Sixth Framework Program
Priority 5
FP6-2005-IST-5

Project Acronym: **WISDOM**
Contract number: **033847**
Contract for: **Specific targeted research project**

Workpackage 3	**New Security Algorithm Design**
Deliverable 3.1	**Security Applications Partitioning**

Contractual Date of Delivery: 31st May 2007
Actual Date of Delivery: June 2007
Delivery Security Class: CO (for consortium only)

Workpackage Leader:

George Kopidakis
Foundation for Research and Technology - Hellas
Institute of Computer Science
P.O. Box 1385, GR-711 10, Heraklion, Crete, Greece
Tel: +30 2810 391497
E-mail: kopidaki@ics.forth.gr

Project Co-ordinator:

Graeme Maxwell
Centre for Integrated Photonics
B55, Adastral Park
IPSWICH, IP5 3RE, UK
E-mail: Graeme.Maxwell@ciphotonics.com
Telephone: +44 1473 663247
Facsimilie: +44 1473 663295

TABLE OF CONTENTS:

1. Participant list

Role*	No.	Participant name	Short Name	Country	Date enter Deliverable	Date exit Deliverable
	1	Centre For Integrated Photonics	CIP	UK	Month 6	Month 12
	2	British Telecom	BT	UK	Month 6	Month 12
	3	University College Cork	UCC	Eire	Month 6	Month 12
WP Leader	4	Foundation for Research and Technology – Hellas	FORTH	Greece	Month 6	Month 12

2. Abstract

WP3: "New Security Algorithm Design"
Deliverable 3.1: "Security Applications Partitioning"

The aim of Workpackage 3 is to develop algorithms for security applications that take advantage of all-optical processing capabilities. The main objectives are to:

- identify critical security application components which can be efficiently implemented in the optical domain
- characterise constraints to algorithmic components and develop novel analytical techniques for simplified pattern matching
- design a Security Application Programming Interface (SAPI) which will be the interface between high-level security applications and low-level optical implementation.

The first of these objectives is achieved through the "Security Applications Partitioning" described in this report. Taking into account the boundary conditions imposed by optical hardware and by security algorithms, we identify security application components which can be effectively and efficiently implemented in the optical domain. These components can, in principle, operate almost entirely in the optical domain using optical processing capabilities already developed within the project. We also describe a software platform which simulates optical device operations. We develop this software tool in order to use it for achieving the forementioned objectives of WP3 by facilitating (i) the performance evaluation of components implemented in optics, (ii) the design of more complex "optical" algorithms, (iii) the interface between electronic and optical operations (later in the project).

3. Introduction

The objectives of the WISDOM project, the landscape in terms of both optical hardware and network security applications, and guidelines on the approaches to be used in the project have been described at some length in Deliverables D2.1 [1] and D2.2 [2] of WP2. The main objective of WP2 was to identify boundary conditions for the development of security applications that are imposed by the different physical architectures available for the opto-electronic subsystems. More specifically, D2.1 defined the optical firewall physical architecture and concomitant D2.2 considered security applications, some implications on physical hardware architecture and limitations that optical hardware imposes on security algorithm design. From WP2 it became clear that a critical component to be implemented in the project is an all optical pattern recognition system. Within WP4, the design of such a system and an all-optical pattern matching algorithm was introduced and presented in detail in D4.1 [3].

The main challenge of this project is to demonstrate that it is possible to bridge the gap between the very fast (at least 40 Gbps) optical data transmission and the much slower electric data security checks by pushing critical security functions all the way down to the optical domain. Having a general view from WP2 in regards to what is feasible and practically achievable in the optical domain, compared to what is usually adapted in the electronic domain, and having also the design of a working tool from D4.1, which is able to match a bit pattern in an optical bit stream, this document presents real security applications that can be deployed in the optical domain. In this report we outline the main considerations that were made in order to identify efficient operations in the optical domain, which are related to basic Firewall and Network Intrusion Detection/Prevention System (NIDS/NIPS) functionality, as well as transmitted data packet structure and decoding (Section 4) and to optical data and hardware restrictions (Section 5). We then describe security applications in the optical domain followed by specific examples which illustrate that they can be readily implemented with components developed in the project (section 6). Moreover, we describe deployable infrastructures that combine electrical and optical components in order to extend the state of the art of modern Firewalls and NIDS/NIPS with new systems that are able to cope with fast network data transmission, since a part of their functionality is implemented natively in the optical domain. Finally, a software framework, designed especially for the WISDOM project and which is able to simulate optical devices is presented (Section 7). This software framework is able to model various aspects of optical building blocks, such as SOA or SOA-MZI gates analysed in detail in D4.1. Using this software tool, specific applications that combine optical parts can be modelled and profiled. We finish with conclusions and remarks (Section 8).

4. Security Applications

An extensive overview of modern network security applications (with some generic examples) currently used in the electronic domain has been presented in D2.2 [2]. Security platforms against malicious attacks include Firewalls, which prevent communication for specific servers and services, and NIDS/NIPS, which form an additional line of defence and perform more sophisticated signature or anomaly based attack detection. Here, we reiterate some characteristics of these well known technologies, which are used in the electric domain for security purposes, with emphasis in features that are amenable to optical implementation.

4.1 Standard Firewalls

The term *firewall* is one of the first notions in Computer Science in the field of System and Information Security. The basic principle which drives firewall operation is a list of rules, according to a series of policies, specified by the organisation that runs a system. This list of rules defines what is allowed and what is not, in terms of network traffic, to enter or exit the system. In other words, the firewall controls the network interaction of a computer with other computers.

This kind of control offered by a simple firewall is usually minimal. The information that a firewall can use is limited to computer identities and protocol/service identities. More precisely, a firewall can inspect the headers of a network packet and decide if this packet is allowed to enter or leave the system according to a list of rules, composed by computer identities and protocol/service identities.

A typical usage of a firewall is the following. A system administrator can configure a firewall to deny any computer of the outer (internal) network to access any protocol/service in the internal (outer) network except the Web service. This can be translated to the simple following rules:

```
iptables -P INPUT DROP
iptables -A INPUT -p tcp -m tcp --syn --dport 80 -j ACCEPT
```

The above rules are examples of *iptables' rules*. The iptables component is the standard firewall technology used in the Linux kernels during the last few years. The meaning of the above rules is the following. The first rule forces the firewall to drop any incoming network packet from any computer of the outer network that targets any service in the administrative system. In other words, this rule defines the default behaviour of the firewall: *drop any incoming network packet*. The second rule instructs the firewall to differentiate from the original behaviour and allow network packets that have as *destination port* the port with number 80. This kind of identity (destination port 80) is typical for the Web service. The majority of modern Web servers operate by accepting incoming requests in the port with number 80. Thus, this rule permits to any computer in the outer (internal) network any access targeting any Web server in the internal (outer) network. The combination of the two rules is the product of the firewall configuration we mentioned in the previous paragraph:

"A system administrator can configure a firewall to decline to any computer of the

outer (internal) network to access any protocol/service in the internal (outer) network except the Web service."

The ability to accept or drop incoming (and outgoing) network traffic based on the identities of computers and protocols is valuable and sometimes it can be used as a shield for more complicated attacks. For example a vast amount of incoming SYN requests (that is, incoming requests that seek establishment of connection) in a short time period, might be the sign of an upcoming Denial of Service (DoS) attack (typically directed to select IP addresses). The importance of the firewall technology is that it depicts the policies of an organisation into rules that apply in the network level. This is important, because the firewall rules do not depict the organisation's policies in security aspects only, but they compose the organisation's regulation as a whole. For example, an organisation may want to decline access to services operated by another organisation, in another country or of a specific content. A firewall is able to apply such policies at the network level.

