International Journal of Computing and Network Technology
ISSN (2210-1519)
Int. J. Com. Net. Tech. 5, No. 2 (May-2017)

SPIMN Stateful Packet Inspection for Multi Gigabits Networks

Amr Ibrahim

Systems Engineering and Computer Department, Al-Azhar University, Faculty of Engineering, Cairo, Egypt

Received: 25 Jan. 2017, Revised: 10 Apr. 2017, Accepted: 23 April 2017, Published: (1 Ma. 2017)

Abstract: Stateful Packet Inspection (SPI) is the most important area of Network Intrusion Detection Systems (NIDS (However it must be operated in multi-Gigabit speeds, to trace and reassemble every connection, and examine every packet flow. I proposed Stateful Packet Inspection for Multi Gigabits Networks (SPIMN). It is customized hardware to achieve a more efficient and faster online inspection system. A generic architecture of SPIMN is based on using FPGA and Header Inspection. Therefore, Xilinx ISE 14.1 used in the design and in the simulation to test the design before the implementation on FPGA Virtex-7 by creating a Test-bench circuit. Finally, the testing and evaluation of SPIMN indicate that this model can work with more than 2,000 Snort rules on 100 Gigabit Ethernet networks.

Keywords: Stateful Packet Inspection (SPI), Header Inspection, Header Parser, Packet Reassembly, Field Programmable Gate Array (FPGA)

1. Introduction

Recent years have witnessed rapid growth in both internet penetration and bandwidth due to huge improvements in telecommunication infrastructure, the proliferation of competitively priced computers and internet-capable mobile devices, and the reduced cost of internet access, resulting from increased competition [1]. The AV-TEST Institute registers over 390,000 new malicious programs evsery day [2]. One of the most-used solutions for protecting networks is the use of Network Intrusion-Detection Systems (NIDS) such as Snort [3]. NIDS depends on the network packet inspection (NPI). There are many categories for the NPI, depending on how many network layers are reference inspected. Although there is no definite category, Parsons divided it into three levels [4]. This study adopted the categorization by Parsons. The three levels of packet inspection are divided into shallow packet inspection (or Stateful packet inspection, SPI), medium packet inspection (MPI) and deep packet inspection (DPI). Figure 1 shows the levels of SPI, MPI and DPI in the OSI architecture described above.

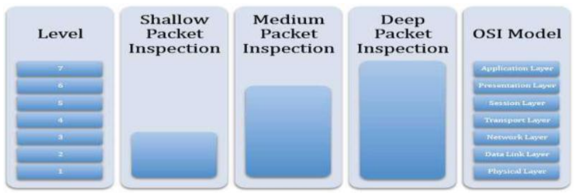

Figure 1. OSI 7 Layers and Packet Inspection Level [5]

The inspection (flow monitoring) can also be categorized according to the numbers of fields (context information) into different types [5]. The description of these fields and the header categorize are shown in Table 1.

This work aims to design and implement Stateful Packet Inspection for Multi Gigabits Networks (SPIMN) based on FPGA to: - 1-header Inspection, 2- Intrusion Protection, and 3- 100 Gigabit network. The operational goals of SPIMN are: - 1-dropping the invalid packet, 2- Applying two techniques in the detection: header analyzing, and matching, 3- filtering and preventing the infected packet, 4-generating reports, 5- Parallel processing to increase the speed, and 6- No modification to hardware, operating system, or run time environment. In addition, the proposed architecture is based on the following assumptions: - 1- using a fixed parser, and 2-

E-mail: amr.ibrahim67@gmail.com

working with IP4, and 3- Snort rules generating: It is difficult and time consuming to write these rules, so in we satisfy by a limited number of these rules. But in the future we will add a module to generate these rules automatically.

TABLE I. RELATIONSHIP BETWEEN TCP CONTEXT INFORMATION AND THE DEGREE OF TCP FLOW ANALYSIS

Field	*Size*	*Description*	*Type*	*Inspection Type*				
Prot	8 bits	Protocol (TCP, UDP,…)	Not Calculated	Flow Definition	Flow Statistics	Flow Establish (Three Way Handshake)	Flow Control (Sliding Window)	Flow Deep Monitor (Stateful TCP Inspection)
SA	32 bits	Source IP Address						
DA	32 bits	Destination IP Address						
SP	16 bits	Source Port Number						
DP	16 bits	Destination Port Number						
CS	4 bits	Client State: 0: Closed 1: Listen 2: SYN_RCVD 3: SYN_SENT 4: Established 5: Close_ Wait 6: Last_Ack 7: Fin_Wait_1 8: Closing 9: Fin_Wait_2 10: Time Wait		Flow State				
SS	4 bits	Server State: The same as Client State Field						
ConS	32 bits	Connection State: State is tracking TCP (L4) as well as Application Protocol (L7). For example, Http, FTP, Telnet...						
ISNC	32 bits	Client Sequence Number	Calculated					
ISNS	32 bits	Server Sequence Number						
AckC	32 bits	Client Acknowledge Number						
AckS	32 bits	Server Acknowledge Number						
WSC	16 bits	Client Window Size						
WSS	16 bits	Server Window Size						
TTLC	8 bits	Client TTL						
TTLS	8 bits	Server TTL						
Flags	Six control bits	The six control bits are as follows: URG - ACK - PSH - RST - SYN - FIN	Not Calculated					

This paper is organized as follows: Section 2 emphasizes the related work. SPIMN architecture is explained in Section 3. Section 4 describes its realization (Design and Implementation). Section 5 presents verification and validation of SPIMN. Finally, the paper concludes in Section 6 with opening the scope for further research.

2. Related Work

Most of the published researches on packet classification have focused on IP address lookup for routing and matching rule sets, with initial packet parsing [8]. For instance, Prasanna et al., [9] have demonstrated IP address lookup at up to 100Gbps rates by using FPGA implementation. Packet matching researches are typically based on the Snort rule-based intrusion detection technology [10]. Attig et al., demonstrated a 100Gbps line rate by coupling the parsing module with a key lookup module, in order to perform complete packet classification [8]. This packet classification subsystem was in turn coupled with a traffic management subsystem to demonstrate 400Gbps network processor by using a dual Xilinx Virtex-7 implementation. Kangaroo is a programmable parser that parses multiple headers per cycle. Kangaroo buffers all header data before parsing, which introduces latencies that are too large for switches today [11]. Weirong et al., presented a novel decision-tree-based linear multi-pipeline architecture of FPGAs for wire-speed and multi-field packet classification [9]. Extensive simulation and FPGA implementation results demonstrated the effectiveness of the solution. The FPGA design supported 10K rules or 1K Open Flow-like complex rules and sustained over 40 Gbps throughput for minimum size (40 bytes) packets.

