THE STATE COMMITTEE OF COMMUNICATION, INFORMATIZATION AND TELECOMMUNICATION TECHNOLOGIES OF THE REPUBLIC OF UZBEKISTAN
TASHKENT UNIVERSITY INFORMATION TECHNOLOGIES

The right of manuscript

ALIEV HAYOTJON UZAKOVICH

DEVELOPMENT OF SYSTEM SECURITY FOR IP-BASED PACKET NETWORKS BY USING DPI-TECHNOLOGY

Specialty: 5A311301
«Data transmission equipments and systems»

DISSERTATION

Master's thesis

Scientific adviser:

Assoc

iated

PhD. Abdurakhmanov R.P.

TASHKENT 2014

THE CONTENT

Introduction

The development and expansion of sphere of services plays an important role in increasing the number of jobs. The communication, information, financial, banking and transport services and those on auto and household electronic goods repairing developed with highest rates. It is especially worth noting the dynamic development of services in the sphere of information and communication technologies, which for over the past four years have been increasing annually on average by 50 percent [1].

Wireless and fixed network broadband service providers face the challenge of increasing return on investment in the face of increasing infrastructure costs to keep up with unprecedented data demand by subscribers while opportunities for increasing revenues remain elusive. Service providers are faced with unprecedented demand for more and higher speed bandwidth in the face of new applications, increased video use, and subscriber growth. Network convergence where all services will be provided over IP on the same technology—whether as LTE in wireless or over a fiber/coaxial/copper connection in fixed—forces the question of “fair use” and prioritization. Voice, video on demand, live video, network DVR, multimedia messaging, videoconferencing, browsing, and later medical, home monitoring, energy use, smart grid, and other technologies both as a bundled offering with a service provider or their partners as well as over the top all share the same infrastructure in IP services to a subscriber. Considering IP is the glue for all communications whether for entertainment, education, employment and work,

payment, and a new round of machine to machine applications to support health, convenience, home security, metering, and efficient energy use, service providers have a unique position to provide the environment—one that ensures good performance and security—that enables this convergence and innovation to all IP. Service providers need an awareness of the applications that use IP in order to provide the key roles of security and fairness and DPI is the cornerstone in enabling those through informed decisions. Key motivations for DPI can be categorized as follows:

Cost Reduction: Through increased knowledge of capacity, performance bottlenecks, and as well through traffic management more subscribers can be served from the same infrastructure. Subscriber retention increases through better management of the network combined with proactive engineering to add capacity to address performance concerns.

Increased Revenues: Through the offerings of tiered services to both subscribers and business partners. New services such as content filtering to limit access to content that is appropriate for a child or employee, network security services to detect and/or prevent malware, crimeware, and sensitive information loss, and services to business partners for increased ad revenues and/or Quality-of-Service guarantees to deliver traffic.

Emerging Regulatory Compliance: Service providers are increasingly faced with lawfully required surveillance and/or assistance for law enforcement and/or national security/defense, requirements to offer content filtering, requirements to report access to unlawful content, and similar, such as interests in detecting access and transfer of pirated content.

The proliferation of Internet and networking applications, coupled with the wide-spread availability of system hacks and viruses, urges the need for network security. The security of digital information systems has an increasing impact on modern societies and economies. Information is most valuable when (safely) circulated and hence, network security is a critical issue with great

financial impact and significant effect on society. Private industries in finance, trade, services, transportation, manufacturing, and public sectors such as medical, vital services, national economy, defense and intelligence depend on computing systems. Consequently, any information and network security failure of these systems may often result in significant economic damage or disasters. Recent analyses show the economic impact of network security.

Today's solutions for network security require deep inspection of packet payload for supporting clear communication. However, performing in-network packet inspection is not easy. To achieve this, we need *deep packet inspection techniques* that scan packet payloads to analyze the meaning and purpose of the network traffic to distinguish malicious packets from normal packets. These inspection techniques are not only useful for detecting and filtering packets containing worm signatures but are also required by other newly emerging edge network services such as high speed firewalls (for protecting end hosts from security attacks), HyperText Transfer Protocol (HTTP) load balancing (smartly redirecting packets to different servers based on their HTTP requests.

As it is told in I.A.Karimov's book «Search of the big and small projects in technical and technological updating of manufacture for maintenance of competitiveness of production, and also means and sources for this purpose should become first of all the major business and a duty of the head and the technical personnel of each enterprise».

That's why according to our President words we should try to improve our environment with all possible methods and technologies that lead us to e-government. So we must give great opportunities to people for their new generation safe communications with new and powerful solutions.

Topicality of the subject of the Master's thesis. Today modern Internet and communication networks are based on the IP-based packet switched technologies that allow sending different types of information like voice, data, video and others. Within increasing number of Internet users, different

problems are going up on the networks. The main problems on the communication networks are quality of service, network security and traffic management on the today's point.

The decision of these problems is connected with deep packet inspection technology, which analyses data packets in "depth". Also, security of communication systems defines with DPI. Using DPI aplications allow security services, like control, blocking unwanted accesses, removing malicious packets, distinguish viruses and other services for avoiding from threads to service providers and network users. In detail, network providers can protect and control their network with DPI.

Developing and implementing DPI security solutions in Uzbekistan improve the national infrastructure of information and communication society. After realizing Electron-government in Uzbekistan DPI takes main role in control and management information resources and supports reliable and secure communication.

Because of reasons defined above now DPI systems are actual issue in the global telecommunication market. Researchers are working to development new methods to improve the performance of deep packet inspection applications.

Therefore, this work is devoted to development of security solution for IP-based packet networks by using DPI.

The main objective of the Master's Thesis is development of security solution for IP-based packet networks by using DPI. Consistent with this objective *the following tasks* have been put:

- analysis of DPI solutions for networks security in ip-based communication networks;
- analyses of deep packet inspection methods and algorithms;
- developing algorithm and program for DPI security system in different operation systems.

– explaining experimental analyses taken from real network with DPI program

The subject of research is developing DPI security system and analyzing taken characteristics from networks.

The object of the research is analyzing packets of IP-based multiservice networks such as local area network, access network. For example, input and output packets in local area network are inspected and filtered with DPI application.

Methodological basis of research is developing algorithm and programming packet inspection processes, mathematical modeling and experimental analysis.

The scientific novelty is developed algorithm and program using DPI for Linux and Windows machines for local using. Also, in work are explained experimental results. In addition, multicore DPI platforms are reviewed and analyzed packet inspection processes in one and 4- core machines.

Practical value and introduction of the **research** ***results.*** The content of the present work is focused on analyzing and developing Deep Packet Inspection systems and applications. Results of work, DPI applications can be offered to internet providers and corporative networks and modified on demand.

Dissertation structure. The dissertation consists of the introduction, four chapters, and conclusions to every chapter and used literature list with final common conclusion.

The first chapter - “Overview of DPI solutions for networks security in ip-based communication networks” consists of theoretical aspects IP-packet networks and analysis of DPI solutions for packet networks security in global telecommunication market.

The second part - “Packet inspection, filtiration analyses and firewall systems” defines packet inspection, filtration, and processes in IP networks.

Therefore, modern and traditional firewall systems of IP-based packet networks are explained.

In the third part - “Analyses of Deep Packet Inspection methods and algorithms” are given. Main algorithms packet inspection and multicore architecture of deep packet inspection are illustrated and experimented.

The fourth part - “Development of security solution for IP-based packet networks by using DPI” defines ITU-T recommendations for developing DPI system in first. Therefore, developing DPI filtering applications on the Linux and Windows machines are explained and program results and source code of program are given.

Chapter I. Overview of DPI solutions for networks security in IP-based communication networks

1. IP based communication basics and IP network architectures

Modern digital technology allows different sectors, e.g. telecom, data, radio and television, to be merged together. This occurrence, commonly known as convergence, is happening on a global scale and is drastically changing the way in which both people and devices communicate. At the center of this process, forming the backbone and making convergence possible, are IP-based networks.

Services and integrated consumer devices for purposes such as telephony, entertainment, security or personal computing are constantly being developed, designed and converged towards a communication standard that is independent from the underlying physical connection. The cable network, for instance, first designed for transmitting television to the consumer, can now also be utilized for sending e-mail, surfing the Web or even monitoring a network camera sending live pictures from another continent. Furthermore, these features are

also available over other physical networks, e.g. telephone, mobile phone, satellite and computer networks.

The Internet has become the most powerful factor guiding the ongoing convergence process. This is mainly due to the fact that the Internet protocol suite has become a shared standard used with almost any service. The Internet protocol suite consists primarily of the Internet Protocol (IP) and the Transport Control Protocol (TCP); consequently, the term TCP/IP commonly refers to the whole protocol family.

IP-based networks are of great importance in today's information society. At first glance, this technology might appear a bit confusing and overwhelming. Therefore, we'll start by presenting the underlying network components upon which this technology is built.

A network is comprised of two fundamental parts, the nodes and the links. A node is some type of network device, such as a computer. Nodes are able to communicate with other nodes through links, like cables. There are basically two different network techniques for establishing communication between nodes on a network: the circuit-switched network and the packet-switched network techniques. The former is used in a traditional telephone system, while the latter is used in IP-based networks.

IP-based networks on the other hand utilize a packet-switched network technology, which uses available capacity much more efficiently and minimizes the risk of possible problems, such as a disconnection. Messages sent over a packet-switched network arefirst divided into packets containing the destination address. Then, each packet is sent over the network with every intermediate node and router in the network determining where the packet goes next. A packet does not need to be routed over the same links as previous related packets. Thus, packets sent between two network devices can be transmitted over different routes in the event of a link breakdown or node malfunction (Figure 1.1).

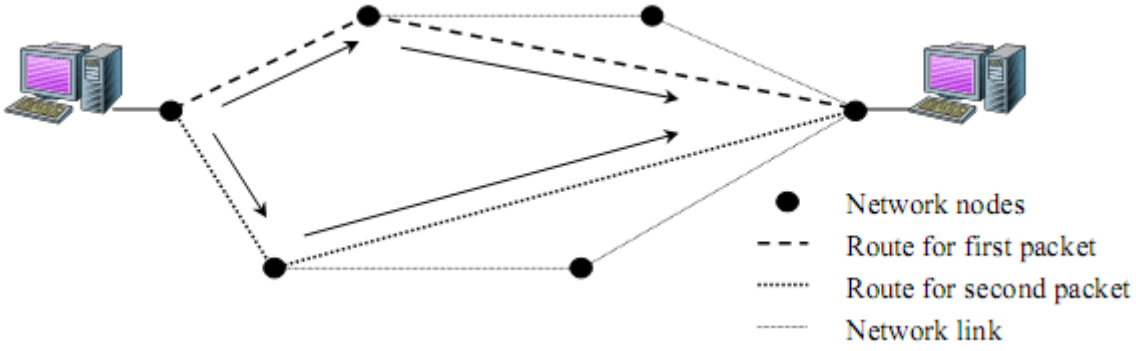

Figure 1.1. A packet-switched network routes each packet independently

IP-based network solutions are both flexible and economical substitutes for solutions that utilize old network technologies. The diverse properties between these technologies result from how information is represented, transmitted and managed. Information is simply structured collections of data, and thus takes its meaning from the interpretation we give it. There are two fundamental types of data, analog and digital, and both possess different behaviors and characteristics.

The Internet protocol suite is a layered protocol family where each layer builds upon the layer below it, adding new functionality. The lowest layer is concerned purely with sending and receiving data utilizing the transmission layer. At the top are protocols designed for specific tasks, such as sending and receiving motion pictures, sound and control information. The protocols in between handle things such as dividing the message data into packets and forwarding them reliably between network devices.

The Internet Protocol (IP) is the basis of the Internet protocol suite and is the single most popular network protocol in the world. IP enables data to be transmitted across and between local area networks, hence the name: Internet Protocol. Data travels over an IP-based network in the form of IP packets (data

units). Each IP packet includes both a header and the message data itself, where the header specifies the source, the destination, and other information about the data.

IP is a connectionless protocol where each packet is treated as a separate entity, like a postal service. Any mechanisms for ensuring that sent data arrives in a correct and intact manner are provided by higher-layer protocols in the suite.

Each network device has at least one IP address that uniquely identifies it from all other devices on the network. In this manner, intermediate nodes can correctly guide a sent packet from the source to the destination.

The Transport Control Protocol (TCP) is the most common protocol for assuring that an IP packet arrives in a correct and intact manner. TCP provides reliable transmission of data for upper layer applications and services in an IP environment. TCP offers reliability in the form of a connection-oriented, end-to-end packet delivery through an interconnected network.

The Internet Protocol suite provides an adaptation to the transmission layer protocols and offers a standardized architecture for communication over an interconnected collection of LANs, i.e. a WAN. This is a tremendous advance, mainly because we're able to connect and communicate over different physical connections in a standardized way. With IP as the basis, the Internet Protocol suite provides the third building block for successful digital communications, the IP layer (Figure 1.2).

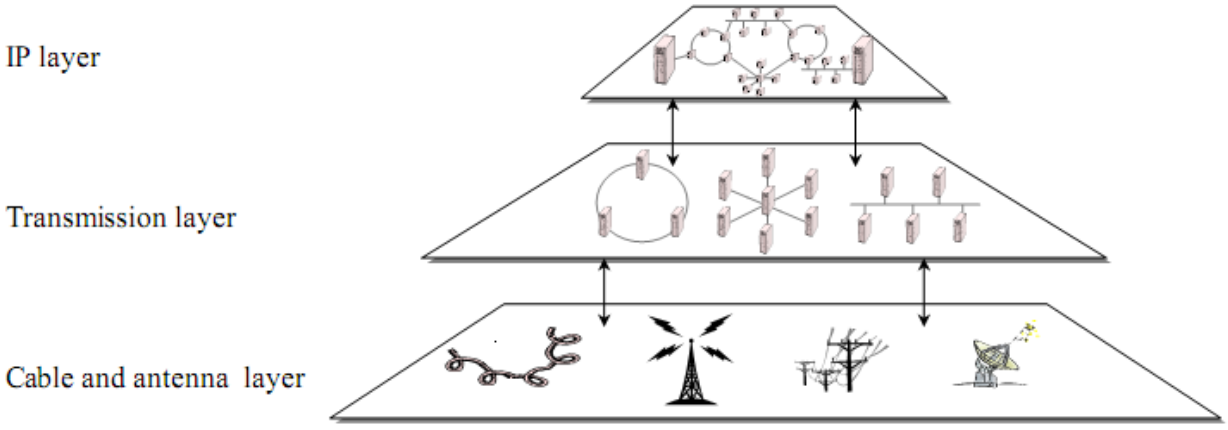

Figure 1.2. IP layer − the third building block

The Internet Protocol suite brings together all transmission layer protocols into a single, standardized protocol architecture, which can be utilized by applications for different communication purposes. As a direct result, any application that supports TCP/IP will also be able to communicate over any IP-based network.

It should be easy to see that this standardized architecture has revolutionized network communication. An ever-increasing number of applications that transfer text, sound,live pictures and more utilize IP-based architecture. All these applications and application protocols constitute the application layer and provide the fourth, and final, building block for successful digital communications (Figure 1.3).

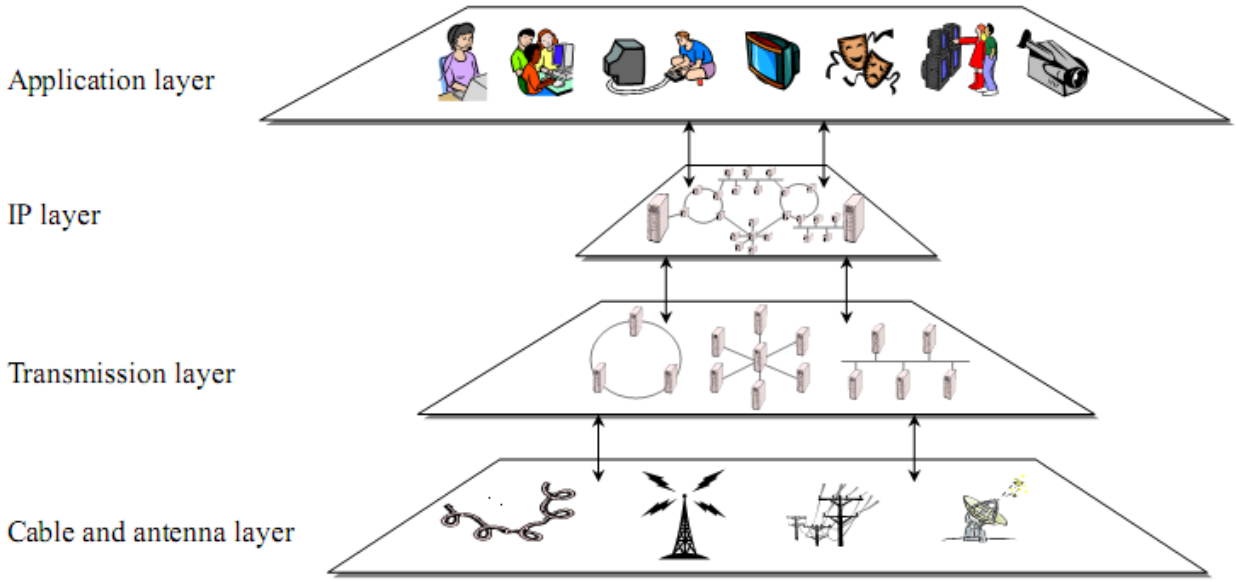

Figure 1.3. IP-based network architecture

Modern IP networks allow for convergence where different services, and combinations of these services, can be provided through infrastructures that formerly accommodated only one type of service. There are three major factors that create the conditions for convergence: digital technology, transmission technology and standardized communication protocols. Digital technology allows all information—text, sound and motion pictures, for example—to be represented as bits and transmitted as sequences of ones and zeros. Transmission technology enables better utilization of available capacity in different infrastructures. Consequently, services that require high capacity can be provided by infrastructures previously able to deliver only simpler services.

We have already seen how IP-based technology provides an excellent architecture for the process of ongoing convergence. At the heart of the Internet

Protocol suite is the Internet Protocol, which represents the building block that uniformly connects different physical networks with a variety of applications. In addition, presently available IP-based solutions can be fully integrated with other available systems[2],[3], [4].

Also we must emphasis that data security on the IP networks is important issue for today's point.

The security techniques are combined to provide security services in the Internet at the network layer, transport layer, and application layer. But now it is not efficient for security of network users. So in the next part are reviewed packet and data content inspection trends that fully describe to support security, management and QoS on the packet switched networks.

2. Review and analysis of DPI solutions for packet networks security

The usage of the Internet changed dramatically in the last few years. The Internet now transports traffic generated by many different users and applications including financial transactions, e-business, entertainment and more, which is definitely different from the traffic we had 30 years ago when the network was engineered for email, telnet and FTP. Often, the only way to keep the pace of the new traffic trends is to have an effective infrastructure for real-time measurements in the network, which allow discovering changes in the traffic patterns as soon as they appear and adapt to them quickly.

The capability to recognize and inspect which application generated the traffic is perhaps one of the most important challenges in network measurements. Several technologies have been proposed so far. First was port-based traffic classification, which is now considered extremely imprecise due to the large amount of applications using non-standard ports in order to escape network limitations (e.g. bandwidth shaping enforced by network providers). Next step was Deep Packet Inspection (DPI), which analyzed also

data in the application-layer payload and is usually extremely effective. On the downside, this technology is considered extremely demanding in terms of processing requirements and memory occupancy. Recently, newer techniques have been proposed that base on statistical methods, whose main advantage is the capability to detect also encrypted and tunneled traffic, but whose precision is acceptable only if we target a few protocols (often less than a dozen) out of the hundreds available on nowadays networks. Moreover, their processing (and memory) requirements may be comparable with packet-based DPI.

Due to the experimental nature of the statistical techniques and their current limitations, the mostly used technology for traffic inspection is still DPI. Interesting, the biggest criticism to this technique is not related to its difficulties in classifying encrypted or tunneled traffic, but to its supposed high processing cost. In fact, DPI is extensively used in security applications such as Intrusion Detection Systems (IDS) and firewalls, which have strict requirements in terms of precision. In other words, a single misclassification and not inspection in such these applications could let an attacker to compromise even an entire network, and is therefore a risk that people do not want to run. In order to minimize the risk of misclassifications, all these DPI implementations tend to privilege the accuracy, without taking the processing cost into much consideration.

However our scenario is different because we focus on applications that use traffic classification and inspection mainly for statistics, quality of service (QoS), and monitoring, and can accept some compromises in terms of accuracy. For instance, misclassified and not inspected traffic may update the byte counter of the wrong protocol but this does not represent a threat such as a misclassification into an IDS. In this environment, the DPI implementation can optimize, for example by getting rid of expensive algorithms (e.g. the ones that reconstruct the entire stream at the application level) that are a must if the

precision is a major concern, but that contribute substantially to the processing costs and memory requirements [5].

The past few years have seen a radical evolution in the nature and requirements of network security. There are many factors contributing to these changes, the most important of which is the shift in focus from so-called 'network-level' threats, such as connection oriented intrusions and Denial of Service (DoS) attacks, to dynamic, content-based threats such as Viruses, Worms, Trojans, Spyware and Phishing that can spread quickly and indiscriminately, and require sophisticated levels of intelligence to detect [6].

To protect against modern network threats, there are essentially two deployment architectures that are available to the telecommunication engineers:

1. Deploy a next-generation DPI firewall that performs traditional SPI firewall functionality, as well as DPI application security, or
2. Deploy a DPI Content Security appliance that sits behind an existing SPI firewall

Both of these approaches are illustrated in the Figure 1.4 below. In the latter example, the Content Security appliance is typically configured in transparent mode, where the device sits 'invisibly' between the firewall and the switch such that subnets do not have to be re-mapped. The device examines all traffic in a 'promiscuous' mode, where it makes forward/drop/log/quarantine decisions based on what services are activated (e.g. Anti-Virus, Intrusion Prevention, Anti-Spam, Anti-Spyware, URL Filtering and Spam Filtering) [6].

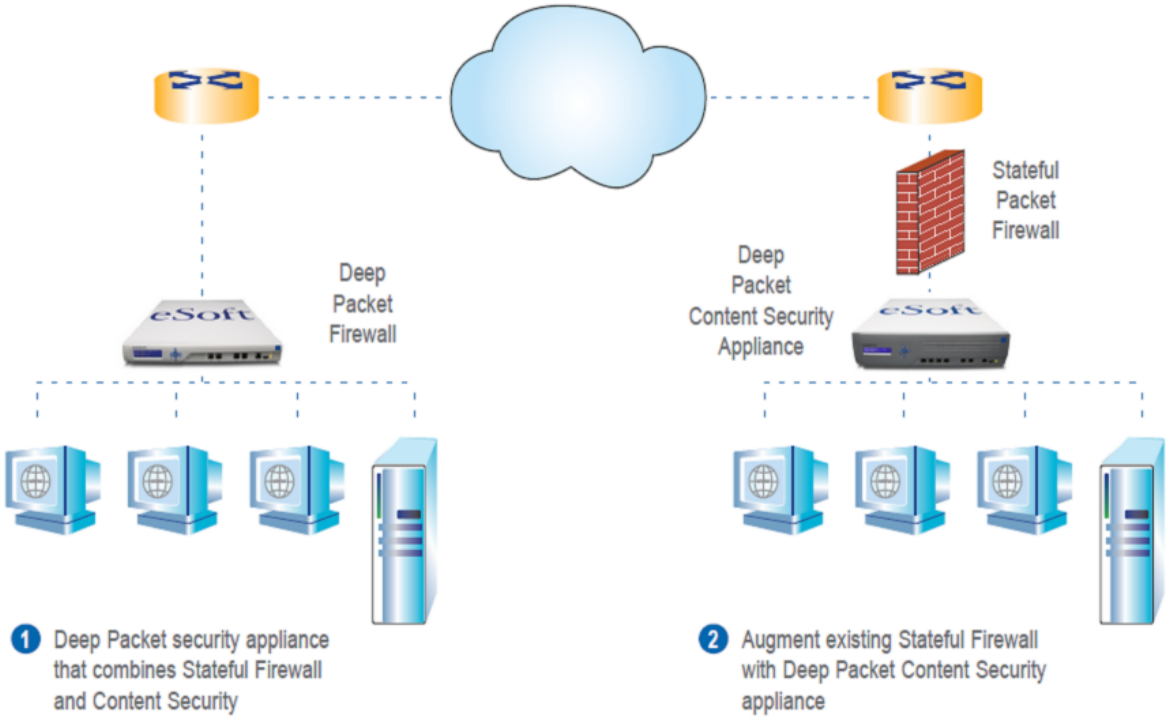

Figure 1.4. Modern alternatives for Deep Packet security

Recently, mobile traffic has increased due to the advance of mobile telecommunication technologies and the advent of the various connected devices. According to the research of Sandvine company, mobile data has the peak traffic around 2PM in Europe (See Fig. 1.5) [7]. The Internet service providers (ISP) and the mobile carriers spend their network management budgets by decreasing the traffic in congestion time [8]. Therefore, the mobile carriers need to adopt the Network Intelligence Technology for efficiently managing and distributing the mobile traffic that constantly occurs in the mobile network under limited spectrum resource environment [9, 10].