As already stated, the basic information a firewall needs in order to operate is a list of rules which is composed by computer identities or protocol identities. Although a computer identity is a well defined property (a 32-bit integer, usually represented in a 4-dot notation, for example 192.168.0.1, called IP address), a protocol identity has evolved over the last 10 years. Approximately 10 years ago, a Port number was sufficient to denote a protocol or a service (for example Port 80 maps to the Web service and Port 25 to the incoming e-mail service). However, lately the Port number is not a sufficient protocol indicator, in most cases. This is mostly related to the trend of many modern protocols – especially peer-to-peer protocols focusing in media exchange among users – to tunnel their traffic through other popular protocols. For example, a lot of File Sharing applications or Instant Messaging applications can be tunnelled via Port 80, using the HTTP protocol.

For these reasons, in order to be able to identify correctly a protocol or a more sophisticated attack we need further inspection in the application payload of incoming and outgoing network packets.

4.2 Network Intrusion Detection and Prevention Systems (NIDS/NIPS)

NIDS is usually a passive monitoring system that receives copies of data packets and performs a signature-based or an anomaly-based detection of attacks. In signature-based detection, each packet is checked against the NIDS ruleset (set of signatures), thus leading to identification of already known attacks. Anomaly-based detection leads to intrusion alarm when data traffic behaviour falls outside "normality" (where normality is defined by a model built inside the system). NIDSs protect against DoS attacks, protocol violations, attacks carried out over a sequence of packets which require stateful inspection for their detection, infections by viruses and worms, etc. While NIDSs are passive systems, meaning that unlike Firewalls they are not located in the packet's path and do not directly stop attacks but raise alarms instead, modern NIPS combine NIDS and Firewall functionalities by being able to both detect and stop suspect traffic, thus providing multi-functional all-in-one security platforms.

4.3 Packet Structure and Decoding

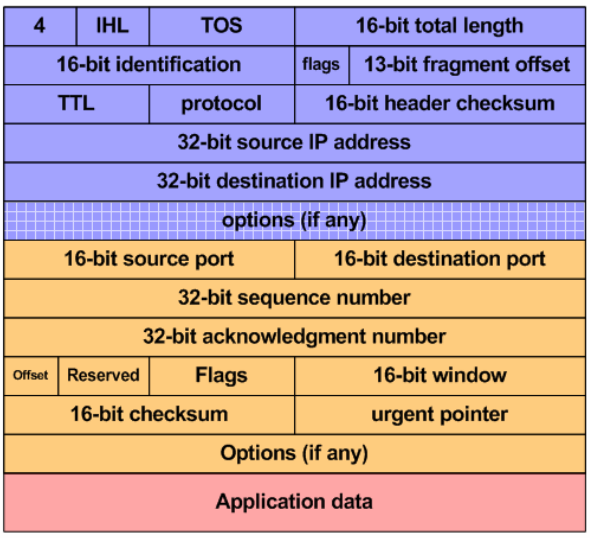

***Figure 1**-The IP protocol header along with TCP header. IP header is colored blue and TCP header orange.*

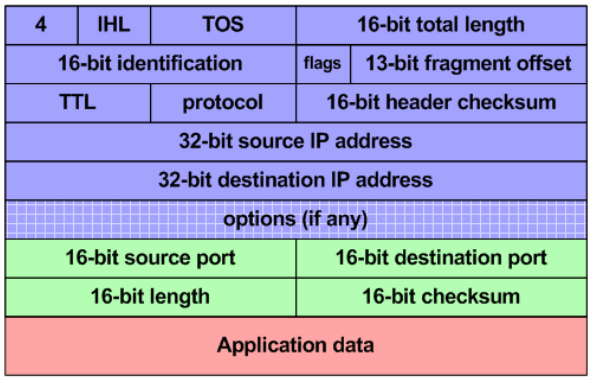

***Figure 2** - The IP protocol header along with UDP header. IP header is colored blue and UDP header green*

Packet headers have fixed length, typically of the order of a few hundred bits (Figures 1 and 2). However, payload length is variable and may be of the order of many thousand bits. Consequently, a decision was made within WP2 to focus on header inspection only during this project [1,2]. This decision is further justified by taking into account the fact (documented in D2.2) that, currently, while less than 10% of the rules of a typical NIDS ruleset require header inspection only, more than 90% of security alerts are triggered by header inspection only [2].

The position and the length of the fields that identify IPs, protocols, port numbers, etc, in the packet headers are well determined and fixed. Only the length of the options field (between IP and TCP or UDP headers, Figures 1 and 2, respectively) is variable, but it is previously specified in the appropriate field of the header.

4.4 Example Scenarios

This section presents some typical configurations of modern digital firewalls operating in the electrical domain. Thus, this section presents example scenarios of real world firewall deployments. More precisely we try to answer the following question:

What are the "top 10" firewall rules in modern firewalls?

This question does not have a trivial answer, since there are a lot of different types of firewall, depending on its use. For example, a firewall configuration used in a home PC is very different from the one used in the gateway of a company and significantly different from the one used in the gateway of an ISP. In order to address this issue, we searched for the default configurations of known firewalls in

the industry. These configurations cover a wide range of usage and are quite complex, since they do not employ header only rules.

4.4.1 Default Configurations of Industry Leading Firewalls

The default configuration of the Sophos Client Firewall can be found at:

http://www.sophos.com/support/knowledgebase/article/16608.html

The default configuration of the Sophos small business solutions Firewall can be found at:
http://www.sophos.com/support/knowledgebase/article/14464.html

The default configuration of the Norton Personal Firewall can be found at:

http://service1.symantec.com/SUPPORT/sunset-c2002kb.nsf/672c231f89ff479085256ee600556cc3/ade2747ea2a9741685256ede00518db1?OpenDocument

4.4.2 Basic Firewall Rules

The following is a list of some very common firewall rules used in modern firewalls. These firewalls could belong to organisations that run common services such as:

- Web Server
- E-mail Server (Incoming/Outgoing)
- DNS Server
- DHCP Server

#	Rule	Description
1	Deny all *incoming* traffic with IP addresses matching internal IP addresses (part of the network the firewall operates in).	Incoming traffic must not look like it originates from internal machines of the network. This is a sign for IP Spoofing: a packet embeds a fake IP address.
2	Deny all *incoming* **ICMP** traffic.	ICMP has been used for Denial of Service attacks (i.e. *ping of death*, etc.).
3	Deny *incoming* traffic (TCP/UDP 135, 137, 138 139/445).	These ports are related to RPC and Microsoft sharing and should be used only by internal computers of the network. These ports are used very often for attacks by malicious worms.
4	Allow *incoming* traffic (TCP 80/443) from any external address destined to the IP address of a hosted Web Server.	Port 80 is used for HTTP (Web Service) and port 443 is used for HTTPS (Secure equivalent).
5	Deny *incoming* traffic (TCP 25) originated from IP addresses that do not belong to the organisation (excluding cooperative e-mail servers).	Port 25 is used for e-mail delivery (SMTP). The e-mail server should be used only by computers that belong to the organisation, which hosts the mail server. Access should be allowed to cooperating e-mail servers, too.
6	Deny *incoming* traffic (TCP 110) originated from IP addresses that do not belong to the organisation.	Port 110 is used for incoming e-mail (POP3). The incoming e-mail server should be used only by computers that belong to the organisation, which hosts the mail server.
7	Deny *incoming* traffic (TCP 143/993) originated from IP addresses that do not belong to the organisation.	Port 143 is used for incoming e-mail (IMAP). Port 993 is used for SSL/TLS authentication with the IMAP server. The incoming e-mail server should be used only by computers that belong to the organisation, which hosts the mail server.
8	Deny *incoming* traffic from IP addresses that are black listed as open SPAM relay proxies.	These machines are used to send SPAM e-mails.
9	Allow *incoming* traffic (UDP 53) to the internal DNS server.	The DNS Server should be able to be contacted from any computer in order to resolve correctly IP addresses to domain names (and vice versa). It is safe to accept only UDP traffic and not both UDP and TCP, since the latter may increase sufficient its load.
10	Deny incoming traffic (UDP 67/68) originated from IP addresses that do not belong to the organisation.	Port 67/68 is used by the DHCP Server, which is responsible for the assignment of IP addresses. This service should be used exclusively by computers that belong to the organisation.
11	Deny incoming/outgoing traffic (TCP 6666/6667).	Port 6666/6667 is used by the IRC protocol. This service is heavily used by botnets.