Stateful Packet Inspection (SPI) was originally developed for Firewall. However, recently, there have been various applications such as Virtual Private Network (VPN), NIDS, Traffic Monitoring and so on. Published works on the packet classification can be divided into two categories: software solutions based on novel classification algorithms running by normal CPU or network processor and hardware solutions based on Ternary CAM, ASIC or FPGA. In general, a software-based solution is flexible but normally CPU time-consuming, which is intolerant when a network traffic load is heavy. For the high-speed realization of packet classification (above 10Gbps line rate), hardware solutions are currently the best [12]. Abhishek et al designed an FPGA-based architecture for anomaly detection in network transmissions [13]. They first developed a feature extraction module (FEM) to summarize the network information to be used at a later stage, and its throughput could reach to 40Gbps. Ashok proposed architecture of a Distributed Intrusion-Detection System (DIDS), which could be implemented in more than one FPGA and MGTs (Multi-Gigabit Transceivers support serial communication speeds up to 10Gpbs per channel) [6]. Ioannis et al., introduced an original design for reconfigurable hardware by implementing the packet pre-filtering technique and it could process packets at 2.5 to 10Gbps [14].

3. SPIMN ARCHITECTURE

Figure 2, consists of six main components that work together.

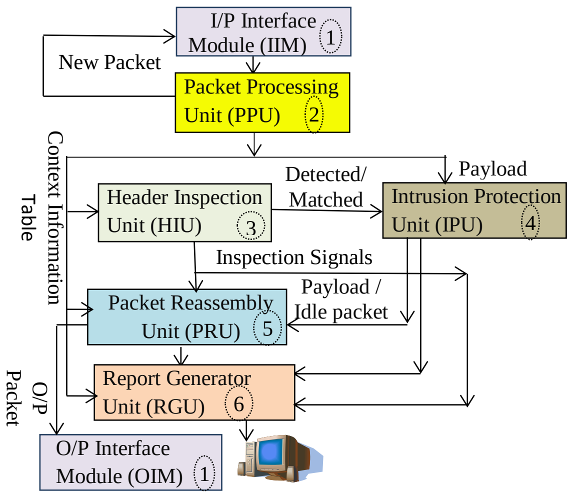
I/P Interface Module (IIM) 1
New Packet
Packet Processing Unit (PPU) 2
Context Information Table
Detected/ Matched
Payload
Header Inspection Unit (HIU) 3
Intrusion Protection Unit (IPU) 4
Inspection Signals
Payload / Idle packet
Packet Reassembly Unit (PRU) 5
O/P Packet
Report Generator Unit (RGU) 6
O/P Interface Module (OIM) 1

Figure 2. General Architecture of IDP System

A. *Input Output Interface Unit (IOIU)*

The function of IOIU is to establish the connection between SPIMN and the physical network. It provides low-level services for interactions with physical network standards. Such architecture consists of two modules as shown in Figure 3:

1) Input Interface Module (IIM)

The functions of IIM are: - Receive the traffic from the network, Convert the traffic into packets, Queue the packets, and send the packets to the Packet Processing Unit (PPU).

2) Output Interface Module (OIM)

OIM performs the following functions Receive the packets from Packet Reassembly Unit (PRU) and Report Generator Unit (RGU), Queue the packets, Convert the packets into traffic, and send the traffic to the network.

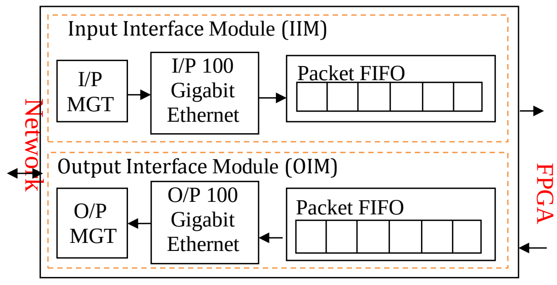

Figure 3. The Block-diagram of Input Output Interface Unit (IOIU)

I/P and O/P MGT (Multi Gigabit Transceiver): provide low-level services for interactions with physical network standards such as Gigabit Ethernet. I/P and O/P 100 Gigabit Ethernet. Convert the Ethernet frames into packets and check the validation of the frames. Packet FIFO (first input first output) buffers: store and pass the packet.

B. *Packet Processing Unit (PPU)*

The functions of PPU are to extract the packet into header and payload. After that, it sends the payload to the Intrusion Protection Unit (IPU) and decodes the header to generate a header fields table classified by the protocol (Context information table).

PPU mainly consists of: - Packet Extractor Module (PEM), and Header Parser Module as shown in Figure 4.

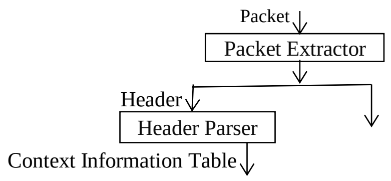

Figure 4. The Block-diagram Packet Processing Unit (PPU)

1) Packet Extractor Module (PEM)

The main functions of the packet extractor are the following: - Extract the packets that received from IIM into header and payload, Send the header to the Header Parser, and send the payload to the Intrusion Protection unit (IPU).

It consists of a set of buffers, comparators, decoders, and state machines that sequencing identifies as the Ethernet frame elements within a packet. It is specified as a text file containing a name and size of all elements.

2) Header Parser Module (HPM)

The functions of the header parser are: -check and identify the protocol, Extract fields for processing by subsequent stages of the system, classify the header fields by the protocol (Context Information table), and send this table for Header Inspection Unit (HIU), Packet Reassembly Unit (PRU) and Report Generator Unit (RGU).

There are two types of a header parser: - 1) Fixed header parser (Our Work), and 2) Programmable (Dynamic) header parser (The future work).

Mainly it consists of 2-parts as shown in the following Figure 5.

