

**TOPCIT**  
ESSENCE

TOPCIT ESSENCE Technical Field 03 Understanding of Network

# TOPCIT

## ESSENCE Ver.2

Technical Field  
03 Understanding of Network



# TOPCIT

## ESSENCE Ver.2

Technical Field

03 Understanding of Network



TOPCIT ESSENCE is published to provide learning materials for TOPCIT examinees.

The TOPCIT Division desires the TOPCIT examinees who want to acquire the necessary practical competency in the field of ICT to exploit as self-directed learning materials.

For more information about TOPCIT ESSENCE, visit TOPCIT website or send us an e-mail.

As part of the TOPCIT ESSENCE contents feed into authors' personal opinions, it is not the TOPCIT Division's official stance.

---

Ministry of Science, ICT and Future Planning  
Institute for Information and Communications Technology Promotion  
Korea Productivity Center

**Publisher** TOPCIT Division  
+82-2-398-7649 www.topcit.or.kr/en helpdesk@topcit.or.kr

**Date of Publication** 1<sup>st</sup> Edition 2014. 12. 10  
2<sup>nd</sup> Edition 2016. 2. 26

Copyright © Ministry of Science, ICT and Future Planning  
All rights reserved.  
No part of this book may be used or reproduced in any manner whatever without written permission.

# TOPCIT

## ESSENCE Ver.2

Technical Field  
03 Understanding of Network

# TOPCIT

ESSENCE Ver.2



Technical Field

03 Understanding of Network

<b>Introduction to Network</b>	<b>12</b>		
01 What is Protocol?	16		
02 OSI Reference Model and TCP/IP Protocol Layer Structure	17		
03 Internet Address Structure	19		
MAC Address, IP Address, Port Number	19		
IPv4 Address Structure	20		
IPv6 Address Structure	23		
04 Internet Standards	24		
<b>Data Link Layer and Physical Layer</b>	<b>28</b>		
01 Basic Idea about Data Link Layer	30		
02 Data Link Layer Encapsulation	30		
Data Link Layer Encapsulation	30		
Frame Header and Trailer	31		
03 Structure of Data Link Layer	31		
Sub-layer of Data Link Layer	31		
Logical Link Control	31		
MAC	32		
04 MAC Address Search	32		
IP Address and MAC Address Resolution Protocol	32		
		MAC Address Search Scenario	33
		<b>05 Error Detection and Correction in Data Link Layer</b>	<b>34</b>
		Definition of Error Control	34
		Error Detection and Error Correction	35
		<b>06 Physical Layer</b>	<b>36</b>
		Operation Scenario	36
		Signal Delivery Method	37
		<b>07 Classification of Physical Layer Medium</b>	<b>37</b>
		Transmission Medium	37
		Guided Medium	38
		Wireless Medium	41
		<b>08 IEEE 802 Standard</b>	<b>41</b>
		Basic Concept of IEEE 802	41
		IEEE 802,3 Standard	42
		IEEE 802,11 Standard	42
		IEEE 802,15 Standard	43
		<b>Understanding Routing Protocols and the IPv4 Address System and to Utilize the Same</b>	<b>47</b>
		01 Outline of Network Layer and Device	49
		What is Network Layer?	49
		Function of Network Layer	49
		Internetworking Device	50

# CONTENTS

What is Router?	50	IPv4 Addressing Structure	62
What is Switch?	52	Special IPv4 Address	63
VLAN (Virtual Local Area Network)	53	Subnetting	64
<b>02 Encapsulation of Network Layer</b>	<b>53</b>	CIDR (Classless Inter-Domain Routing)	64
Encapsulation of Network Layer	53	Supernetting	65
IPv4 Header	54	How IPv4 Address is Assigned?	66
<b>03 Packet Switching and Network Layer Protocol/Command</b>	<b>54</b>	<b>Transport Layer Protocol</b>	<b>69</b>
Packet Switching	54	<b>01 Concept of Transport Layer Protocol</b>	<b>70</b>
Network Layer Protocol/command	55	<b>02 TCP</b>	<b>71</b>
Network Layer Command	55	Characteristics of TCP	71
<b>04 Network Service Quality</b>	<b>56</b>	Scenario for TCP Operation	73
QoS(Quality of Service)	56	TCP Protocol	74
Techniques Used to Secure Quality of Service	56	<b>03 UDP (User Datagram Protocol)</b>	<b>83</b>
<b>05 Routing Protocols and Algorithms</b>	<b>58</b>	Characteristics of UDP	83
What is Routing Protocol?	58	UDP Protocol	84
What is Routing Algorithm?	58	Use Case of Multicast Sockets via UDP	86
Types of Routing Protocols	59	<b>04 SCTP(Stream Control Transmission Protocol)</b>	<b>88</b>
Types of Routing Protocols	60	Features of SCTP	88
<b>06 Outline of IPv4</b>	<b>60</b>	SCTP Protocol	88
What is IPv4 (Internet Protocol Version 4)?	60	<b>Application Layer Technologies, including Web Applications</b>	<b>94</b>
An Example of a Network Using the IPv4	61	<b>01 What is Application Layer Protocol?</b>	<b>97</b>
<b>07 IPv4 Addressing and Subnetting</b>	<b>62</b>		
IPv4 Expression	62		