Table 1 - *Example of configuration of a digital firewall.*

4.4.3 WISDOM oriented firewall

Based on the above, a “photonic” firewall for the WISDOM project may contain:

1) Rules to forbid incoming access to ports used commonly by modern worms: 445, 135 (Windows Sharing and RPC).
2) Rules to forbid outgoing/incoming access to ports 6666, 6667. These ports are related to the IRC protocol and they are heavily used from botnets.
3) Rules to block certain ICMP traffic, or all ICMP traffic (ICMP traffic is commonly blocked by large sites, by default).
4) Rules to deny access to black listed IP address (such as open relay mail servers used for the delivery of SPAM messages).

The first two rules require simple Port checking in the packet's header.
The third rule requires the identification of an ICMP packet (or certain properties of an ICMP packet).
The fourth rule requires IP checking in the packet's header.

The choice of rules to be implemented in the optical domain should be made by considering both their practical significance and the limitations in optics (Section 5). More rules included (and others not included) in Table 1 may be implemented in a “photonic” firewall, as we illustrate in Section 6 through some example application scenarios.

5. Capabilities and Restrictions in the Optical Domain

State-of-the-art in the optical domain and some basic optical circuits that are to be developed in this project have been described in D2.1 [1] and D4.1 [3]. The base line data rate will be 40 Gbps (25 ps bit period) and data format will be Return-to-Zero pulses. On chip delays should be able to provide a buffer memory capacity of at least 40 bits (below 1 ns latency) with the possibility to increase up to 200 bit capacity (5 ns latency). The main optical processing units are XOR/AND gates, consisting of Semiconductor Optical Amplifiers (SOAs). An optical pattern recognition system with a Mach-Zehnder Interferometer (MZI) consisting of SOAs has been introduced and has already been described in detail (D4.1). When n incoming bits are compared with N-bit target, the latency for the pattern matching operation is nN. The use of a recirculating buffer for the n-bit sequence within this scheme means that sequence length should be equal to the recirculating loop capacity (which is not readily variable). Optical switches are available to the project which can gate packets according to inspection, with sub-nanosecond switching times and reconfigurable within ns.

Even though the available optical components are superior to their electronic counterparts in terms of processing speed, they fall behind in terms of memory and scalability. Some basic limitations of the optical domain make the process of implementing a number of security applications (described in D2.2 and in Section 4 of this document) a hard process. More specifically, the following constraints that are imposed by the optical domain characteristics are critical for the deployment of security solutions.

5.1 Absence of state

One of the fundamental constraints in the optical domain is the absence of state, which is a consequence of the fact that having memory in the optical domain is not as trivial as it has become in the electrical domain. More precisely, the notion of *memory* in the optical domain is to use serial, time-of-flight storage so that the light pulses are delayed for a relatively short period of time. Optical memory capacity is thus relatively limited compared to electronic memory, with integrated optical devices being able to store a few hundreds of bits of information (all be it at high information data rates). The difficulty in having a flexible storage medium in the optical domain results in a system that lacks almost completely the notion of state. On the other hand, almost all systems and algorithms developed in the electrical domain reside to a great extent in having a controllable state. Therefore to develop security algorithms in the optical domain requires a new method of thinking and a learning time to understand how these algorithms may be implemented.

5.2 Configurability

The lack of memory in the optical domain also causes limited flexibility in the configuration of a device and the dynamic re-adjustment of its setup. Since there is no memory – in the sense of the term used in the electrical domain – to hold a system's state, it is hard to modify a system's behaviour by altering at run-time its properties. As far as the implementation of a security purpose device in the optical domain is concerned, the limited configurability reduces the flexibility in

implementing trivial features, enabled by default in all similar devices of the electrical domain, such as adding, deleting or modifying the *rule-set* of the device. However, increasing the future reconfigurability of the optical system is one aspect of scalability that will be examined in WP6, and with modelling in WP3.

6. Security Applications in the Optical Domain

This section is dedicated to specific security applications and systems that can be initially implemented – partially or fully – in the optical domain. We list some example application scenarios. Each example promotes some of the functionalities we deploy in the optical domain. These features can be combined for building more sophisticated systems and security solutions that operate in a hybrid environment of optical and electrical components later in the project.

6.1 Filtering out specific services

As we stated in Section 4, the basic operation of a standard firewall is to act as a filter, so as to distinguish wanted network traffic from the unwanted one. Having a similar filter in the optical domain is beneficial, since the all-optical filter will be able to cope better with high rates of network traffic, expected in current and future optical networks.

In order to deploy an all-optical filter we need to be able to perform a pattern matching algorithm so as to be able to inspect, and decide accordingly, based on the properties of a network packet. UCC has designed and implemented a pattern matching algorithm, which operates entirely in the optical domain (D4.1) [3].

Using the all-optical pattern matching algorithm, we can search the headers of incoming network packets for specific fields of the TCP/UDP headers. In order to filter out network traffic that targets a specific service, we need to check the destination Port number of an incoming network packet. The destination Port number of any incoming or outgoing network packet is located in a specific byte offset of the network packet. If the destination Port number matches the service we are looking for, we can then further process the network packet using standard technology in the electrical domain or drop the packet, according to the policy defined by the organisation that runs the system.

In Figure 3 we depict the basic architecture of the optical filtering component that matches a specific service. Specifically, we illustrate an example of filtering out network traffic that targets the e-mail service. The identity of the service, the Port number that is assigned commonly for outgoing e-mail delivery, is 25.

More specifically, in the pure optical domain, we apply a mask in the incoming network traffic and, using the optical pattern matching algorithm developed for the WISDOM project, we check if the packet's destination Port number is equal to 25. We also apply a delay in the incoming flow in order to perform the conditional operation. If the packet's destination is Port 25 we classify the network flow as an e-mail targeting network flow and we further process it in the digital domain. At this point, the processing can continue in the digital domain – as illustrated in the example – or be dropped in the optical domain.

In the example of Figure 3 we sketch in a simple way the proposed architecture, without placing any optical components. It is vital to note that the APPLY MASK box is implemented natively in the optical domain (as is the general box that hosts it).

Having as a basic building block the diagram depicted in Figure 3, we can construct devices that are able to filter out whatever network flow that targets a specific service by inspecting, in the optical domain, the destination Port number of an incoming packet, which is located in the header of the packet.

Furthermore, the idea can be extended to match targeted computer machines instead of targeted services. In Figure 4 we depict how this can be done using the same basic architecture, by substituting the service's identity (Port number) with a computer's identity (IP address).