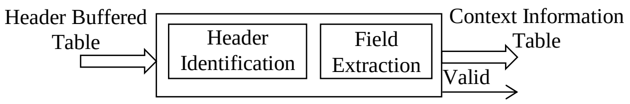

Figure 5. Abstract Architecture of Heer Parser Model (HPM)

The functions of Header Identification: - Receive the header and store it in a buffer, and identify the type and the length of the header by the aid of the parse graph (Parse table), and the function of Field Extraction: - Extract the header by using the parse table into the fields, Store the extracted fields in a context information table, Send the Context Information table into Header Inspection Unit (HIU), and Send the valid signal to the HIU, IPU, and RGU.

C. Header Inspection Unit (HIU)

HIU is the main part of SPIMN. It inspects the packet by two techniques: - 1) checking the anomaly values (Anomaly Detection) via checking the header fields values, and 2) detecting the intrusion through matching the context information table with the header Snort rules. After the detection it sends the result to the Packet Reassembly Unit (PRU), Intrusion Protection Unit (IPU), and Report Generator Units (RGU) to complete the operations. It can be used as a filtering unit to speed up the detection and if any anomaly occurs, this packet is not sent to the matching circuit (i.e. filtering the packet) to speed the inspection process.

So it consists of two modules as illustrated in Figure 6.

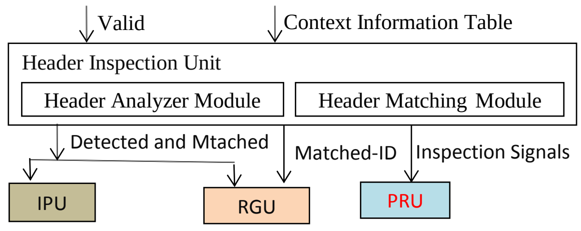

Figure 6. The Block-diagram of Header Inspection Unit (HIU)

1) Header Analyzer Module (HAM)

HAM is used to check the values of the header fields via a set of flow testers. The main roles of the Header Analyzer are: - inspect the header fields, send the result (inspection signals) to Packet Reassembly Unit (PRU), Generate the detected signal from the inspection signals, and send the detected signal to the Header Parser Module, Report Generator Unit (RGU), and Intrusion Protection Unit (IPU).

It consists of a flow tables generators and a set of flow tables' testers. The block-diagram of the Header Analyzer Module is shown in Figure 7. The fields of flow tables' are combinations of the fields of Table 1. The relationship between the testers and the flow tables is one-to-one to speed up the inspection.

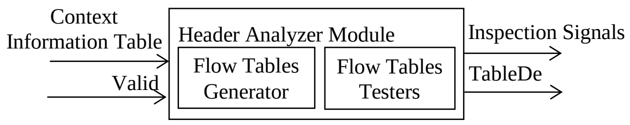

Figure 7. The Block-diagram of Header Analyzer Module

2) Header Matching Module (HMM)

The Header Matching Module is a rule-matching module to find out if the header information matches with any of the given header snort rules. These rules may contain checking of the flow definition.

It consists of a set of comparators arrays, and a header snort rule detector. Figure 8 shows the block-diagram of the HMM.

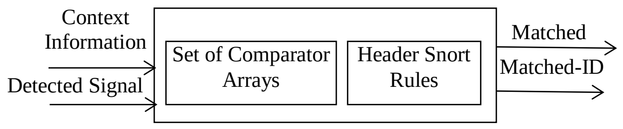

Figure 8. The Block-diagram of the Header Matching Module (HMM)

D. Intrusion Protection Unit (IPU)

The functions of IPU are: check the detected and matched signals, and replace the anomaly or infected packet by the idle pattern to filter and prevents the intrusion.

It consists of: - checking, replacement, and sending modules. Figure 9 shows the block diagram of the IPU.

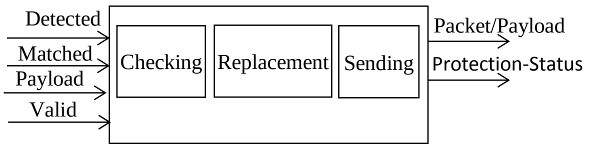

Figure 9. The Block-diagram of the Intrusion Protection Unit (IPU)

E. Packet Reassembly Unit (PRU)

PRU is one of the main tasks that the monitored TCP flow should accomplish. By parallelizing the tasks of reassembling TCP packets on the server and the client side of the FPGA, the performance of the stateful TCP inspection can be greatly improved. It gathers the header and the payload (or idle packet in the case of infected) to pass it to the server or the client.

It consists of three modules: - Dispatcher, Streaming, and Tracking as shown in Figure 10.

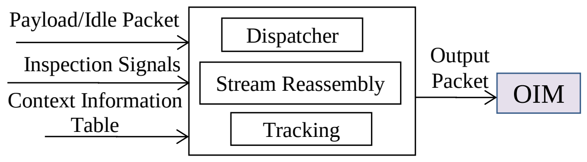

Figure 10. The Block-diagram of The Packet Reassembly Unit (PRU)

The functions of Dispatcher are: - Receive the packet, or the idle packet from the IPU, Receive header fields from the extractor module, receive the C/S Direction signal from header analyzer, check the C/S signal to define the output direction (to the client or to the server), Enable the stream reassembly as a result of the checking of C/S, and Send the packet or the idle packet to the stream reassembly. The functions of Stream Reassembly are: - Pack-up the header and data into a packet, and sending the packets to the tracking. The functions of tracking module are queuing the packet and send it to the output Interface Module (OIM).

F. *Report Generator Unit (RGU)*

Report generator unit (RGU) counts and tabulates the frequency for each rule and accumulates the numbers of matched and unmatched packets. These results are reported to the host PC about the detection and protection processes.

RGU consists of a set of data gathering buffers, searching module and a set of tables generators as shown in Figure 11.

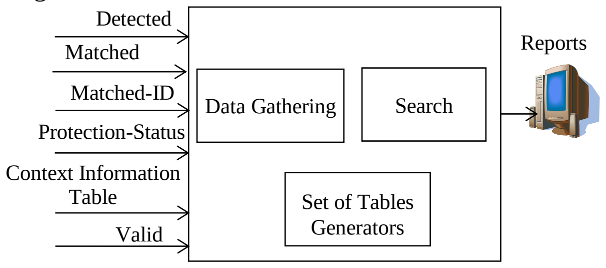

Figure 11. The Block-diagram of the Report Generator Unit (RGU)

In practice, SPIMN is quite challenging: - 1- Throughput: must run at line-rate (100Gbps). (2) Parallelism: especially in header inspection and packet reassembly units.