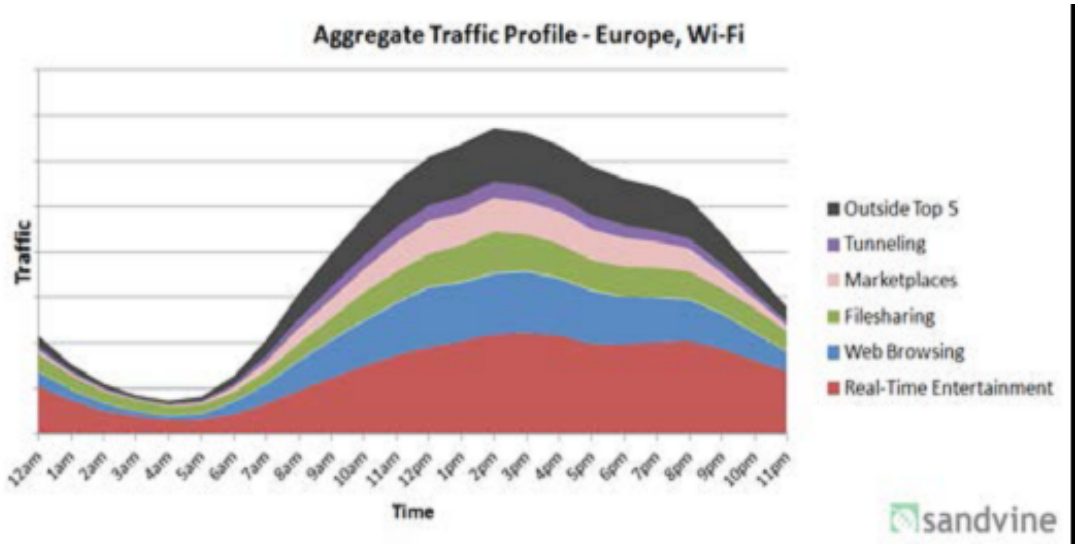

Figure 1.5. Mobile Data Traffic Profile (Europe, Wi-Fi) (Source: Sandvine, 2012)

In order to enhance the mobile traffic situation in the world, the Electronics and Telecommunications Research Institute (ETRI) has developed the Deep Packet Inspection (DPI) platform as part of the development of Network Intelligence Technology [11]. So In the below I describe the DPI platform which is developed which respected to software and hardware.

The DPI technology was first introduced for network security. As the representative security technology prior to the introduction of DPI, Stateful Packet Inspection (SPI) technology was widely used, which is the technology for detecting abnormal packets by inspecting the packet headers (Layer 2-4) [12]. However, SPI is not able to detect new network attacks, for example, NIDS Evasion and DDoS. Accordingly, DPI was introduced and it became a new technology to inspect the Payload, which means the contents of packets, by inspecting the entire layer of packets (Layer 2-7) (Figure 1.6) [13]. These days,

the network enterprises embed the technology into their equipment. The DPI provides operators with visibility into the data traffic needed to enable new service models and pricing plans and to support new architectures like software-defined networks (SDNs) [14].

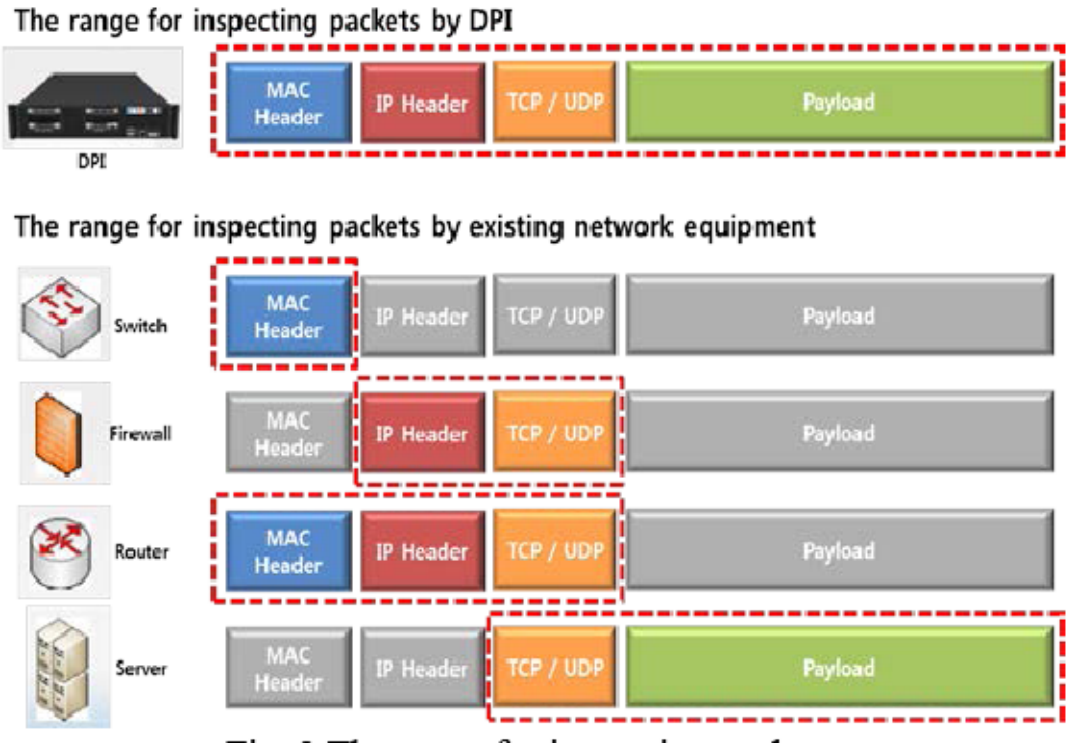

Figure 1.6. The range for inspecting packets (Source: ETRI, 2012)

According to Infonetics Research, the DPI market is expected to grow at a 30.4% compound annual growth rate (CAGR) from 2011 to 2016, driven by the increased use of DPI in wireless networks [15]. The representative DPI equipment developers are Sandvine, Cisco, Arbor Networks and Procera. They

support Internet traffic management and the real-time processing of the massive traffic by DPI standalone equipment.

The DPI equipment of Sandvine, the leader in the DPI market, has unique DPI signature analysis functions (Figure 1.7) [14, 15]. The PTS 24500 of Sandvine provides 80Gbps process capacity per box by the Policy Traffic Switch. With this function, it also provides network reporting, management of congestion, the creation of private services and management of malicious traffic [16]. The SCE 800 developed by Cisco is the router embedding DPI modules and it provides 30Gbps process capacity, multi-clustering supporting 10Gbps and concurrent processing for 1 million subscribers [17].

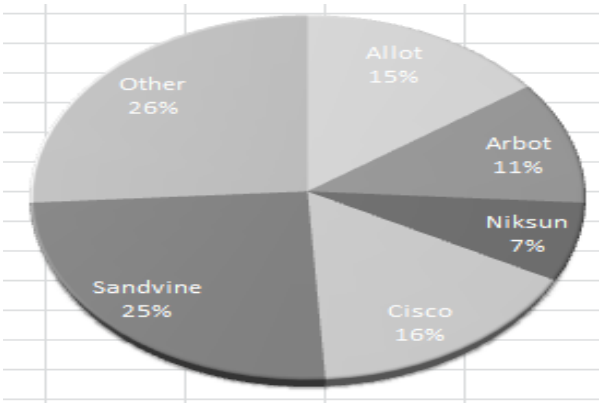

Figure 1.7. The DPI market share (Source: Infonetics Research, 2013)

The ETRI developed the DPI platform which is the High-speed Internet Traffic Control and Analysis Platform (HITCAP) with small enterprises and universities in Korea. HITCAP is the platform including the hardware and software. It supports the 40Gbps processing capacity, service virtualization, and individual policy controls [18].

In below table shows the specification of DPI equipments that is used in the wide range of world telecommunication network companies.

Table 1.1 The specification of DPI equipment

	Sandvine	Cisco	Arbor networks	Procera	ETRI	
Product Name	PPTS 24500	SCE8000	E100	PL-20000	HITCAP-10	HITCAP-40
Type	STANDALONE	STANDALONE	STANDALONE	STANDALONE	STANDALONE	STANDALONE
Height	4U	5U	5U	14U/15U	2U	2U
Interface	8 ports*10GE 8 ports*1GE	2 or 4 ports*10GE 8~16 ports*10GE	4 ports*10GE 12 ports*10GE	36 ports*10GE 2 ports*100GE	3 ports*10GE(2ea) 2 ports*10GE(2ea)	4 ports*10GE(4ea)

Continuation

Blades/Sash Performance	40Gbps / 60Gbps	15Gbps / 30Gbps	10Gbps / 20Gbps	320Gbps	20Gbps / 80Gbps	40Gbps / 160Gbps(Estimated)
The umber of Subscribers	5M	1M		10M	2M	5M(Estimated)
Features	- Recognition of various applications - 3G network billing	- Routing based on DPI technology	- The management and control system for centralized traffic	- Policy technology based on DRDL - Supporting 100GbE interface	- Service virtualization (Multitasking) - eFlowC (Easy development environment) - Controlling individual policies - High-performance DPI pattern recognition - Flexible Add-on (Multi-core with co-processor)	

The features of HITCAP are service virtualization for multitasking functions,

eFlowC, which is the easy development environment, controlling individual policies, high-performance DPI pattern recognition, and flexible add-on, which has multi-core with co-processor.

The DPI platforms that are developed in these days are expected to apply to the following communication network areas:

1) Additional management services for subscribers based on personalized policies;

2) The charge per packets and service traffic control;

3) Real-time accurate monitoring of massive traffic and controlling of the service traffic;

4) Real-time integrated control of the network traffic for providing a clean Internet environment.

Conclusion for chapter I

So far we have discussed the structure of the IP-based architecture and network security issues and especially DPI solutions for modern telecommunication market. IP-based communication architecture creates great opportunities for new application domains. Hence, applications that previously could not be realized can now be successfully implemented.

Additionally, application domains built upon older technologies derive increased functionality when utilizing IP-based technology. For illustration, consider an application domain that has clearly taken advantage of IP-based architecture: visual surveillance systems.

An IP-based solution also allows images to be remotely stored and monitored over any interconnected network, such as the Internet. This alone creates huge advantages for enterprises that wish to outsource the monitoring of their offices and facilities to a third party surveillance and monitoring center. This center simply needs a password and the IP-address to access live pictures, via the Internet, from a camera placed anywhere in the world. Moreover, the IP-based architecture creates a new world in which different applications can be

completely integrated. For instance, motion pictures can be distributed to other network solutions, such as

The primary benefits of an IP-based network strategy are the cost savings and operational improvements from using one converged network instead of several smaller networks dedicated to specific purposes, like data, voice and motion pictures. The second most important group of benefits from network convergence is in enabling new applications. New applications not only drive cost reductions; they can also be a source of new revenue as they provide value essential to enterprises and users.

Convergence is here and the benefits are real. Now it's time to pick strategic partners those who understand the broad scope of needs and are committed to meeting them and take the first step towards an IP-based future.

The evolution of IP based networks has significantly altered the requirements for modern network security architecture. Just a few years ago, a simple Stateful Packet Inspection (SPI) Firewall was sufficient to stop basic attacks such as port scans and DoS attacks. Now, application-layer buffer overflow attacks, Spam, Spyware, Polymorphic Trojans, Blended Threats, Phishing and Pharming are causing unprecedented amounts of financial damage and loss of productivity.

To detect and prevent these threats, a completely new kind of security system is required. This system still performs all of the classic functions of the firewall, but much more. This system is based on Deep Packet Inspection (DPI) technology, where the appliance has the brains - and the horse power- to inspect every byte of every packet… even across multiple thousands of streams of packets. The modern DPI security appliance resembles its SPI predecessor only by its looks… inside is a completely new system, designed from the ground up to deal with the rigors of in-depth packet inspection.

Due to the dramatic increase of data traffic in mobile networks, lots of service providers are adopting the DPI equipment for securing the visibility of

the data traffic. The DPI technology has become one of the key network intelligent technologies in order to manage the data traffic. Recently, Sandvine and Cisco, which are major network equipment enterprises, have released DPI equipment. The ETRI has also developed the DPI platform, i.e., HITCAP, as the representative Korean DPI platform.

The developed DPI platforms will contribute to improving the mobile network environment by controlling the data traffic on wired and wireless networks. This is one of the important goals of telecommunication enterprises and the platform is expected to have a significant role in improving the networks.

Chapter II. Packet inspection, filtiration analyses and firewall systems

1. The definition of packet filtering

Packet filtering is a network security mechanism that works by controlling what data can flow to and from a network. The basic device that interconnects IP networks is called a router. A router may be a dedicated piece of hardware that has no other purpose, or it may be a piece of software that runs on a general-purpose computer running Unix, Linux, Windows or another operating system (MS-DOS, BSD, Embedded OS, Macintosh, or others). Packets traversing an internetwork (a network of networks) travel from router to router until they reach their destination. The Internet itself is sort of the granddaddy of internetworks - the ultimate "network of networks".

A router has to make a routing decision about each packet it receives; it has to decide how to send that packet on towards its ultimate destination. In general, a packet carries no information to help the router in this decision, other than the IP address of the packet's ultimate destination. The packet tells the router where

it wants to go but not how to get there. Routers communicate with each other using routing protocols such as the Routing Information Protocol (RIP) and Open Shortest Path First (OSPF) to build routing tables in memory to determine how to get the packets to their destinations. When routing a packet, a router compares the packet's destination address to entries in the routing table and sends the packet onward as directed by the routing table. Often, there won't be a specific route for a particular destination and the router will use a default route; generally, such a route directs the packet towards smarter or better-connected routers.

In determining how to forward a packet towards its destination, a normal router looks only at a normal packet's destination address and asks only "How can I forward this packet?" A packet filtering router also considers the question "Should I forward this packet?" The packet filtering router answers that question according to the security policy programmed into the router via the packet filtering rules.

Some machines do packet filtering without doing routing; that is, they may accept or reject packets destined for them before they do further processing.

Packet filtering systems route packets between internal and external hosts, but they do it selectively. They allow or block certain types of packets in a way that reflects a site's own security policy, as shown in Figure 2.1. The type of router used in a packet filtering firewall is known as a screening router.

As we discuss in below, every packet has a set of headers containing certain information. The main information is:

• IP source address;

• IP destination address;

• Protocol (whether the packet is a TCP, UDP, or ICMP packet);

• TCP or UDP source port;

• TCP or UDP destination port;

• ICMP message type;

• Packet size.

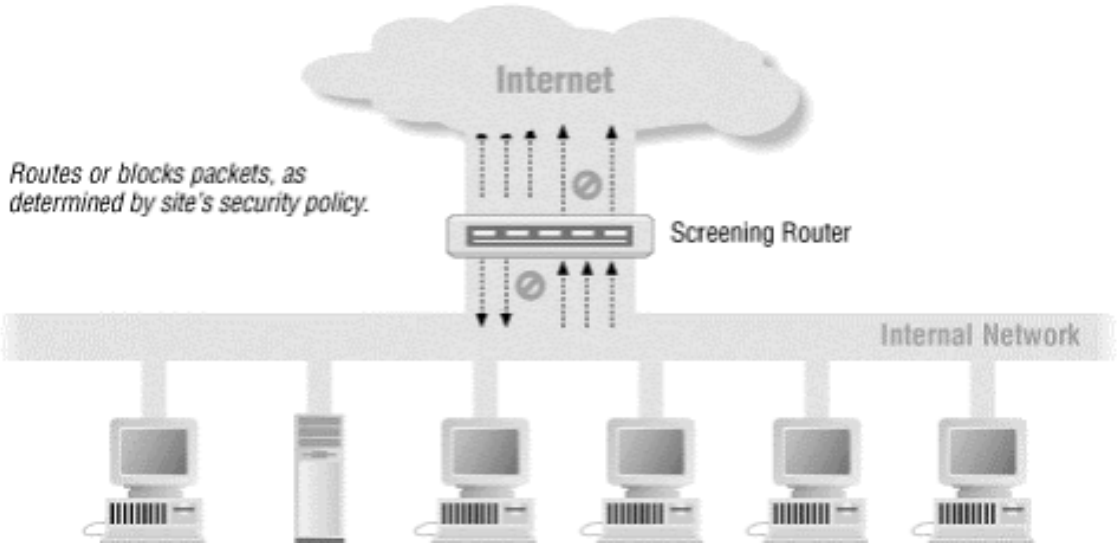

Figure 2.1. Using a screening router to do packet filtering

The router can also look past the packet headers at data further on in the packet; this allows it, for instance, to filter packets based on more detailed information (like the name of the web page that somebody is requesting) and to verify that packets appear to be formatted as expected for their destination port. The router can also make sure that the packet is valid (it actually is the size that it claims to be and is a legal size, for instance), which helps catch a number of denial of service attacks based on malformed packets.

In addition, the router knows things about the packet that aren't reflected in the packet itself, such as:

• The interface the packet arrives on;

• The interface the packet will go out on.

Finally, a router that keeps track of packets it has seen knows some useful historical facts, such as:

- Whether this packet appears to be a response to another packet (that is, its source was the destination of a recent packet and its destination is the source of that other packet);
- How many other packets have recently been seen to or from the same host;
- Whether this packet is identical to a recently seen packet;
- If this packet is part of a larger packet that has been broken into parts (fragmented).

To understand how packet filtering works, let's look at the difference between an ordinary router and a screening router.

An ordinary router simply looks at the destination address of each packet and picks the best way it knows to send that packet towards that destination. The decision about how to handle the packet is based solely on its destination. There are two possibilities: the router knows how to send the packet towards its destination, and it does so; or the router does not know how to send the packet towards its destination, and it forgets about the packet and returns an ICMP "destination unreachable" message, to the packet's source.

A screening router, on the other hand, looks at packets more closely. In addition to determining whether or not it can route a packet towards its destination, a screening router also determines whether or not it should. "Should" or "should not" are determined by the site's security policy, which the screening router has been configured to enforce.

Packet filtering may also be performed by devices that pay attention only to "should" and "should not" and have no ability to route. Devices that do this are packet filtering bridges. They are rarer than packet filtering routers, mostly because they are dedicated security devices that don't provide all the other

functions that routers do. Most sites would rather add features to routers that they need anyway, instead of adding a dedicated device. However, being a dedicated device provides advantages for packet filtering bridges; in particular, they are harder to detect and attack than packet filtering routers. They provide the same general features that we discuss for packet filtering routers.

Once it has looked at all the information, a straightforward packet filtering router can do any of the following things:

- Send the packet on to the destination it was bound for;
- Drop the packet - just forget it, without notifying the sender;
- Reject the packet - refuse to forward it, and return an error to the sender;
- Log information about the packet;
- Set off an alarm to notify somebody about the packet immediately.

More sophisticated routers might also be able to do one or more of these things:

- Modify the packet (for instance, to do network address translation);
- Send the packet on to a destination other than the one that it was bound for (for instance, to force transactions through a proxy server or perform load balancing);
- Modify the filtering rules (for instance, to accept replies to a UDP packet or to deny all traffic from a site that has sent hostile packets).

The fact that servers for particular Internet services reside at certain port numbers lets the router block or allow certain types of connections simply by specifying the appropriate port number (e.g., TCP port 23 for Telnet connections) in the set of rules specified for packet filtering.

Here are some examples of ways in which you might program a screening router to selectively route packets to or from your site:

• Block all incoming connections from systems outside the internal network, except for incoming SMTP connections (so that you can receive electronic mail);

• Block all connections to or from certain systems you distrust;

• Allow electronic mail and FTP services, but block dangerous services like TFTP, the X Window System, RPC, and the "r" services (*rlogin*, *rsh*, *rcp*, etc.).

Packet filtering devices that keep track of packets that they see are frequently called Stateful packet filters (because they keep information about the state of transactions). They may also be called dynamic packet filters because they change their handling of packets dynamically depending on the traffic they see. Devices that look at the content of packets, rather than at just their headers, are frequently called intelligent packet filters. In practice, almost all Stateful packet filters also are capable of looking at the contents of packets, and many are also capable of modifying the contents of packets, so you may see all these capabilities lumped together under the heading "Stateful packet filtering". However, something can legitimately be called a "Stateful packet filter" without having the ability to do advanced content filtering or modification.

A packet filtering system is also a logical place to provide virtual private network or network address translation services. Since the packet filter is already looking at all of the packets, it can easily identify packets that are intended for a destination that is part of the virtual private network, encrypt those packets, and encapsulate them in another packet bound for the appropriate destination[1], [2].

2. The main types of packet inspection and filtering

Many of the functions provided by packet inspection technology have been available before to limited extent depending on the level of packet analysis. Packet inspection technologies that have been in use in networking environments can be classified in three classes. These three classes are ‘shallow’, ‘medium’, and ‘deep’ packet inspection. Figure 2.2 provides a visual representation of the depth of packet inspection each of these technologies allows for.

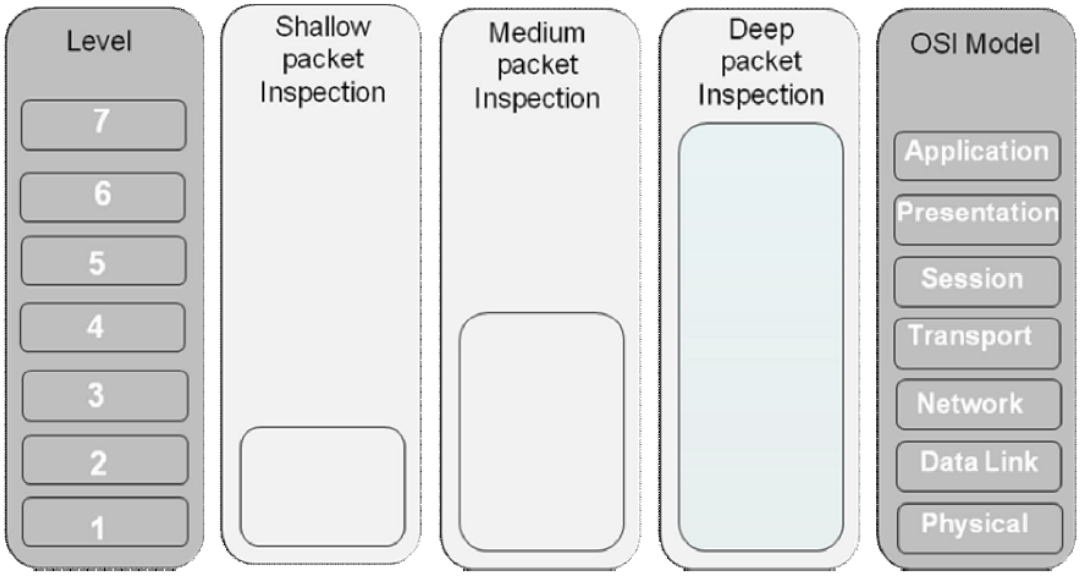

Figure 2.2. Packet Inspection levels

Shallow packet inspection (SPI) examines the headers of the packets (which is the information placed at the beginning of a block of data, such as the sender and recipient's IP addresses), as opposed to the body or “payload” of the packet.

This kind of packet inspection allows the communications to remain 'virtually anonymous' since the content of the packets is not observed, and the information in the header is used only to route the packet.

SPI technologies drive the (relatively) simplistic firewalls found in the recent generations of operating systems, such as Windows XP, Windows Vista, and OS X. These firewalls stand between a particular client computer and the network that it is attached to. They limit user-specified content from leaving, or being received by, the client computer.

Medium Packet Inspection (MPI) is typically used to refer to ‘application proxies’, or devices that stand between end-users’ computers and ISP/Internet gateways. These proxies can examine packet header information against their loaded parse-list. When a packet enters the proxy, it is analyzed against a parse-list that system administrators can easily update. A parse-list allows specific packet-types to be allowed or disallowed based on their data format types and associated location on the Internet, rather than on their IP address alone.

MPI devices can read the presentation layer of the packet’s payload and identify facets of the application layer. Using MPI devices, administrators could prevent client computers from receiving flash files from YouTube, or image files from social networking sites. MPI technologies can prioritize some packets over others by examining the application commands that are located within the application layer and the file formats in the presentation layer.

MPI devices suffer from poor scalability which limits their usefulness for ISPs, where tens of thousands of applications can be transmitting packets at any given moment.

Deep Packet Inspection (DPI) technologies are intended to allow network operators precisely to identify the origin and content of each packet of data that passes through the networking hubs. DPI devices are designed to determine

what programs generate packets, in real-time, for hundreds of thousands of transactions each second. About DPI I explain in detail in next chapter.

The most straightforward kind of packet filtering and inspection lets you control (allow or disallow) data transfer based on:

• The address the data is (supposedly) coming from;

• The address the data is going to;

• The session and application ports being used to transfer the data.

Basic packet filtering systems don't do anything based on the data itself; they don't make content-based decisions. Straightforward packet filtering will let you say:

- Don't let anybody use the port used by Telnet (an application protocol) to log in from the outside.

Or:

Let everybody send us data over the port used for electronic mail by SMTP (another application protocol).

Or even:

That machine can send us data over the port used for news by NNTP (yet another application protocol), but no other machines can do so.

However, it won't let you say:

This user cans Telnet in from outside, but no other users can do so.

Because "user" isn't something a basic packet filtering system can identify. And it won't let you say:

You can transfer these files but not those files.

Because "file" also isn't something a basic packet filtering system can identify. It won't even let you say:

Only allow people to send us electronic mail over the port used by SMTP.

Because a basic packet filtering system looks only at the port being used; it can't tell whether the data is good data, conforming to the protocol that's supposed to use that port, or whether somebody is using the port for some other purpose.

More advanced packet filtering systems will let you look further into the data of a packet. Instead of paying attention only to headers for lower-level protocols, they also understand the data structures used by higher-level protocols, so they can make more detailed decisions.

Slightly more advanced packet filtering systems offer state tracking and/or protocol checking (for well-known protocols). State tracking allows you to make rules like the following:

Let incoming UDP packets through only if they are responses to outgoing UDP packets you have seen.

Or:

Accept TCP packets with SYN set only as part of TCP connection initiation.

This is called stateful packet filtering because the packet filter has to keep track of the state of transactions. It is also called dynamic packet filtering because the behavior of the system changes depending on the traffic it sees. For instance, if it's using the preceding rule, you can't look at an incoming UDP packet and say that it will always be accepted or rejected.

Different systems keep track of different levels of state information. Some people are willing to call something a stateful packet filtering system if it enforces TCP state rules (which control the flags used during startup and teardown of TCP sessions), even if the packet filtering system provides no further stateful features. While TCP state enforcement is nice to have (it helps to prevent some forms of port scanning and denial of service), it does not allow you to support additional protocols, and we do not consider it stateful packet filtering.

Figure 2.3 illustrates dynamic packet filtering at the UDP layer.