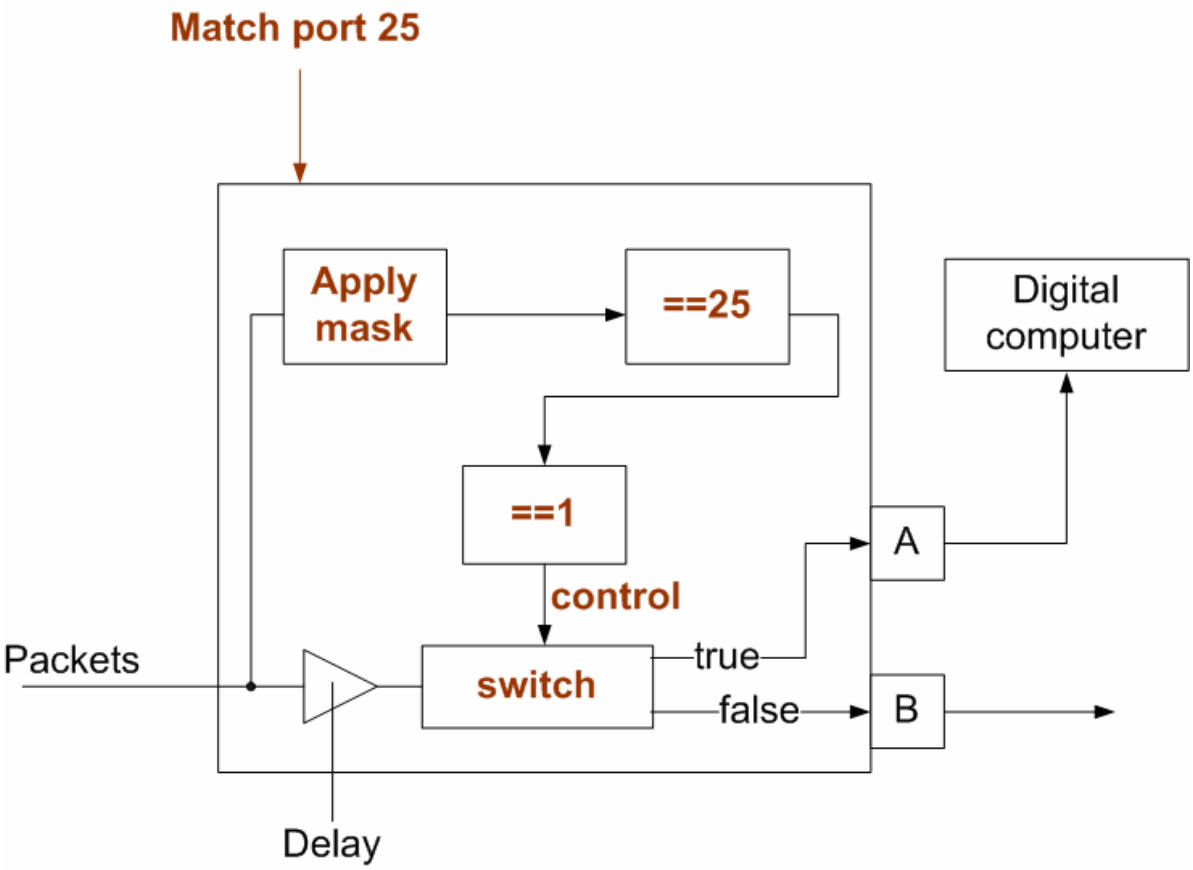

Figure 1 - *The fundamental WISDOM's architecture layout for matching a network packet that targets a specific service, based on the identity of the service (Port number). This example illustrates an application of filtering out e-mail traffic. The central box is entirely implemented in the optical domain, using state of the art all optical pattern matching algorithms (D4.1).*

Using the basic architecture's layout, which is inspecting specific fields in the headers of a network packet, we can capture network activities that target a specific service, target a specific computer, that originate from specific computers and so on. The complete set of capabilities is presented in Table 2.

Filtering Capability	Matched Identity	Application Example
Network packet targeting a specific service	Destination Port Number	Filtering out e-mail traffic
Network packet originating from a specific service	Source Port Number	Filtering out a Web server's response
Network packet targeting a specific computer	Destination IP Address	Preventing contact with a computer
Network packet originating from a specific computer	Source IP Address	Preventing access from a computer
Network packet with specific properties	IP protocol header field	Filtering out ICMP traffic

Table 2- *The filtering capabilities of the WISDOM architecture*

6.2 Filtering out SPAM e-mails

Having described the basic filtering mechanism based on service targeting, we are able to build more sophisticated and practical applications using a hybrid of optical and digital architecture. Since we are able to identify e-mail traffic in the optical domain, we can further proceed on identifying SPAM e-mails.

SPAM e-mails are considered as messages that are massively sent to Internet users from companies that would like to advertise their products [4]. Although, SPAM e-mail traffic is considered commonly as annoying and unwanted e-mail traffic, sometimes it may also become dangerous for a user. A significant fraction of the current e-mail traffic contains fraud messages, which try to delude a user to enter a malicious Web site, or reply to an e-mail with private information, such as a password or a credit card number. Thus, in this document we address the SPAM effect from its security perspective; we consider a SPAM filter as a security application.