4. Design and Implementation

In this section, I presented the SPIMN procedure, SPIMN design and implementation.

A. *SPIMN Procedure*

In the following, the SPIMN sequence of operation is pointed out.

↓ Traffic

Input Interface Module (IIM)

1. Receive the traffic from the network.
2. Convert the traffic to packets.
3. Queue the packets.
4. Send the packets to PPU.

Packet Processing Unit (PPU)

1. Receive the packets from Input Interface Module (IIM).
2. Extract the packets into header and payload.
3. Parse the header into the context information table classified by the protocol
4. Send the table to HIU, PRU and RGU.
5. Send the payload to IPU.
6. Send a request to new packet

Header Inspection Units (HIU)

1. Receive the context information table from PPU.
2. Analyze the values of header fields to know if there are any anomalous values or not.
3. Send the inspection signals of table 2 to PRU.
4. Send the detected signal to IPU, and RGU.
5. Compare (match) the context information table with header snort rules table to detect the intrusion.
6. Send the Matched signal to IPU and RGU.

Intrusion Protection Unit (IPU)

1. Listen to Header Inspection Unit (HIU).
2. Receive matched and detected signals.
3. Receive the payload from the PPU.
4. Send the payload or the idle packet as a result of the detected and matched signals to the PRU.
5. Send the protection-Status to the RGU.

Report Generator Unit (RGU)

1. Receive the number of Inspection, detected, and matched signals from HIU.
2. Receive the Protection-Status from IPU
3. Generate reports about the infection and the Protection-Status.
4. Send these reports to the OIU

Packet Reassembly Unit (PRU)

1. Receive inspection signals from the HIU.
2. Receive the context information table from PPU.
3. Receive the payload or the idle packet from IPU.
4. Marshal the header and the payload into packets.
5. Direct the packets into client or server (via OIM).

The SPIMN operational steps are shown in Figure 12.

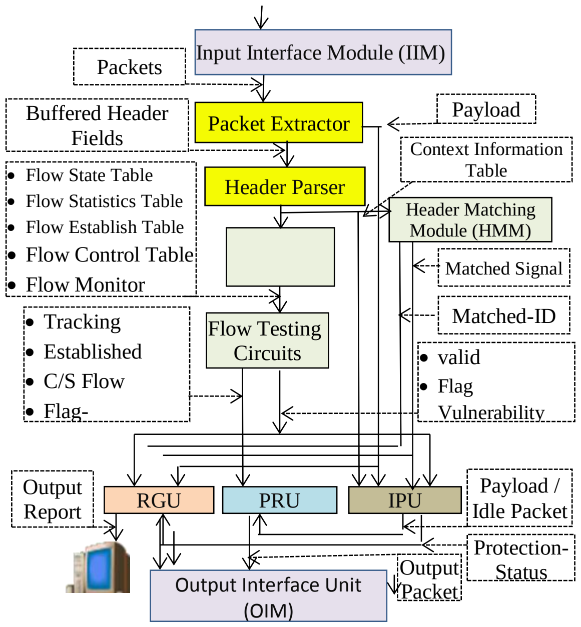

Figure 12. SPIMN Operational Steps

B.SPIMN Design and Implementation

I used Xilinx ISE 14.1 in both the design and implementation of SPIMN to satisfy the requirements and the architecture those given in section (3). The design and implementation steps are shown in Figure 13.

In the load onto FPGA step, I select Xilinx Virtex-7 690T FPGA device. Because Virtex-7 FPGA series has integrated features that include FIFO and ECC logic, DSP blocks, PCI-Express controllers, Ethernet MAC blocks, and high-speed transceivers (setting in SW). The Modules are designed and developed in VHDL by using Xilinx ISE 14.1.

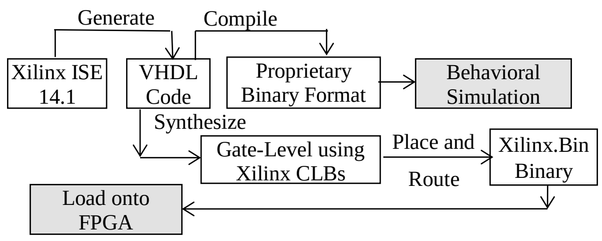

Figure 13. Implementation Steps

1. Input Output Interface Unit IOIU

IOIU is designed to support 512-bit width in 312 MHz operation (to enable the work over 100 Gbps), full-duplex 100Gb/s with physical interfaces CGMII (100 Gigabit per Second Media Independent Interface) and CAUI-10 (100 Gigabit per Second Attachment Unit Interface) connects to the I/O of the FPGA. It was created by Xilinx Ethernet multi-Giga Generator. Because of The registered Xilinx 14.1 ISE generates Ethernet LogiCORE for an integrated facility to work with Ethernet level. It offers an integrated 100 Gigabit per second (Gbps) Ethernet Media Access Controller (EMAC) and Physical Coding Sub-layer (PCS) core of high-performance interconnect technologies for communications equipment and flexible implementation of the IEEE 802.3ba (standard includes 100GBase-SX transmission over Multimode fiber). The PCS portion of the IP can be configured in CAUI-10 (10 lanes x 10.3125G. Also Xilinx offers two wrappers for the integrated block: AXI4-Stream and AXI4-non-stream and high-speed GTH transceivers. We used AXI4 stream and GTH because it is high-end low-power transceivers. Also Xilinx 14.1 used to design and implement the other units of the SPIMN and also in the simulation to test the system.

