Comparison of DOD and OSI Model in the Internet Communication

Ei Ei Khaing

FCST Department, University of Computer Studies, Thaton, Myanmar

ABSTRACT

The Internet protocol suite is the computer networking model and set of communications protocols used on the Internet and similar computer networks. It is commonly known as TCP/IP, because it's most important protocols, the Transmission Control Protocol (TCP) and the Internet Protocol (IP), were the first networking protocols defined in this standard. Often also called the Internet model, it was originally also known as the DoD model, because the development of the networking model was funded by DARPA, an agency of the United States Department of Defense. TCP/IP provides end-to-end connectivity specifying how data should be packetized, addressed, transmitted, routed and received at the destination. This functionality is organized into four abstraction layers which are used to sort all related protocols according to the scope of networking involved. From lowest to highest, the layers are the link layer, containing communication technologies for a single network segment (link); the internet layer, connecting hosts across independent networks, thus establishing internetworking; the transport layer handling host-to-host communication; and the application layer, which provides process-to-process application data exchange. Our aim is describe operation & models of TCP-IP suite in data communication networking.

KEYWORDS: TCP-IP, OSI, PROTOCOL, LAYERS, STACK

I. INTRODUCTION

The development and growth of the internet required specific protocols for internet communication. Two protocols – Internet Protocol (IP) and Transmission Control Protocol (TCP) – were developed as a result. Known collectively as the TCP/IP suite, each protocol is responsible for different aspects of communication: TCP/IP is a two-layer program. The upper layer, Transmission Control Protocol, manages the assembling of a message or file into smaller packets that are transmitted over the Internet and received by a TCP layer that reassembles the packets into the original message. The lower layer, Internet Protocol, handles the address part of each packet so that it gets to the right destination. Each gateway computer on the network checks this address to see where to forward the message. Even though some packets from the same message are routed differently than others, they'll be reassembled at the destination [1]. TCP/IP uses the client/server model of communication in which a computer user requests and is provided a service by another computer in the network. TCP/IP communication is primarily point-to-point, meaning each communication is from one point in the network to another point or host computer. TCP/IP and the higher-level applications that use it are collectively said to be "stateless" because each client request is considered a new request unrelated to any previous one.

Being stateless frees network paths so that everyone can use them continuously Any Internet users are familiar with the even higher layer application protocols that use TCP/IP to get to the Internet. These include the World Wide Web's

Hypertext Transfer Protocol (HTTP), the File Transfer Protocol (FTP), Telnet (Telnet) which lets you logon to remote computers, and the Simple Mail Transfer Protocol (SMTP). These and other protocols are often packaged together with TCP/IP as a "suite." Personal computer users with an analog phone modem connection to the Internet usually get to the Internet through the Serial Line Internet Protocol (SLIP) or the Point-to-Point Protocol (PPP). These protocols encapsulate the IP packets so that they can be sent over the dial-up phone connection to an access provider's modem [2]. Protocols related to TCP/IP include the User Datagram Protocol (UDP), which is used instead of TCP for special purposes. Other protocols are used by network host computers for exchanging router information. These include the Internet Control Message Protocol (ICMP), the Interior Gateway Protocol (IGP), the Exterior Gateway Protocol (EGP), and the Border Gateway Protocol (BGP).
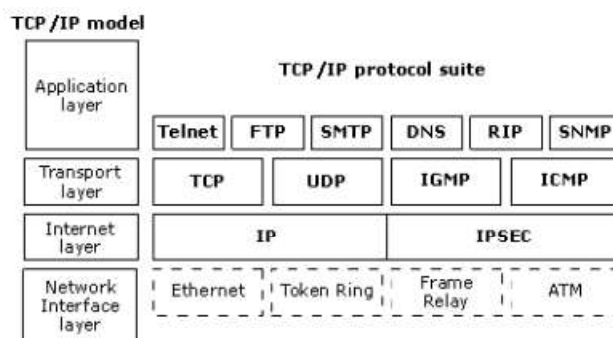

Figure 1: TCP-IP Protocol Suite.