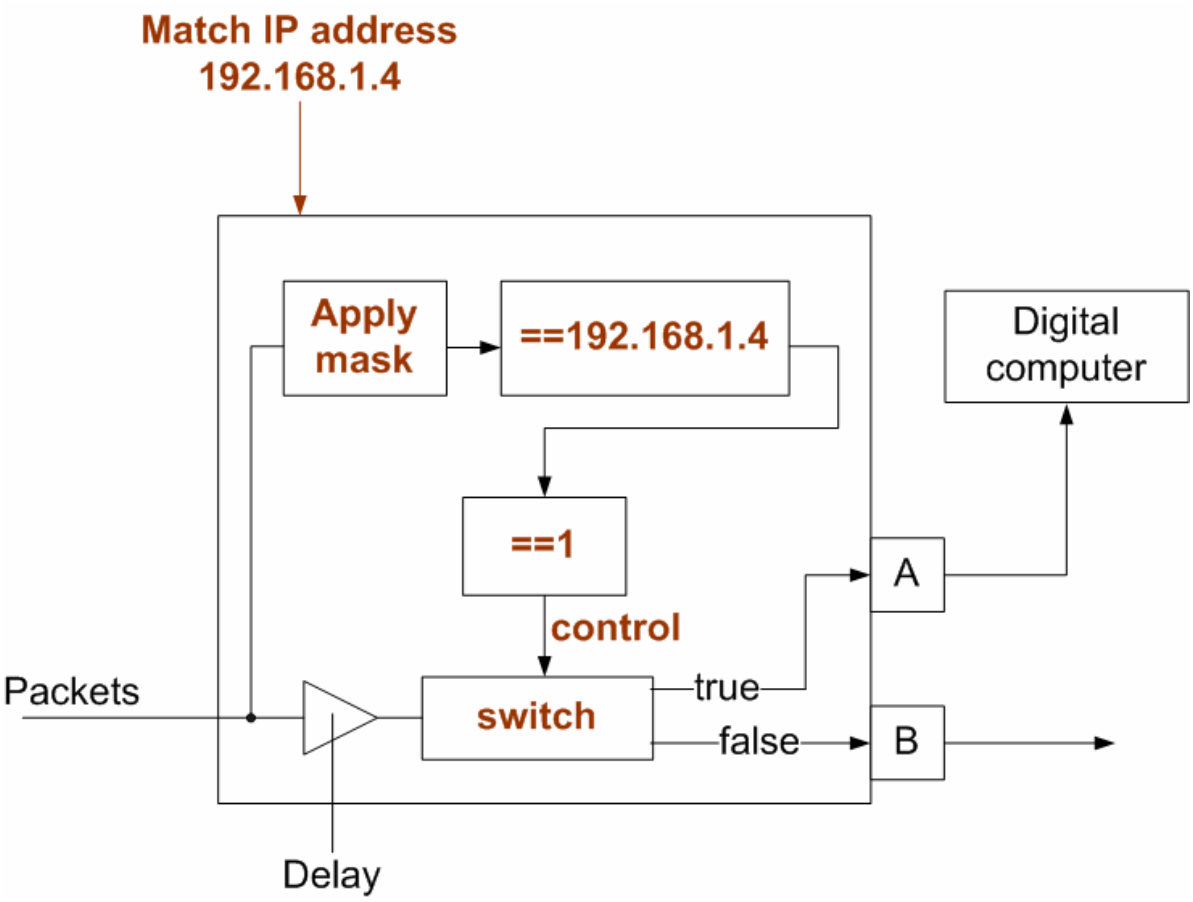

Figure 2 - *The fundamental WISDOM architecture layout for matching a network packet that targets a specific machine with IP address 192.168.1.4. This diagram extends the one depicted in Figure 3**,** by substituting the target service's identity (Port number) with a machine's identity (IP address).*

There are numerous ways to identify SPAM e-mails in the digital domain. Most commonly used are the Bayesian filters, which perform some machine learning algorithms to the body of an e-mail message and try to investigate if it is compliant to some popular heuristics that promote common properties of a SPAM e-mail. For example, words repeated in the body of an e-mail message or capitalisation of sentences that are frequently found in SPAM e-mails are checked out by the Bayesian filter in order to judge if an e-mail looks like a SPAM e-mail.

Performing such a sophisticated computation in the body of the e-mail message is out of the scope of WISDOM. Luckily, there is another technique to identify SPAM e-mail traffic, which has proved to be applicable in the digital domain [5]. Since SPAM e-mails are sent massively by e-mail servers that are operated by the SPAM senders, there are black lists that host the addresses of these e-mail servers. These servers are commonly called *Open Relaying Proxies*. They are e-mail servers which are configured to accept the delivery of an arbitrary e-mail message, whatever its sender, its recipient(s) and its payload. These servers *relay* whichever e-mail traffic they encounter.

The architecture depicted in Figure 3 can be extended in order to identify incoming SPAM e-mail traffic by performing some additional pattern matching in the headers

of an incoming network packet. Beyond checking that a network packet targets the e-mail service, we further apply a new mask in the source IP address field, which is mapped to the sender of the e-mail. If this IP address is found in a pre-configured in the device black list of open relaying proxies, the packet is classified as a packet belonging to a SPAM e-mail delivery.

6.3 Fighting with simple Denial of Service attacks

Denial of Service (DoS) attacks are considered as a major Internet threat [6]. There is a rich variety of DoS attacks nowadays. The basic feature of a DoS attack is that an attacker uses enough resources, or vulnerabilities, in order to exhaust the resources of a victim computer. Resources that DoS attacks target are memory, CPU and bandwidth resources. Although there are plenty of ways to perform a DoS attack, from simple massive generation of network traffic to quite elaborate misuse of another system as a DoS platform [7], we should be able to fight against some of them purely in the optical domain.

The DoS attack we will be able to detect in the optical domain is the well known SYN flood attack. An attacker sends a massive number of network packets that request a connection establishment to the victim. A connection establishment is related to the TCP family of the Internet Protocol (IP), which encapsulates the notion of state. In order for a host A to achieve a connection establishment with a host B, a three way handshake is required. Host A sends a SYN packet to host B, receives a SYN_ACK in advance from host B, and finally replies back to host B with an ACK packet. This mechanism can be exploited by an attacker in the following way. The attacker issues a series of SYN packets to the victim, but never completes the connection establishments. For each SYN packet received, the victim must allocate a new structure in order to prepare the upcoming connection. However the attacker is not going to complete the connection handshake. Actually, the attacker can enhance the attack by constructing SYN packets that embed as source IP address a random IP address, not the IP address of the attacker. Thus, the victim will reply to each SYN packet with a SYN_ACK targeting a random IP address. These SYN_ACK packets, also known as *backscatter traffic* [6], will reach hosts that never issued a SYN packet (hosts that may not even be on-line, since the attacker picks up IP addresses in a random fashion), so eventually they will be dropped. But, the victim must continuously allocate resources in order to serve the new incoming connections seeking establishment. Eventually, if the attacker achieves to send a massive collection of SYN packets, the victim will be overwhelmed and become out of resources.

The main idea of the TCP SYN flood attack is the following:

Send as many as possible TCP SYN packets to the victim in a short period of time.

In order to defeat the TCP SYN flood attack in the WISDOM project, we need to maintain a counter of the incoming TCP SYN packets. Using the developed pattern matching techniques, we may identify SYN packets natively in the optical domain by inspecting the SYN flag of a packet. A TCP SYN packet has the SYN flag on, which is a field of fixed location in the TCP headers of the packet.

The absence of state in the optical domain does not allow, in principle, maintaining the counter in the optical domain, so we need to check the counter in the digital domain. If the counter exceeds specific digitally configurable thresholds, then the optical firewall will be notified for an upcoming TCP SYN flood attack. Eventually, the optical firewall will start dropping incoming TCP SYN packets and protect the victim from the DoS attack.

An alternative optical detection of DoS attacks with the use of optical switches and photodiodes will be explored in the project. Detection of DoS attacks appears to be more challenging to implement in the optical than the rest of the security applications mentioned in this document.

6.4 Preventing access to specific subnets

In Section 6.1 we described how the WISDOM architecture may prevent access to specific services in the operational network or to specific computers. In Figure 4 we illustrate the basic outline of the WISDOM component that performs this operation.

It is a common tactic for modern organisations to accept or deny traffic from networks operated by other organisations. This kind of policy can be applied by configuring the central firewall of the organisation with subnets that are forbidden to communicate with. This firewall rule can be applied natively in the optical domain, by performing pattern matching in the incoming and outgoing network packets and specifically in the source IP and destination IP addresses located in the headers of the network packets. The sketch illustrated in Figure 4 can be extended by substituting the IP address with a subnet, that is, a range of IP addresses.

Thus, the WISDOM architecture can be deployed in an organisation's infrastructure in order to: (a) deny communication with the network of another organisation and (b) prohibit access in the internal network to computers of another organisation. In order to deploy (a) the WISDOM firewall must be configured to inspect the **destination IP address** of an **outgoing network packet**. If it matches the subnet that is occupied by the organisation with which interaction is prohibited, then the packet is dropped. The whole processing and the decision making are implemented natively in the optical domain, without employing any assistance of digital components.

In the same fashion, in order to deploy (b), the WISDOM firewall must be configured to monitor every **incoming network packet,** and perform optical pattern matching in the **source IP address** field. If it matches the subnet from which the organisation wishes to prevent access to the internal network, then the network packet is dropped in the optical domain.

6.5 Reducing ICMP traffic

The ICMP (Internet Control Message Protocol) protocol is one of the fundamental IP protocols, along with TCP and UDP. It is widely used in order to exchange messages between two hosts. Its most widely adapted usage is to check if a host is alive. By issuing a series of ICMP packets destined to a host A, host B can judge if host A is visible and reachable. If host A receives the ICMP packet, it will send a reply back to host B. By correlating the time that the original ICMP packet left host B and the time it took for the reply of host A to reach back host B, assumptions for the quality of the communication channel that interconnects A and B can be made. Although a similar action could be deployed using TCP, the light nature of ICMP is more suitable and designed specifically for such tasks. The best known tool that uses the ICMP protocol for tasks like the ones mentioned above is the ping [8] tool.