Algorithm 1: Input Interface Module (IIM)

```
Receive a new frame from the Ethernet MAC Wrapper.
Convert the input traffic format from CAUI to CGMII to
    handle “idle” patterns easily.
Strips off Preamble, SFD, Pad
Verify the FCS.
Send enable signal to FIFO (start of frame)
Store the frame remaining in a FIFO and frames marked
    as invalid are dropped.
Store the packet in FIFO
Send EoF (End of Frame) to FIFO and PPU
Out the packet from FIFO to PPU
```

Algorithm 2: Output Interface Module (OIM)

```
Receive a new packet from the PRU.
Convert the packet into frame
Convert the format from CGMII to CAUI, to send into
    the Ethernet MAC Wrapper.
Send enable signal to FIFO (start of frame)
Store the packet in FIFO
Send (end of frame) to FIFO
Out the packet from FIFO to network
```

IOIU actually consists of two sub-layers: 100Gb/s MAC and 100GBASE- PCS support for CAUI-10 interfaces, and two Ethernet FIFO buffers as shown in Figure 14. Each FIFO is two Dual Ports block RAMs with 4096 bytes in each FIFO.

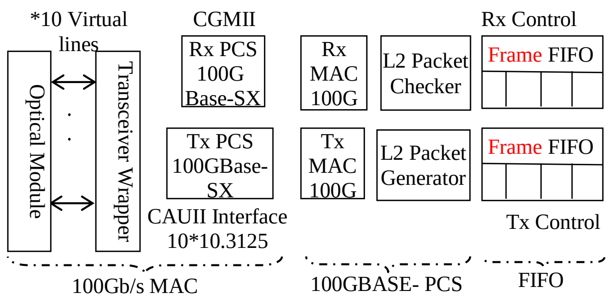

Figure 14. The Gate-level of Input-Output Unit

2. Packet Processing Unit

PPU extracts the packets into header and payload and after that useful header information from MAC Layer, network layers (IPv4, ICMP), and transport layers (TCP, UDP). Any other layer is not being decoded.

a. Packet Extractor Module (PEM)

The extractor divides the packets into header fields and data after that it sends the header fields to the parser and the data to the Intrusion Protection Unit (IPU) as shown in Figure 15. It strips some frame elements as preamble, start of frame delimiter (SFD), Pad and FCS. If SFD did not receive, it generates a reset to the frame if not, it generates a valid signal to enable the parser module.

Algorithm 3: Packet Extractor Module (PEM)

```
Wait and listen
If Receive EoF=1 then
  Enable the MAC state machine
  Receive the data from the FIFO
  Generate the frame elements
  If (SFD $\neq$ 0xD5)
    Set valid=0
    Exit
Else
    Set valid=1
    Store the Source address and destination address and
      Ether-Type in the header fields' buffer
    Store the data in the payload buffer
    Send the buffered header fields to the header parser
    Send the payload into the IPU
    Set new-frame=1
    Send new frame to a FIFO buffer
    Set EOE=1
    Send EOF to the parser
Send valid to HPM, IPU, and RGU
```

It consists of a state machine, Demultiplexer, and numbers of buffers as shown in Figure 15.

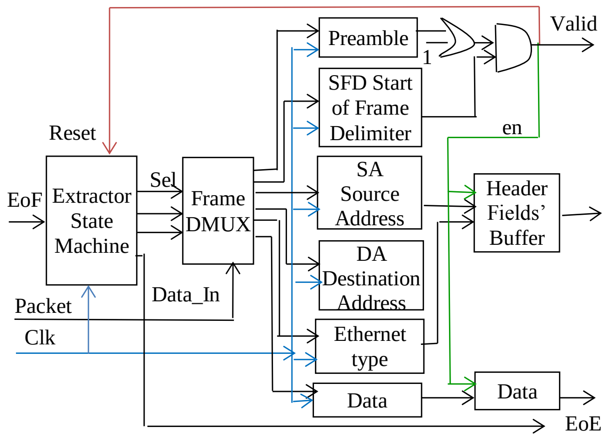

Figure 15. The Gatelevel of Packet Extractor Module (PEM)

b. Header Parser Module (HPM)

Header parser module receives EoE (End of Extortion), valid frame, and the buffered header fields from the extractor module to extract useful header information from MAC layer, network layers (IPv4, ICMP) and transport layers (TCP, UDP).

So it consists of a set of decoders for IP, TCP, UDP, and ICMP. All of the generated header fields are stored in registers (Context Information Table). A registered header data output (width 120 bits) is created. The operations states are illustrated in Figure 16.

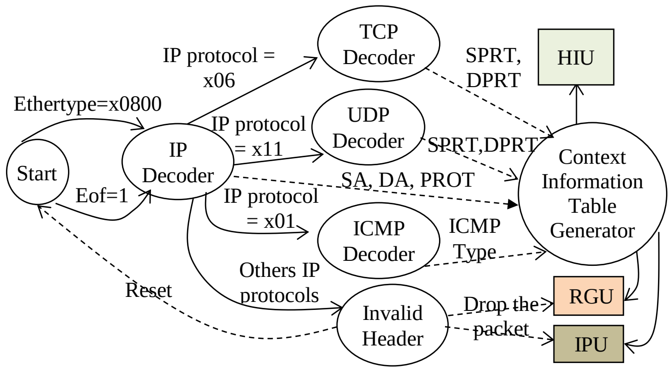

Figure 16. Operational Steps of the Header Parser Module (HPM)

Algorithm 4: The parser algorithm

```
Waite and listen
If (EOF =1 && Valid=1)
  Receive the buffered header fields from the FIFO
  // Check if it IP4
  If Ether-Type = 0x800          // Extract IP4 elements
    Valid =1
    For i=0, i< 32, i++
    Read the buffered header fields
    Store in the source address register
    For i=0, i< 32, i++
    Read the buffered header fields
    Store in the destination address register
    For i=0, i< 8, i++
    Read the buffered header fields
    Store in the protocol register
    // Check the protocol type
    If protocol = 0x06              // Extract TCP elements
      For i=0, i< 16, i++
      Read the buffered header fields
      Store in the source port register
      For i=0, i< 16, i++
      Read the buffered header fields
      Store in the destination port register
    Else If protocol = 0x11      // Extract UDP elements
      For i=0, i< 16, i++
      Read the buffered header fields
      Store in the source port register
      For i=0, i< 16, i++
      Read the buffered header fields
      Store in the destination port register
    Else If protocol = 0x01     // Extract ICMP elements
      For i= 0, i< 8, i++
      Read the buffered header fields
      Store in the ICMP type register
    Else
      Break
Else
    Valid = 0
Send valid to HAM, IPU, and RGU
```