II. A TCP/IP communication example

If we want to view a webpage on the internet, the TCP/IP suite handles the communication between our device and the webpage's web server:

1. Our browser uses the Hyper Text Transfer Protocol (HTTP) to send a request to the Application Layer.
2. The Application Layer protocol (HTTP) sends the request to the Transmission Control Protocol (TCP) in the Transport Layer. The TCP communicates with the Internet Layer to establish a connection with the web server across the network(s).
3. The Internet Protocol in the Internet Layer establishes the address of the web server and converts the request into packets. The packets are sent to the Network Layer.
4. The Network Layer uses its protocols to send the packets over the internet to the web server.
5. At the web server the process is reversed. The packets are sent up through the protocols in the layers, and re-assembled into the request. The request is passed through the Application Layer protocols for the web server to service.
6. In the same manner the webserver uses the protocols in the layers to send the webpage data back to our device.

III. What is a protocol?

Modern computers are designed with communication in mind. At a basic level, computers may need to communicate with peripherals such as a printer, or to send messages to other computers, which may be connected across a local area network (such as in a business, or a school), a mobile phone cellular network, or the internet.

Before one computer can communicate with another, there must be a set of agreed rules in place that manage how the communication is to take place. These rules are known collectively as a protocol.

To illustrate the need for rules when two or more objects communicate, we will look at some rules used when humans talk to each other. For example, we:

➤ use sound to send messages
➤ use a common language that we both understand
➤ listen when the other person is talking, so we can hear them speak
➤ speak at a rate that allows us to clearly transmit words and receive them
➤ have ways of checking that the other person has heard what we have said.

Without such rules, communication becomes difficult, or even impossible.

Similarly to human communication, computers use rules, and these form the protocols that are used for computer communication. Protocols need to specify several aspects of communication, for example:

➤ the communication medium to be used, such as wired or wireless
➤ the transmission type, such as duplex or simplex
➤ whether the handshake is hardware or software in nature
➤ the method of error checking to be used
➤ the bit rate
➤ the character set that will be used.

The protocols allow messages to be transmitted in a structured, specific manner, so that the receiving device is able to process the data sent from the sending device. Protocols therefore make sure that communication between any two devices is successful. Different communication protocols exist to specify rules for different types of communication, for example:

➤ Hyper Text Transfer Protocol (HTTP) – handles transmission of data to and from a website
➤ Simple Mail Transfer Protocol (SMTP) – handles email transmission
➤ File Transfer Protocol (FTP) – handles transmission of files across a network
➤ TCP/IP – handles communication over the internet.

IV. EARLY RESEARCH ON TCP-IP

The Internet protocol suite resulted from research and development conducted by the Defense Advanced Research Projects Agency (DARPA) in the late 1960s. After initiating the pioneering ARPANET in 1969, DARPA started work on a number of other data transmission technologies. In 1972, Robert E. Kahn joined the DARPA Information Processing Technology Office, where he worked on both satellite packet networks and ground-based radio packet networks, and recognized the value of being able to communicate across both. In the spring of 1973, Vinton Cerf, the developer of the existing ARPANET Network Control Program (NCP) protocol, joined Kahn to work on open-architecture interconnection models with the goal of designing the next protocol generation for the ARPANET [3]. By the summer of 1973, Kahn and Cerf had worked out a fundamental reformulation, in which the differences between network protocols were hidden by using a common internetwork protocol, and, instead of the network being responsible for reliability, as in the ARPANET, the hosts became responsible. Cerf credits Hubert Zimmermann and Louis Paulin, designer of the CYCLADES network, with important influences on this design. The design of the network included the recognition that it should provide only the functions of efficiently transmitting and routing traffic between end nodes and that all other intelligence should be located at the edge of the network, in the end nodes. Using a simple design, it became possible to connect almost any network to the ARPANET, irrespective of the local characteristics, thereby solving Kahn's initial problem. A computer called a router is provided with an interface to each network. It forwards packets back and forth between them. Originally a router was called gateway, but the term was changed to avoid confusion with other types of gateways.

V. ARCHITECTURAL PRINCIPLE

End-to-end principle: This principle has evolved over time. Its original expression put the maintenance of state and overall intelligence at the edges, and assumed the Internet that connected the edges retained no state and concentrated on speed and simplicity. Real-world needs for firewalls, network address translators, web content caches and the like have forced changes in this principle. Robustness Principle: "In general, an implementation must be conservative in its sending behavior, and liberal in its receiving behavior. That is, it must be careful to send well-formed datagrams, but must accept any datagram that it can interpret .The second part of the principle is almost as important: software on other hosts may contain deficiencies that make it unwise to exploit legal but obscure protocol features.