Beyond the healthy use of the ICMP protocol, an adversary may issue a vast amount of ICMP packets in order to saturate the communication channel of a victim. There is a significant amount of attacks totally based on the misuse of the ICMP protocol [9]. Thus, a common configuration of a modern digital firewall is to deny incoming ICMP packets from every computer in the Internet.

A network packet is classified as an ICMP packet if the protocol field in the IP header is set to the ICMP id. Note the *protocol* field in the IP header (blue fraction) in Figure 1 and Figure 2. Thus, using the basic architecture depicted in Figures 3 and 4, we may isolate ICMP traffic natively in the optical domain.

6.5.1 Real world ICMP based attacks

In this Section we present all the ICMP based rules used by a modern version of the Snort IDS system. These rules can all be captured by the WISDOM architecture natively in the optical domain. Note that all these rules rely in pattern matching in the headers of a network packet for specific properties of the ICMP protocol; there is no need for further examination of the packet's payload.

In
Table *3* we present rules specifically constructed for capturing attacks which take advantage of certain misuses of the ICMP protocol. In
Table *4* we list the keywords used by Snort in order to describe the attacks.

Rule	ICMP options
ICMP PING NMAP	dsize:0; itype:8
ICMP icmpenum v1.1.1	dsize:0;icmp_id:666;icmp_seq:0;id:666;itype:8
ICMP redirect host	icode:1; itype:5;
ICMP redirect net	icode:0; itype:5;
ICMP Source Quench	icode:0; itype:4;
ICMP Broadscan Smurf Scanner	dsize:4; icmp_id:0; icmp_seq:0; itype:8;
ICMP Destination Unreachable Communication Administratively Prohibited	icode:13; itype:3;
ICMP Destination Unreachable Communication with Destination Host is Administratively Prohibited	icode:10; itype:3;
ICMP Destination Unreachable Communication with Destination Network is Administratively Prohibited	icode:9; itype:3;
ICMP PATH MTU denial of service	itype:3; icode:4; byte_test:2,<,576,2;

Table 3 - *Rules for ICMP based attacks used by latest versions of the Snort IDS tool.*

Field Type	Explanation
dsize	The *dsize* keyword is used to find the length of the data part of a packet.
icmp_id	The *icmp_id* option is used to detect a particular ID used with ICMP packet.
icmp_seq	The *icmp_seq* option is used to detect a particular sequence used with ICMP packet.
itype	The *itype* keyword is used to detect attacks that use the type field in the ICMP packet header. In the ICMP packet header the type may take values from 0 to 16, indicating: Echo Reply, Destination Unreachable, Source Quench, Redirect, Echo Request, Time Exceed, Parameter Problem, Timestamp Request, Timestamp Reply, Information Request and Information Reply respectively.
icode	The *icode* option is used to detect a particular value of the code field in the ICMP header.
byte_size	This keyword is used to test a specific byte in the network stream for a particular value.

Table 4 - *A list of keywords used by Snort to describe ICMP specific rules. More details can be found in http://www.phptr.com/articles/article.asp?p=101171&seqNum=6&rl=1.*

7. Software Framework for Modelling Security Application

This Section presents the software platform for modelling optical components. It is crucial to have a software platform that can model in time constraints the behaviour of the optical components and perform a series of software simulations. We have developed *rwsim*, a modular software engine, capable of testing the operation of optical components over time.

7.1 Generic Overview

The framework, *rwsim,* has been developed in the Ruby scripting language. Ruby is an Object Oriented modern scripting language with high levels of dynamism. We choose Ruby for the framework's implementation, because of its lightweight syntax and its pure Object Oriented nature.

The key properties of the simulator are that it is highly modular and extensible. The user has the ability to insert modules or combination of modules and to inspect their operation over time. The core engine of the simulator has the ability to host various modules force them to interact, if necessary, and output the results of the various operations.

The time granularity of the engine is expressed in *ticks*. Each tick maps to the time needed to perform a fundamental operation. A fundamental operation is considered as the minimum operation that can not be further divided. For example, reading a bit from a bit stream is considered as a fundamental operation.

The component granularity can be varied. A user may construct complex modules or simpler ones. It is even possible to build a module that it is a combination of simpler ones as a whole, or build the simple modules and let the simulator to outcome their combination.

For example, it is feasible to construct a SOA-MZI component, or combine two SOA gates to inspect the behaviour of the SOA-MZI device.

7.2 Architecture

The core of the framework relies in the Object Oriented philosophy. In Figure 5 we depict the generic architecture of the framework. You may observe three different classes: *WSimulator()*, *WModule()* and *WNetworkFlow()*. These three classes compose the core engine of the whole framework.

The main class *WSimulator()* is responsible for the instrumentation of the whole framework's operation. It encapsulates all needed properties to have a simulator run. In order a *WSimulator()* instance to start running, it needs some information to process. For this operation there is a WNetworkFlow() class which simulates a source of information in terms of a bit stream. That is, a WNetworkFlow() must be attached to the WSimulator() instance in order to feed it with bits.

The WNetworkFlow() class simulates a source of information by encapsulating the operation of different media sources. For example it may feed the simulator with

bits originating from a packet trace stored in the hard disk or with actual bits arriving to a network interface attached in a computer. For initial testing purposes, the WNetworkFlow() embeds a pseudo random generator which feeds the main simulator with random bits.

After attaching a WNetworkFlow() instance to a WSimulator() instance various modules simulating optical components may be attached in the simulator. The simulator orchestrates the operation of each module by feeding it with information produced by the WNetworkFlow() instance and by maintaining accounting information for each module. After a successful run, the simulator outputs the result of each module, as well as the time, in terms of ticks, elapsed from the initial start of the simulator run. If it is desirable, the simulator may output partial information during the whole operation, so as to be feasible to track the progress of each module.

The core architecture of the rwsim is depicted in Figure 5.

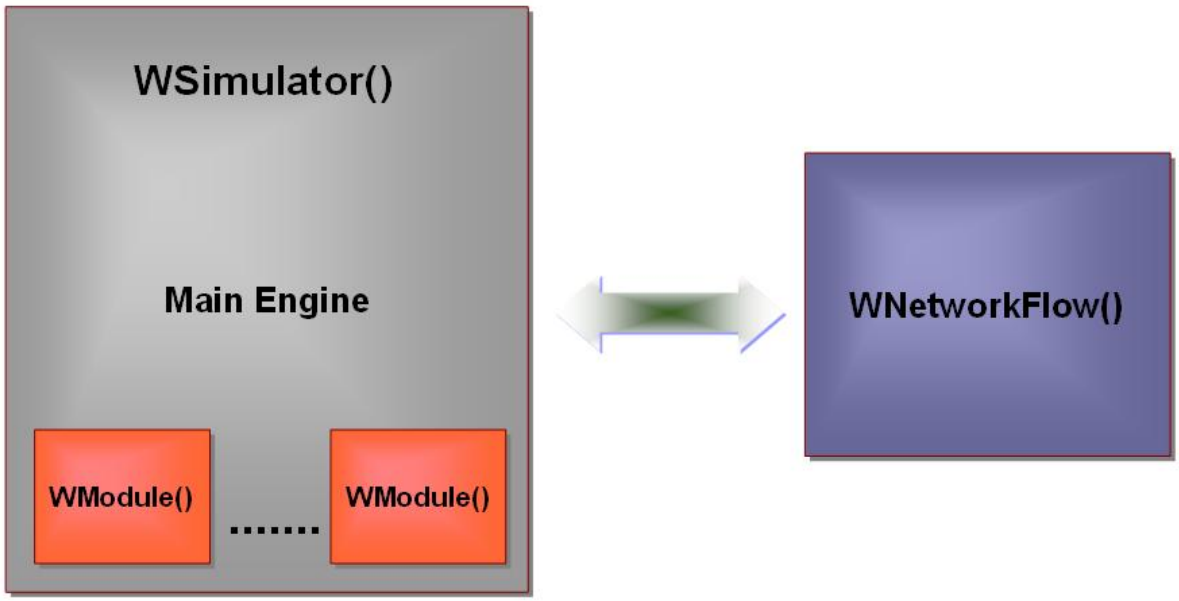

Figure 3 - *The core engine of the rwsim software platform.*

7.3 Implemented Modules

In this Section we present the implementation of modules for the rwsim platform, which simulate specific optical features.