# CONTENTS

<b>02 HTTP</b>	<b>97</b>		
Characteristics of HTTP	97		
HTTP Protocol	97		
<b>03 File Transmission Protocol</b>	<b>101</b>		
Characteristics of FTP	101		
FTP Protocol	102		
Example of Actual FTP Protocol	105		
Example of FTP Implementation with Library	112		
<b>04 JSP Web Programming</b>	<b>118</b>		
Characteristics of Web Programming	118		
Examples of Web Programming	119		
<b>Keep up with the Recent Trends in Network Technology</b>	<b>131</b>		
<b>01 Multimedia Network</b>	<b>132</b>		
Type of Image Compression	132		
Multimedia Data	132		
QoS(Quality of Service)	133		
<b>02 Basic Idea about VoIP and Call Signaling Protocol</b>	<b>135</b>		
What is Voice over Internet Protocol (VoIP)?	135		
VoIP call Signaling Protocol	136		
H.323	138		
<b>03 Media Transport Protocol</b>	<b>139</b>		
Types of Media Transport Protocol	139		
		RTP(Real-time Transport Protocol)	139
		RTCP(Real-time Transport Control Protocol)	141
		IMS (IP Multimedia Subsystem)	141
		<b>04 Network Technologies for IoT (Internet of Things)</b>	<b>144</b>
		Introduction to IoT	144
		Trend of IoT standardization	145
		Core Technology of IoT	147
		Major Protocols for IoT	147
		<b>05 Software Based Network</b>	<b>149</b>
		Limitation of Conventional Communications Environment and Paradigm Shift	149
		SDN (Software Defined Network)	150
		NFV (Network Function Virtualization)	153
		SDN & NFV	154

# Introduction to Network

## ▶▶▶ Latest Trends and Key Issues

The network technology has been developed from 56kbps wired data communications to tens of Mbps level wireless network. The technology has become an integral part of our lives not only in business domains but also in our daily lives because of the emergence of diverse connected terminals. Hence, any software used in a terminal or in a system has close interaction with network and bi-directional communications feature comes as default. Hence, it is necessary to understand the standard protocols used in the network so that we can design and build agile and optimized software out of the basic principles about the network.