VI.   MODEL LAYERS OF TCP-IP

The Internet protocol suite uses encapsulation to provide abstraction of protocols and services. Encapsulation is usually aligned with the division of the protocol suite into layers of general functionality. In general, an application uses a set of protocols to send its data down the layers, being further encapsulated at each level. The layers of the protocol suite near the top are logically closer to the user application, while those near the bottom are logically closer to the physical transmission of the data [5]. Viewing layers as providing or consuming a service is a method of abstraction to isolate upper layer protocols from the details of transmitting bits over, for example, Ethernet and collision detection, while the lower layers avoid having to know the details of each and every application and its protocol. Even when the layers are examined, the assorted architectural documents—there is no single architectural model such as ISO 7498, the Open Systems Interconnection (OSI) model have fewer and less rigidly defined layers than the OSI model, and thus provide an easier fit for real-world protocols.

One frequently referenced document, RFC 1958, does not contain a stack of layers. The lack of emphasis on layering is a major difference between the IETF and OSI approaches. It only refers to the existence of the internetworking layer and generally to upper layers; this document was intended as a 1996 snapshot of the architecture: "The Internet and its architecture have grown in evolutionary fashion from modest beginnings, rather than from a Grand Plan. While this process of evolution is one of the main reasons for the technology's success, it nevertheless seems useful to record a snapshot of the current principles of the Internet architecture."

RFC 1122, entitled Host Requirements, is structured in paragraphs referring to layers, but the document refers to many other architectural principles not emphasizing layering. It loosely defines a four-layer model, with the layers having names, not numbers, as follows:

➢ The Application layer is the scope within which applications create user data and communicate this data to other applications on another or the same host. The applications, or processes, make use of the services provided by the underlying, lower layers, especially the Transport Layer which provides reliable or unreliable pipes to other processes. The communications partners are characterized by the application architecture, such as the client-server model and peer-to-peer networking. This is the layer in which all higher level protocols, such as SMTP, FTP, SSH, HTTP, operate. Processes are addressed via ports which essentially represent services.

➢ The Transport Layer performs host-to-host communications on either the same or different hosts and on either the local network or remote networks separated by routers. It provides a channel for the communication needs of applications. UDP is the basic transport layer protocol, providing an unreliable datagram service. The Transmission Control Protocol provides flow-control, connection establishment, and reliable transmission of data.

➢ The Internet layer has the task of exchanging datagrams across network boundaries. It provides a uniform networking interface that hides the actual topology (layout) of the underlying network connections. It is therefore also referred to as the layer that establishes internetworking, indeed, it defines and establishes the Internet. This layer defines the addressing and routing structures used for the TCP/IP protocol suite. The primary protocol in this scope is the Internet Protocol, which defines IP addresses. Its function in routing is to transport datagrams to the next IP router that has the connectivity to a network closer to the final data destination.

➢ The Link layer defines the networking methods within the scope of the local network link on which hosts communicate without intervening routers. This layer includes the protocols used to describe the local network topology and the interfaces needed to effect transmission of Internet layer datagrams to next-neighbor hosts [6].

➢ The Internet protocol suite and the layered protocol stack design were in use before the OSI model was established. Since then, the TCP/IP model has been compared with the OSI model in books and classrooms, which often results in confusion because the two models use different assumptions and goals, including the relative importance of strict layering.