State tracking provides the ability to do things that you can't do otherwise, but it also adds complications. First, the router has to keep track of the state; this increases the load on the router, opens it to a number of denial of service

attacks, and means that if the router reboots, packets may be denied when they should have been accepted. If a packet may go through redundant routers, they all need to have the same state information. There are protocols for exchanging this information, but it's still a tricky business. If you have redundant routers only for emergency failover, and most traffic consistently uses the same router, it's not a problem. If you are using redundant routers simultaneously, the state information needs to be transferred between them almost continuously, or the response packet may come through before the state is updated.

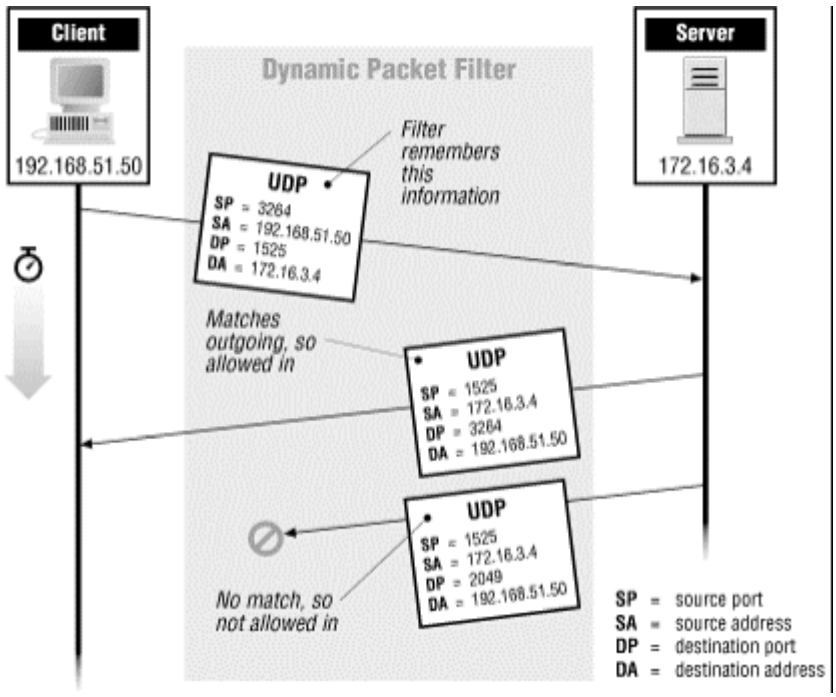

Figure 2.3. Dynamic packets filtering at the UDP layer

Second, the router has to keep track of state without any guarantee that there's ever going to be a response packet. Not all UDP packets have responses.

At some point, the router's going to have to give up and get rid of the rule that will allow the response. If the router gives up early, it will deny packets that should have been accepted, causing delays and unneeded network traffic. If the router keeps the rule too long, the load on the router will be unnecessarily high, and there's an increased chance that packets will be accepted when they should have been denied. Some protocol specifications provide guidelines, but those are not necessarily useful. For instance, DNS replies are supposed to arrive within 5 seconds, but reply times for name service queries across the Internet can be as high as 15 seconds; implementing to the protocol specification will almost always deny a response that you wanted to accept.

This sort of filtering is also vulnerable to address forging; it is validating that packets are responses based on their source addresses, so an attacker who intercepts an outgoing packet can forge the appropriate source address and return an acceptable "reply" (or, depending on the implementation, a whole bunch of packets all of which will be accepted as replies). Nonetheless, this provides a reasonable degree of security for some UDP based protocols that would otherwise be extremely difficult to protect [1], [2].

3. Advantages of packet inspection and filtering

The main advantage of packet inspection and filtering is leverage: it allows you to provide, in a single place, particular protections for an entire network. Consider the Telnet service as an example. If you disallow Telnet by turning off the Telnet server on all your hosts, you still have to worry about someone in your organization installing a new machine (or reinstalling an old one) with the Telnet server turned on. On the other hand, if Telnet is not allowed by your filtering router, such a new machine would be protected right from the start, regardless of whether or not its Telnet server was actually running.

Routers also present a useful choke point for all of the traffic entering or leaving a network. Even if you have multiple routers for redundancy, you probably have far fewer routers, under much tighter control, than you have host machines.

Certain protections can be provided only by filtering routers, and then only if they are deployed in particular locations in your network. For example, it's a good idea to reject all external packets that have internal source addresses - that is, packets that claim to be coming from internal machines but that are actually coming in from the outside - because such packets are usually part of address-spoofing attacks. In such attacks, an attacker is pretending to be coming from an internal machine. You should also reject all internal packets that have external source addresses; once gain, they are usually part of address-spoofing attacks. Decision-making of this kind can be done only in a filtering router at the perimeter of your network.

Only a filtering router in that location (which is, by definition, the boundary between "inside" and "outside") is able to recognize such a packet, by looking at the source address and whether the packet came from the inside (the internal network connection) or the outside (the external network connection). In figure 2.4 is illustrated this type of source address forgery.

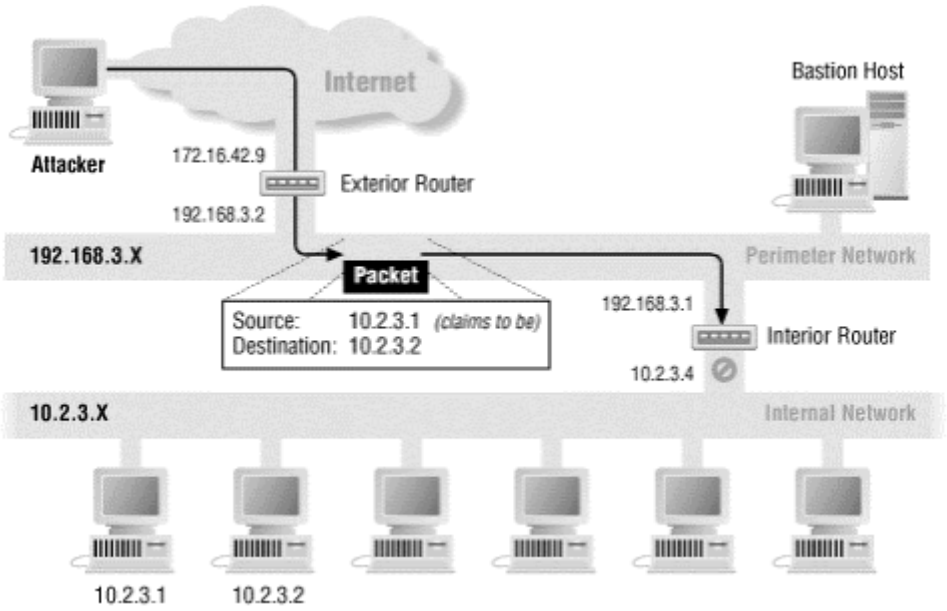

Figure 2.4. Source address forgery

Filtering routers are also good at detecting and filtering out illegal packets. Many denial of service attacks depend on sending misformatted packets of one sort or another. Routers in general have very reliable TCP/IP implementations (so they are not vulnerable to these attacks) and are well placed to prevent these attacks.

General-purpose computers being used as packet filters are more likely to be vulnerable to these attacks, but at least it is easier to fix them than it is to fix all your internal machines [1].

In general, packet filtering gives a number of advantages that we can count.

- *One screening router can help protect an entire network.* One of the key advantages of packet filtering is that a single, strategically placed packet filtering router can help protect an entire network. If only one router connects

your site to the Internet, you gain tremendous leverage on network security, regardless of the size of your site, by doing packet filtering on that router.

- *Simple packet filtering is extremely efficient.* Because simple packet filtering requires paying attention only to a few packet headers, it can be done with very low overhead. Proxying is a fairly time-consuming operation, and adding proxying means directing connections through another program, usually on a machine that otherwise wouldn't be necessary to the routing process. Packet filtering takes place on a machine that was already in the critical path, and introduces a much smaller delay.

However, there is no free lunch; the more work your packet filters do, the slower they will be. If your packet filters behave like proxies, doing complicated data-driven operations that require keeping track of multiple packets, they will tend to perform like proxies as well.

- *Packet filtering is widely available.* Packet filtering capabilities are available in many hardware and software routing products, both commercial and freely available over the Internet. Most sites and corporative networks have packet filtering capabilities available in the routers they use.

Most commercial router products include packet filtering capabilities. Packet filtering capabilities are also available for a number of general-purpose computers, servers and other network devices.

There are some disadvantages to using packet filtering as well. Packet filtering reduces router performance. Doing packet filtering places a significant extra load on a router. As we discussed previously, more complex filters place more load on the router, but in some cases, simply turning on packet filtering on a given interface can also cost you a lot of performance on some routers, because the filtering is incompatible with certain caching strategies commonly used for performance enhancement. Cisco's "fastpath" functionality is an example of this; normally, fastpath can perform basic routing functions completely on the interface card, without involving the main CPU, but using

some forms of filtering requires involving the main CPU for each packet, which is much slower.

4. Firewall systems and their main services

A firewall is a mechanism used to control and monitor traffic to and from a network for the purpose of protecting devices on the network. It compares the traffic passing through it to a predefined security criteria or policy, discarding messages that do not meet the policy's requirements. In effect, it is a filter blocking unwanted network traffic and placing limitations on the amount and type of communication that occurs between a protected network and other networks (such as the Internet, or another portion of a site's network).

A firewall can come in many different designs and configurations. It can be a separate hardware device physically connected to a network (such as the Cisco PIX® or the Symantec Security Gateway® firewalls), a hardware/software unit with OS-based firewall capabilities (such as 'iptables' running on a Linux® server), or even a completely host-based software solution installed directly on the workstation to be protected (such as Norton Personal Firewall® or Sygate Personal Firewall®).

Separate hardware or hardware/software units are often referred to as network firewalls and typically provide the most secure solution for the separation of the PCN from corporate network. They are dedicated-function units that can be hardened to resist all but the most ingenious assaults. In addition, network firewalls generally offer the best management options since they are typically designed to allow remote management.

Host-based firewalls must generally accept compromises since the primary function of the host device is not security, but workstation or server tasks such as database access or web services. As well, host-based firewall solutions are currently only available for Windows or Unix-based platforms and can do little

to regulate traffic destined for embedded control devices, such as PLCs, on the network. Host-based firewalls may have a place on the any data communication network. Thus with a few exceptions, data communication network firewalls will be a dedicated hardware or hardware/software solution that, through a series of rules, allows or denies network traffic to pass on to the control network.

Network traffic is sent in discrete groups of bits, called a packet. Each packet typically contains a number of separate pieces of information, including (but not limited to) items such as the:

- Sender's identity (Source Address);
- Recipient's identity (Destination Address);
- Service to which the packet pertains (Port Number);
- Network operation and status flags;
- Actual payload of data to be delivered to the service.

A firewall, upon receiving a packet, inspects and analyses these characteristics, and determines what action to take with the packet. It may choose to drop the packet, allow it through immediately, buffer it momentarily to bandwidth limit a class of service or forward it to a different recipient than was initially intended - whatever the network security policy deems is an appropriate action to take.

These decisions are based on a series of rules commonly referred to as Access Control Lists (ACLs). Different classes of firewalls exist, each with increasingly sophisticated analysis and action capabilities.

2.4.1 Packet Filter Firewalls

The simplest class of firewall is known as a packet filter firewall. It has a series of static rules and uses them to take action upon received packets on an individual basis. The following sample rules can easily be handled by a packet filter firewall:

- Accept Domain Name Service (DNS) response packets on User Datagram Protocol (UDP) port 53;
- Block any traffic to or from Internet Protocol (IP) address 24.116.25.21;
- Prevent web surfing by blocking outgoing packets from accessing Transmission Control Protocol (TCP) ports 80, 443, 3128, 8000 and 8080, unless they are directed to the corporate intranet web server's IP address;
- Drop source routed packets;
- Accept TCP port 23 (telnet) traffic to a specific Distributed Control System's (DCS) IP address, from a specified range of IP addresses for engineering consoles.

Unfortunately, a packet filter firewall lacks the ability to understand the relationships between a series of packets. For example, the broad rule "Accept DNS response packets on UDP port 53" contains a serious flaw. What if no DNS query was ever issued, but a spoofed DNS "response" packet came in instead? This simplistic firewall would accept the packet and deliver it to the "requesting" host, possibly causing it to become confused. A moderately intelligent hacker could use these firewall weaknesses to compromise internal systems.

The advantages of packet filtering firewalls include low cost and low impact on network performance, often because only the IP addresses and port numbers in the packet are examined. This method is also sometimes called Static Filtering [3]. It is often directly deployed in layer-3 switches or routers, rather than dedicated "firewalls".

2.4.2 Stateful Firewalls

A more sophisticated firewall, known as a Stateful firewall, intelligently tracks the interrelationships between packets allowed to flow through it. By keeping a history of accepted packets and the state of current connections, it can

accept only “anticipated” traffic. Stateful firewall rule sets can be made conditional due to the firewall's intelligence. For example:

- Accept DNS response packets on UDP port 53 only if they match an outgoing DNS query with the same DNS request identification;
- Allow File Transfer Protocol (FTP) data channel traffic through only after successful control channel negotiation;
- Block all incoming web traffic originating from TCP port 80 unless it is specifically in response to a previous outgoing Hyper-Text Transfer Protocol (HTTP) request.

These types of firewalls offer a high level of security, good performance, and transparency to end users, but are more expensive. Due to their complex nature, they can be less secure than simpler types of firewalls if not administered by competent personnel. This method is also sometimes called Dynamic Packet Filtering [4].

To provide better performance, many stateful firewalls abbreviate examinations by performing only an application-level examination of the packet that initiates a communication session, which means that all subsequent packets are tracked through the state table using Layer 4 information and lower. This is an efficient way to track communications, but it lacks the ability to consider the full application dialog of a session. In turn, any deviant application-level behavior after the initial packet might be missed, and there are no checks to verify that proper application commands are being used throughout the communication session.

However, because the state table entry will record at least the source and destination IP address and port information, whatever exploit was applied would have to involve those two communicating parties and transpire over the same port numbers. Also, the connection that established the state table entry would not be properly terminated, or the entry would be instantly cleared.

Finally, whatever activity transpired would have to take place in the time left on the timeout of the state table entry in question. Making such an exploit work would take a determined attacker or involve an accomplice on the inside.

Another issue with the way stateful inspection firewalls handle application-level traffic is that they typically watch traffic more so for triggers than for a full understanding of the communication dialog; therefore, they lack full application support. As an example, a stateful device might be monitoring an FTP session for the port command, but it might let other non-FTP traffic pass through the FTP port as normal. Such is the nature of a stateful firewall; it is most often reactive and not proactive. A stateful firewall simply filters on one particular command type on which it must act rather than considering each command that might pass in a communication flow. Such behavior, although efficient, can leave openings for unwanted communications types, such as those used by covert channels or those used by outbound devious application traffic [7].

2.4.3 Application Proxy Firewalls

Application Proxy firewalls open packets at the application layer, process them based on specific application rules, then reassemble them and forward them to the desired target device. Typically they are designed to concentrate on a variety of application protocols (such as Telnet, FTP, HTTP, etc.) through a single machine, but then forward the traffic to individual host computers for each service. Instead of connecting directly to an external server, the client connects to the proxy firewall which in turn initiates a connection to the requested external server. Depending on the type of proxy firewall used, it is possible to configure internal clients to perform this redirection automatically, without the user's knowledge. Others might require that the user connect directly to the proxy server and then initiate the connection through a specified format [5].

Proxy firewalls offer some significant security features for controlling protocols that the firewall can recognize. For example, it is possible to apply access control lists against the application protocols, requiring users or systems to provide additional levels of authentication before access is granted. In addition, rules sets can be created which understand specific protocols and can be configured to block only subsections of the protocol. For example, a firewall rule for HTTP could be created to block all inbound HTTP traffic containing scripts. By contrast, a filtering router could either block all HTTP traffic or none, but not a subset.

These types of firewalls can offer a high level of security, but can also have a significant impact on network performance. In addition, most application proxy firewalls products only support common Internet protocols such as HTTP, FTP or SMTP (Simple Mail Transfer Protocol). As a result, messages containing industrial application layer protocols such as Common Industrial Protocol® (CIP) or MODBUS/TCP® will require the firewall to process the traffic in stateful or packet-filter mode [5]. In the past few years, most of firewalls on the market have utilized a combination of Stateful and Application-Proxy techniques and are often referred to as hybrid firewalls.

2.4.4 Deep Packet Inspection Firewalls

The firewall market is currently moving to a fourth technique that is referred to as "deep packet inspection" (DPI) or "application firewalling". This typically offers filtering deeper into the application layer than the traditional application proxy and yet does not perform a full proxy on the TCP connection. For example, DPI firewalls will be able to inspect Simple Object Access Protocol (SOAP) objects in Extended Markup Language (XML) on web connections and enforce policy on what objects are allowed/disallowed into the network. This market is still in development.

Microsoft, Cisco, Checkpoint, Symantec, Nortel, Sonic Wall, NAI, Juniper/ Netscreen, and others, have, in the past eighteen months started manufacturing

firewall appliances that implement Deep Packet Inspection (DPI). In general, the DPI engine scrutinizes each packet (including the data payload) as it traverses the firewall, and rejects or allows the packet based upon a ruleset that is implemented by the firewall administrator. The inspection engine implements the ruleset based upon signature-based comparisons, heuristic, statistical, or anomaly-based techniques, or some combination of these.

Deep Packet Inspection promises to enhance firewall capabilities by adding the ability to analyze and filter SOAP and other XML messages, dynamically open and close ports for VoIP application traffic, perform in-line AV and spam screening, dynamically proxy IM traffic, eliminate the bevy of attacks against NetBIOS-based services, traffic-shape or do away with the many flavors of P2P traffic (recently shown to account for ~35% of internet traffic), and perform SSL session inspection.

Deep Packet Inspection essentially collapses Intrusion Detection (IDS) functionality into the firewall appliance so that both a firewall and in-line IDS are implemented on the same device. Many of these products have recently been shown to be vulnerable to exploitation of software defects in their DPI inspection engines, however. These data suggest that the addition of these enhanced functions to firewalls may, in fact, weaken, rather that strengthen network perimeter security [6].

2.4.5 Other Firewall Services

In addition to the core service of traffic inspection and filtering, most modern firewalls offer other network based security services. These may include:

1. Acting as an intrusion detection system (IDS) by either logging packets that are denied access, recognizing network packets specifically designed to cause problems, or reporting unusual traffic patterns;

2. Deploying “frontline” anti-virus software on a firewall. If the characteristics of infected traffic can be formulated, such traffic can be blocked before entering the network;
3. Authentication services requiring users wishing to connect to devices on the other side of the firewall to authenticate to the firewall using either passwords or strong two-factor authentication methods such as public key encryption;
4. Virtual Private Network (VPN) gateway services where an encrypted tunnel is set up between firewall and remote host device;
5. Network Address Translation (NAT) where a set of IP addresses used on one side of the firewall are mapped to a different set on the other side.

Additional services offered will depend on the particular make and model of the firewall purchased. Obviously these additional functions can mean additional costs, increased complexity and reduced performance. However by utilizing them, one can often significantly improve the overall security of data communication networks [5], [6].

Conclusion for chapter II

Packet filtering systems are important module of network devices for inspecting, filtering packets between internal and external hosts. They allow or block certain types of packets in a way that reflects a site's own security policy.

An important point that should be considered when discussing perimeter security is the concept of a firewall as a network chokepoint. A *chokepoint* is a controllable, single entry point where something is funneled for greater security.

As seen above, Firewalls have been used extensively to prevent access to systems from all but a few, well defined access points (ports), but they cannot eliminate all security threats nor can they detect attacks when they happen. Stateful inspection firewalls are able to understand details of the protocol that

are inspecting by tracking the state of a connection. They actually establish and monitor connections until they are terminated. However, current network security needs, require a much more efficient analysis and understanding of the application data. Content-based security threats and problems occur more frequently, in an everyday basis. Virus and worm inflections, SPAMs (unsolicited e-mails), email spoofing, and dangerous or undesirable data, get more and more annoying and cause innumerable problems. Therefore, next generation firewalls should support Deep Packet Inspection properties, in order to provide protection from these attacks. Network Intrusion Detection Systems (NIDS) are able to support DPI processing and protect an internal network from external attacks. NIDS check the packet header, rely on pattern matching techniques to analyze packet payload, and make decisions on the significance of the packet body, based on the content of the payload.

In this chapter I provide some background information regarding Intrusion firewalls and Detection Systems that capture, inspect and filtrate data packets on the networks.

Chapter III. Analyses of deep packet inspection methods, algorithms and approaches

1. Technical Background of DPI and requirements for DPI systems

At first glance, a technical definition of deep packet inspection is straightforward to write down and in fact very simple. DPI systems inspect entire packets traveling the network as part of a communication, looking not only at packet headers like legacy systems, but also at the packet's payload. The central point of this definition is the inspection of packet payload. While this seems to be quite clear, both terms require a closer look, not least because this payload inspection constitutes the main draw for criticism of DPI technology.

DPI enables such as traffic shaping, admission, content access restrictions, information extraction about subscribers from their packet traffic, and so on and at a higher level the DPI applications which form the basis of useful services to a subscriber or capabilities to a service provider.

The convergence of traditional network services to a common IP infrastructure has resulted in a major paradigm shift for many service providers. IP networks were originally used to carry best effort Internet traffic; however,

today many service provider IP networks carry core network services including Internet, Voice over IP (VoIP), Broadcast TV (IPTV), and Video-on-Demand (VoD). IP convergence has resulted in both business opportunities and challenges for service providers.

The development of new IP services and applications has led to a new set of traffic management and security problems for service providers. In order to address these problems, first generation Deep Packet Inspection (DPI) products have been installed in networks. These products typically solve problems such as Peer-to-Peer (P2P) traffic management. While first generation DPI products are adept at managing P2P traffic, which is an urgent problem for service providers, they are not always capable of solving new problems or facilitating new services. A one-problem, one-box approach to DPI leads to deployment of armies of appliances which is not a viable long term strategy. As a result a second generation of DPI products has emerged to solve a broader set of problems. Designed as multi-purpose L2-L7 traffic management systems, these products can mitigate current and future security threats, manage traffic from specific subscribers and applications, and craft new IP services. This 'Advanced DPI' employs comprehensive and broad-based traffic management and security capabilities to solve current and future problems, plus these products operate at 10 Gbps. So, DPI functionality is a critical component of converged services IP networks [1], [3].

Minimizing negative impact to the network due to rogue applications or attacks is the second fundamental business driver for DPI. Because IP networks have become critical components of network infrastructure, outages and/or performance problems due to rogue applications and network attacks contain serious financial implications. Unlike discrete service networks, each service in a unified network is subject to all the ills of the other services.

First generation DPI and IDP products focus on solving specific and important problems and have been widely deployed in networks as a result.

However, the nature of application services and threats from intelligent adversaries is such that the capabilities of layer 2-7 content processing products must constantly change to address new needs. The current approach to solving this problem is deploying additional L2-L7 point products into the network. Clearly this is not an efficient long term solution to the problem of managing changing requirements. Because of this, a new generation of DPI technology has emerged with increased content and application processing capabilities designed to perform a wide range of functions, and execute several applications simultaneously. By deploying Advanced DPI with broad, line-speed content processing capabilities, service providers avoid deploying multiple point products for every problem, as shown in Figure 3.1.

The processing capacity of Advanced DPI devices afford network operators leverage for their DPI investment, while the functional flexibility and easy repurposing capabilities promises protection of that investment against future needs and threats.

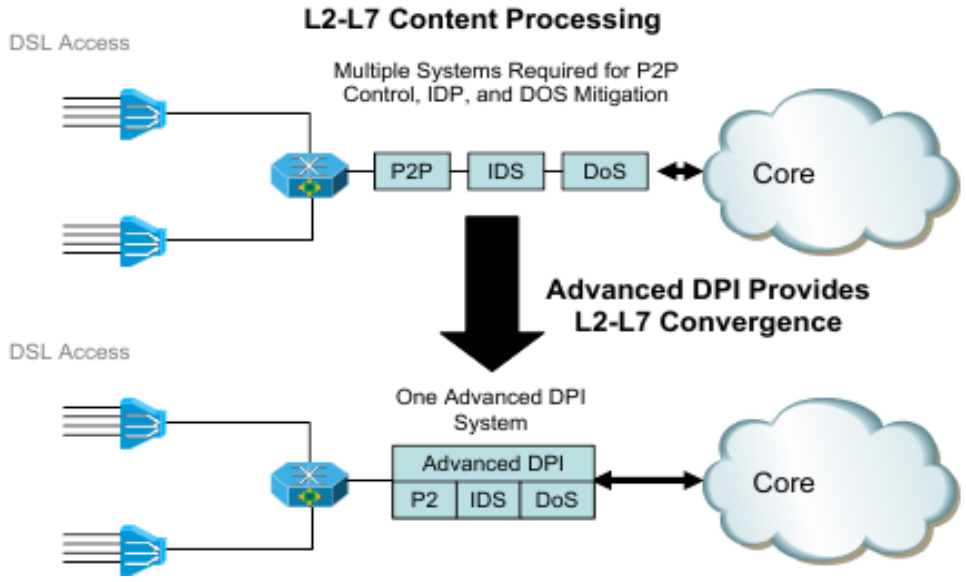

Figure 3.1. Convergence using Advanced DPI

DPI convergence makes good business sense; however, it is a complex problem requiring robust and flexible content processing technologies [1].

DPI baseline requirements are table stakes for any vendor to play in the DPI game. These are necessary conditions but not sufficient for deployment in a service provider network. The DPI baseline requirements are presented below.

*Scan Packet Payloads at all layers (L2-L7).*In order to monitor and control traffic without limitation, it is necessary to scan packets from the first bit to the last. Only this way can you ensure all layers of the protocol stack are addressed. In fact, this is the definition of Deep Packet Inspection.

For maximum applicability, payload scanning capabilities should include application content and transactions that traverse multiple packets. When appropriate, DPI products must be able to reassemble content before scanning.