## ▶▶▶ Study Objectives

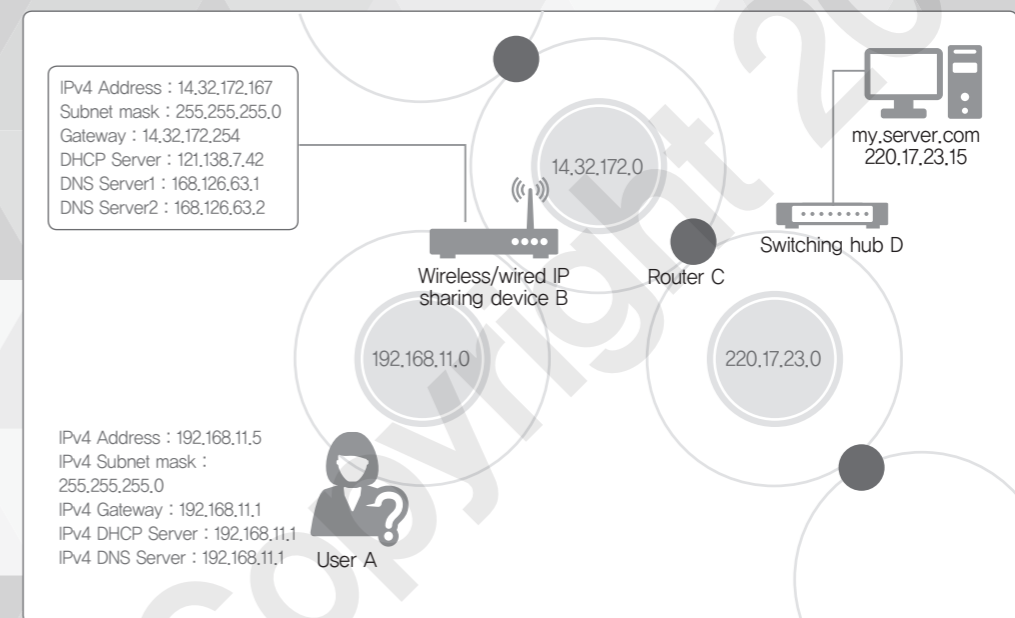
- \* To be able to explain about the basic theory of the Internet and protocol layers
- \* To be able to understand and utilize MAC addresses, IP addresses, and port numbers that are used for the Internet address structure.
- \* To be able to explain how data is transmitted on the network and to identify which standard protocol is used during the transmission

## ▶▶▶ Practical Importance Medium

## ▶▶▶ Keywords

Protocol, internet, IETF, 3GPP, 3GPP2, ITU-T, OSI reference model, Application layer, Presentation layer, transport layer, Network layer, Data link layer, Physical layer, Internet protocol layer, Internet layer, Network interface layer, Internet address system, MAC address, IP address, Port number, IPv4 address, IPv6 address, Private network, NAT, CIDR, RFC

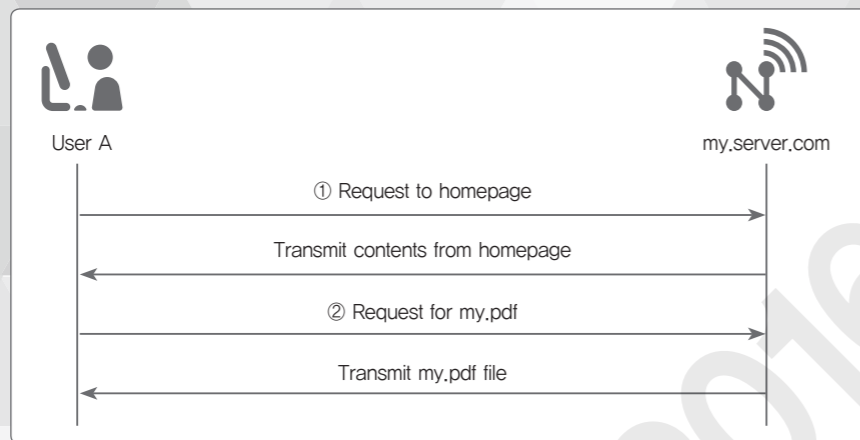
## + Practical tips How to understand network?



(Figure 1) Networking between a user and a server

(Figure 1) displays how network connection works when User A wants to search for information after having an access to a web server (my.server.com). This kind of connection frequently takes place at home, school, or company but not many users are fully aware of the network information they are using. Let's take a specific example of how network service is provided.





(Figure 2) User A's web usage sequence

(Figure 2) shows a case where User A is connected to a web server (my.server.com) and tries to search for the information, whereas (Table 1) provides a step-by-step explanation about what kind of changes are happening on the network corresponding to the activities shown in (Figure 2). The standard protocol used in each step is written in parenthesis and the definition and mechanism of action for each of them will be explained in the following chapters.

Let's take (Figure 2) as an example and look into how a simple network service can be provided.