VII.   Data Encapsulation and the TCP/IP Protocol Stack

The packet is the basic unit of information that is transferred across a network. The basic packet consists of a header with the sending and receiving systems' addresses, and a body, or payload, with the data to be transferred. As the packet travels through the TCP/IP protocol stack, the protocols at each layer either add or remove fields from the basic header. When a protocol on the sending system adds data to the packet header, the process is called data encapsulation [7]. Moreover, each layer has a different term for the altered packet, as shown in the following figure.
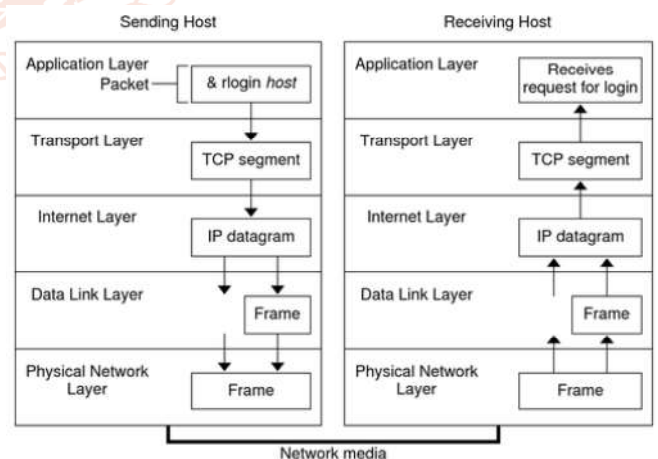
Figure2: How a Packet Travels Through the TCP/IP Stack

IP attaches an IP header to the segment or packet's header, in addition to the information that is added by TCP or UDP. Information in the IP header includes the IP addresses of the sending and receiving hosts, the datagram length, and the datagram sequence order. This information is provided if the datagram exceeds the allowable byte size for network packets and must be fragmented. TCP/IP provides internal

---

trace support by logging TCP communication when an RST packet terminates a connection. When an RST packet is transmitted or received, information on as many as 10 packets, which were just transmitted, is logged with the connection information.

VIII. TCP PROTOCOLS WITH INTERNETS

TCP/IP is most commonly associated with the UNIX operating system. While developed separately, they have been historically tied, as mentioned above, since 4.2BSD UNIX started bundling TCP/IP protocols with the operating system. Nevertheless, TCP/IP protocols are available for all widely-used operating systems today and native TCP/IP support is provided in OS/2, OS/400, all Windows versions since Windows 9x, and all Linux and UNIX variants [8]. When the user accesses a Web site on the Internet, the NAT server will translate the "private" IP address of the host (192.168.50.50) into a "public" IP address (220.16.16.5) from the pool of assigned addresses. NAT works because of the assumption that, in this example, no more than 27 of the 64 hosts will ever be accessing the Internet at a single time.
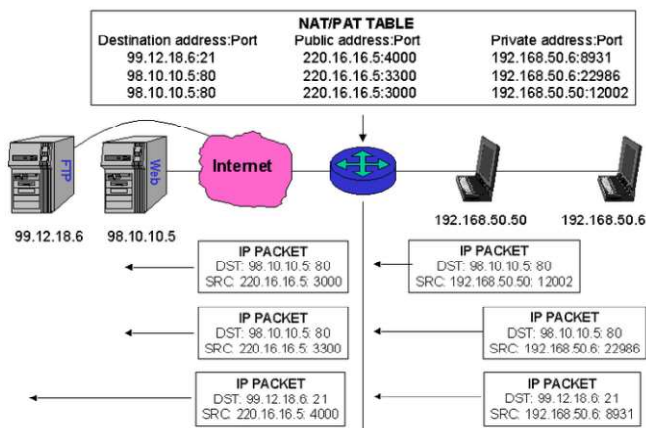
Figure3: TCP-IP

A pool of IP addresses can be shared by multiple hosts using a mechanism called Network Address Translation (NAT). NAT, described in RFC 1631, is typically implemented in hosts, proxy servers, or routers. The scheme works because every host on the user's network can be assigned an IP address from the pool of RFC 1918 private addresses; since these addresses are never seen on the Internet, this is not a problem. Consider the scenario shown in Figure 3. Port numbers are used by higher layer protocols (e.g., TCP and UDP) to identify a higher layer application. A TCP connection, for example, is uniquely identified on the Internet by the four values (aka 4-tuple) <source IP address, source port, destination IP address, destination port>. The server's port number is defined by the standards while client port numbers can be any number greater than 1023. The scenario in Figure 3 shows the following three connections:

- The client with the "private" IP address 192.168.50.50 (using port number 12002) connects to a Web server at address 98.10.10.5 (port 80).

- The client with the "private" IP address 192.168.50.6 (using port number 22986) connects to the same Web server at address 98.10.10.5 (port 80).

- The client with the "private" IP address 192.168.50.6 (using port number 8931) connects to an FTP server at address 99.12.18.6 (port 21).