The SOA-MZI optical construct is used by the optical pattern matching algorithm developed by UCC. We have developed a module that simulates the behaviour of the SOA-MZI component. In Figure 7 we depict the schematic operation of the SOA-MZI module.

For detailed information of the theoretical operation of a SOA-MZI component, refer to D4.1.

As you can see in Figure 7, the simulator (gray box) hosts a network flow (blue box)

and a SOA-MZI module (orange box). In a successful operation the network flow feeds the simulator with random bits, which in turn are fed to the module. In addition, the module has set up a C control pulse and a B pulse – which is the actual pattern we are looking for in the incoming bit stream – required by the SOA-MZI operation.

Upon a successful run, the module outputs the result Y to the simulator, which in turn is outputted to the user, along with accounting information in terms of time ticks.

In Figure 6 we present the code in the Ruby programming language, required in order to test the operation of the SOA-MZI module. The character '#' denotes comments. Observe that it is almost straightforward for a novice user to test the functionality of the module.

The code is separated in three distinct phases: (a) the initialisation of the simulator, (b) the initialization of the module and (c) the attachment of the module and launching of the simulator.

The network flow can be also configured by the user. In this example, the network flow is implicitly set up in the default configuration – an information source producing a stream of random bits - by the simulator instance.

```
Require 'rwsim'
require 'soa-mzi'

# Create the simulator.
sim = WSimulator.new()

# Create the SOA-MZI module and initialize it.
m = SOAMZIModule.new()
m.SetPatSize(4)
m.SetWinSize(15)
m.SetPattern(9)

# Attach module and run
sim.AttachModule(m)
sim.Run()
```

Figure 4 - *The code in the Ruby programming language required in order to test the SOA-MZI module.*

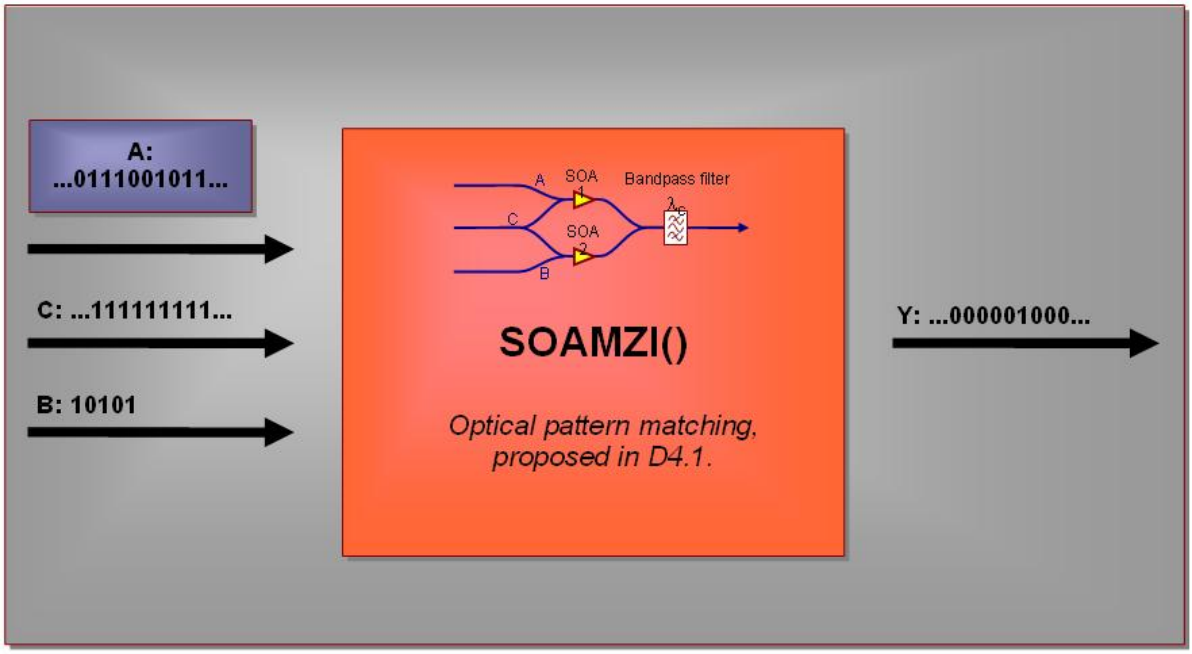

Figure 5 - *The schematic diagram of a typical operation of the SOA-MZI module.*

7.4 Examples of Usage

The distribution of rwsim comes ready to be tested without the requirement of technical knowledge and skills. The only requirement is that there is a recent version of the Ruby interpreter installed in the host, which will be used for running the rwsim simulator.

After unpacking the distribution to a directory, the following hierarchy appears:

- **rwsim directory**
  Contains the whole distribution
  - **core directory**
    Contains the source code of the simulator and modules
    - **test directory**
      Contains test files for testing the various modules

In order to test the SOA-MZI module, a user must simply issue the following command:

```
% ruby test/test-soa-mzi.rb
```

This command must be issued from the directory **`rwsim/core`**.

The output of a testing session of the SOA-MZI module is depicted in Figure 8.