(Table 1) User A's web usage scenario

- Step 1** User A types in `http://my.server.com/` on a web browser and hits the enter key.
- Step 2** The web browser wants to send out the request of **User A** to the web server (**my.server.com**). However, **my.server.com** is a domain name that can only be easily recognized by humans, so the browser cannot send out the request. Therefore, the web browser has to ask for an IP address given to **my.server.com** to a Domain Name Server.(DNS)
- Step 3** The DNS gives back the IP address (220.17.23.15) for **my.server.com** to the web browser. The web browser gains access to the web server using this IP address and asks for the information in the document root.(HTTP)
- Step 3-1** To deliver the request to **my.server.com**, the web browser hands over the request information to the TCP protocol. This information is encapsulated properly and delivered to a web server program of **my.server.com**. (TCP)
- Step 3-2** The information generated from TCP protocol goes through various network hops and is encapsulated again so that the information can be routed to 220.17.23.0 where **my.server.com** is connected. (IP)

- Step 3-3** A packet is generated, including the information for inter-network routing. However, this packet cannot move from one network to another and needs some more preparation within the network. User A's computer has the IPv4 address of 192.168.11.5, which cannot be directly delivered to **my.server.com**. Hence, the computer tries to deliver the address to its gateway: **wireless/ wired IP sharing device B** (192.168.11.1). However, the IP address cannot be used for transmission within the network, so it is necessary to get the MAC address information which is designed for transmission within the network. The MAC address matched to the gateway's IP address (192.168.11.1) should be found out by conducting broadcasting within the 192.168.11.0 network, and the packet is sent to **IP sharing device B**. (ARP, Ethernet, private network, NAT)
- Step 3-4** The **wireless/wired IP sharing device B** receives the packet from **User A's** computer and looks into the final destination of the packet because the packet is not directed to the IP sharing device. The IP header contains the final destination (**my.server.com**) and the **IP sharing device B** can recognize that the final destination is not for 192.168.11.0 or 14.32.172.0 where the IP sharing device belongs. The IP sharing devices passes the packet to **Router C** where it belongs. (Routing protocol)
- Step 3-5** Router C can recognize that the packet is directed to the network 220.17.23.0 where it belongs and tries to send the packet to the web server's IP address 220.17.23.15. Just as in the case of step 3.3, the packet movement within the network requires a physical address of **my.server.com** or MAC address. Hence, the broadcasting job is conducted within the network to acquire the MAC address corresponding to **my.server.com's** IP address (220.17.23.15) and the packet can be delivered to the address.
- Step 3-6** **My.server.com** receives the packet, and looks into the IP address to identify whether the packet is directed to the right address. After that, the server removes the header information and sends the remaining data to a higher protocol layer. (IP)
- Step 3-7** The data is delivered to the application of the web server (**my.server.com**) after identifying to which application of the web server the data should be delivered and then unnecessary information is removed.
- Step 3-8** The web server application recognizes that the message is intended to request information from the document root and transmits the relevant information to **User A's** computer.
- Step 4** The web browser, which receives the information in the document root from the web server, visually presents the information for **User A**. If a hyperlink within the document is selected, it is possible to view a video clip on the web browser. (HTTP, HTML)

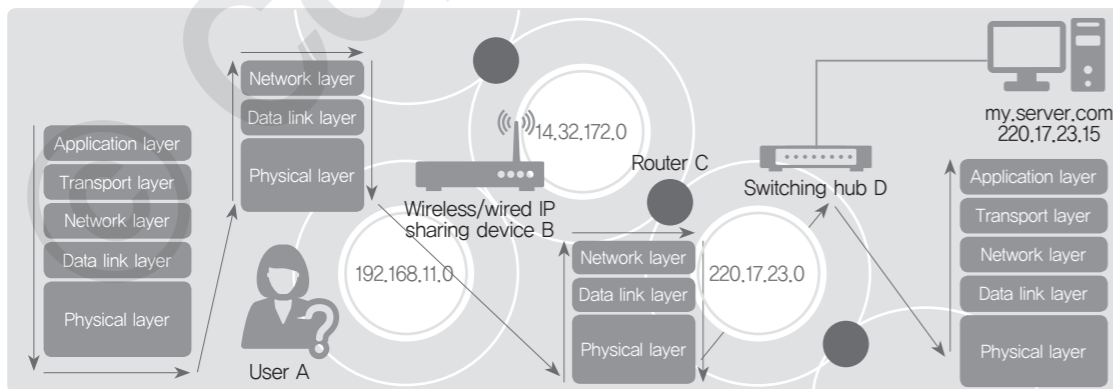
## 01 What is Protocol?

In the previous (Table 1) ARP, IP, TCP, HTTP, DNS, NAT and many other standard protocols were introduced, A standard protocol can be defined as a standardized communication rule which is intended to send and receive data through the network. In general, the Internet has a wide network wherein multiple networks come together and each of the networks is connected with many PCs and servers.