PAT works in this scenario as follows. The router (running PAT software) can assign both local hosts with the same "public" IP address (220.16.16.5) and differentiate between the three packet flows by the source port. A final note about NAT and PAT. Both of these solutions work and work fine, but they require that every packet be buffered, disassembled, provided with a new IP address, a new checksum calculated, and the packet reassembled. In addition, PAT requires that a new port number be placed in the higher layer protocol data unit and new checksum calculated at the protocol layer above IP, too. The point is that NAT, and particularly PAT, results in a tremendous performance hit. One advantage of NAT is that it makes IP address renumbering a thing of the past. If a customer has an IP NET_ID assigned from its ISP's CIDR block and then they change ISPs, they will get a new NET_ID. With NAT, only the servers need to be renumbered [9].

IX. Layering and the OSI model
1. The OSI Model Stack
The OSI model divides the complex task of computer-to-computer communications, traditionally called internetworking, into a series of stages known as layers. Layers in the OSI model are ordered from the lowest layer to highest. Together, these layers comprise the OSI stack. The stack contains seven layers in two groups:

Upper Layers:
- 7: Application
- 6: Presentation
- 5: Session

Lower Layers:
- 4: Transport
- 3: Network
- 2: Data Link
- 1: Physical

2. Upper Layers of the OSI Model
OSI designates the Application, Presentation, and Session stages of the stack as the *upper layers*. Generally speaking, software in these layers perform application-specific functions like data formatting, encryption, and connection management.

Examples of upper layer technologies in the OSI model are HTTP, SSL, SCP, NetBIOS, SMTP, FTP, RPC, DNS, and NFS.

3. Lower Layers of the OSI Model
The remaining *lower layers* of the OSI model provide more primitive network-specific functions like routing, addressing, and flow control.

Examples of lower layer technologies in the OSI model are TCP, UDP, IP, Ethernet, RDP, ICMP, IPsec, IPv4 and IPv6, RIP, toking ring, Bluetooth, USB, DSL, and GSM.

4. A Practical Example of the OSI Model
The seven layers of the OSI model work together, one by one, to complete a full task. All of this work, of course, happens behind the scenes, so even when you're using the concept of the OSI model, you might not even realize it. For example, when your computer requested this web page, the device may have used an Ethernet connection to relay the

information through a router, and the request ultimately landed on an HTTP server where the page resides. In this example, the OSI model was used from the lower layers to the upper layers, from layer one through layer seven.

Many types of computer – for example, personal computers, laptops, servers, tablets and smartphones – exist, and they are made by many different manufacturers. As a result, it is important to have standards in place that designers and equipment manufacturers can follow to make sure their products can communicate with others'. The Open Systems Interconnection model (OSI model) was introduced to form standards for computer communication. The model takes the various elements of communication and conceptually separates them into layers. Each layer can be treated individually, and is independent from the others. Each layer contains different protocols and handles a different part of the transmission.

The model specifies seven layers:

Application Layer:
The interface between applications and the network. Data/messages from applications are passed down to / received from the Presentation Layer.

Presentation Layer:
Converts the data/message passed down from the Application Layer into a format usable by the layers below. Applications use data in different ways and in different formats. This layer makes sure all data/messages are standardized into one format.

Session Layer:
Controls the communication. Every communication has a beginning and an end. The Session Layer initiates the communication and maintains it until it is terminated.

Transport Layer:
Orders and assembles data packets for transmission across the network.

Network Layer Delivers packets:
This layer determines the address of the destination computer and works out a route for packets to be sent across the network.

Data Link Layer:
Aids error-free transmission. This layer contains the error-checking rules and attempts to correct any transmission errors, as well as determining the flow rate. Devices may not be able to communicate at the same speed, so this layer makes sure data flows at a rate acceptable for both devices.

