pdf

[23] Xilinx, "UG191: Virtex-5 FPGA Configuration User Guide," Xilinx, Oct. 19, 2012.

[24] ——, "NetFPGA 10G," Jan. 2013. [Online]. Available: http://netfpga.org/site/#/systems/3netfpga-10g/details/

[25] ——, "UG761: AXI Reference Guide," Xilinx, Mar. 7, 2011.

[26] G. Antichi, "The open source network tester," 2012. [Online]. Available: http://osnt.org/



**Tran Ngoc Thinh** received the B.E. degree in Computer Engineering from the Ho Chi Minh City University of Technology, Vietnam, in 1999. He received his M.E. and Ph.D. from the King Mongkut's Institute of Technology Ladkrabang, Thailand in 2006 and 2009, respectively. He is now a lecturer at the Department of Computer Engineering, Faculty of Computer Science and Engineering, Ho Chi Minh City University of Technology, Vietnam. His research interests include network security and bioinformatics on reconfigurable devices. He is a member of the IEEE.

**Cuong Pham-Quoc** is currently a lecturer at the Faculty of Computer Science and Engineering, Ho Chi Minh City University of Technology, Vietnam. He received his BEng degree in Computer Science and Engineering and MEng degree in Computer Science form the Ho Chi Minh City University of Technology in 2007 and 2009, respectively. In 2015, he got his PhD degree in Computer Engineering from the Delft University of Technology, The Netherlands. His research interest include: multi-/many-core architectures, high performance computing, heterogeneous hardware accelerators, on-chip interconnect, and hardware/software co-design. He is a member of the IEEE.

**Biet Nguyen-Hoang** received his B.Eng. Degree in Information Technology from Nong Lam University, Ho Chi Minh City, Vietnam in 2008. He is currently working as System Engineer in IBM Vietnam. He is also working towards the M.Eng. Degree in Computer Science from the Ho Chi Minh City University of Technology, Vietnam. His research interests include network security, software-defined networks and applications on reconfigurable devices.

**Thuy-Chau Tran-Thi** received her B.Eng in Computer Engineering from Ho Chi Minh University of Technology in 2016. Currently, she is an assistant lecturer in the Faculty of Computer Science and Engineering, Ho Chi Minh University of Technology. Her research includes network security and applications on the reconfiguration devices

**Chien Do-Minh** received his Honor Bachelor in Computer Engineering from Ho Chi Minh City University of Technology, Vietnam, in 2016. He is currently working at the assistant position in Faculty of Computer Science and Engineering, Ho Chi Minh University of Technology, Vietnam. His research interests include network security and applications on reconfigurable devices.

**Quoc Nguyen-Bao** received his Honor Bachelor of Engineering in Computer Engineering from Ho Chi Minh City University of Technology, Vietnam, in 2016. He is currently working as an assistant lecturer in the Faculty of Computer Science and Engineering, Ho Chi Minh University of Technology. His research interests include network security, advanced networks and applications on reconfigurable devices.

**Nguyen Quoc Tuan** is a senior lecturer at the Faculty of Computer Science and Engineering, Ho Chi Minh City University of Technology, Vietnam. He got his bachelor degree from Military Technical Academy (currently, Le Qui Don Technical University) in 1982. In 1999, he received his master degree in Computer Science from Ho Chi Minh City University of Technology. His research interests include micro-processor-based applications, sensor signals processing, hardware description languages, and reconfigurable computing.