Application Classification, Measurement, and Reporting. DPI products must have the ability to classify applications, measure throughput, and create application traffic reports. This provides network managers with the visibility essential for strategic traffic planning, tiered broadband access, and usage based billing.

Set Policies for Prioritization, Blocking, or Shaping. Based on packet payload scans, application classification, and a set of policies for managing traffic, DPI products must prioritize, block, or rate limit traffic. This function provides one of the fundamental controls that used for offering tiered services and for controlling problem applications such as P2P.

Session vs. Packet Identification. One of the premium requirements for Advanced DPI is session identification and state change tracking. Moreover, Advanced DPI products must be able to analyze session behavior along with packet behavior. This is important because many applications or attacks can only be analyzed in the context of a session.

While the DPI baseline requirements are table stakes, DPI premium requirements are necessary to create flexibility for handling both current and future problems and providing flexible service control. DPI premium requirements are the key to DPI equipment convergence resulting in capital and operating cost savings.

Modification of the Packet Envelope. DPI engines must be capable of modifying packet envelopes in order to implement new services and prevent attacks. This capability must be configurable, programmable, and sufficiently granular to allow maximum flexibility in packet processing. Packet envelope modification examples include:

- Modify routing information in support of content-based routing policies;
- Expand the packet length to add VLAN/MPLS tags/labels;
- Contract packet length to enable removal of tunnel labels;
- Support multiple layers of labeling/tagging as needed.

As Carrier Ethernet and MPLS networks become more prevalent in service provider networks the ability to modify packet envelopes becomes more critical.

Modification of Packet Payload Content. In addition to modifying the packet envelope, DPI engines must be capable of modifying payload content. Based on dynamic packet and session monitoring, DPI packet content modification can carry out functions such as:

- Removing viruses;
- Transforming content to support gateway functions (e.g. IPv4-to-v6);
- Enforcing network use policy (e.g., no high bandwidth Codecs, or IPv6 only);
- Solving problems related to the rapid pace of change in service protocols (e.g., SIP);
- Addressing future unknown problems.

Sometimes problems can only be resolved by modifying packet content.

Generating Packets. In some instances, the preferred method for securing services infrastructure or providing gateway functions requires the generation of packets. A premium DPI requirement is the ability to build a packet with all the appropriate content and envelope information to be injected into the network. Example uses of packet generation include:

- Building a protocol or application proxy;
- Gracefully resetting a TCP session from an unwanted host;
- In-band signaling between network infrastructures (e.g. DPI device-to-DPI device).

Adaptable Functionality. Future services and threats are unknown and potentially more complex than current services and threats. Therefore, it is essential that algorithms in next generation DPI systems that analyze the content and behavior of packets in a session and take appropriate action based on that behavior and content are adaptable. Moreover, this function 'adaptability' should be accessible directly by network operators and their solutions partners.

This creates the flexibility necessary for quickly crafting new service offerings and the ability to defend the network against new attacks.

DPI System Security Requirements. Because DPI systems are essential to network security, it is critical that Advanced DPI products themselves are completely secure. All DPI devices must protect themselves from attacks. The following DPI security requirements provide a strong foundation to achieve this objective.

- DPI systems must operate in stealth mode such that they are not visible in the network;
- The DPI device management plane and data path must be separated by design, with only a tightly controlled, protected interface between them;
- All DPI provisioning should be done out-of-band ;
- DPI products should have security accreditations [1] [2].

DPI generally implies broad capabilities as determined by the goals of the service provider. DPI technology is deployed in a service provider context, and as such, will often co-reside with other network functions that are already providing communications services to subscribers, such as access gateways, or are in the communications path of traffic, such as a router, border gateway, security or VPN gateway, firewall.

Nowadays DPI applications are quite diverse and span a broad range of technical segments and capabilities.

Policy enforcement as traffic shaping and prioritization, access controls and admission, and content filtering attract the most attention and are at the heart of both fixed and wireless broadband network evolution given their obvious importance in providing fair use, the option for new services, and the ability to enhance subscriber experience through traffic management.

Network security applications ranging from basic firewalls to network-based antivirus, intrusion detection and prevention, data leak prevention, anti-spam, and anti-SPIT & SPIM (spam internet telephony and

spam instant messaging), web-application firewalls are emerging both in conjunction with endpoint security software (to address gaps where endpoint software simply cannot be updated fast enough to address an outbreak) and in network-based security approaches where devices may not have endpoint security capability (such as in embedded or machine to machine applications).

Network and subscriber analytics applications are commonplace—these aid the service provider in gauging the overall health of the network by pinpointing performance and capacity as well as provide the service provider with a greater understanding of their subscriber's behavior that may be used to enhance marketing revenues.

Monitoring and interception—whether for purposes of lawful intercept or other regulations or to support problem diagnosis in live systems—are major DPI technical segments.

Content optimization through proxying and modifying content to better adapt content—such as by reducing still and video image quality, reformatting web pages for mobile devices, and other techniques—given bandwidth and device constraints have been applied to allow more users to enjoy content with acceptable performance than otherwise.

Billing and Metering applications to count the volumes or rates of traffic, but with more complex schemes to account for a mixture of both free, partner, and paid traffic to support schemes where a subscriber's traffic may be capped to a certain volume, may be paying by the byte, or other schemes (such as payments between a service provider and content provider for certain types of traffic and/or application usage).

Content caching applications where a service provider may elect to serve cached content via intercepting traffic and returning the content directly.

Application Distribution and Load Balancing. These employ DPI technology to examine packet content (possibly far into the packet) and re-write

the packet to direct the packet to a different destination for purposes of load balancing, fault tolerance, or other uses.

Modification and Injection applications examine content and modify packet content for a variety of purposes, such as to insert tracking ids, rewrite packet headers (such as TOS), and may inject new packets or traffic as a result (for example: injecting TCP resets to interfere with P2P traffic)[2].

2. Deep packet inspection methods and system implementation analyses

High speed and always-on network access is commonplace around the world creating a demand for more sophisticated packet processing and increased network security. The answer to this sophisticated network processing and network security can be provided by Deep Packet Inspection (DPI). In essence, deep packet inspection is able to accurately classify and control traffic in terms of content and applications. In other words, it analyzes packets content and provides a content-aware processing. The most challenging task in DPI is content inspection, since the body (payload) of each packet needs to be scanned. In general, DPI systems should provide the following:

- High processing throughput;
- Low implementation cost;
- Flexibility in modifying and updating the content descriptions;
- Scalability as the number of the content descriptions increases.

The above goals become more difficult to achieve due to two reasons. First, the gap between network bandwidth and computing power is growing. Second, the database of known attack patterns becomes larger and more complex.

Currently, several network functions need a more efficient analysis and information about the content and the application data of the processing packets.

DPI is used in network applications such as:

- Network Intrusion Detection/Prevention Systems: As opposed to traditional firewalls, NIDS/NIPS scan the entire packet payload for patterns that indicate hazardous content. A combination of packet classification (header matching) and content inspection is used to identify known attack descriptions. Previous techniques such as Stateful inspection are still required to provide efficient security;
- Layer 7 Switches1: authentication, load balancing, content-based filtering, and monitoring are some of the features that layer 7 switches support. For example application aware web switches provide transparent and scalable load balancing in data centers;
- Traffic Management and Routing: Content-based routing and traffic management may differentiate traffic classes based on the application data.

This dissertation addresses the above Deep Packet Inspection challenges focusing on DPI for Network Security (NIDS/NIPS). Although, the principles followed in all DPI network applications remain unchanged, we can note that in NIDS2 the content descriptions may be more complex and more in number, creating significant performance limitations and implementation difficulties compared to other network applications such as content-aware traffic management and switching.

So, Deep Packet Inspection (DPI) is the core component of many networking devices on the Internet such as Network Intrusion Detection (or Prevention) Systems (NIDS/NIPS), firewalls, and layer 7 switches. In DPI, in addition to examining the packet headers, the entire contents of each packet is compared against a set of signatures to check if any signature is found in the packet or not. For instance, for security applications, each individual virus or attack threat is represented using one signature. The payload of each packet passing through the network device is compared against the set of signatures,

and a match indicates the corresponding threat is found. Necessary action to neutralize the threat can then be taken. Application level signature analysis is also used for providing advanced QoS mechanisms, detecting peer-to-peer traffic, and in general application protocol identification.

In the past, DPI typically used string matching as the core operation, in which signatures are specified as simple strings. Today, DPI typically uses Regular Expression (RE) matching as the core operation, in which signatures are specified as REs. REs are used instead of simple string patterns because REs are fundamentally more expressive and thus are able to describe a wider variety of attack signatures. Most open source and commercial intrusion detection and prevention systems such as Snort [8, 10], Bro [9], HP Tipping Point and Cisco networking appliances use RE matching. Likewise, some operating systems such as Cisco IOS and Linux [11] have built RE matching into their layer 7 filtering functions.

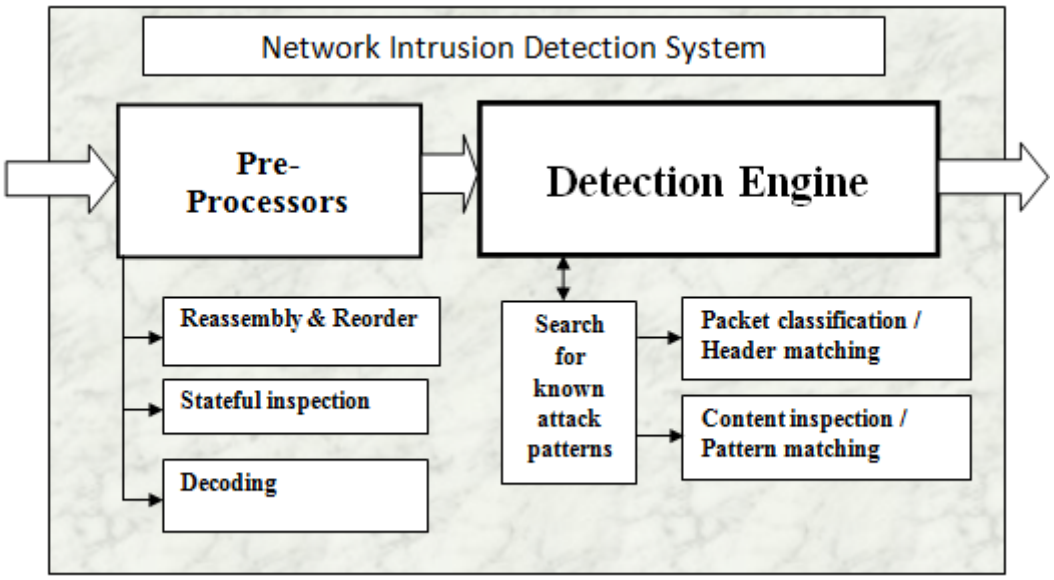

Figure 3.2. A Network Intrusion Detection System consists of several preprocessors and the detection engine

In the DPI Intrusion Detection Systems (IDS) use several preprocessors and a ruleset-based detection engine which performs packet classification and content inspection. Figure 3.3 illustrates a break-down of an intrusion detection system. It is worth noting that the described.

IDS generates per packet alerts and subsequently correlations between multiple alerts may indicate a complete attack plan. An IDS rule such as the ones of Snort and Bleeding open source IDS, consists of a header matching part and a payload matching part. The first one checks the header of each incoming packet using packet classification techniques. The second examines the payload of each packet performing content inspection. Content Inspection involves matching packet payload against predefined patterns either described as static patterns or regular expressions. Additional restrictions concerning the placement of the above patterns introduce further complexity to the processing of the IDS tasks. Below, each IDS task is discussed in detail.

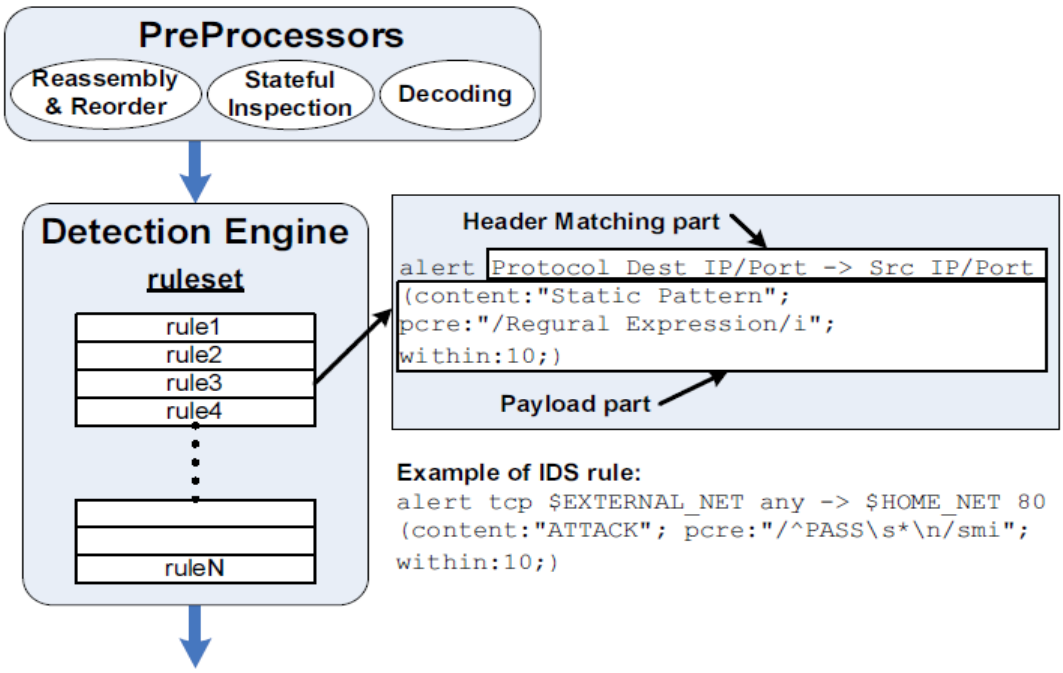

Figure 3.3. NIDS decomposition

In figure incoming traffic is scanned first by a series of preprocessors that perform packet reordering and reassembly, stateful inspection and decoding of specific kinds of traffic. Subsequently, packets are examined against rules that describe malicious activity. Each rule has a packet header and payload description. An example of IDS rule is depicted at the right bottom part of the Figure.

Preprocessors: The IDS preprocessors implement the necessary functions that allow the subsequent detection engine to correctly examine incoming traffic against predefined attack descriptions. Preprocessors are responsible for three kinds of tasks. First, they reassemble and reorder TCP packets into larger ones. This is necessary in order to detect attacks that span across multiple packets.

Second, they perform stateful inspection functions such as flow tracking or portscan detection; that is, functions related to the protocol level that keep track of different connections/flows. Stateful inspection can also be seen as a module which has an overview of the traffic -at a higher level than the content inspection- checking for abnormal events such as buffer overflows or Denial of Service (DoS) attacks. Third, preprocessors perform specialized inspection functions, mostly decoding of various kinds of traffic, e.g., Telnet, FTP, RPC, HTTP, SMTP, packets with malicious encodings, etc.

After the preprocessors comes the detection engine which uses a rule database (ruleset) to describe malicious packets. Each rule has a packet classification and a content inspection part. Furthermore, content inspection includes static pattern matching, regular expression matching and pattern placement restrictions.

Packet Classification: The header part of each NIDS rule describes the header of a potentially dangerous packet. As depicted in Figure 3.3, the header description may consist of some or all the following: Protocol, Destination IP and Port and Source IP and Port. The IP and Port fields of a rule may specify ranges of values instead of a specific address or port. This makes packet classification more challenging than a simple comparison of numerical values.

Many researchers in the past have proposed different techniques for packet classification and IP lookup such as, while some of them also use reconfigurable hardware [4], [5].

Below main packet analyzing and matching methods I will explain.

Static Pattern Matching: Matching the literal meaning of predefined (static) patterns is certainly the most significant IDS task. Static patterns are used to describe malicious payload contents and provide an insight of the packet application data. An IDS rule may contain one or more static patterns of a few bytes up to several hundreds of bytes long. Figure 3.3 illustrates an IDS-rule example which contains a static pattern. The static pattern ATTACK is

indicated as a malicious payload pattern using the statement: content:"ATTACK".

Matching thousands of payload patterns in parallel per incoming packet creates fundamental difficulties in IDS performance. Initially, IDS systems described malicious contents only with static patterns; however, recently they started using both static patterns and regular expressions.

Regular Expression Matching: We next discuss the regular expressions used in IDS packet payload scanning. More precisely, we describe the features of the regular expressions included in Snort and Bleeding Edge IDS. Snort and Bleeding Edge open source IDS, used in the remainder of this thesis, adopted the Perl-compatible regular expression syntax (PCRE). The IDS rule example of Figure 3.3 uses the statement pcre:"/^PASS\s*\n/smi"; to describe malicious content in regular expression format. Besides the remaining part of the rule, in order to identify a dangerous packet based on this rule, a string that matches the regular expression "/^PASS\s*\n/smi" needs to be included in the payload. Apart from the well known features of a strict definition of regular expressions, PCRE is extended with new operations such as flags and constrained repetitions. Table 3.1 describes the PCRE basic syntax supported by our regular expression pattern matching engines. Matching regular expressions is considered substantially more efficient and, at the same time, more complex and computationally intensive. Even if malicious contents are described in regular expression formats alone, some parts of them usually contain static patterns which are more efficient to match separately.

Pattern Placement and Other Payload Restrictions: Restrictions regarding packet payloads and payload pattern placement are features that create additional difficulties in IDS implementation. Table 3.2 depicts some of the Snort syntax features which make the IDS rules more complex. The above commands change the original meaning of the payload content rule parts (either static patterns or regular expressions) adding extra constraints regarding the

placement of the matching patterns in the packet payload. Consequently, rules might specify the packet payload part where a pattern should be matched, relative either to the beginning or the end of a packet or relative to a previously matched pattern. In addition, commands such as byte test select and test a byte payload field using several numerical or logical operators. Each IDS rule might specify different payload constraints to describe a suspicious packet using the above syntaxes. For example, the IDS rule of Figure 3.3 uses the statement within:10; to state that the second payload pattern (the regular expression /∧PASS\s*\n/smi) needs to be matched within 10 bytes after matching the first pattern (ATTACK). The above restrictions create significant implementation difficulties, making each rule to possibly require a separate module (engine, thread, etc.) to keep track of the satisfied conditions, specify the parts of the payload which are valid for each pattern to match, and store payload byte fields to be tested using the byte test and byte jump commands. The above features introduce significant cost and limit performance both in software and hardware NIDS implementations [4], [5].

Table 3.1. Snort-PCRE basic syntax

Feature	Description
a	All ASCII characters, excluding meta-characters, match a single instance of themselves
[\^$.—?*+()	Meta-characters. Each one has a special meaning
\?	Backslash escapes meta-characters, returning them to their literal meaning
[abc]	Character class. Matches one character inside the brackets. In this case, equivalent to (a|b|c)
[a-fA-F0-9]	Character class with range.
[^abc]	Negated character class. Matches every character except each non-Meta character inside breackets.
RegExp*	Kleene Star. Matches zero or more times the RegExpr

RegExp+	Plus. Matches one or more times the RegExpr.
RegExp?	Question. Matches zero or one times the RegExpr.
RegExp{N}	Exactly. Matches N times the RegExpr.

Continuation

RegExp{N, }	AtLeast. Matches N times or more the RegExpr.
RegExp{N,M}	Between. Matches N to M times the RegExpr.
\xFF	Matches the ASCII character with the numerical value indicated by the hexadecimal number FF.
\000	Matches the ASCII character with the numerical value indicated by the octal number 000.
\d, \w and \s	Shorthand character classes matching digits 0-9, word chars and whitespace, respectively.
\n, \r and \t	Match an LF char, CR char and a tab char, respectively.
(RegExp)	Groups RegExprs, so operators can be applied.
RegExp1RegExp2	Concatenation. RegExpr 1, followed by RegExpr 2.
RegExp1 | Reg-Exp2	Union. RegExpr 1 or RegExpr 2.
^RegExp	Matches RegExpr only if at the beginning of the string.
RegExp$	Dollar. Matches RegExpr only if at the end of the string.
(?=RegExp), (?!RegExp), (?<=text), (?<!text)	Lookaround. Without consuming chars, stops the matching if the RegExp inside does not match.
(?(?=RegExp) then |else)	Conditional. If the lookahead succeeds, continues the matching with the “then” RegExp. If not, with the “else” RegExp.
\1, \2. . .\N	Backreferences. Have the same value as the text matched by the corresponding pair of capturing parenthesis, from 1st through Nth.
Flags	**Description**
i	Regular Expression becomes case insensitive.
s	Dot matches all characters, including newline.

m	^ and $ match after and before newlines.

Table 3.2. Current SNORT syntax features which make IDS tasks more computationally intensive

Feature	**Description**
depth	specifies how far into a packet Snort should search for the specified pattern.
offset	specifies where to start searching for a pattern within a packet.
distance	specifies how far into a packet Snort should ignore before starting to search for the specified pattern relative to the end of a previous pattern match.
within	makes sure that at most *N* bytes are between pattern matches.
isdataat	verifies that the payload has data at a specific location, optionally looking if data relative to the end of the previous content match.
byte test	tests a byte field against a specific value (with operator i.e. less than (<), greater than (>), equal (=), not (!), bitwise AND (&), bitwise OR (ˆ) and various options such as value, offset, relative, endian, string, and number type). Capable of testing binary values or converting representative byte strings to their binary equivalent and testing them.
byte jump	allows rules to be written for length encoded protocols. By having an option that reads the length of the portion of data, then skips that far forward in the packet, rules can be written that skip over specific portions of length-encoded protocols and perform detection in very specific locations. Several options are supported such as byte to convert, offset, relative, multiplier <value>, big/little endian, string, HEX/DEC/OCT, align and from beginning
dsize	tests the packet payload size.

There are several different implementation platforms for DPI IDS, each having advantages and disadvantages. The first IDSs were built in GPPs, while, other commercial products implement mostly only parts of an IDS in fixed function/dedicated ASICs. Network processors can also be used for IDS offering some dedicated modules for network functions, while reconfigurable hardware may provide the increased flexibility that such systems require.

There is a tradeoff between performance and flexibility in these solutions. General-purpose microprocessors are very flexible, but do not have adequate performance. Network processors are less flexible but have slightly better performance. Reconfigurable hardware provides some flexibility and better performance.

Finally, dedicated ASICs are not flexible but can process packets at wire rates. This tradeoff is shown in Figure 3.4. Next we discuss each alternative in more detail.

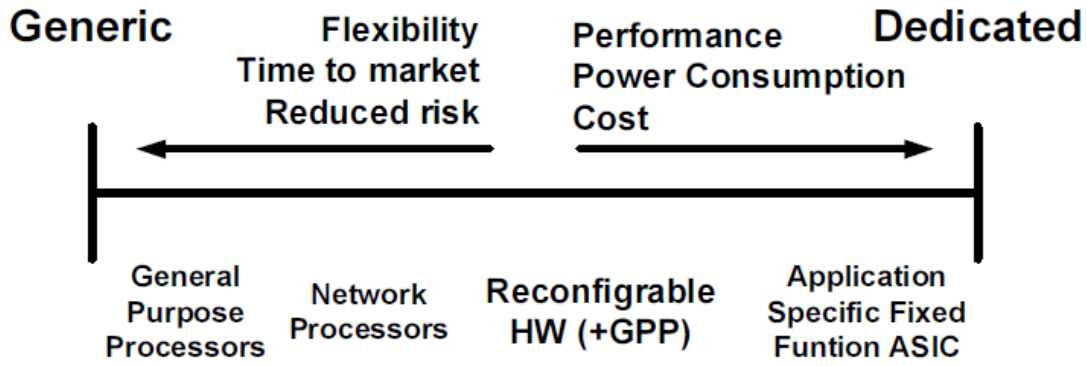

Figure 3.4. Performance-Flexibility tradeoff between different IDS implementation solutions

General purpose processors (GPPs) are used for their flexibility to adapt to IDS ruleset changes and their short time to complete the software development. An ID implemented for GPPs does not require running the code of every IDS rule for each packet. Based on packet classification a specific subset of rules may apply and can be called. This "on the fly" flexibility is another significant GPP advantage. On the other hand, GPPs fall short in performance and cannot process data at wire rates.

On the contrary, dedicated ASIC**s** are designed to process packets at wire rates, however are not flexible. Hardwired (custom) chips are difficult and expensive to modify, to add features, fix bugs or adapt to the rapidly changing IDS features. Moreover, ASICs require massive product volumes to amortize their high NRE cost (non-recurring expenses). In order to provide the required IDS flexibility and update IDS ruleset, ASICs are forced to follow memory-based designs where the contents of an IDS rule are compiled into memory contents that may indicate payload patterns or states of an FSM-like engine. This memory-based architecture restricts the design alternatives and limits performance. Systems' performance is restricted by the memory which, at best, may require a single access per operation and in other cases multiple accesses. Although ASICs are currently the fastest implementation platform for IDS, their performance is not as high as it could be expected compared to e.g., reconfigurable hardware. It can be presumed that reconfigurable platforms are about 5- 10× slower than ASICs in absolute operating frequency, since current FPGAs can operate at the order of 400-500 MHz, while ASICs at 2-4 GHz.

Network processors (NP) combine the GPP flexibility including one or multiple microprocessors and employ dedicated blocks for network functions such as packet classification and memory management in order to improve performance. The NP architectures can be also viewed as powerful GPPs or programmable engines combined with application-specific, fixed-function coprocessors.

Current NPs are not prepared for IDS processing and in particular content inspection. Such functions need to be processed in the NP microprocessor(s) and, therefore, inherit the GPP performance limitations. As a consequence, new content inspection coprocessors/modules need to be designed. They need to somehow provide flexibility in order to update the IDS rules. This can be achieved by either using fixed-function, memory-based modules (as in ASICs) or seek the required flexibility in a different technology and/or implementation platform.