Figure 17 shows the Gate-Level of Header Parser

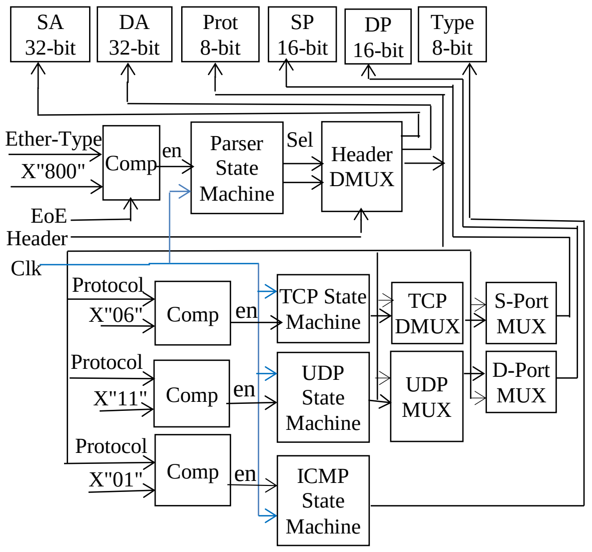

Figure 17. The Gate-level of the Header Parser Module (HPM)

3. *Header Inspection Unit (HIU)*

Such as architecture of HIU described in section 3.3,

a. Header Analyzer Module (HAM)

Header Analyzer checks the header fields' values from the points of validation, direction and the tracking. It consists of a set of flow tables generators, and flow testers (flow state, flow statistics, flow established, flow control, and flow monitor).

The input is the context information table (described in the Table 1) and the outputs are the inspection signals (set of status and control signals as shown in a Table 2) and detected signal. Figure 18 shows the Gate-level of HAM.

Algorithm 5: Header Analyzer Module HAM

```
Wait and listen
Receive the valid signal
If (valid =1)
    Receive the context information table
    Generate the flow tables
    Send the flow tables to flow testers
    Generate the inspection signals table from the testers
        result
    Generate detected signal
    Send the inspection signals to PRU
    Send the detected signal to the HMM, IPU
```

TABLE II. TABLE 2 INSPECTION SIGNALS

Field	*Size*	*Description*
Establishment	1 bit	Connection Establishment Status 0: Not Established 1: Established
Half-Closed	1 bit	Half Closed Status 0: Half-Closed 1:Not Half-Closed
C/S Direction	1 bit	1: Packets are sent from the client side 0: Packets are sent from the server side
Flag-vulnerability	1 bit	1: Infected packet 0 : Uninfected packet
Tracking	1 bit	Flow the packet to the output latch
Down the Connection	1 bit	TTL
	2 bits	TCP flags (RST – FIN)

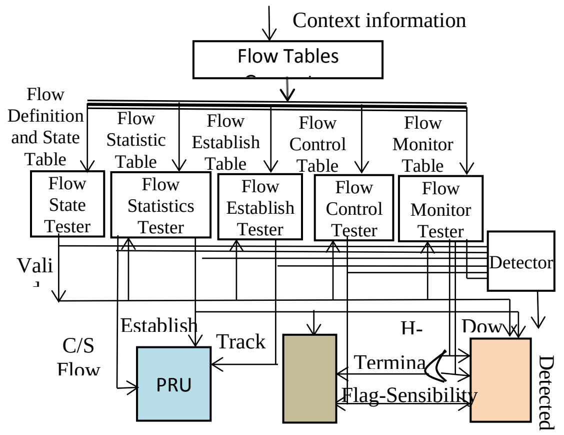

Figure 18. The Gate-level of the Header Analyzer Module (HAM)

b. Header Matching Module (HMM)

HMM generates the rule detector for each rule by using the table index. So the header table generator is used to repeat the header vector to 20 bit register comparator to compare to the rules in the parallel, and the comparator module compares the context information table with the header snort rules. It is implemented by FPGA because it is more suitable to implement thousands of pattern comparators operating in parallel. The table index indicates whether the header is matching results corresponding to the Snort rule. The outputs of comparator module are Matched signal and Matched-ID (matched identification).

Algorithm 6: Header Matching Module (HMM)

```
Generate the header snort rules patterns
Distribute the snort rule pattern to the comparators
Receive the context information table
Receive the detected signal
If (Detected ==1)
    Exit
Else
  Generate the header vector
  Repeat the header and distribute to 20 comparators
  Compare the header vectors with the snort pattern rules
  If match
    Matched = 1
    Encode the comparators results
    Send the Matched-ID to RGU
  Else
    Matched signal = 0
Send a matched signal to the IPU, and RGU.
```

Figure 19 shows the Gate-level of Header Matching Module.

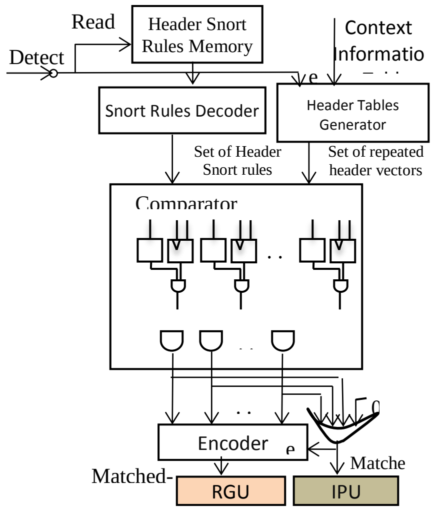

Figure 19. The Gate-level of the Header Matching Module (HMM)

4. Intrusion Protection Unit Gate-Level

Figure 20 shows the processes states in the IPU.

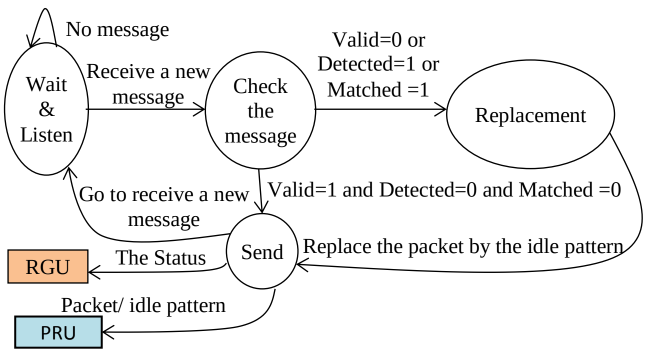
No message
Wait & Listen
Receive a new message
Check the message
Valid=0 or Detected=1 or Matched =1
Replacement
Valid=1 and Detected=0 and Matched =0
Go to receive a new message
Send
Replace the packet by the idle pattern
RGU
The Status
Packet/ idle pattern
PRU

Figure 20. Processes State of the Intrusion Protection Unit (IPU)

The intrusion-protection unit is used to prevent the matched attacks and intrusions. The inputs are the payload, idle packet, and detecting, matching signal and terminating. The output is the packet or the idle pattern as shown in Figure 21.