Physical Layer:
Handles the physical characteristics of the hardware that handles the communication, such as the medium (e.g. wired or wireless) and transmission type.
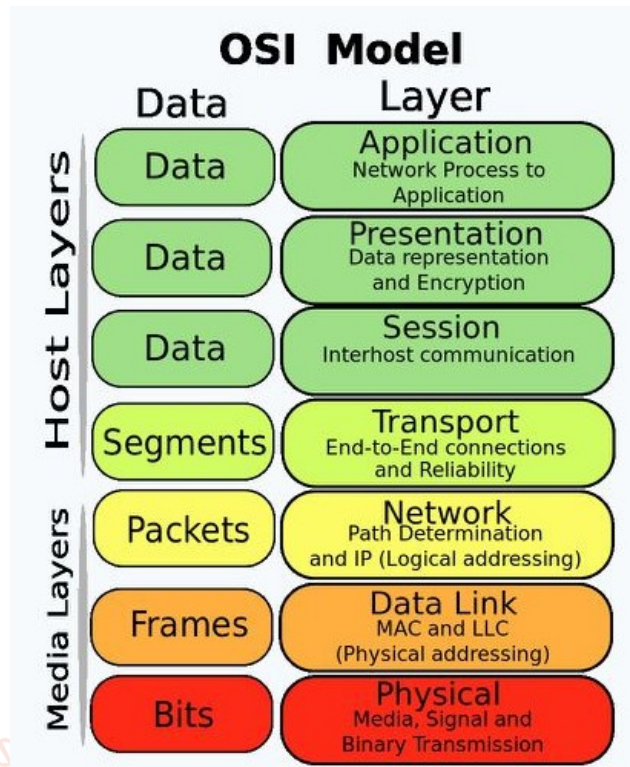
The layers are numbered and sit one above each other, and can therefore be thought of as a stack. The output from one layer provides the input for the next. Each layer interacts only with the layer directly above it and/or below it, and at the bottom of the stack, the Physical Layer transmits data between the source and destination devices:

Dividing protocols into layers provides several benefits:

➢ Different media (e.g. wired and wireless) are able to use the same protocols in most layers above the Physical Layer. This makes it easier for a manufacturer to introduce a new transmission medium.

➢ Similarly, when testing new equipment, developers only have to test the layers that contain protocols specific to that equipment (e.g. for a new router, testing would be needed at the Network Layer, but not the Transport Layer).

➢ Protocols are sometimes updated, and as the layers are separate and independent, the rules within them can be changed without affecting the rules contained in the other layers, as long as the interface to the layers above and below is kept the same.

➢ Transmission faults are more easily traced, as the type of error is often specific to a layer. A tester only has to investigate that layer to find the cause of the problem.

X.   OSI Model Benefits
By separating network communications into smaller logical pieces, the OSI model simplifies how network protocol designed. The OSI model was designed to ensure different types of equipment (such as network (adapter, hub and router) would all be compatible even if built by different manufacturers. A product from one network equipment vendor that implements OSI Layer 2 functionality, for example, will be much more likely to interoperate with another vendor's OSI Layer 3 product because both vendors are following the same model. The OSI model also makes network designs more extensible because new protocols and other network services are generally easier to add to a layered architecture than to a monolithic one.

XI.   COMPARISON BETWEEN TCP-IP & OSI

The three top layers in the OSI model, i.e. the application layer, the presentation layer and the session layer, are not distinguished separately in the TCP/IP model which only has an application layer above the transport layer. While some pure OSI protocol applications, such as X.400, also combined them, there is no requirement that a TCP/IP protocol stack must impose monolithic architecture above the transport layer. For example, the NFS application protocol runs over the external Data Representation (XDR) presentation protocol, which, in turn, runs over a protocol called Remote Procedure Call (RPC). RPC provides reliable record transmission, so it can safely use the best-effort UDP transport. Different authors have interpreted the TCP/IP model differently, and disagree whether the link layer, or the entire TCP/IP model, covers OSI layer 1 (physical layer) issues, or whether a hardware layer is assumed below the link layer. Several authors have attempted to incorporate the OSI model's layers 1 and 2 into the TCP/IP model, since these are commonly referred to in modern standards. This often results in a model with five layers, where the link layer or network access layer is split into the OSI model's layers 1 and 2. The IETF protocol development effort is not concerned with strict layering. Some of its protocols may not fit cleanly into the OSI model, although RFCs sometimes refer to it and often use the old OSI layer numbers. The IETF has repeatedly stated that Internet protocol and architecture development is not intended to be OSI-compliant. RFC 3439, addressing Internet architecture, contains a section entitled: "Layering Considered Harmful". For example, the session and presentation layers of the OSI suite are considered to be included to the application layer of the TCP/IP suite. The functionality of the session layer can be found in protocols like HTTP and SMTP and is more evident in protocols like Telnet and the Session Initiation Protocol (SIP). Session layer functionality is also realized with the port numbering of the TCP and UDP protocols, which cover the transport layer in the TCP/IP suite. Functions of the presentation layer are realized in the TCP/IP applications with the MIME standard in data exchange. Conflicts are apparent also in the original OSI model, ISO 7498, when not considering the annexes to this model, e.g., the ISO 7498/4 Management Framework, or the ISO 8648 Internal Organization of the Network layer (IONL). When the IONL and Management Framework documents are considered, the ICMP and IGMP are defined as layer management protocols for the network layer. In like manner, the IONL provides a structure for "sub network dependent convergence facilities" such as ARP and RARP