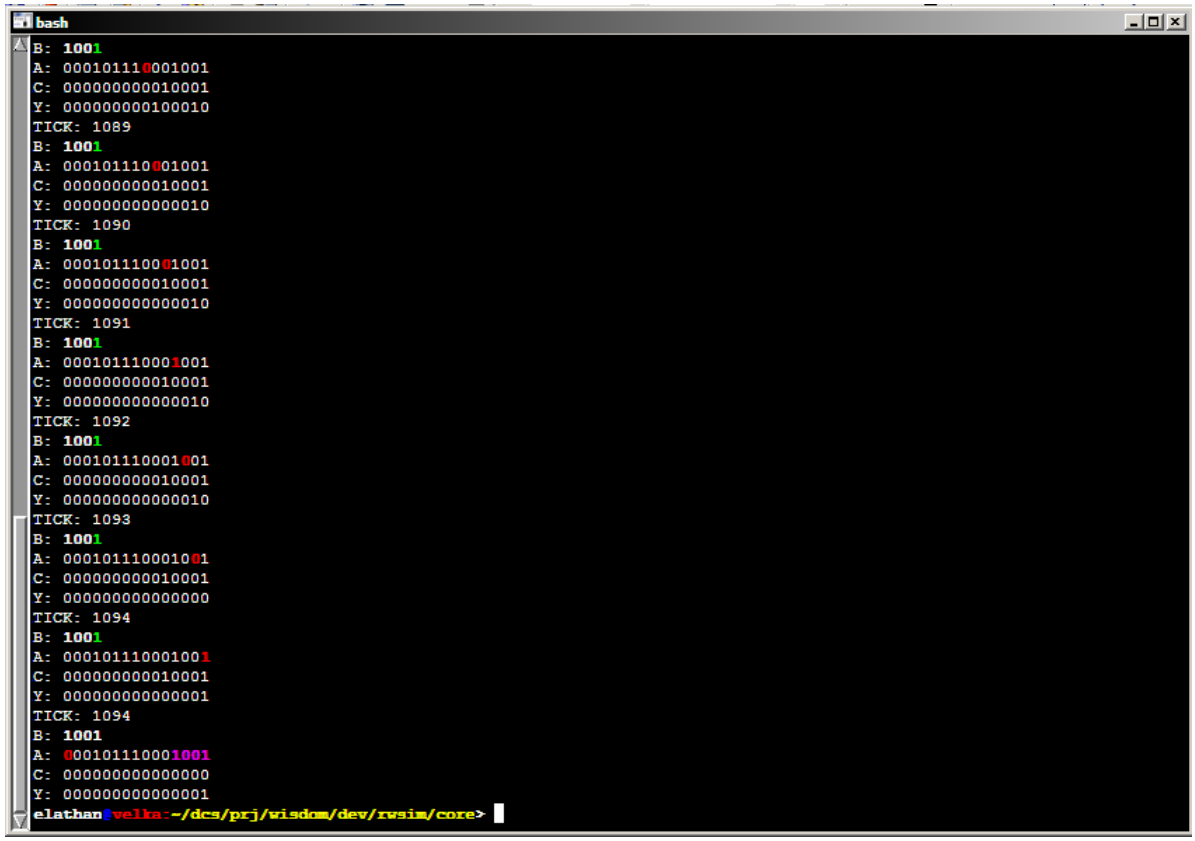

Figure 6 - *A simulator run, which tests the SOA-MZI module. Observe that in the 1094th tick the simulator has found an actual match of the searched pattern in the bit-stream. In each step of the simulator the bit in process is highlighted.*

8. Conclusions and Remarks

This document describes actual network security operations that can be efficiently and effectively implemented in the optical domain. After considering the restrictions imposed by optical hardware and basic security applications functionality, we conclude that the most appropriate applications to implement in the optical domain, at this stage of the WISDOM project, are basic firewall operations and DoS attacks prevention. We present signature-based real applications that rely on header inspection of data packets in the optical domain using the optical pattern matching scheme introduced within this project. Our capability to optically process specific fields contained in the packet headers, opens up many possibilities for performing critical “photonic” firewall operations. We show how to optically control network traffic for specific services (by inspecting port numbers), specific computers (by inspecting IP addresses), and specific protocols (by inspecting IP protocol fields). In addition, we describe how it is possible to perform such inspections combined, in case of important applications which require this approach. Some applications outlined in this report require additional conventional processing in the electronic domain, but the bulk of data handling is in the optical and the interface between optics and electronics is simple. Even though stateful inspection cannot be performed in the optical domain, optical anomaly detection appears to be practical and possibly beneficial in some cases, such as DoS attacks detection. Table 5 summarises the main network security applications described in this report which can be implemented in the optical domain. We also describe a simulation tool developed for the project which is based on functional models of optical devices and which is useful for optical algorithm design and validation.

It is worth pointing out that in the context of WP3 and WP4, we further explore possibilities for the design of more sophisticated optical security algorithms that use both optical and electric domain to perform efficient operations. Many issues that arise during these efforts are also related to scalability and optimum balance between optical and electronic processing (WP6). The use of tree-like structures and hash functions in the optical domain seem to be promising directions for the immediate future, once applications outlined in this report are implemented and demonstrated to perform as expected. Other considerations, such as parallel/distributed use of optical devices, more complex algorithmic components (Bloom filters, heuristics, etc) are being made in order to further increase the processing power for header inspection and possibly extend it to packets payload inspection. Combining pre-processing at low-level optics and algorithms at high-level electronics is the subject of a parallel line of work that will be described in D3.2.

Security Operation	Inspection	Application Example
Match network packet targeting a specific service	Destination Port Number	Filtering out e-mail traffic
Match network packet originating from a specific service	Source Port Number	Filtering out a Web server's response
Match network packet targeting specific computer(s)	Destination IP Address	Preventing contact with a computer
Match network packet originating from specific computer(s)	Source IP Address	Preventing access from a computer
Match network packet with specific properties	IP protocol header field	Filtering out ICMP traffic
Match network packet targeting a specific service and originating from specific computers	Destination Port Number and Source IP Address	SPAM filter
Denial of Service attack detection	SYN flag	Preventing TCP SYN flood attacks

Table 5 – *Security applications for optical implementation*

9. References

1. Project WISDOM, Deliverable 2.1, “Optical Firewall Architecture with Target Specifications for the Optical Subsystems”.
2. Project WISDOM, Deliverable 2.2, “Boundary Condition Algorithm Design”.
3. Project WISDOM, Deliverable 4.1, “Optical Sub-System Boundary Conditions”.
4. CHANNEL MINDS. Spam Traffic Risen 40% Since November Says Email Systems, Feb 2005. http://www.channelminds.com/article.php3?id_article=2464.
5. Andreolini, M., Bulgarelli, A., Colajanni, M., and Mazzoni, F. 2005. HoneySpam: honeypots fighting spam at the source. In Proceedings of the Steps To Reducing Unwanted Traffic on the internet on Steps To Reducing Unwanted Traffic on the internet Workshop (Cambridge, MA). USENIX Association, Berkeley, CA, 11-11.
6. David Moore, Geoffrey M. Voelker, and Stefan Savage, "Inferring Internet Denial-of-Service Activity," Usenix Security Symposium, 2001. http://citeseer.ist.psu.edu/moore01inferring.html.
7. Elias Athanasopoulos, Kostas G. Anagnostakis and Evangelos P. Markatos. Misusing Unstructured P2P Systems to Perform DoS Attacks: The Network that Never Forgets. In Proceedings of the 4th International Conference on Applied Cryptography and Network Security (ACNS'06). June 2006, Singapore.
8. The Story of the Ping Program. Mike Muuss. http://ftp.arl.mil/~mike/ping.html
9. RFC 792 (rfc792) - Internet Control Message Protocol. http://www.faqs.org/rfcs/rfc792.html

10. Appendix: Acronyms

Acronym	Expansion	Description
NIDS	Network Intrusion Detection System	A software program that passively inspects incoming network traffic for a possible attack.
NIPS	Network Intrusion Prevention System	A software program that passively inspects incoming network traffic for a possible attack.
IP	Internet Protocol	The core protocol family of the Internet.
TCP	Transport Control Protocol	TCP is built over IP and provides reliable communication between two Internet hosts.
UDP	User (or Universal) Datagram Protocol	UDP is built over IP and provides unreliable communication between two Internet hosts.
ICMP	Internet Control Message Protocol	ICMP is built over IP and provides a way for two Internet hosts to exchange control messages.
HTTP	HyperText Transfer Protocol	HTTP is the protocol, which is used for the communication of a Web browser and a Web server.
SMTP	Simple Mail Transfer Protocol	SMTP is the protocol, which is used for the delivery of outgoing e-mail.
POP3	Post Office Protocol version 3	POP3 is the protocol, which is used for receiving incoming e-mail.
IMAP	Internet Message Access Protocol	IMAP is similar to POP3, but it is able to store e-mail in the remote e-mail server and not in the user's host.
DNS	Domain Name System	The distributed architecture, used to map IP addresses in user friendly names, like www.google.com.
RPC	Remote Procedure Call	A programming model that supports remote execution of code.
DoS	Denial of Service	An attack that aims on the exhaustion of a host's resources (RAM, CPU, Network Bandwidth).
SPAM	Not an acronym	It characterizes unsolicited e-mail.

Table 6 – *Table of acronyms.*