Algorithm 7: Intrusion Protection Unit (IPU)

```
Wait and listen
Receive payload
Receive detected signal
Receive the Matched signal
Receive valid signal
If Matched signal=0 and detected=0 and valid=1
  Send the payload to PRU
  Protection-Status = 0
Else
  Replace the packet with idle pattern
  Send the idle pattern to the PRU
  Protection-Status =1
Send the Protection-Status to RGU
Return
```

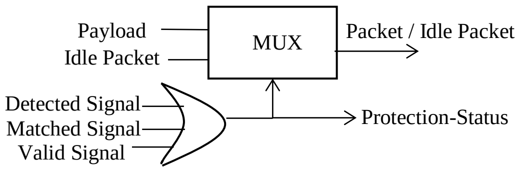

Figure 21. The Gatalevel of the Intrusion Protection unit (IPU)

5. Packet Reassembly Unit Gate Level

The packet reassembly unit is the opposite of the packet extraction. It generates a packet, and also routes it to the server or to the client side.

Algorithm 8: Packet Reassembly Unit (PRU)

```
Receive the inspection signals from the HIU
Receive the header and header information from the PEM
Receive the payload or idle packet from the IPU
```

```
If C/S =0
    Reassemble the TCP stream to the server side
Else
    Reassemble the TCP stream to the client side
```

PRU consists of four parts: two Dispatchers (one for the header and the other for the data), Server Stream, Client Stream, and two Block-Memories as shown in Figure 22.

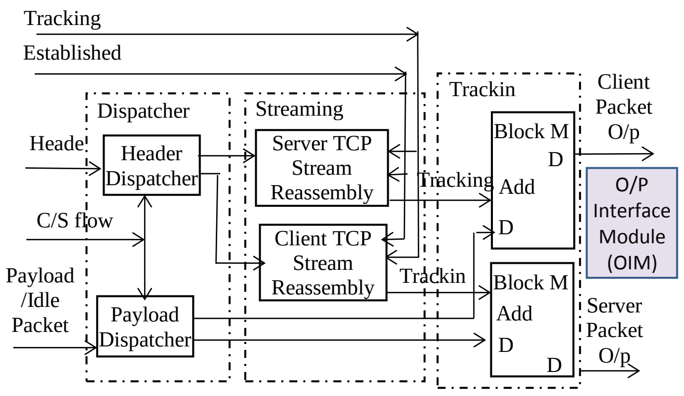

Figure 22. The Gatalevel of the Packet Reassembly Unit (PRU)

6. Report Generator Unit (RGU) Gate-Level

Report Generator Unit generates a status table that contains the number of invalid, anomalous, infected packets, and the type of infection. Also the report shows the intrusion protection status. Figure 23 shows the operation states of the RGU. The search module used to search in the index table by the Matched-ID to find out the snort type (Snort Identification, Alert).

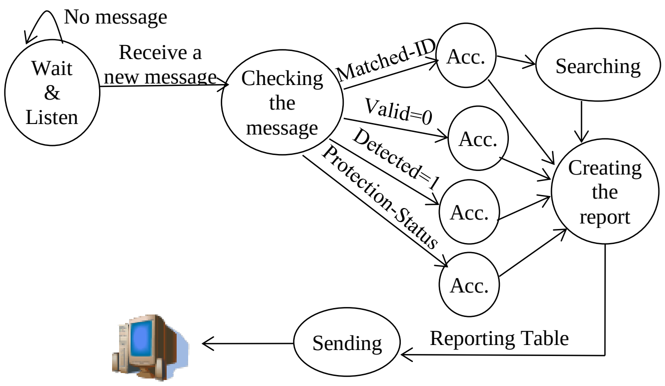

Figure 23. The Operation States of the Report Generator Unit (RGU)

The inputs are Context Information Table and the signals (Matched, Matched-ID, Valid, Detected, and Protection-Status). The output is a status report about detection and the protection. The report generator unit consists of a lookup table, signal checker, table generator. SPIMN exports the monitoring data to an external host through a standard RS-232 serialed approach. This approach requires minimal hardware. I use snort because it is a popular open source in network intrusion detection and easy to customize.

Algorithm 9: Report Generator Unit (RGU)

```
Receive the Valid signal
Receive the Detected signal
Receive the Matched signal
Receive the Matched-ID signal
Receive the Protection-Status
Receive the Header Buffered Table
If valid = 0
    Accumulate the invalid_ packet.
If Detected =1
    Accumulate the detected-packet
If Matched =1
    Accumulate the matched-packet
    Search in L.U.T by Matched-ID to get the alert
  message
Store in a table
Send the table to the monitoring PC (Output The Report)
```

5. SPIMN VERIFICATION AND VALIDATION

To test the design and implementation of section (VI), two steps are carried out: - one for simulating SPIMN, as such, using ModelSim ISE6.0, and the second is verifying it.

1. SPIMN Verification

SPIMN as an entire prototype, implemented experimentally by making use of the above components. Figure 24 shows the realization of the information flow.

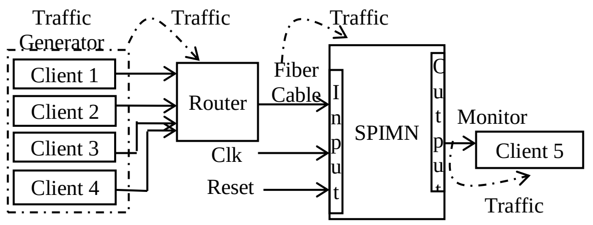

Figure 24. SPIMN Realization

In SPIMN:
1) The input section consists of:
I. One port connects to the router connected with 4 PCs generated the traffic.
II. Clock: The clock signal is used as input for each component in the SPIMN to perform an operation of the synchronization.
III. Reset: The reset signal is used as a control input signal for each component of the SPIMN.
2) Internal SPIMN components

3) The output section consists of one port connects to the monitored PC.