XII.   CONCLUSION

The Internet protocol suite does not presume any specific hardware or software environment. It only requires that hardware and a software layer exists that is capable of sending and receiving packets on a computer network. As a result, the suite has been implemented on essentially every computing platform. A minimal implementation of TCP/IP includes the following: Internet Protocol (IP), Address Resolution Protocol (ARP), Internet Control Message Protocol, Transmission Control Protocol (TCP), and User Datagram Protocol. In addition to IP, Internet Protocol version 6 requires Neighbor Discovery Protocol and IGMPv6 and is often accompanied by an integrated IPsec security layer. Application programmers are typically concerned only with interfaces in the application layer and often also in the transport layer, while the layers below are services provided by the TCP/IP stack in the operating system. Most IP implementations are accessible to programmers through sockets. Unique implementations include Lightweight TCP/IP, an open source stack designed for embedded systems a stack and associated protocols for amateur packet radio systems and personal computers connected via serial lines. Microcontroller firmware in the network adapter typically handles link issues, supported by driver software in the operating system. Non-programmable analog and digital electronics are normally in charge of the physical components below the link layer, typically using an application-specific integrated circuit chipset for each network interface or other physical standard. High-performance routers are to a large extent based on fast non-programmable digital electronics, carrying out link level switching.

REFERENCES

[1] S. Zaman S., F. Karray. Fuzzy ESVDF approach for Intrusion Detection System. The IEEE 23rd International Conference on Advanced Information Networking and Applications (AINA-09). May 26-29, 2009.

[2] I. Onut and A. Ghorbani. A Feature Classification Scheme for Network Intrusion Detection. International Journal of Network Security, Page(s): 1-15, July 2007.

[3] A. Tamilarasan, S. Mukkamala, A. Sung, and K. Yendrapalli. Feature Ranking and Selection for Intrusion Detection Using Artificial Neural Networks and Statistical Methods. 2006 International Joint Conference on Neural Networks (IJCNN'06), Page(s):4754-4761, July 16-21, 2006.

[4] A. Sung, S. Mukkamala. Identifying Important Features for Intrusion Detection Using Support Vector Machines and Neural Networks. Symposium on Application and Internet (SAINT'03), Page(s): 209-216, 27-31 Jan. 2003.

[5] V. Golovko, L. Vaitsekhovich, P. Kochurko and U. Rubanau. Dimensionality Reduction and Attack Recognition using Neural Network Approaches. International Joint Conference on Neural Networks, 2007, Page(s): 2734-2739, 12-17 Aug. 2007.

[6] S. Srinoy. Intrusion Detection Model Based On Particle Swarm Optimization and Support Vector Machine. The 2007 IEEE Symposium on Computational Intelligence in Security and Defense Applications (CISDA 2007), Page(s): 186-192, 1-5 April 2007.

[7] H. Gao, H. Yang, X. Wang. Ant Colony Optimization Based Network Intrusion Feature Selection and Detection. The Fourth International Conference on Machine Learning and Cybernetics, Guangzhou, Page(s): 18-21, August 2005.

[8] Kh. Shazzad, J. Sou Park. Optimization of Intrusion Detection through Fast Hybrid Feature Selection. The Sixth International Conference on Parallel and Distributed Computing, Applications and Technologies, 2005, (PDCAT'05), Page(s): 264 - 267, 05-08 Dec, 2005.

[9] M. Yasin, and A. Awan. A Study of Host-Based IDS using System Calls. INCC 204, International Conference on Networking and Communication 2004, on page(s): 36-41, June 2004.