There are tens of thousands or hundreds of thousands of PCs, servers, and routers which connect these PCs and servers. For data communication among these many devices, it is absolutely necessary to have a common protocol that can be understood from any devices. That is why there are many international organizations that make efforts to standardize these protocols.

- IETF(Internet Engineering Task Force) usually takes care of Internet-related protocols
- 3GPP(Third Generation Partnership Project) and 3GPP2 cover wireless communications protocol such as GSM, CDMA, UMTS, LTE, LTE-A
- ITU-T(International Telecommunication Union Telecommunication Standardization Sector) works on the standardization for telecom communications

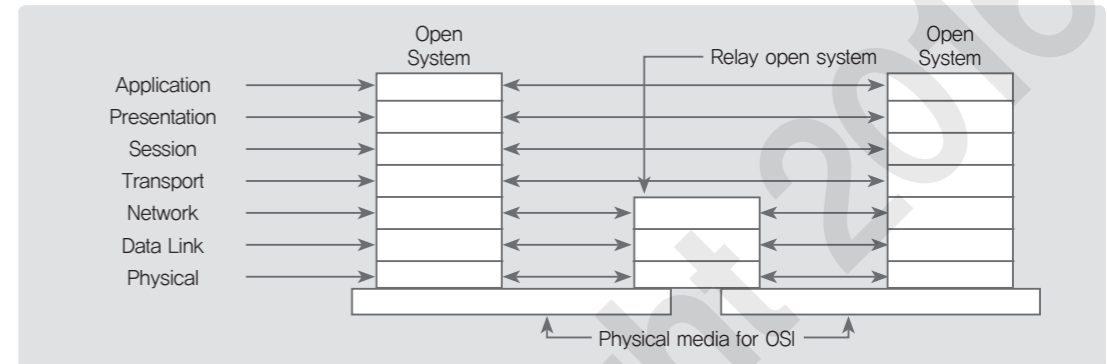
Therefore, in order to develop the software for network / telecommunications industries, it is necessary to understand the standard protocols established by these standardization bodies and to be able to accommodate those protocols during (Figure 3) showcases the basic ideas about the network protocol discussed above and to describe the contents of (Figure 1). If a user searches for information via a web browser, the request and response data are moving over the network based on the protocols that are relevant to the user's computer, wireless or wired IP sharing device, switching hubs, and routers. The physical layer is a directly connected with hardware such as physical wires or switch and is operated by an electrical signal, but the protocol layers above the data link layer (including the data link layer) is implemented and operated with software. If one can understand how many software components interact with each other, it is possible to develop application program or system software via network. There will be further step-by-step explanation for each of the protocol layers shown in (Figure 3): especially how they are operated and interconnected.



(Figure 3) A virtual network that presents basic ideas about protocol layer

## 02 OSI Reference Model and TCP/IP Protocol Layer Structure

(Figure 4) is a conceptual diagram for communications based on the OSI reference model, including relay open system; described in ISO / IEC 7498-1: 1994 (E). This figure is cited often when talking about the so called ISO's OSI reference model or OSI 7 layers. ISO originates from the Greek word 'ισο'(Romanization isos), which means "equal". One must be careful not to be confused with 'International Organization for Standardization (ISO)'. OSI stands for Open Systems Interconnection.



(Figure 4) Conceptual diagram for communications based on OSI reference model, including relay open system