2. *Validation*

Finally, 1 test and evaluate the SPIMN model to make a comparative study between SPIMN and the other systems. The main result of this comparative study is, it can work with 100 Gigabit Ethernet network without any modification. Rule match module used a sub-set of 100 Snort rules in testing the design and implementation of SPIMN (because of the difficulty of generated the snort rules manually so it is planned to recover it in the future).

The steps of SPIMN test as the following :- 1) Generate traffic that includes certain types of attacks and intrusions, 2) Send this traffic to PC1 which contains a standard Snort detection system and notice the report, 3) Send this traffic also to SPIMN and note the output report, which is will sent to the PC2, 4) Compare these reports to check the accuracy (Inspection) and 5) Resend the output traffic from SPIMN to PC to detect again to see that the detected intrusion are removed or not (to satisfy from the protection unit). It is shown in the figure 25.

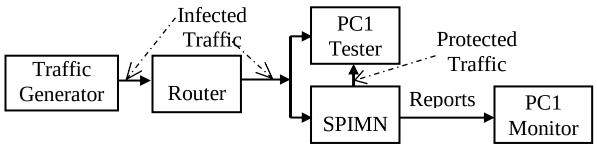

Figure 25. SPIMN Test

The performance of an SPI-based intrusion-detection system mainly depends on the performance of the processing context information table. SPIMN can also do stateful packet inspection in real time and perform two states of detections (header analyzer to detect anomaly values – header matching with the snort rules) to allow more efficient generation of detection. These are the main contributions in SPIMN improving. SPIMN implemented on Xilinx ISE 14.1 Virtex-7 XC7V2000T with '–2' speed grade FPGA device. This model can achieve the throughput over 100 Gbps with dual port memory while it can support more than 2,000 Snort rules. Processing the data flow on the Server side and the Client side in parallel and fully considering context information on the TCP connection are our main contributions to improving the processing of TCP connections in NIDS.

6. CONCLUSION

In the present paper, we described the architecture, design and hardware implementation of 100 gigabits Stateful inspection using FPGA. The proposed model (SPIMN) presents intrusion detection, protection, and report generation.

SPIMN is experimentally tested via two steps. The first step is based on a Xilinx simulation environment for ensuring the correctness of the system architecture before the implementation and by creating a test bench circuit. It is also used for measuring the average response time of SPIMN at 100Gbps, where the system performance is reported and evaluated. The second step is an experimental verification for SPIMN through a network. In the third step, we test and evaluate SPIMN to perform a comparative study between the proposed system and the others. In particular, we demonstrated performance improved by optimized, efficient memory access in FPGA logic unit and parallel processing in both the parts of header inspection unit and also in the unit of packet reassembly. And also the processes in both PRU and HIU are working in parallel. The use of FPGA made it easy to modify and develop the system.

It is planned to extend SPIMN in three directions. The first direction takes place by replacing the header parser with a programmable parser to work with any protocol and to generate the rules of matching automatically (Dynamic SPIMN) and the second direction by improving the Stateful packet inspection to inspect the payload and header together (Deep Packet Inspection DPI). The third direction is to create a circuit to handle regular expressions and by developing a sequencer to handle non-pattern-matching rules.

REFERENCES

[1] CRONIN, Brendan. Hardware acceleration of network intrusion detection and prevention. 2014. PhD Thesis. Dublin City University.

[2] AV-Test Institute (2013). Malware statistics. AV-Test Institute, Germany; Obtained through the internet:Web: http://www.av-test.org/en/statistics/malware/ [accessed 4 Oct. 2013]

[3] Seungyong Yoon, Jintae Oh, Jongsoo Jang "Design of SPI module in large-scale network"Conference: Advanced Communication Technology, 2006. ICACT 2006. The 8th International Conference, Volume: 3

[4] Parsons, Christopher. Deep Packet Inspection in Perspective: Tracing its lineage and surveillance potentials. Queen's University, Surveillance Studies Centre, 2008.

[5] Chang-Su Moon and Sun-Hyung Kim "A Study on the Integrated Security System based Real-time Network Packet Deep Inspection" International Journal of Security and Its Applications Vol.8, No.1 (2014), pp.113-122

[6] Ashok Kumar Tummala and Parimal Patel "Distributed IDS using Reconfigurable Hardware" IEEE 2007.

[7] Sampath V.P "An FPGA-Based Network Intrusion Detection System " World Journal of Science and Technology, www.worldjournalofscience.com 1 (8): (100-102) 2011

[8] M. Attig and G. Brebner. 400Gbps Programmable Packet Parsing on a Single FPGA. In Proc. ANCS '11, pages 12 {23, 2011.

[9] Yang, Y., and Prasanna, V. 2010. High Throughput and Large Capacity Pipelined Dynamic Search Tree on FPGA. In Proceedings of the 18th ACM/SIGDA International

Symposium on Field Programmable Gate Arrays (Monterey, CA, USA, Feb. 2010), 83-92.

[10] SNORT Network Intrusion Detection System, http://www.snort.org

[11] C. Kozanitis, J. Huber, S. Singh, and G. Varghese. Leaping Multiple Headers in a Single Bound: Wire-Speed Parsing Using the Kangaroo System. In Proc. INFOCOM 2010, pages 1 {9, Mar. 2010.

[12] Wang Yong-gang, Zhang Tao, Zheng Yu-Feng, Yang Yang "Realization of FPGA-based Packet Classification in Embedded System" Technology Conference, Singapore, I2MTC 2009 International Instrumentation and Measurement 5-7 May 2009

[13] Abhishek Das, David Nguyen, Joseph Zambreno, GokhanMemik, and AlokChoudhary," An FPGA-based Network Intrusion Detection Architecture" IEEE Transactions On Information Forensics And Security, Vol. 3, No. 1, March 2008

[14] Ioannis Sourdis "Designs & Algorithms for Packet and Content Inspection" Thesis Printed in The Netherlands Copyright 2007 Ioannis SOURDIS electronic and computer engineer Technical University of Crete Geborente Corfu, Griekenland 2007