Reconfigurable Technology may be the answer to the above flexibility. In this thesis, reconfigurable hardware is proposed as a solution for both IDS flexibility and performance. However, this does not imply that an intrusion detection system should be entirely built with reconfigurable logic. Several parts of the system can be fixed-function or reprogrammable (e.g., microprocessor, GPP, ALU) instead of reconfigurable. To explain the difference between reprogrammable and reconfigurable, a reconfigurable device can support directly in hardware *arbitrary* functions on demand, while a reprogrammable device can choose only between its *predefined* (and committed at fabrication), *finite* number of functions.

Reconfigurable hardware has the flexibility to update it functionality on demand and can support high performance. It achieves worse performance than an ASIC, yet not as much as expected. Furthermore, it is less flexible than GPPs (software), but still flexible enough to update the IDS rulesets.

The difference in flexibility between software and reconfigurable hardware lies in the speed of changing the functionality; not considering the time to develop the software program or hardware design. Currently, software can change its functionality substantially faster than hardware can be reconfigured. This permits

to dynamically call in software a different function per packet, while in reconfigurable hardware we can only change functionality per IDS ruleset.

That is, in software, based on the packet classification each packet may need only a (different) subset of rules to be checked, changing the executed routine (functionality) from packet to packet. Obviously the routines of all IDS rules need to be available in the memory hierarchy, however, only the necessary ones are executed. On the contrary, current reconfiguration times do not allow something similar in FPGAs. The hardware of every IDS rule needs to be "installed" in the device and process every packet. The available reconfigurable devices cannot be reconfigured for each incoming packet, they can however update (statically, before the IDS execution) the implemented rules whenever a new ruleset is released. In order to allow the software properties described above, reconfigurable technologies would require finer-grain reconfiguration area and higher reconfiguration speeds. FPGA technologies such as Xilinx allow partial (dynamic) reconfiguration of areas which may span the entire length of a device and a fraction of one column requiring a few msecs.

This is prohibitive for per-packet reconfiguration. It may be sufficient to update the reconfigurable parts of an IDS system (content inspection part, packet classification part, etc.) each time a new ruleset is released, however, this would require a fast design and implementation flow.

A new ruleset is released every few weeks and needs to be installed relatively fast. Consequently, it is inefficient to implement a new design manually each time. Automatic design generators would be more efficient to output a new design ready to be implemented and downloaded in the FPGA device.

Several solutions can be envisioned to speed up the implementation phase of a design, such as patches of additional rules installed via dynamic partial reconfiguration, incremental implementation flow, and implementation guide files.

Currently, the implementation phase of a complete design takes a few hours. However, the system is used to process only parts of incoming traffic

requiring a subsequent GPP to run Snort IDS and possibly being vulnerable to DoS attacks. SIFT puts together in a brute force way string matching and header matching without any attempt to reduce the overall processing load and optimize at the rule-level. Therefore, each packet needs to be processed against every IDS rule [8], [10].

3. Designing Traffic Analyzer Module (TAM) for DPI systems

Deep Packet Inspection (DPI) systems aim at analyzing, indentifying and classifying the traffic that is going through the network. Such systems accomplish these tasks by comparing messages within the packet's payload against strings, which represent signatures of applications, web attacks or some desired content to be scanned. Commonly, the used signatures rely their description on the use of RegEx. This has improved their expressiveness and can represent a set of signatures in a single string. This type of classification, with comparisons performed using RegEx, is often seen as the most reliable of currently available techniques for traffic classification. One strong reason for that is that currently, traffic classification cannot rely on information given by the port-application tuple, because of a variety of applications that use well-known ports (e.g. port 80) to pass through firewalls, e.g. P2P applications. Hence, port classification can lead to misleading results.

The DPI system described on this part, called TAM (acronym for Traffic Analyzer Module), uses RegExes signatures provided by L7-filter DPI system, non RegEx signatures from IPP2P project [12] and other signatures that were created by packet's payload inspection. Those signatures represent more than 60 applications falling into 9 different application's classes, which are listed at Table 3.3. Additionally, TAM was developed using the C language relying on the traditional libpcap library, for packet capture, and the C library *regex* for RegEx matching.

TAM has four modules, 1) the capture module, which is responsible for the online packet capture from an Ethernet device (Figure 3.5), 2) the aggregation module, which is responsible for aggregating the packets in flows (Figure 3.6), 3) the classification module, which compares the packet's payload (only packets that belongs to unclassified flows) with the RegExes signatures, aiming to classify that packet's flow (Figure 3.7) and 4) the cleanup module, which is responsible to go through every entry in the Hash Table and verify whether the flow has expired (Figure 3.8).

The Capture Module, presented in Figure 3.5, dequeues the IP packets from the capture buffer, checks the layer 4 protocol and forwards them to the Aggregation Module. Additionally, it waits until the other modules finish their operations to resume his activities, dequeuing the next packet.

Table 3.3. Recognized Classes and Applications

Class	**Application**
P2P	eDonkey
	BitTorrent
	KaZaA
	Gnutella
	SoulSeek
	Ares
	Mute
	EarthStation
	Xdcc
	Direct Connect
	Waste
	GoBoogy
	Soribada

	WinMX Napster MP2P
Web	HTTP
Chat	AIM Yahoo Messenger MSN Messenger IRC

Continuation

Network Management	DNS NetBios NBDS NBNS BootStrap
Streaming	Video Over HTTP RTSP Video Over HTTP (QuickTime) Audio Over HTTP
E-mail	IMAP POP3 SMTP
Data	FTP MySQL
VoIP	Skype
Secure	SSH

	SSL

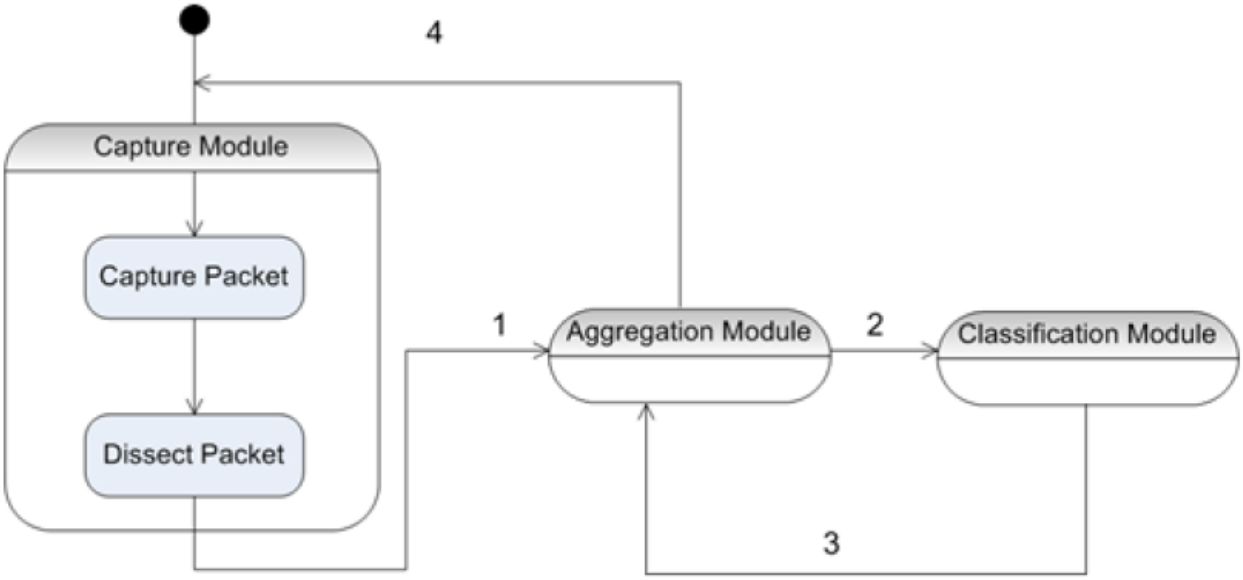

Figure 3.5. TAM's Capture Module

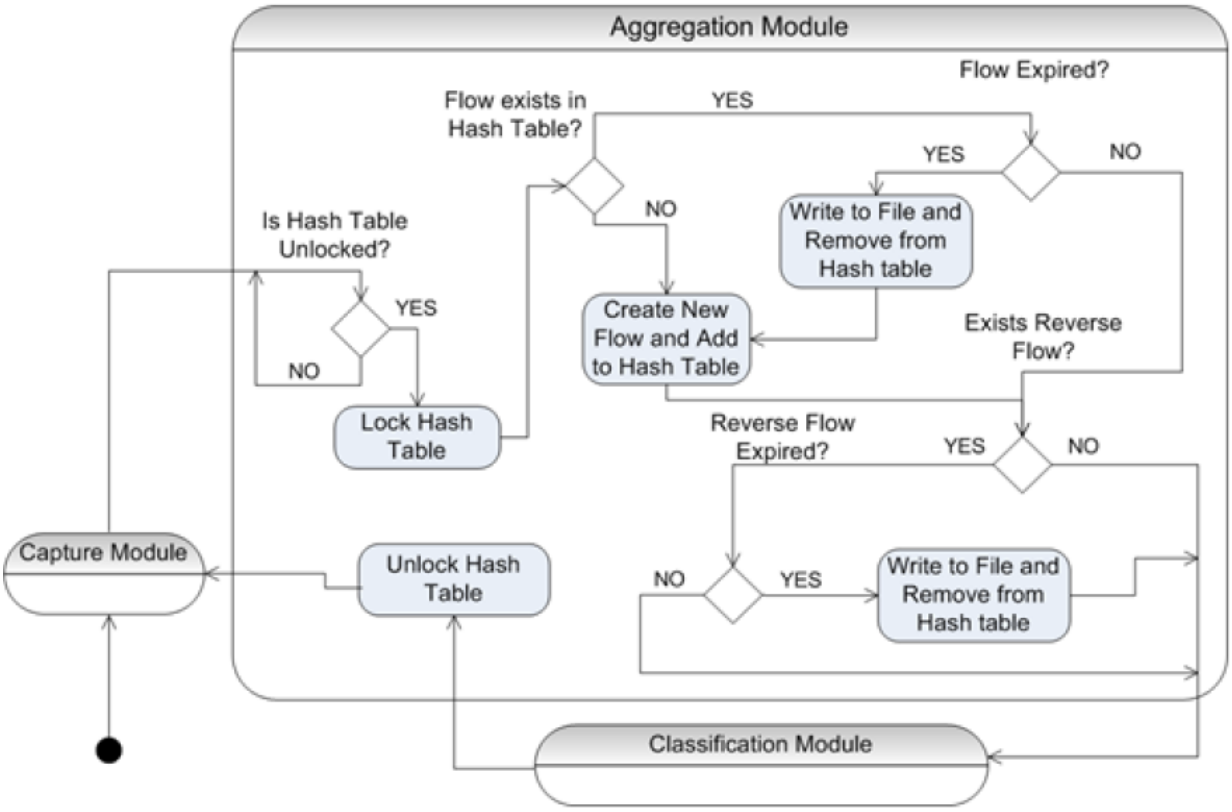

Figure 3.6. TAM's Aggregation Module

As seen at Figure 3.6, the Aggregation module synchronizes the Hash Table with the analysis thread, by the locking operation. Then it checks whether the flow that represents the received packet exists in the Hash Table, creating a new one if it does not exist. If the flow exists and has expired, the module writes it to an output file and creates a new flow with the incoming packet. After that, the flow's reverse flow is sought in the Hash Table and the same

expiration check is performed. Finally, those information, flow and reverse flow, are forwarded to the Classification Module.

The Classification Module, depicted at Figure 3.7, checks whether the flow is already classified. If its reverse flow exists and it was previously classified, then the flow that is being analyzed receives that classification, if it is unclassified. However, if the flow already has a classification, it only updates its information (byte volume, timestamp and number of frames). When the Classification Module finishes its execution, it returns to the Aggregation Module to unlock the Hash Table.

Finally, the Cleanup Module, depicted at Figure 3.8, synchronizes the Hash Table by the locking operation and checks each flow. If the flow has expired, it is removed from the Hash Table and written to an output file. Otherwise, the Cleanup Module retrieves the next flow from the Hash Table.

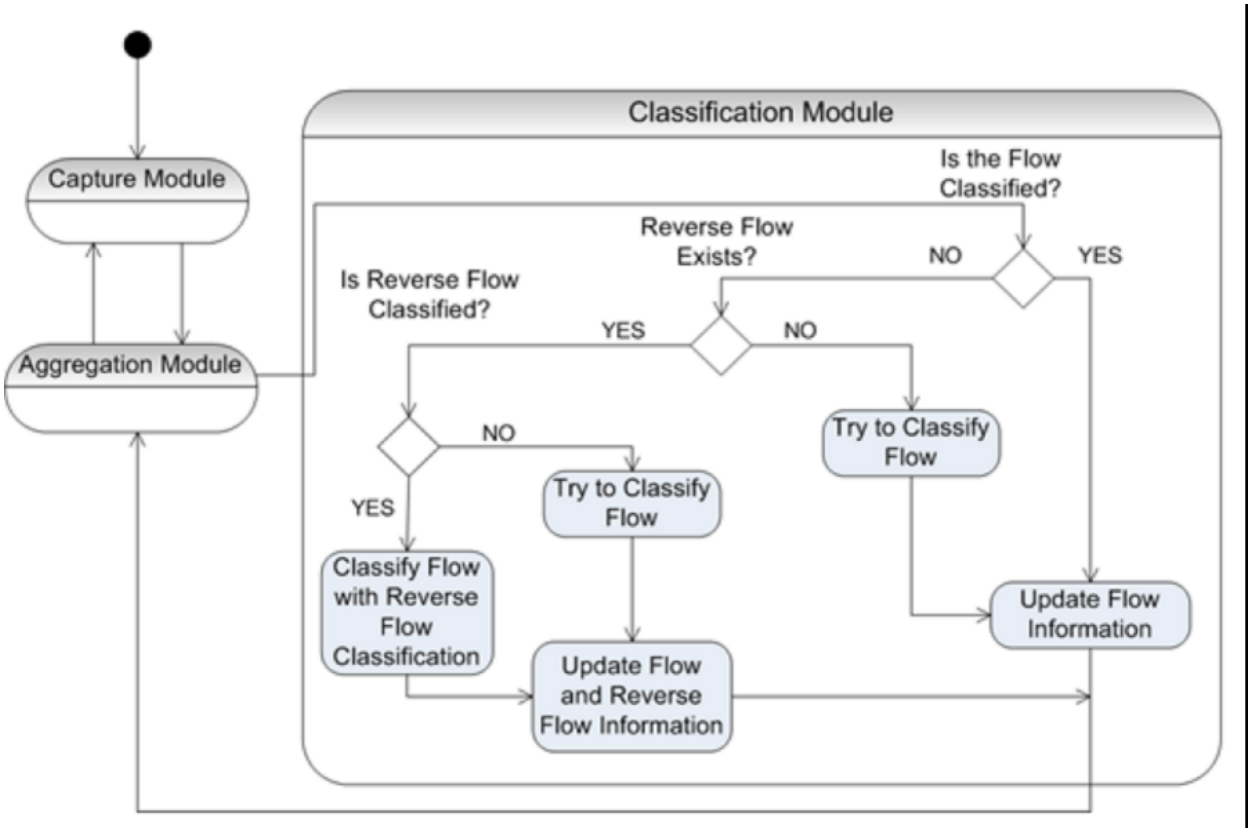

Figure 3.7. TAM's Classification Module

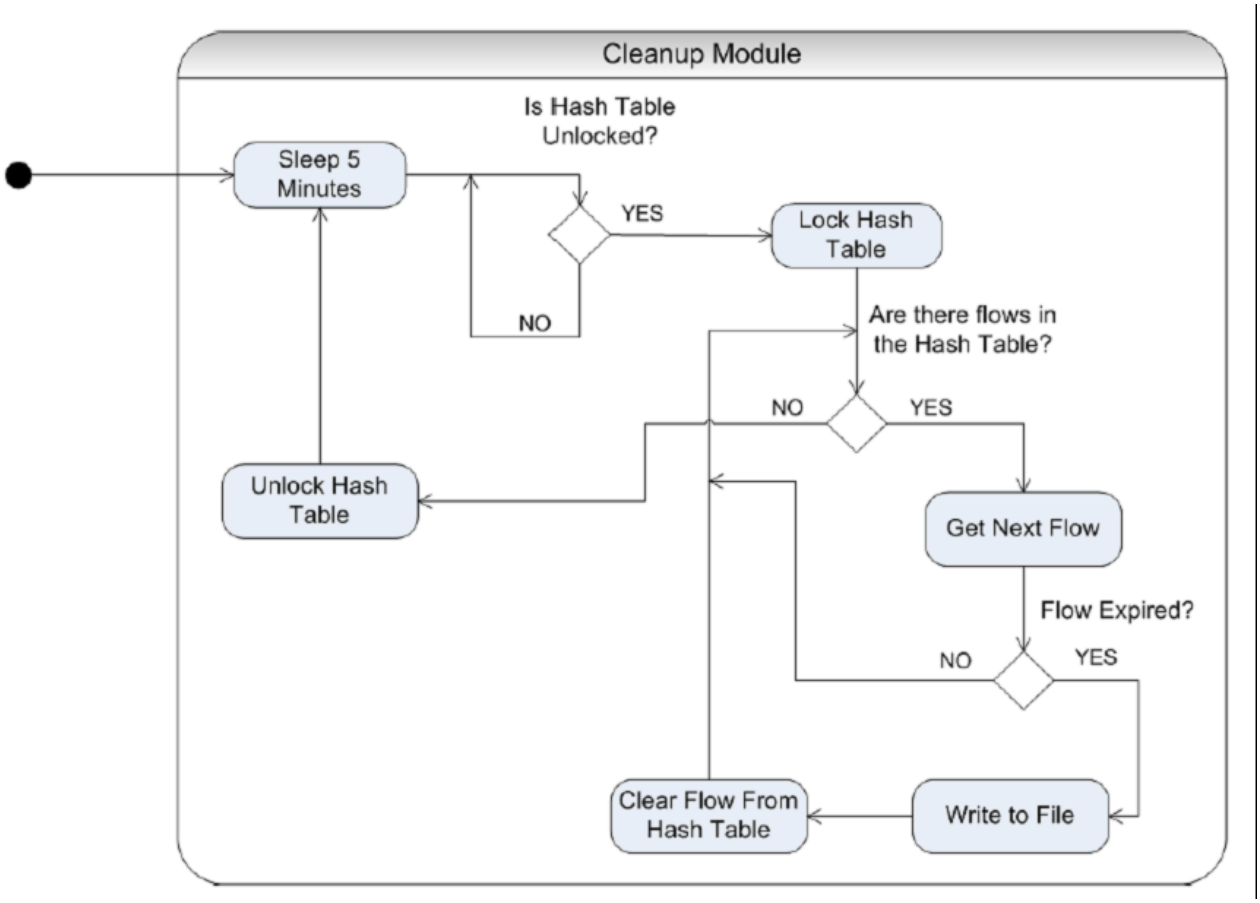

Figure 3.8. TAM's Cleanup Module

The main objective of TAM is to capture all the packets that are traversing the network, classify those packets on the fly, aggregating them in flows with the lowest packet loss rate. Initially, TAM was developed to run in a 34Mbit/s backbone, at the Point of Presence of Pernambuco (PoP-PE), linked with the Point of Presence of Rio de Janeiro (PoP-RJ), working with no loss of packets at this speed. But TAM's requirements have changed; now it has to run at high

performance networks, reaching 1Gbit/s of speed. Now, TAM has to capture, aggregate and classify packets losing the less number of packets possible without decreases in classification completeness.

Those Flows, constantly mentioned in this thesis, consist of a unidirectional series of IP packets with the same source and destination addresses, port numbers and layer 3 protocol (this set of information is commonly called the flow's 5-tuple). Additionally, the aggregation in the format of flows can group all relevant information about a set of packets in a unique registry and, consequently, reduces the storage space occupied by the traffic related information.

In order to reduce memory consumption by the identified flows, TAM establishes a timeout of 64 seconds to write out the expired flows. So if a flow does not receive any packets in an interval of 64 seconds, its entry will be deleted from the hash table and it will be written to an output file. Therefore, there is a thread in TAM that is triggered in intervals of 5 minutes and it scans the whole hash table, searching for expired flows. This thread must be synchronized with the analysis thread, to handle some problems that are going to be described later on this section.

The present system faces some problems, which cause poor packet capture performance. One of the main problems is the packet capture model performed by libpcap. This poor performance can be explained by the time spent with unnecessary packet copies, performed by Linux-based systems, from kernel space to user space. Another problem is the general purpose Hash Table used by this version, which resolves collisions by chaining, i.e. each Hash Table bucket has a single linked list, so objects that have the same key are inserted on the list. Such Hash Table implementation has some bottlenecks, which were identified:

With a single linked list, every removal operation, of an element X, has to perform a search operation in the table, in order to get the element that is placed before X in the list and makes it "next" pointer point to the element after X.

Particularly in this Hash Table, when an insertion is going to be performed, it first checks whether the element, which is going to be inserted, exists in the table. Such check is unnecessary, since the element is searched at the Hash Table before its insertion, so the information about its existence in the Hash Table is known beforehand.

Another problem, which really compromises the performance of TAM, is the time that the system will stay without analyzing any packets, because of the Hash Table cleanup thread. This occur because the cleanup operation has to be synchronized with the main analysis thread, since the cleanup module can be removing an expired flow from the hash table at the same time that the analysis thread is accessing such flow. Even this module resulting on some packet losses, this cleanup operation is really necessary because of some reasons:

Memory consumption monitoring is an essential task in real time systems (so is in TAM) and therefore flushing out expired flows to an output file is very important, in order to avoid memory leaks.

Performing a periodically Hash Table cleaning can lead to a shorter search time, since there will be less flows in the linked lists, leading to reduced flow searching time.

4. Analysis of Multicore approach to Deep packet inspection. Parallelization of deep packet inspection processes and algorithms

Nowadays network security applications such as intrusion detection systems (IDSs), firewalls, anti-virus/spyware systems, anti-spam systems, and security visualization applications are being in this communication sphere.

These applications all heavily rely on deep packet inspection, which is to examine the content of each network packet's payload. Today these security applications cannot cope with the speed of broadband Internet that has already been deployed, that is, the processor power is much slower than the bandwidth

power. Recently the development of multi-core processors brings more processing power. Multi-core processors represent a major evolution in computing hardware technology. While five years ago most network processors and personal computer microprocessors had single core configuration, the majority of the current microprocessors contain dual or quad cores and the number of cores on die is expected to grow exponentially over time. The purpose of this thesis is to discuss the research on using multi-core technologies to parallelize deep packet inspection algorithms and how such an approach will improve the performance of deep packet inspection applications [16].

Multi-core provides a network security application with more processing power from the hardware perspective. However, there are still significant software design challenges that must be overcome. Today the difficulty is not in building multi-core hardware, but programming it in a way that lets applications benefit from the continued growth in CPU performance [17]. From the server or router side, if the network security software is not fast enough, it can be very difficult to process every incoming packet then it would slow down the traffic. From the client side, it can also be very difficult to run network security applications without any interruption to normal applications because those computing intensive applications significantly slow down other simultaneously running applications. Taking advantage of the full power of multi-core processor requires an in-depth approach to realize the speedups by parallelizing the traditional deep packet inspection applications. In The idea of using multi-core processors to enhance the performance of network security applications is promising. However, the research in this area is just emerging and thus requires intensive exploration.

The development of multi-core processors has a significant impact on software applications. To take advantage of multi-core, software requires migration to a multi-threaded software model and necessitates incremental validation and performance tuning. Although kernel or system threads managed

by the operating system can enhance the application performance, it is essential to have multiple user threads maintained by programmers to improve the performance of traditional applications.

Most of current research is still focused on automatically mapping general-purpose applications onto multi-core systems with instruction, data, or thread level parallelization techniques [18] or relying on virtualization technologies such as VMware (VMware). Most of them are essentially extensions of utilizing shared-memory multiprocessors and can only execute coarse-grained threads. Network security applications have their own unique behavioral characteristics such as frequent memory or disk access, complex data structures, and high bandwidth and high speed requirements. There is a distinct mismatch between current Multicore hardware development and high performance demand from network security applications. There has been very little preliminary research done in this area [19]. In short, there is an imperative need for specifically re-designing network security applications from a software perspective based on multi-core hardware architecture.

Deep packet inspection refers to the process of checking packet payload and header in a network device. The applications of deep packet inspection include, for example, network security applications that filter out packets containing certain malicious Internet worms or computer viruses; content-based billing systems that analyze media files and bill the receiver based on the material transferred over the network; and content forwarding applications that look at the hypertext transport protocol headers and distribute the requests among the servers for load balancing [15]. Contrastingly, shallow packet inspection refers to the process of only checking packet header in a network device. Deep packet inspection requires much more processing power than shallow packet inspection.

Most deep packet inspection applications have a common requirement for string matching. For example, the presence of a string of certain byte sequences

in packet payloads can identify the presence of a virus, such as the well-known Internet worms Nimda, Code Red, and Slammer. One requirement of deep packet inspection applications is that the applications must be able to detect strings of different lengths starting at arbitrary locations in the packet payload because the location of such strings in the packet payload and their length is normally unknown. The other requirement of deep packet inspection applications is that they must be able to process network packets at line speed, otherwise it will causes the delay of network traffic, or the incompleteness of deep packet inspection.

The idea of using multi-core processors to enhance the performance of network security applications is promising. However, the research in this area is just emerging and thus requires intensive exploration.

It faces many challenges such as:

• How can we actually use multi-core to continue running the network security applications while keeping the overall system performance?

• How can we efficiently partition and distribute the workload of network security applications between the different cores in the multi-core processor?

•How can we split network data and solve the data dependency problem?

• As multi-core uses shared off-chip memory, how can we smartly utilize the memory then it will bring less memory access latencies?

• How can we synchronize and coordinate different threads of the applications when it is parallelized on multi-core?

To best solve these challenges, scientists propose the new system architecture as in figure 3.9.