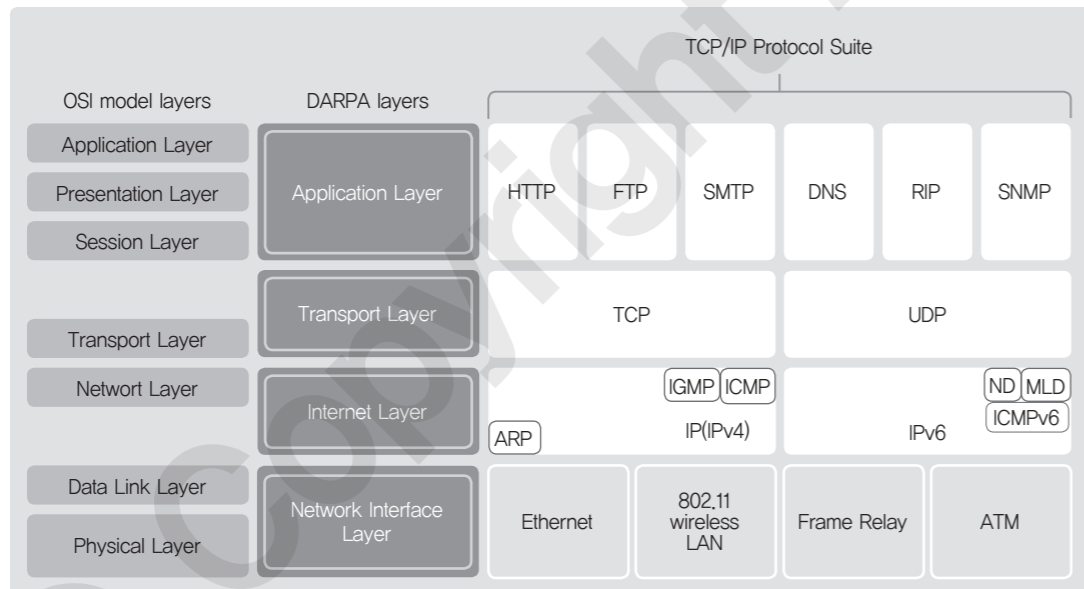
The protocols and their functions for each layer of the OSI reference model are summarized in Table 2. [1]

(Table 2) Protocols and relevant functions for each layer of OSI reference model

Layer	Protocol	Function
Application layer	HTTP, SMTP, SNMP, FTP, Telnet, etc.	Providing services such as user interface, e-mail, database management, etc.
Presentation layer	JPEG, MPEG, XDR, etc.	Supporting encoding translation and encryption for syntax and semantics exchanged between two systems.
Session layer	TLS, SSH, RPC, NetBIOS, etc.	The layer that builds sessions for communications: establishing and maintaining interaction among communication devices and synchronizes them.
Transport layer	TCP, UDP, SCTP, etc.	Providing reliable message transmission between end-to-end processes and error control
Network layer	IP, IPX, ICMP, X.25, ARP, OSPF, etc.	Supporting packet transmission between networks from starting point to destination.
Data link layer	Ethernet, Token Ring, wireless LAN, etc.	Providing functions which transports frames between hops without errors.
Physical layer	Radio wave, optical fiber, PSTN, etc.	Transmitting actual bit stream through physical medium.

The Internet protocol layer is based on the ARPANET reference model, as shown in (Figure 5), which came out earlier than the OSI layer model [3] [9] [10]. The term and expression can slightly vary from one source to another, but in this chapter, we use the most commonly used term.

- ① **Application layer** : allows applications (including web (HTTP), DNS, Telnet, FTP, e-mail transmission (SMTP / POP3 / IMAP4)) to gain access to the service of other layers. This layer covers not only the application layer but also the presentation layer and session layer of an OSI reference model.
- ② **Transport layer** : also called a host-to-host transport layer and responsible for data exchanges among abstract ports such as TCP, UDP, and SCTP which are managed by application programs. This layer is equivalent to the transport layer of an OSI model.
- ③ **Internet layer** : also known as a network layer and responsible for addressing and routing. This layer corresponds to the network layer of an OSI model.
- ④ **Network interface layer** : also called a network access layer and responsible for actual transmission and reception of TCP/IP packets through physical medium, such as IEEE 802.3 Ethernet or IEEE 802.11 WiFi. When compared to the OSI model, this layer is equivalent to the data link layer that takes care of MAC function and Physical layer that defines electrical signal.



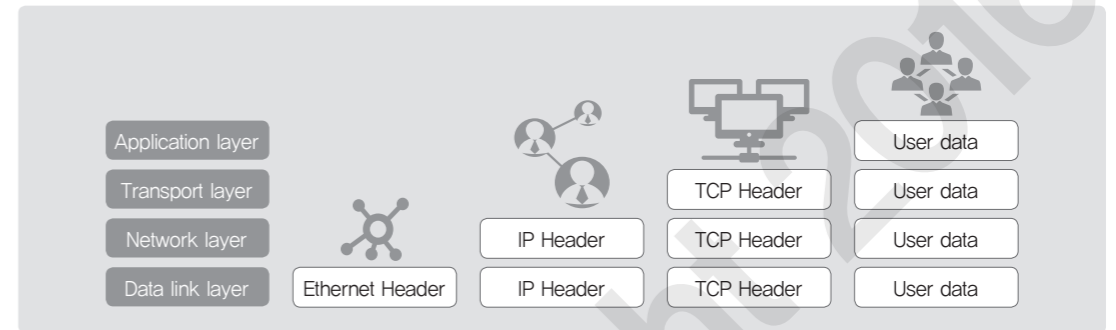
〈Figure 5〉 TCP / IP protocol layers: based on ARPANET reference model [10]

As was expressed in 〈Figure 4,〉 these protocols define the rules for communications between one protocol layer and the equivalent protocol layer of the other host which are connected through a network. In general, in order to make communications possible, it is necessary to obtain the header information so that the same level layers can communicate with each other and to obtain the data passed down from the layer one level higher.

Just as in 〈Figure 6〉, the user data to be sent from the application layer can be transmitted through the **transport layer protocol TCP (or UDP)**, which is connected to the host of the other end. In this process, it is necessary to obtain a TCP header (or UDP header) to carry out the service.

The network layer protocol (IP), which contains the routing information that can cover the network where the target host resides, is used to transport user data (including TCP header information) and such IP information is included within the header.

In the network layer, the TCP header and user data received from the transport layer are all regarded as data. As for the data link layer, which contains information necessary for physical delivery to the other host within the same network, Ethernet protocol is usually used. In this process, the pair of Ethernet header and IP header passed down from the network layer (TCP header and user data) is regarded as data. It is good to remember that each layer only uses the header that is relevant to the given role of each layer.



〈Figure 6〉 Example of network protocol header and data composition

### 03 Internet Address Structure

#### MAC Address, IP Address, Port Number

You will encounter terminologies that contain various numbers such as MAC address, IP address, and port number when you are using the Internet. It can be helpful if you understand the meaning of the protocols of each layer rather than just memorizing them. Just as in the case of the Internet address, there are numbers that comprise MAC address, IP address, and port number.

- ① **MAC Address** : MAC (Medium Access Control) is an addressing system used in the data link layer, which is usually used for frame transport between the nodes that are physically connected. The information is stored in the LAN expansion card (also known as NIC (Network Interface Card)) in the form of hardware, and MAC address is also called Physical Address or EHA (Ethernet Hardware Address). The address has 48 bits, wherein the first 24 bits are used for unique ID of a hardware manufacturer and the last 24 bits are for NIC-dependent address.
- ② **IP address** : IP address refers to the address system used in the network layer, which is used in transporting datagram between two hosts/routers. IP address helps right data be transported from the start node to the destination node through the various networks that belong to the Internet. While MAC address is called a physical address, IP address is called a logical address. The existing IP address structure or the IPv4 address is composed of 32 bits, but there has been an exponential increase of the Internet usage globally and the address structure did not have enough vacancy. Hence, there has been discussion for the adoption of the IPv6 which has

128 bits, but still the IPv4 address is mainly used. IANA (Internet Assigned Numbers Authority) used to take the responsibility of IP address allocation, but ICANN (International Corporation for Assigned Names and Numbers) is doing the role these days.

- ③ **Port number** : Port number is used to transport messages between the two processes (the application that is currently running) and it can be simply said the port number connects the web browser and web server program of my.server.com shown in (Figure 1). It is composed of 16 bits and can identify up to 65535 application programs. The number from 0 to 1023 are also called **well-known ports** and allocated by the IANA. These numbers are usually used for the application program that has been used for a long time such as Telnet or e-mail communications. The number from 1024 to 49,151 are called registered ports and usually used for servers just as in the case of well-known ports. Typical examples are the Internet chatting protocol IRC (TCP port number 6667) and Git version control system (TCP port number 9418). The number from 49,152 to 65,535 are usually used on the client side on a temporary basis. They are also called **dynamic ports** or **ephemeral ports**.

### IPv4 Address Structure

IPv4 address, composed of 32 bits, is used in the IP layer, and provides a unique but universally used identifier for a router or a host to gain access to the Internet. In general, the IPv4 address is classified into five classes as shown in (Figure 7). Class A, B, and C are for the Internet, while Class D is for multicast and Class E is for research and development.

IPv4 address can be divided into two parts: **network identifier** and **host identifier** within the network. The length of the network identifier for Class A, B, and C is 8 bits, 16 bits, and 24 bits respectively. Out of the whole 32 bits, the remaining bits other than the network identifier are used as host identifier.