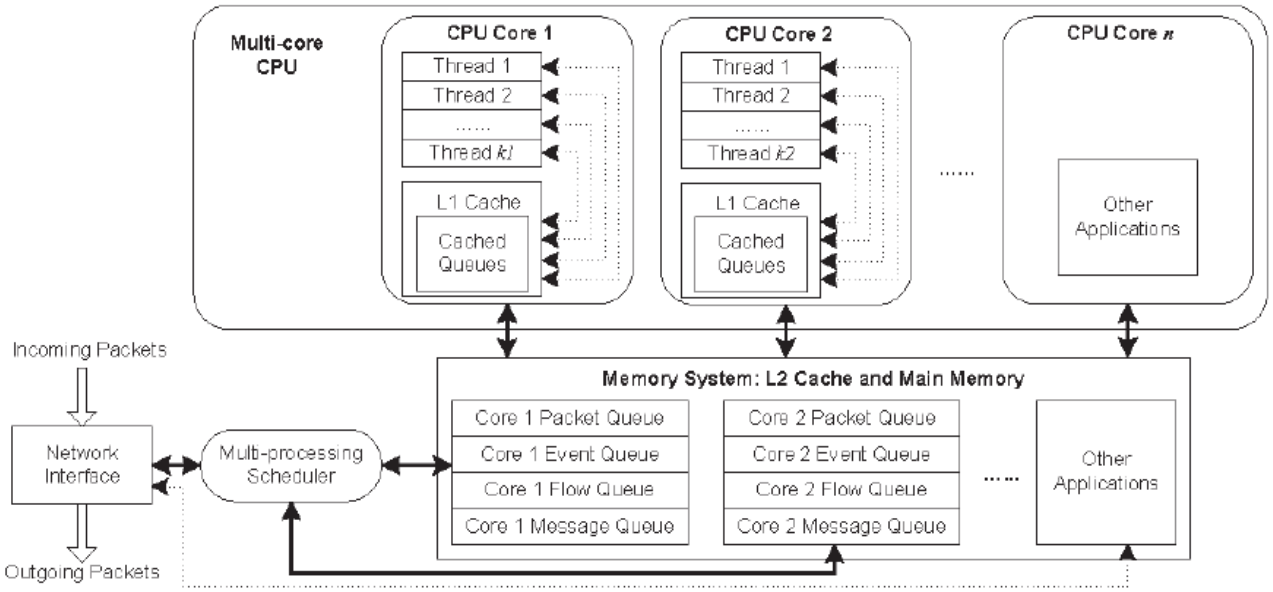

Figure 3.9. System architecture of using multi-core processors to support deep packet inspection

The essential ability of this architecture is that it can process network packets in parallel and thus meet the real-time requirement. As is shown in the figure, the multi-processing scheduler coordinates and distributes the workload to different cores. The information from packets, events, flows, and messages are processed in the multi-core processor in parallel. The processor has spare cores to run other applications.

As is illustrated in figure 3.9, the proposed architecture must be able to process network packets at line speed. In another words, it must keep its performance in normal applications, such as forwarding packets in a router. To fully utilize the potential of multi-core, this system architecture will use different level of parallelization such as instruction-level parallelization, memory parallelization, loop-level parallelization, and fine-grained thread-level

parallelization. High performance can be achieved through interaction between algorithms, strategies, and architectural design, from high-level decisions on data allocation and task partitioning to low-level micro-architectural decisions on instruction selection and scheduling. For each network security application, we also need to identify what the potential bottlenecks are and how to possibly avoid them. The potential bottlenecks could be packet processing, data normalization, data correlation, pattern generation, and pattern matching in such a parallel computing environment.

As the instance of the aforementioned multi-core based deep packet inspection architecture, we test two levels of parallelization on multi-core: packet level and flow-level parallelization.

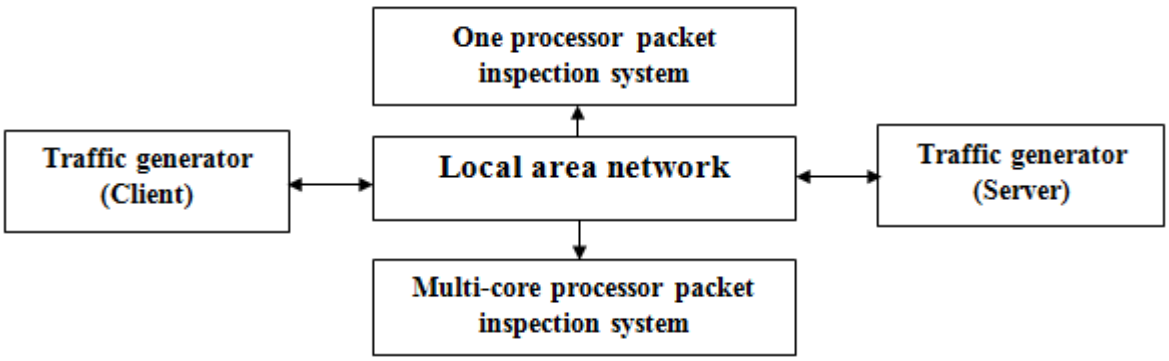

Figure 3.10. Experiment environment

As current network traffic speeds and volume are increasing at an exponential rate, the processing speed required by a deep packet inspection application is high. In this experiment, we evaluate the performance of multi-core parallel deep packet inspection methods based on Snort, an open source intrusion detection system. The test environment is shown in Figure 3.10. To simulate the large volumes of network traffic, network traffic generator is used to test the capability of the intrusion detection system Snort to handle

continuous high traffic loads. We use TG2.0 [14] to generate high volumes of network traffic. A core-2 duo computer with a 2.2GHz Intel Pentium processor and 1GB RAM is used in the experiment. The network adaptors used are 10/100M PCI Ethernet adapters. We first use the packet-level parallelization to evaluate the multi-core supported intrusion detection system's performance on deep packet inspection. The TG's client is set to open a UDP socket to send packets to the TG server waiting at 192.168.20.1 and port 1000, with packet data length being 52 and packet number is 1500. On the multi-core system, the odd number packets are processed by one core and the even number packets are processed by another core. The detailed procedure is specified in the basic algorithm as follows.

Algorithm 1. Algorithm of packet-level parallelization

```
n: the number of parallel threads;
int i=0; // all threats start
while(true)
{
// receive packet from network interface;
I++;
if (i>n)
i = 1;
// sent the packet to the i-th thread;
}
```

The testing results under different inter-packet transmission time are shown in Figure 3.11, and the dropping rate (the percentage of dropped packets in all packets) is shown in Figure 3.12. Because the system needs to capture the incoming packets from the network adapter and analyze these packets for possible attacks, depending on the packet capturing and analyzing speed (processing speed), and the speed of incoming packets (network speed), the system may be able to process all incoming packets, or have to drop some

packets. If the processing speed is slower than the network speed (this may happen when the system is under heavy load), the system may drop some packets thus may lose some useful information, which will increase false positive and false negative rate.

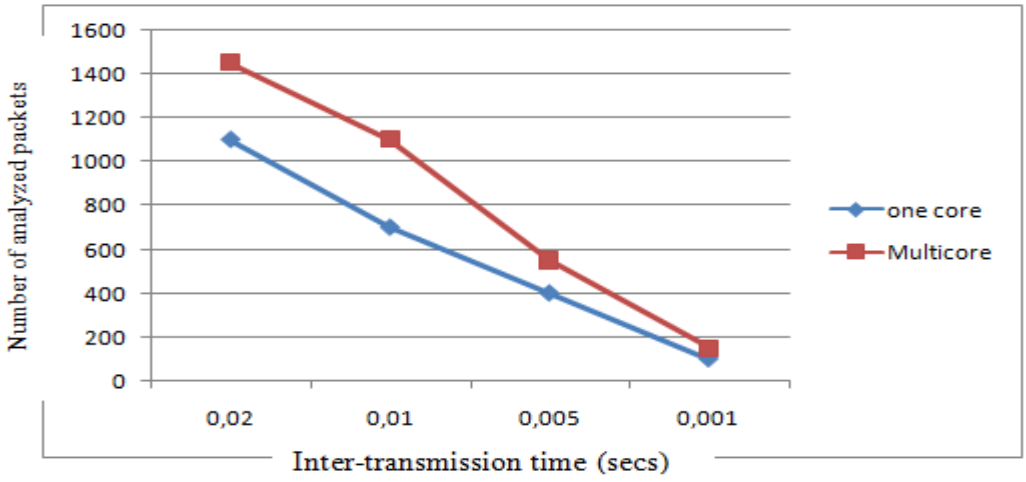

Figure 3.11. The number of packet analyzed on packet-level parallelization

The figure shows the trend that if the system is parallelized at packet-level by using more than one core, the dropping rate can be slightly decreased. The number of analyzed packets thus can also be slightly increased. This proves that multi-core can increase the processing speed of the whole system.

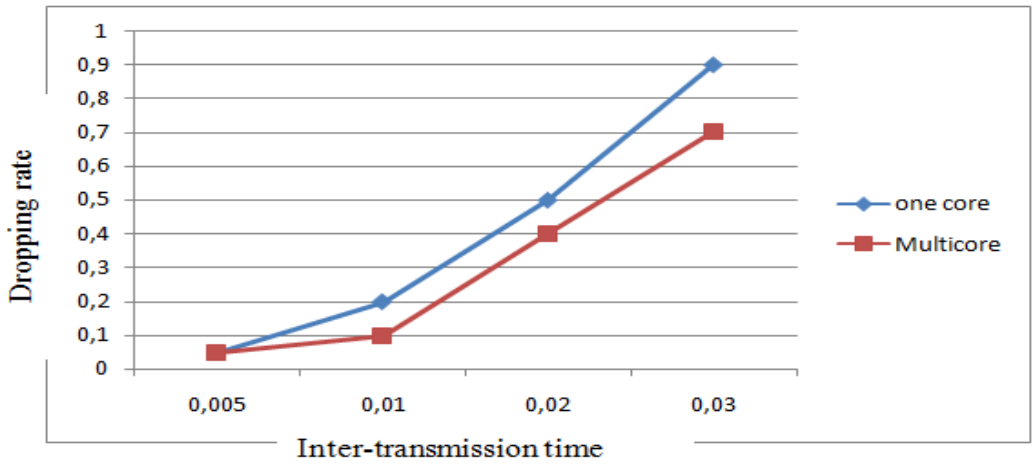

Figure 3.12. Dropping rate on packet-level parallelization

For packet-level parallelization to be practical, no resource should actually share common information across separate packets. Although deep packet inspection applications like Snort receive input packet by packet, they must aggregate distinct packets into flows such as TCP streams to prevent an attacker from disguising malicious communications by breaking the data up across several packets. Since packets from one flow will not affect the states of another flow, different flows can be processed by independent processing threads with no constraints on ordering. To test the flow-level parallelization performance on multi-core, I conduct another experiment based on flow-level parallelization.

The experiment proves that the parallel strategies can improve the deep packet inspection performance. From the comparison between packet-level parallelization and flow-level parallelization I also find that flow-based parallelization has similar performance speedups as packet-level parallelization. However, it is a practical solution to meaningfully examine each packet in real deep packet inspection applications.

From the above evaluation results we find there are many benefits to use multi-core to support deep packet inspection applications. I emphasis the benefits of using multi-core supported system architecture in deep packet inspection applications as high performance, comprehensive, intelligent, and scalable [16].

Conclusion for chapter III

In conclusion, “DPI” is already being implemented to provide a range of services from content filtering, to web application firewalls, application aware policy enforcement, and others. DPI applications solutions are a mix of compute and packet processing, real-time and non-real time processing, as well as in-the-path/ on-the-wire and off-the-wire/off-the-path processing depending on the DPI application. As advances in computing and packet processing occur, the ability to inspect and act on more traffic to enable a variety of useful services—content-based traffic prioritization, network hosted intrusion and malware prevention systems, dynamic content filtering, and web application firewalls to address increasing mixtures of threats in “web 2.0” spanning image, text, script, java, flash, and other executable objects—is enabled within the network. Faster availability of subscriber analytics enables better planning as well as can enhance a service provider’s ability to participate in advertising revenues. Network analytics enable the service provider to understand the performance of the network and react sooner and more effectively to true capacity and equipment limitations.

With improvements in compute and packet processing performance in the same power/form factor occurs, more applications can be made co-resident enabling service providers to offer more services on the same infrastructure with better cost, reliability, and performance than through deployment of external solutions. External solutions though do have a role, and the combining

or co-residency of a DPI application with a core function must consider other factors, such as the need for frequent updates required by some DPI applications to stay current whereas the core function may be very stable and require only routine patching and upgrading, some DPI applications may require updates several times a day whereas core network functions may go months between upgrades or patches.

It has been indicated that IDS is currently the most efficient solution for network security providing content-aware processing. In this chapter, we described the IDS tasks and discussed the most challenging issues in IDS performance and implementation. The core of IDS is the detection engine which uses a large ruleset of attack descriptions. The main functions are header matching and payload matching (content inspection). Profiling the popular open source Snort IDS shows that payload matching is the most computationally intensive IDS task requiring 40-80% of the total execution time.

The rapid growth of IDS rulesets and especially their payload content descriptions indicate the increase of IDS processing requirements and reveal the need for scalable IDS solutions in terms of performance and implementation cost. I discussed alternative IDS implementation platforms, analyzing their advantages and disadvantages and presenting their flexibility-performance tradeoff. We advocate the use of reconfigurable hardware for the solutions provided in this thesis regarding the IDS detection engine and content inspection tasks.

In conclusion, I present new multi-core supported deep packet inspection architecture and an instance of the architecture. Leveraging the power of multi-core processors can be the answer to many yet-unsolved but crucial challenges in deep packet inspection applications such as isolated security environment, real-time attack detection and attack packets filtering, and real-time visualization of network monitoring. It enables sophisticated and stateful network processing rich in semantics and context as a routine capability

provided by a network's routers. Multi-core supported deep packet inspection applications can provide comprehensive protection against different threats. It will provide significant benefits to the security of future networks and systems.

Chapter IV. Development of security solution for IP-based packet networks by using DPI

1. The architecture of DPI systems recommended by ITU-T

The Recommendation "ITU-T Y.2770. Requirements for Deep Packet Inspection in Next Generation Networks" primarily specified the requirements for Deep Packet Inspection (DPI) entities in NGN, addressing, in particular, aspects such as application identification, flow identification, inspected traffic types, signature management, reporting to the network management system (NMS) and interaction with the policy decision functional entity.

This Recommendation also identifies the requirements for DPI of traffic in non-native encoding formats (e.g., encrypted traffic, compressed data, and transcoded information).

In packet-based networks, it is imperative to identify different kinds of services and apply different control mechanisms to provide differentiated services for its subscribers. As a control point in packet forwarding, DPI is often deployed in the following application scenarios, as illustrated in the figure.

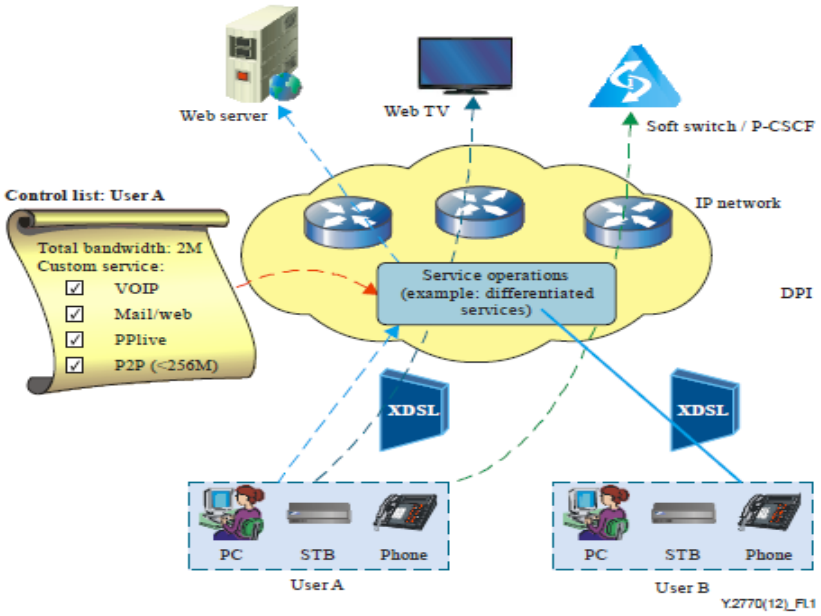

Figure 4.1. Scenario of differentiated services at the example of an IP-based packet network with DPI support

As depicted in Figure 4.1, user A, whose available bandwidth is, e.g., 2 Mbit/s has different network needs including services like VoIP, mail/web, on-line video (e.g., PPlive), peer to peer services and so on. When the user's application traffic flows pass through the DPI device in the network, the DPI may implement differentiated services in accordance with the predefined control list (i.e., the policy rules table for DPI in the packet forwarding path) and the identification results.

DPI may be deployed to provide the capabilities to identify malicious traffic that may degrade user performance, drain network resources, impair infrastructure, and finally make the network unavailable to its subscribers. Most of the malicious traffic disguises itself as normal traffic and is extremely bandwidth consuming, such as: Outgoing spam, IP scanning and port scanning,

etc. Figure 4.2 shows a typical application scenario that when malicious traffic is identified, it will be removed by the DPI component from the traffic thus preventing it from spreading into the network.

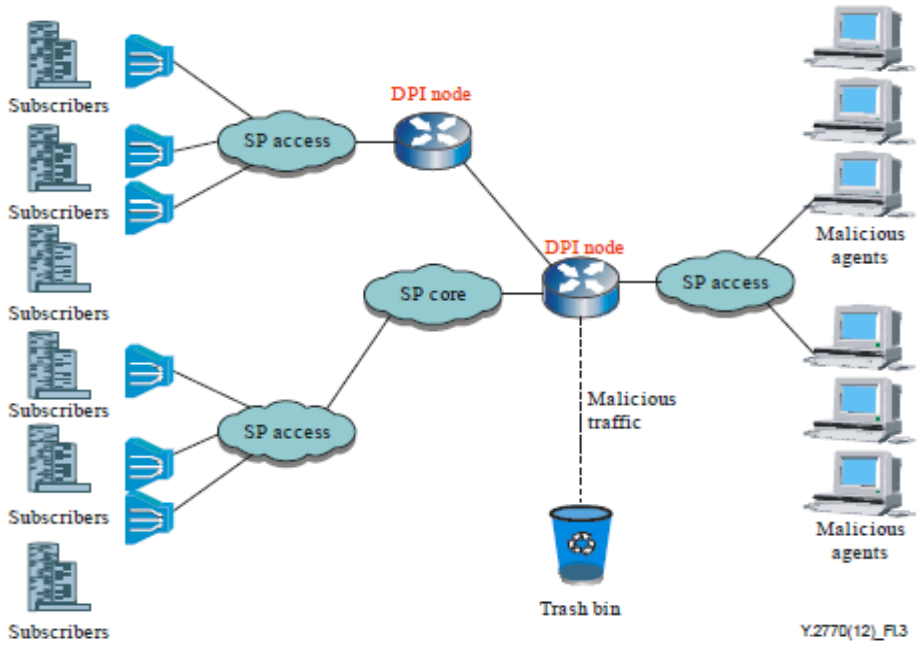

Figure 4.2. DPI deployed to filter out malicious traffic

Therefore, DPI in such kinds of application scenarios provides the packet forwarding process with the following capabilities:

1. On-line monitoring, tracking and analyzing possible connections;

2. Real-time identification of malicious attacks. Through inspecting deep into the traffic, DPI alerts the network administrators of possible attacks and providing a range of monitoring and detection tools to track attack launchers, applications, flows, connections, ports, protocols, trends and other parameters. Meanwhile, some of the attack patterns if possible, should also be feed backed to the DPI signature library to make resources unavailable to the attackers;

3. Real-time reporting of possible attacks. Through an automated and flexible early-warning mechanism DPI is applied to inform network administrators of potential threats in advance, enabling them to take appropriate actions against possible attacks. Thus, DPI may be fundamentally a basic function of an intrusion detection system (IDS).Mitigation of threats through DPI policy rules enforcement. In such scenario DPI is deployed to yield productive measures against possible attacks. Under such circumstances, mitigation of threats would be implemented to avoid the infiltration of the malicious traffic into the network, as it might make the network more vulnerable and even collapse from resource exhaustion [1].

This Recommendation defines the following main terms that connected with security:

Deep Packet Inspection (DPI): a method of packet filtering that functions at Layer 2-7 (Open Systems Interconnection) reference model [2]. The use of DPI makes it possible to find, identify, classify, or control packets with specific data or code payloads that conventional packet filtering, which examines only packet headers, cannot detect.

DPI node: a network node device with DPI function. Note: This network node device could be a router, switch, bridge, border gateway, or any other kind of access devices, telecom transmission device or system, etc.

DPI engine: a DPI processing element including a scan function, analyzer, rule action, rules table.

A rule entry: one of a set of rules (or N-octet string) which are statically predefined or dynamically generated and used to compare the particular overhead or contents octets of the real-time packet flows with a set of the rules (strings) to determine if the string matching is successful or not.

A DPI analyzer: a functional implementation to perform comparison functions between the particular overhead or contents octets of the

real-time packet flows and a set of the rules (strings) to determine what the results are.

A rule action: an action after analyzing according to the related rule actions, which include at least the following:

• Traffic classification, measurement, and reporting and management;

• Resource management, admission control and filtering;

• Policy-based prioritization, blocking, shaping and scheduling

• Dynamic rule building and modification.

Rules table: a database including the multiple rule entries. These rules are defined and classified at the different levels (layer2-layer7), and the different functions at the same layer to meet the carrier class interworking requirements.

The basic DPI function and architecture are presented in Figure 4.3. As the packet passes through the pipeline, the extracted ingress packets from incoming link enter the input queue and are scanned in different window lengths for signatures of different lengths by DPI engine, which is a real-time processing functionality and ability of traffic management based on the particular overhead or content of packet. DPI operates with content awareness on information contained at all layers of the protocol stacks, including the application layer to satisfy new emerging requirements. The DPI engine illustrated in Figure 4.3 contains a scan function, analyzer, rule action, and rules table. A DPI analyzer performs comparison function between the particular overhead or contents octets of the real-time packet flows and a set of the predefined rules (strings) to determine what the result is.

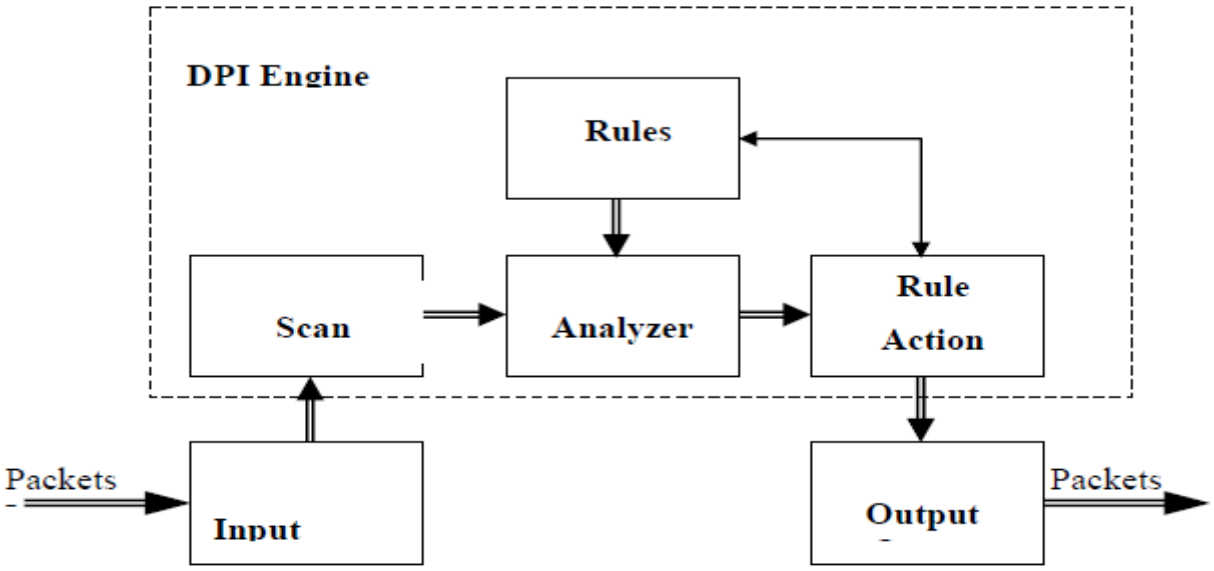

Figure 4.3. The basic DPI function and architecture

In this point from architecture, I must define that rule action is the most critical and reflects the enforcement of the related rule requirements, this action includes at least the following:

•Dynamic rule building and modification;

•Traffic classification, measurement, and reporting and management;

•Resource management, admission control and filtering;

•Policy-based prioritization, blocking, shaping and scheduling.

The important characteristics of DPI are a direct traffic management in the real-time. As an example, if a rule action detects a match occurred, the related content can be blocked and an alert message is generated, or the related content can be modified and a schedule or action is produced. Rules table in the DPI engine presents a table database including the multiple rule members. These

rules are defined and classified as the different levels (including L2, L3, L4, L5, L6, L7 and content), and the different functions at the same layer to meet the carrier class interworking requirements.

Data leaves the content pipeline after the DPI engine processing, flows to the output queue, and then packets are re-injected into the network.

Figure 4.4 illustrates the bi-directional DPI function and architecture. The Mediation Unit is targeted at a mediation function unit of performing the bi-directional DPI. This unit interfaces to the bi-directional rules table, rule action and analyzer to align and deal with the end-to-end pertinences between outgoing stream and the relevant incoming stream.

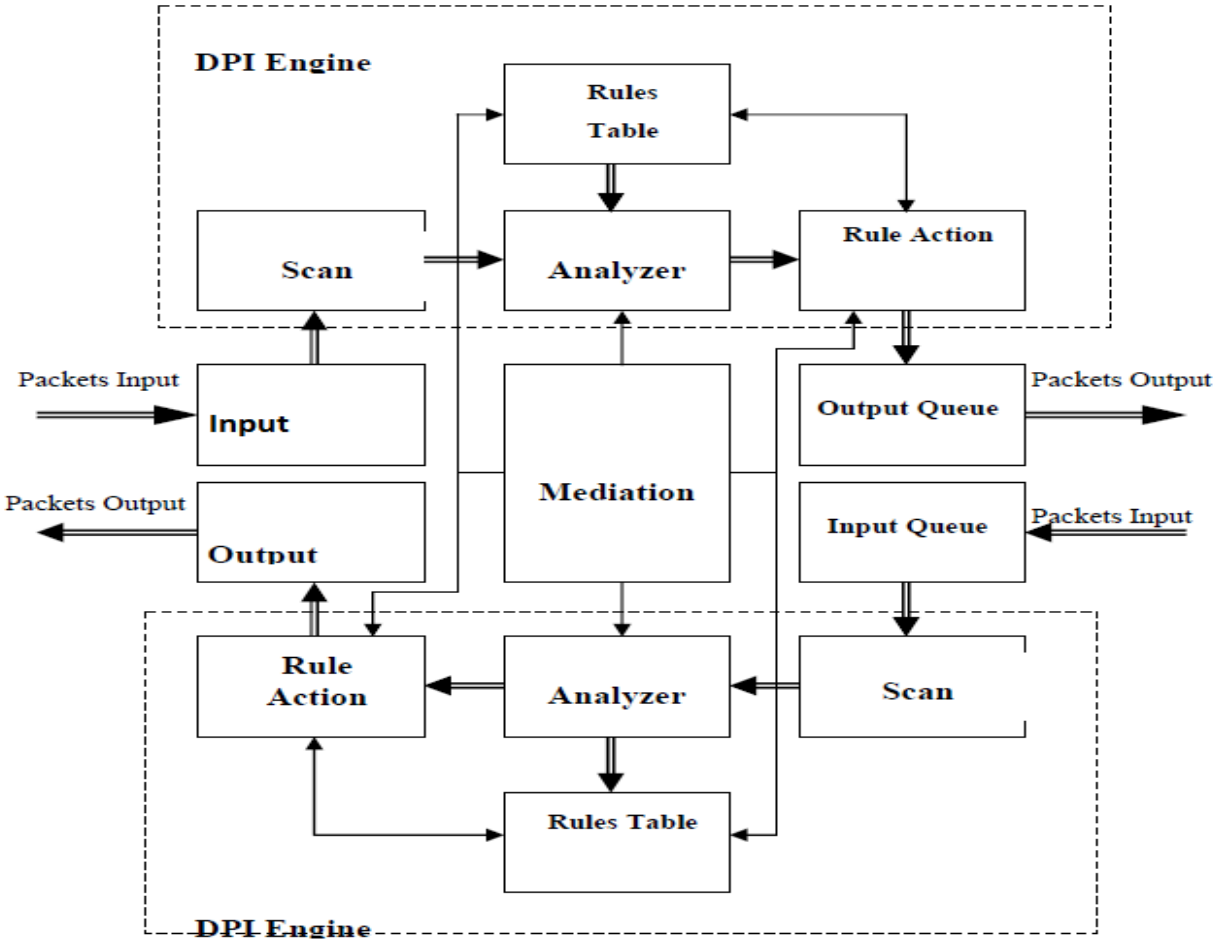

Figure 4.4. The bi-directional DPI function and architecture

DPI offers network operators complete visibility of network applications, flexible traffic control through the real-time comparison and string matching between the particular overhead or contents octets of the packet flows and a set of the octets predefined rules. The string matching is one of the most important functions in applications such as IP address lookup in routers. In a DPI node, it

allows for the node to scan into both the headers and the actual content flowing.

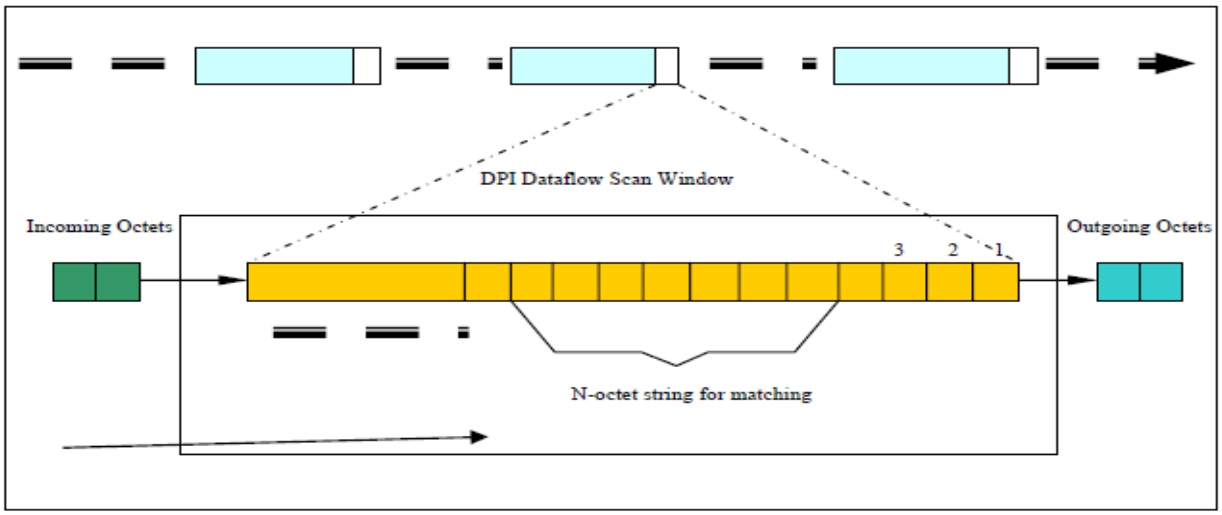

Figure 4.5. DPI Dataflow Window

Figure 4.5 illustrates the DPI Dataflow Window for matching the rule (N-octet string). If a string is identified as a rule member of DPI function, the system can declare the string as a matching signature. Such strings are then sent to a DPI analyzer to determine if the string matching is successful or not. The DPI analyzer uses a particular string-matching algorithm and gives a query result per clock cycle. The tens of thousands of strings can be scanned and analyzed at gigabit per second rates.

The Recommendation identifies the security threats to NGN and defines the requirements for protection against the threats. Since the DPI-related entities are a part of NGN, the conclusions of [ITU-T Y.2701] are applicable to them.

Based on [ITU-T Y.2701] the security threats related to the DPI entities are identified as follows:

- Destruction of DPI-related information;
- Corruption or modification of DPI-related information;
- Theft, removal or loss of DPI-related information;
- Disclosure of DPI-related information;
- Interruption of services.

The information pertaining to the DPI operations include DPI policy rules with their signatures and DPI exported flow and application information. Destruction, corruption or modification, theft, removal or loss of such information may make it unusable for the DPI operations. In many countries, such information is recommended to be treated according to the national regulatory and policy requirements and must not be disclosed. Interruption of services may be result of the DoS attacks. Any entity receiving data can be a target of DoS attack. For example, an attacker can indirectly flood a DPI entity with large volume of traffic causing degradation or interruption of the DPI services for the legitimate users [1].

2. Development of Linux- based deep packet inspection systems

The GNU/Linux family of operating systems includes a packet filtering framework known as netfilter. Netfilter comprises a packet-filtering firewall along with a number of extensions and utilities [4]. One of the most important parts of netfilter is iptables, a command-line utility for viewing, adding, removing, and modifying firewall rules. These rules are applied by the kernel itself to evaluate network traffic. Like other rule-based firewall solutions, iptables allows the construction of sophisticated rule sets based on packets' source and destination addresses, source and destination ports, and other header

information. Furthermore, the set of rules in effect can be dynamically modified, either by the system operator or by automated scripts. Thus, iptables can be used to respond to changing conditions and requirements without rebooting the host or otherwise interrupting the flow of network traffic.

Many commercial vendors, including Microsoft, Cisco, and Symantec, have begun to develop deep filtering tools. These "deep filtering" applications require greater functionality than that supplied by iptables and other traditional packet filters.

Iptables is highly versatile, as rules can be written for any network protocol supported by Linux. Furthermore, it is application-agnostic; the envelope information for traffic following a given protocol is all formatted similarly. Iptables does not examine the actual packet data (the "payload"). There are times, however, when it would be useful to control the flow of packets based on their contents. In these cases, the agnosticism of iptables becomes a weakness.

Some or all of the network traffic arriving at a host may be subjected to deep filtering. Choosing which packets to filter deeply remains the task of conventional rule sets set up and maintained with iptables. In addition to accepting or rejecting packets that meet specified header-based criteria, netfilter allows the kernel to defer certain packets—those that match the appropriate rule(s)—for further processing. Packets deferred in this manner are "queued" until they can be undergoing deep processing [4].

This extra processing is not performed within the operating system. The application-specific nature of deep filtering makes the inclusion of deep filtering code in the operating system kernel inappropriate. Deep filtering schemes are many and varied, and some are even specific to network traffic generated by particular software packages. Custom kernel code, in the form of one or more dynamically loaded modules, would have to be developed for every filtering application. This would be needlessly difficult from a software

engineering perspective. Adding custom code to the kernel also poses security and stability issues, as an operating system is not protected against errors (security or otherwise) in its own kernel, as it is against userland code.

Because it is application-specific and requires dynamic change during a system's runtime, and because we wish to protect the operating system from errors in the filtering software, deep filtering ought to be carried out at the user level. Using the netfilter ipqueue kernel module and the accompanying libipq C library, one can develop a userland packet filtering daemon that partners with the kernel to provide deep packet filtering.

Here, in order to understand how packet capture, using libpcap, and kernel sockets work on Linux-based systems, a brief description about the frame reception and deep inspection processing inside the Kernel is given. The Figure 4.5describes the frame reception flow inside the kernel and the pointed numbers during the text are referring to the steps depicted by the figure.

Whenever a packet arrives at the NIC, it is first copied into memory. It then triggers an interrupt, which will be handled by the NIC driver ((1) from figure 4.6). Therefore, the driver disables future interrupts and allocates a sk_buff (short for socket buffer) structure, which is the packet representation inside the kernel, fetches the data from the NIC buffer into the new sk_buff, in most cases via Direct Memory Access (DMA), and invokes netif_rx, which is the generic network reception handler.

The netif_rx (2) function puts a pointer to the new sk_buff inside the CPU's queue for incoming packets (there is one queue for each CPU), raises a software interrupt (softirq) and returns the congestion level of the queue to the caller. Please note that in the event that this queue is full, the incoming packet is then discarded. The function that processes the softirqs is known as do_softirq (3). It checks a bit mask and calls the appropriate handling routine, which represents the set bit. In this thesis, the interrupt of our interest is the NET_RX_SOFTIRQ (which is softirq raised when a packet has arrived) and the

handle routine is the net_rx_action (4), which basically dequeues the first packet from the current CPU queue and calls the processing functions, present on two protocol handler lists.

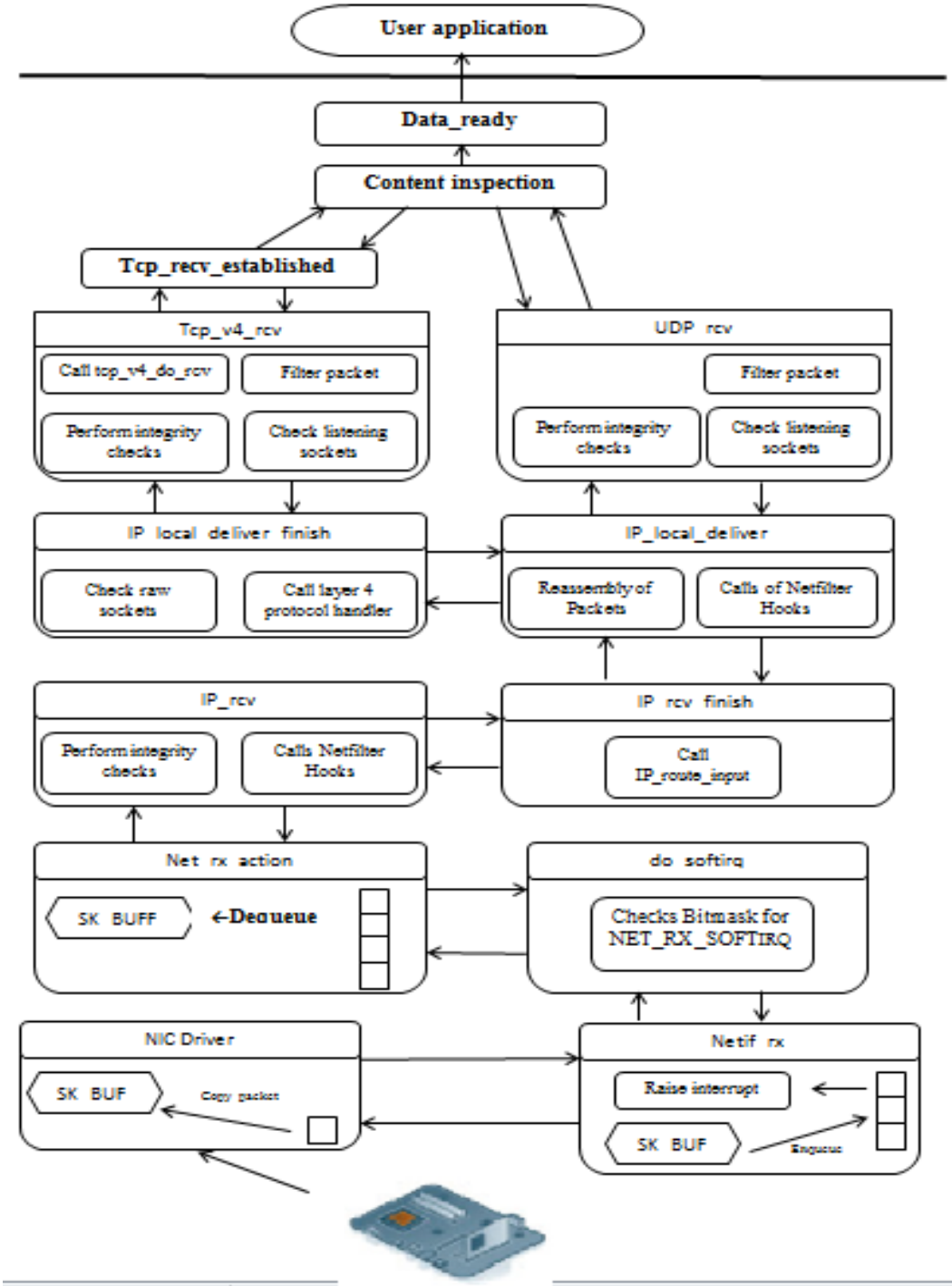
User application
Data_ready
Content inspection
Tcp_recv_established
Tcp_v4_rcv
Call tcp_v4_do_rcv
Filter packet
Perform integrity checks
Check listening sockets
UDP rcv
Filter packet
Perform integrity checks
Check listening sockets
IP local deliver finish
Check raw sockets
Call layer 4 protocol handler
IP_local_deliver
Reassembly of Packets
Calls of Netfilter Hooks
IP_rcv
Perform integrity checks
Calls Netfilter Hooks
IP rcv finish
Call IP_route_input
Net rx action
SK BUFF
←Dequeue
do softirq
Checks Bitmask for NET_RX_SOFTIRQ
NIC Driver
SK BUF
Copy packet
Netif rx
Raise interrupt
SK BUF
Enqueue

Figure 4.6. Packet inspection model for Linux-kernel board

Protocol handlers are functions which are registered into the kernel to handle specifics types of packets, such functions are registered in two lists: ptype_all and ptype_base (containing respectively protocol handlers for all types of packets and for specific protocols). The functions registered inside ptype_all list are those with the ETH_P_ALL flag (for generic packets), the ones at ptype_base are the others with ETH_P_* flags (e.g. ETH_P_IP, for IP packets). So, net_rx_action loops, until there are no packets to be dequeued or a threshold of maximum number of packets has been reached, calling each protocol handler registered, to take care of the incoming packet (NET_RX_SOFTIRQ is enabled again, if net_rx_action leaves a non empty queue of packets).

The protocol handler function responsible for handling IP packets is the ip_rcv (5) function. It performs all the integrity checks on the IP packet (checksum, header fields, etc.) and then, if the packet is considered correct, passes the packet to any Netfilter hook registered and finally calls ip_rcv_finish.

The ip_rcv_finish (6) function deals with packet routing. Within it, the ip_route_input function, decides whether the packet will be forwarded or locally delivered. Finally, the function ip_local_deliver is called.

ip_local_deliver (7) deals with IP defragmentation (reassembling the fragmented packets) and then, similarly to ip_rcv, it calls ip_local_deliver_finish (8), but before that it also calls Netfilter hooks. Next, the ip_local_deliver_finish takes care of the tasks that are still pending for the packet to be passed to upper layers, additionally it also checks if there are any RAW Sockets listening, in order to forward the packets to them, by calling the raw_v4_input function.

Therefore, after ip_local_deliver_finish checks whether there are any raw socket opened, it calculates a hash function based on the protocol number present at the IP header, and gets back the layer 4 function that will handle the packet. Such functions are registered in a Hash Table, called inet_protos, which is filled when the Linux kernel is initialized, with the layer 4 protocol handlers. These handlers are called tcp_v4_rcv, udp_rcv, icmp_rcv and igmp_rcv, representing each of the layer 4 protocols. Those layer 4 functions should return a value indicating if an ICMP Destination Unreachable message has to be returned to the packet sender, which must occur if the layer 4 protocols do not find a socket opened that matches the incoming packet.

If the packet contains a TCP segment, then the tcp_v4_rcv (9) handler is the function that will be called. It performs some header integrity checks, searches whether there are any sockets listening for TCP packets and performs the TCP Finite State Machine processing. If a TCP socket was registered inside the kernel, then the filter expressions, registered with each socket, are evaluated, in order to decide whether the packet should be forwarded to a current socket or not. If the packet passes the filter then the tcp_v4_do_rcv function is called, otherwise a value indicating that an ICMP Destination Unreachable packet should be sent back.

After the filtering process, the corresponding function, which matches the current state of the TCP connection, is called. If the connection has been already established, the function tcp_rcv_established (11) is called, where acknowledgements mechanisms and header processing are going to be performed. Additionally, at tcp_rcv_established, the function data_ready (12) will be called, which will alert the current process, owning the socket, which a packet has arrived and is ready to be consumed.

On the other hand, if the packet contains a UDP segment, the process is much simpler. At udp_rcv (10), some header and integrity checks are performed and then the same tasks as in those for TCP handlers are performed including:

search for a UDP socket matching the current packet (if no socket is found the packet is discarded), filtering (sk_filter function) and finally calls data_ready.

A programmer opens a socket (by invoking the socket system call) for packet reception, internally the kernel calls __sock_create and will create a new a sock structure (the kernel representation of sockets), and registers a protocol handler for the specified protocol.

These sockets are basically represented as a queue which stores pointers to packets inside the kernel and each of these sockets is responsible for a class of packets. The PF_PACKET, for instance, handles all the packets that arrive at the Network Interface Card (NIC) and skips all the processing that is usually performed by the TCP/IP stack inside the kernel (depicted at Figure 4.6). On the other hand, the PF_INET socket only deals with IP packets, but there are other types of sockets that only deal with TCP, UDP packets, etc., at the Linux protocol stack.

When an application is capturing packets using libpcap library, it first registers a PF_PACKET kernel socket, inside the kernel to handle all packets that are arriving in the NIC. Therefore, whenever a packet arrives at the NIC, an interruption is triggered to call the corresponding kernel driver for that NIC. Next, the packet is copied to the driver and a pointer to the packet is stored inside every socket's queue that matches the packet type, therefore the Linux Socket Filter (LSF) registered with such socket will filter the packet and decide whether it will be copied to user space or not. In summary, the capture process performs the following operations: packet copy from the NIC to the driver, pointer storage at PF_PACKET socket queue and another packet copy from kernel to user space.

Experimental results and source code from original implementation of deep packet inspection on the Linux are presented in appendix.

3. Development of Windows OS - based deep packet inspection system and experimental results

In this section I will explain the DPI solution which inspect entering packets to node, define malicious traffic and unauthorized access threats. The architecture of packet inspection system below in figure presented.

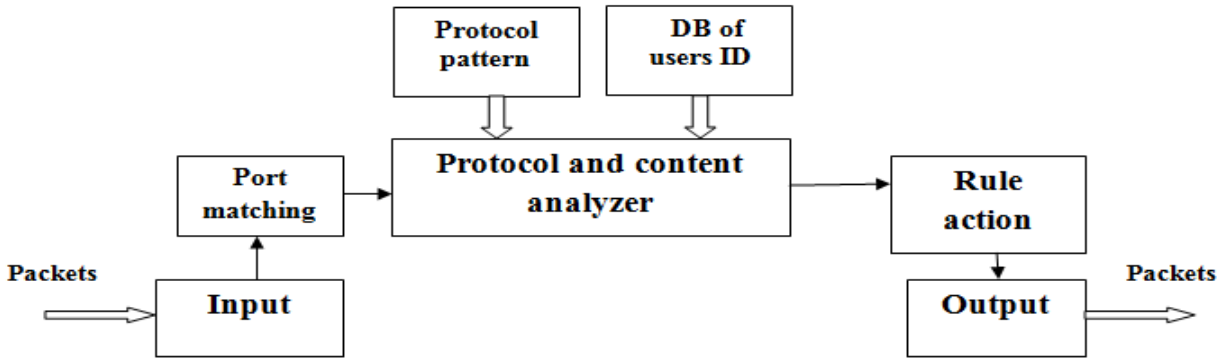

Figure 4.7. Packet inspection software architecture

The DPI software capture packets from Windows network drivers via loopback address 127.0.0.1. Firstly, matching ports are accomplished, next on the protocol and content analyzer block protocol pattern matching and checking and analyzing of content is performed.

For software special data protocol pattern is implemented. The format of protocol is illustrated in table-figure 4.8

Fields	1byte	16 byte for ID	1 byte	1 byte	1byte	2 byte Data length	1 byte	1byte
ASCII	*	-	*		*		*	Data

Hex	2A	-	2A	07	2A		2A	Data
Packet Length	23 byte data packet							

Figure 4.8. Data protocol pattern for testing application

After the matching of protocol pattern and inspecting content, the program defines resolution with rule action block. If packet is true, it sends packet for to their application inside the system or another processes, else it will remove from system. Many users may connect to software in one time, but software services only clients, which is registrated in ID databases.

The software consists of several algorithmic blocks and three main windows. The first window is “Settings form”. In this window port numbers, available user IDs, connection time can be added. This window illustrated in figure 4.9.

Reports of connections, attempts to connection, status of added objects will present main form (figure 4.10).

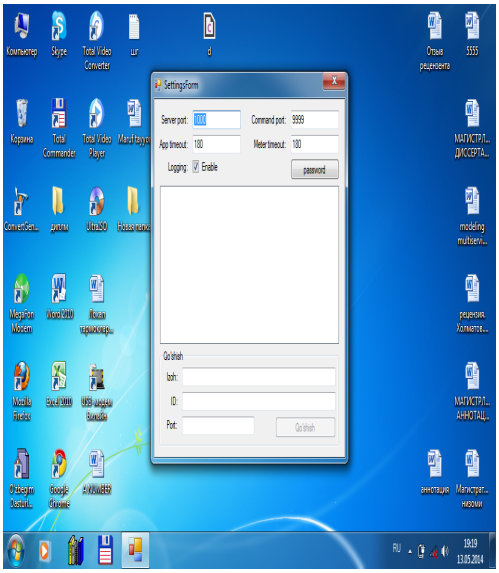

Figure 4.9. Settings window

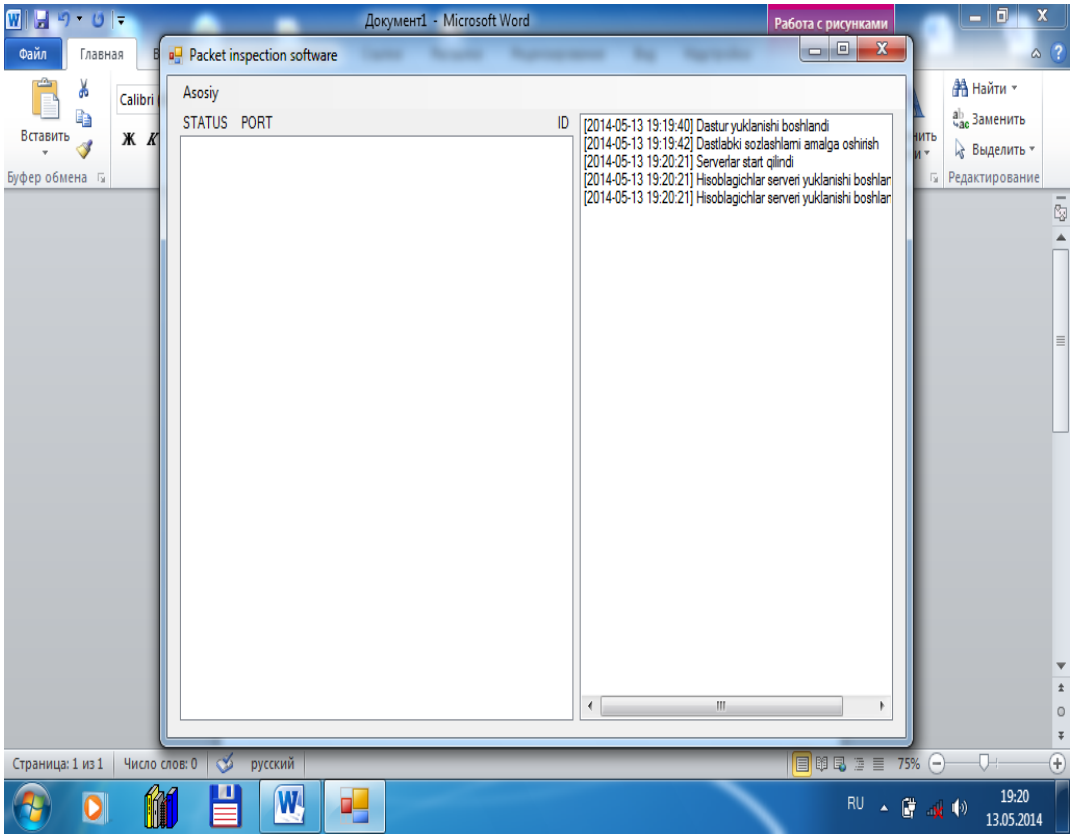

Figure 4.10. Main window of program

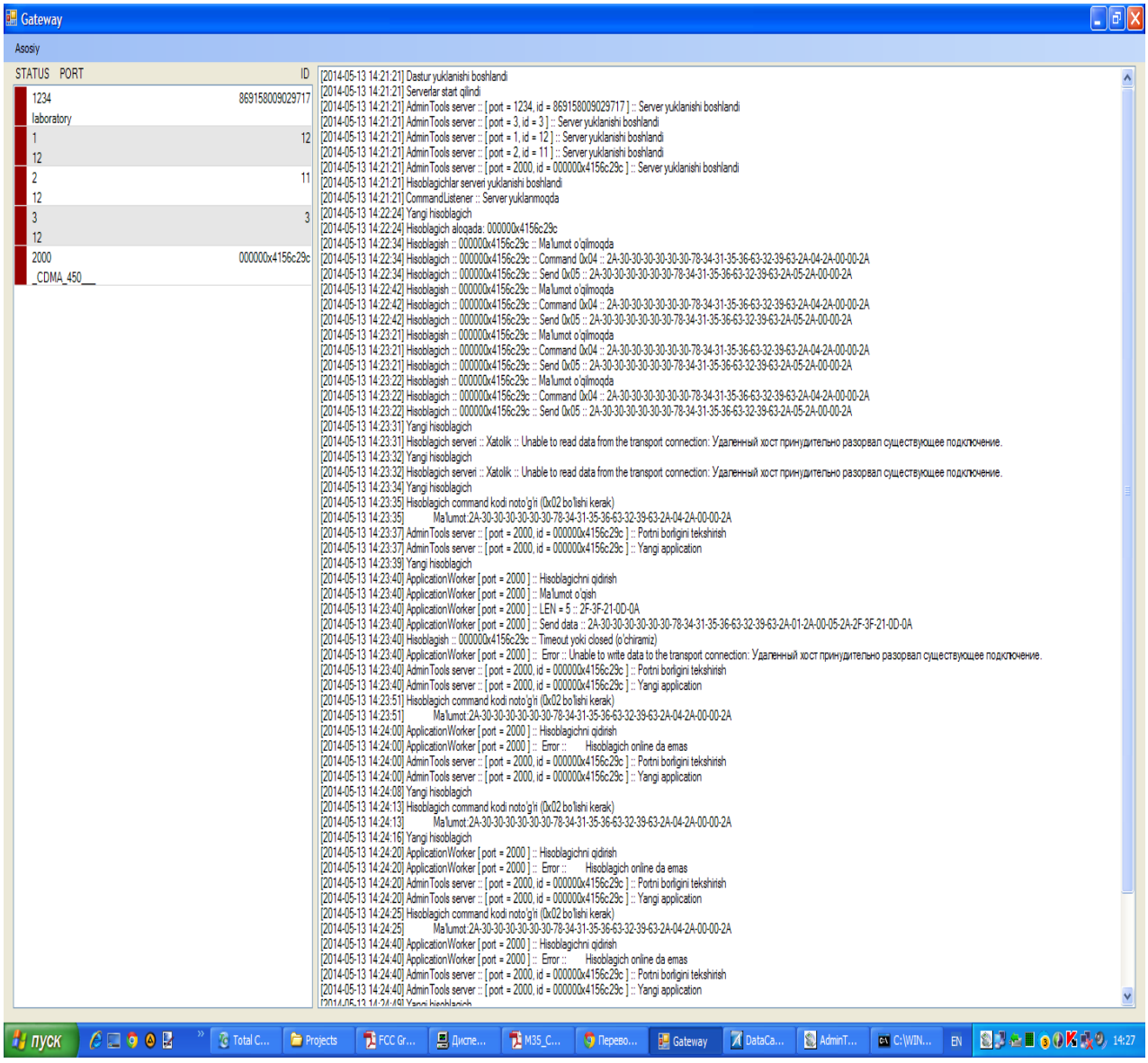

Figure 4.10. Program in running. Connections and object status window

All reports on the running time of program will be saved in ".log" file. From this file we can take statistics like protocol errors, connection times, number of unregistrated users and others.

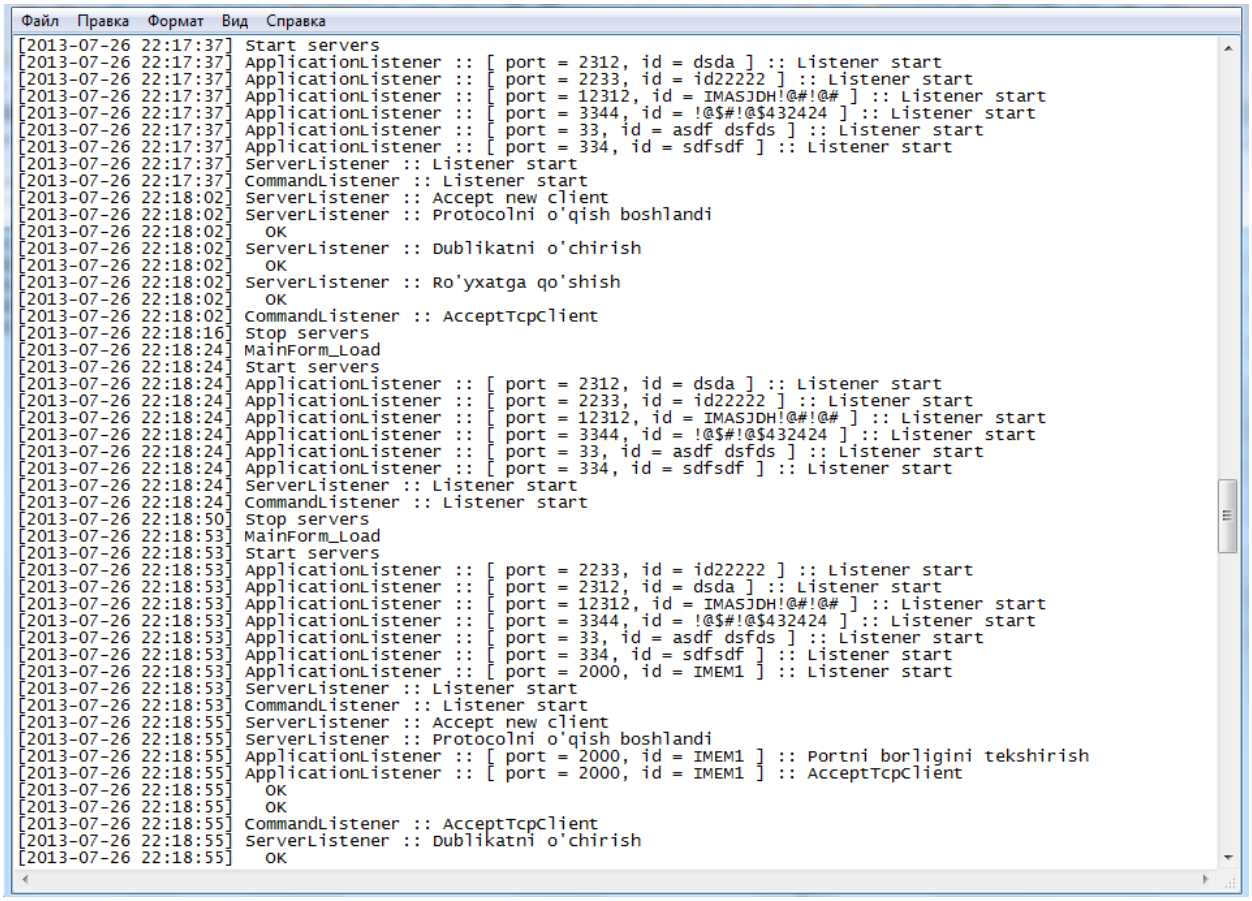

Figure 4.11. Window of program processes reports

In testing of packet inspection application I took results that illustrated below figures.

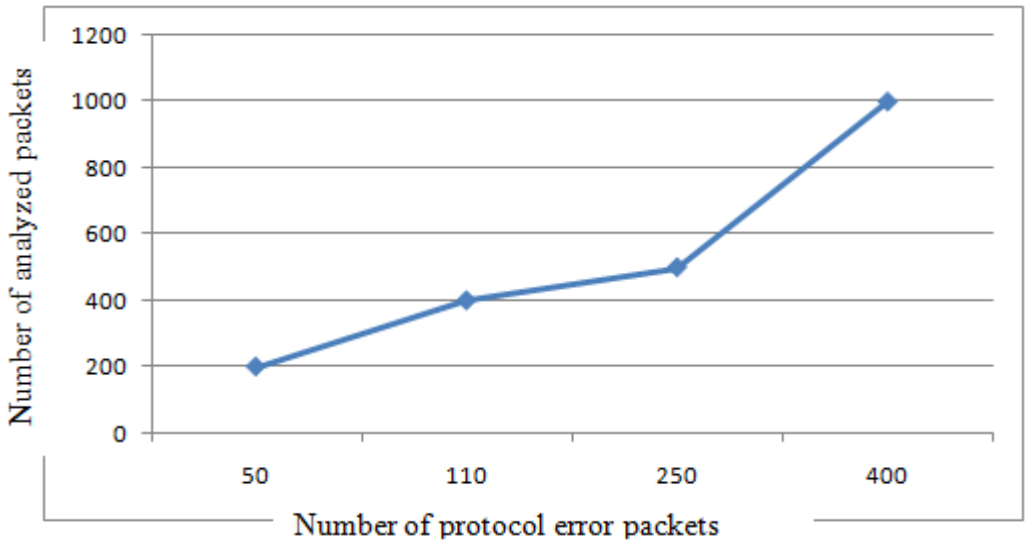

Figure 4.12. Protocol inspection results

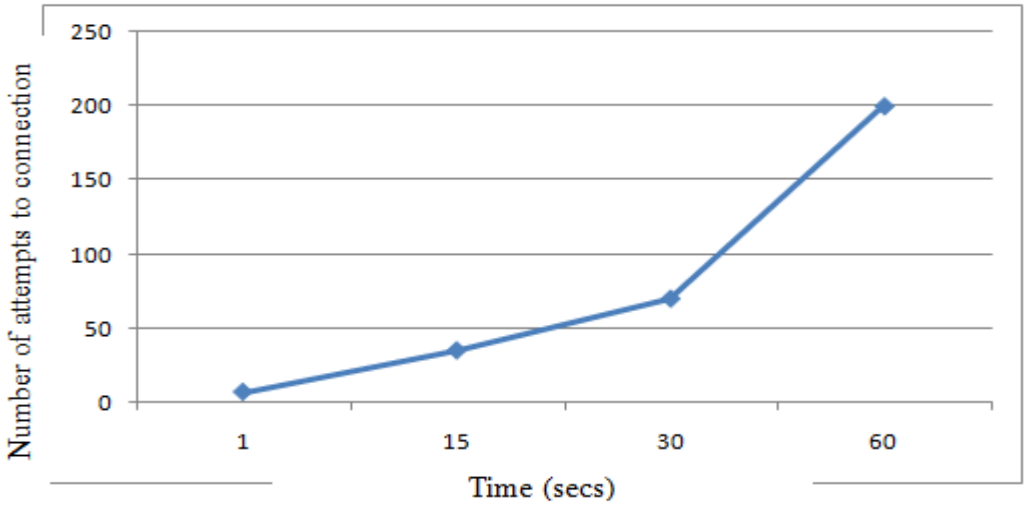

Figure 4.13. Attempts to connection to test server

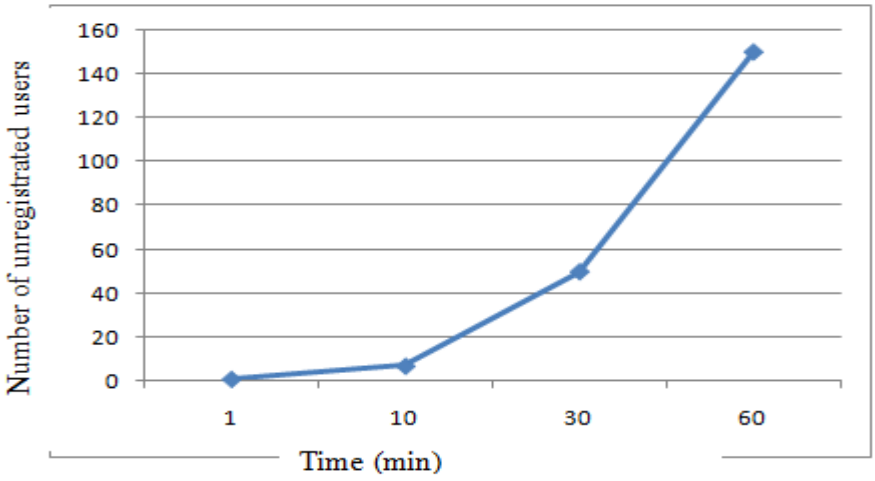

Figure 4.14. Threats of unregistrated users to system

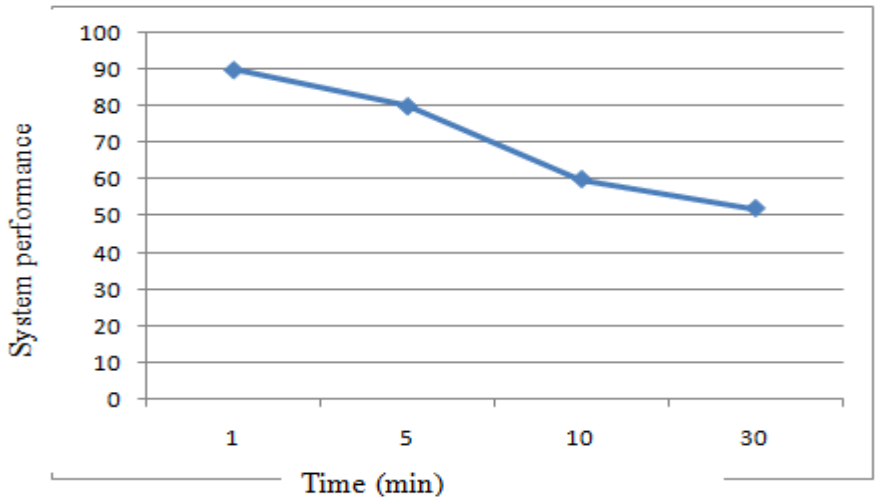

Figure 4.15. The performance of software

Results present, attempts to connect to system and threats to system increase depends on the time, effective performance of system goes down because of unregistrated attempts to connect. Testing results shows that software performance remains around 50% - 60% after a half hour.

This packet inspection software developed in MS Visual studio 2010 and tested Microsoft Windows XP. The main function of program is capturing and analyzing packets from real-time network.

Conclusion for chapter IV

In this chapter the recommendation for “ITU-T Y.2770 Requirements for Deep Packet Inspection in Next Generation Networks” is reviewed and requirements to DPI system development are analyzed. In recommendation DPI architecture and algorithms for system developers are specified in detail.

Using from recommendation, the algorithm and design of DPI system is implemented. Based on algorithm, DPI application is developed on the two different operation systems, Linux and Microsoft Windows.

Windows Deep Packet Inspection application is presented in this chapter and experimental results are explained. Linux DPI filter is installed on the mini router, but architecture - algorithms and work principles are given above.

Testing results show, Linux DPI filter application works better then Windows application. The performance of Linux DPI application is higher and packet inspection, filtering processes are faster in local machine.

In conclusion, developed DPI security systems help you to protect your host and local network, allow controlling data packets which entered to network. Also, they provide the capabilities to identify malicious traffic that may degrade user performance, drain network resources, impair infrastructure, and finally make the network unavailable to its subscribers. Most of the malicious traffic is discarded.

Conclusion

In dissertation, I addressed several challenging issues regarding the design of network intrusion detection and deep packet inspection systems, which performing the content inspection task more efficiently. I explained the developing of DPI system, with low implementation cost solutions as well as flexibility and scalability.

This thesis shows DPI system developers can take advantage of subtle modifications that lead to considerable performance gains. Now, DPI systems that were able to deal with high packet incoming rate can handle such load of traffic with increase in the classification completeness. Additionally, such modifications empower DPI systems, in order to provide ISPs with a wider view of what is passing through their network, aggregating a great amount of traffic information. In this dissertation, I explained high speed packet processing

algorithms for new services such as network intrusion detection, high speed firewalls, Network Address Translation (NAT) and developed own DPI security software for local using. These DPI applications can reconfigure on demand and may add thousands of known protocol patterns for rapid scanning and analyzing data packets.

In research on this thesis, I used deep packet inspection techniques, algorithms and requirements. DPI requirements specify the packet inspection function on the entire packet content rather than just the header. To keep up with high speed packet processing in existing networks, I proposed and used deep packet inspection schemes that are, optimized for new technologies such as Static Pattern Content Matching, Regular Expression Pattern Matching and multi-core processors. Algorithms, I used in that work both on packet headers and packet payload. The developed DPI system is in a cohesive and flexible architecture that can perform high rate packet scanning and inspecting against thousands of sophisticated patterns.

So, consistent with this thesis, the tasks have been solved:

- analyzed of DPI solutions for networks security in ip-based communication networks;
- analyzed of deep packet inspection methods and algorithms;
- developed algorithm and program for DPI security system in different operation systems;
- explained experimental analyses taken from real network with DPI software.

Bibliography

Literatures for chapter I

The books by President of Uzbekistan I.A. Karimov

1. The Global Financial-Economic crisis, ways and measures to overcome it in the conditions of Uzbekistan. –T: Uzbekistan, 2009. -59 p.

Main literatures

2. IP-based Networks: Basics. Axis Communications, 2002.

3. Forouzan, Behrouz A. TCP/IP PROTOCOL SUITE, FOURTH EDITION. Published by McGraw-Hill, New York, NY 10020., 2010.

4. Data communication and networking, 5th edition. Published by McGraw-Hill, New York, NY 10020., 2012.

5. N. Cascarano, L. Ciminiera, F. Risso. Optimizing Deep Packet Inspection for High-Speed Traffic Analysis, 2012

6. White Paper - Modern Network Security: The Migration to Deep Packet Inspection. www.esoft.com

7. Sandvine, "Global Internet Phenomena Report", *Sandvine White Paper,* 2012.

8. Y.J. Lee, J.S. Oh, and B.G. Lee, "Logical Push Framework for Real-time SNS Processing," *The 4th IEEE International Conference on Computational Aspects of Social Networks,* Brazil, Nov. 2012.

9. Infonetics Research, "Service Provider Deep Packet Inspection Products", *Infonetics' biannual DPI report,* 2012.

10. R. Bendrath, "Global technology trends and national regulation: Explaining Variation in the Governance of Deep Packet Inspection," *International Studies Annual Convention,* New York, pp.1-32, Mar. 2009.

11. Y.J. Lee, J.S. Oh, J.K. Lee, D.W. Kang, and B.G. Lee, "The Utilization of Deep Packet Inspection for Real-time Massive Traffic Processing," *The conference on Korean Society for Internet Information,* Korea, Vol.13, No.2, pp.77-78, Nov. 2012.

12. T. Porter, "The Perils of Deep Packet Inspection," *Security Focus,* Mar. 2008.

13. S.W. Shin, D.H. Kang, K.Y. Kim, and J.S. Jang, "Analysis of Deep Packet Inspection Technology," *Electronic Communication Trend Analysis,* Vol.19, No.3, pp.117-124, Jun.2004.

14. R. Bendrath, “Global technology trends and national regulation: Explaining Variation in the Governance of Deep Packet Inspection,” *International Studies Annual Convention*, Vol.15 No.18, Feb. 2009.

15. Shira Levine, “Mobile momentum drives DPI market to 30.4% CAGR through 2016,” *Infonetics Research,* 2012.

16. Sandvine, “Policy Traffic Switch 24000: Platform Datasheet,” *Sandvine* Data Sheet*,* Jul. 2012.

17. Cisco, “Cisco SCE 8000 Service Control Engine,” *Cisco Data Sheet,* 2010.

18. ETRI, “Development of Real-time Traffic‘s Integrated Control Platform for Clean Internet and Fair Interconnection Environment,” *Project Report,* 2012

Literatures for chapter II

1. Elizabeth D. Zwicky, Simon Cooper & D. Brent Chapman. Building Internet Firewalls, Second Edition, June 2006

2. L. Gheorghe. Designing and Implementing Linux Firewalls and QoS using netfilter, iproute2, NAT, and L7-filter, Copyright © 2006 Packet Publishing

3. Technical Report ISA-TR99.00.01-2004: Security Technologies for Manufacturing and Control Systems, *Instrumentation, Systems and Automation Society (ISA),* March 2004

4. ibid - Technical Report ISA-TR99.00.01-2004

5. B. Fraser, “RCF 2196 - Site Security Handbook”, *Internet Engineering Task Force*, September 1997, Pg. 22 10 ibid - Technical Report ISA-TR99.00.01-2004

6. Dr. Thomas Porter. The Perils of Deep Packet Inspection**.** Security Focus, 2006

7. Check Point. “Stateful Inspection Technology Tech Note.” http://www.checkpoint.com/products/security/whitepapers/firewall-1_statefulinspection.pdf. March 2002.

8. Cryptography and network security, 5th edition. 2011, 2006 Pearson Education, Inc., publishing as Prentice Hall.

9. Network Security policy and objectives, http://publib.boulder.ibm.com/infocentr/iseries/securitypolco.htm

10. Packet Filtering Process, http://www.ibm.com/developerworks/linux/ library/s-netip/Packet filtering using IP tables, http://netfilter.org/documentation/ HOWTO /packet-filtering- HOWTO-7.html

11. David W Chadwick, “Network Firewall Technologies”, IS Institute, University of Salford, Salford, M5 4WT, England.

Literatures for chapter III

1. Next Generation Deep Packet Inspection: An Overview of Requirements and Applications. Network Strategy Partners, LLC, 2007.

2. Whitepaper. DPI: Deep Packet Inspection Motivations, Technology, and Approaches for Improving Broadband Service Provider ROI. RadiSys Corporation, 2011.

3. K. Mochalski, H. Deep Packet Inspection. Technology, Applications & Net Neutrality. Whitepaper Ipoque, 2009

4. Ioannis Sourdis. Designs & Algorithms for Packet and Content Inspection

I. Sourdis, D.N. Pnevmatikatos, S. Vassiliadis, Scalable Multi-Gigabit

5. Pattern Matching for Packet Inspection, to appear in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, special section on Configurable Computing Design, 2007/2008.

6. I. Sourdis, V. Dimopoulos, D.N. Pnevmatikatos, S. Vassiliadis, Packet Pre-filtering for Network Intrusion Detection, in 2nd ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS), ACM, pp. 183–192, San Jose, California, December 2006.

7. Application layer packet classifier for linux. http://l7-filter.clearfoundation .com/.

8. Snort. http://www.snort.org/.

9. V. Paxson. Bro: a system for detecting network intruders in real-time. Computer Networks, 31(23-24): 2435{2463, 1999.

10. M. Roesch. Snort: Lightweight intrusion detection for networks. In Proc. 13thSystems Administration Conference (LISA), USENIX Association, pages 229238, November 1999.

11. M. Becchi and P. Crowley. A hybrid finite automaton for practical deep packet inspection. In Proc. ACM Int. Conf. on emerging Networking EXperiments and Technologies (CoNEXT). ACM Press, 2007.

12. IPP2P Project - "A NetFilter extension to identify P2P file sharing traffic." http://www.ipp2p.org, visited in November 15, 2008.

13. J. Levandoski, E. Sommer, and M. Strait, "Application Layer Packet Classifier for Linux." http://l7-filter.sourceforge.net/, visited in November 14, 2008.Mc Kenney, P. E., Lee, D. Y., & Denny, B. A. (2008). Traffic generator software release notes.

14. Dharmapurikar, S., Krishnamurthy, P., Sproull, T. S., & Lockwood, J. W. (2004). Deep packet inspection using parallel bloom filters. *IEEE Micro*, *24*(1), 52–61. doi:10.1109/MM.2004.1268997

15. Yang Xiang, Daxin Tian. Multi-Core Supported Deep Packet Inspection. Hershey - New York, Copyright © 2010 by IGI Global

16. Sutter, H., & Larus, J. (2005). Software and the concurrency revolution. *ACM Queue; Tomorrow's Computing Today*, *3*(7), 54–62. doi:10.1145/1095408.1095421

17. Xiang, Y., & Zhou, W. (2006). Protecting information infrastructure from ddos attacks by mark-aided distributed filtering (madf). *International Journal of High Performance Computing and Networking*, *4*(5/6), 357–367. doi:10.1504/IJHPCN.2006.013491

18. Paxson, V., Sommer, R., & Weaver, N. (2007). Architecture for exploiting multi-core processors to parallelize network intrusion prevention. *Proceedings of IEEE Sarnoff Symposium.*

19. Aho, A. V., & Corasick, M. J. (1975). Efficient string matching: An aid to bibliographic search. *Communications of the ACM*, *18*(6), 333–340. doi:10.1145/360825.360855

Literatures for chapter IV

1. ITU-T Y.2770. Requirements for Deep Packet Inspection in Next Generation Networks. Telecommunications Technology Association, 2012.

2. ITU-T Recommendation Y.2011 (2004), General principles and general reference model for Next Generation Networks. ITU-T Y.2011

3. Draft baseline text of Y.dpireq "Requirements of DPI in packet-based networks and NGN environment". This is the output of draft Recommendation Y.dpireq Requirements of DPI in packet-based networks and NGN environment) from the January 2009 meeting of SG13.

4. The Netfilter Project. http://www.netfilter.org/.

5. T. LACERDA On the optimization of deep packet inspection Recife, PE, 2008

Internet resources

1. http://www.netfilter.org/.

2. http://www.snort.org/.

3. http://www.ibm.com/developerworks/linux

4. www.esoft.com

Appendix