That means Class A network can have 24 bits as a host identifier (after excluding 8 bits for network identifier), so  $2^{24} - 2$  hosts can be utilized. The reason '2' was subtracted because if all the bits of the host identifier equals to zero, the IPv4 address is the **network address**, while if all the bits of the host identifier equals to one, it means **direct broadcast address**. When the network identifier and host identifier have specific value, they have special meaning rather than indicating a specific host. (Table 3) summarizes types of 'special addresses'.

(Table 3) Special IPv4 address types

Special address	Network identifier	Host identifier
Network address	Specific	0
Direct broadcast address	Specific	All= 1
Limited broadcast address	All= 1	All= 1
Host within the network	All= 0	All= 0
Specific host within the network	All= 0	Specific
Loopback address	127	Don't Care

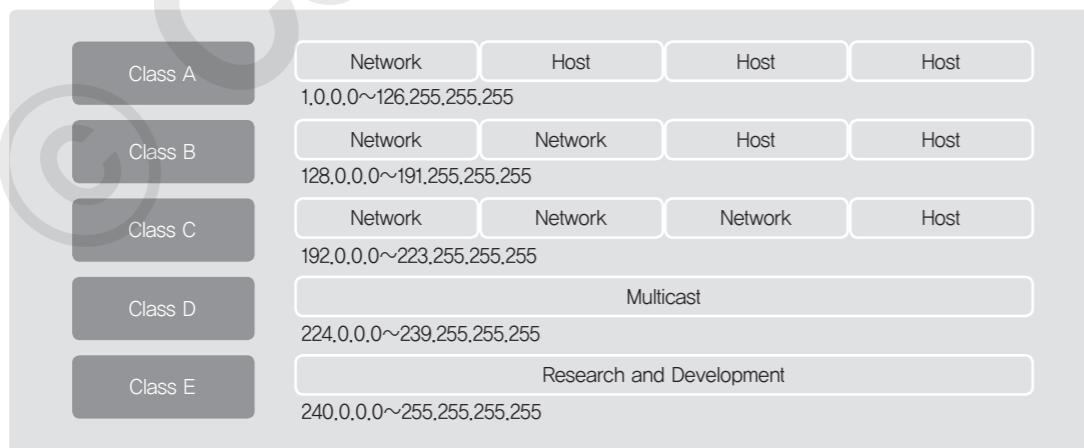
Some of the IPv4 address space is dedicated for **private address space**. The network that uses the private address space is called a **private network** and the network should comply with RFC 1918[11] and RFC 4193[12].

IETF has established RFC 1918 and makes recommendations about Class A, B, and C level address specifically for independent networks that are not linked to the internet.

IPv4 address is allocated either by IANA or ICANN, so an individual cannot choose to use a random IPv4 address. However, if you build your own private network, it is not only easy to configure but also the IPv4 address within the private space can be allocated for individuals. If you use **NAT (Network Address Translation)** you can connect the private network to the public Internet.

(Table 4) Private address space

RFC 1918 name	Range of IP address	Number of addresses
24bit block	10.0.0.0 ~ 10.255.255.255	16,777,216 (Class A: 1)
20bit block	172.16.0.0 ~ 172.31.255.255	1,048,576 (Class B: 6)
16bit block	192.168.0.0 ~ 192.168.255.255	65,536 (Class C: 256)



(Figure 7) IPv4 address classes

**NAT (Network Address Translation)** can provide mapping between private address and global address, supports virtual private network [15], and works as a core technology for networking firewall. In (Figure 1), you can see the network where User A's computer and wireless/wired IP sharing device B are located, the network (192.168.11.0) is an example of a private network. NAT is working based on the ideas of reusing the private address. Such ideas can be implemented within a router, which means the router can receive data from each of the ports and translates IP address field of IP packet's initiation point into unique public IP address, following the NAT mapping rules. The types of NAT are explained in (Table 5).

<Table 5>Types of NAT

NAT type	Detail
Static NAT	One on one mapping between an internal network address and an external network address
Dynamic NAT	Dynamic mapping between the address pool of internal network and one external network address
Port NAT (PAT)	Communications with external network: between a number of internal network addresses and one or multiple external network addresses with different port numbers

<Figure 8> shows the details about network connection within the Windows environment. The IPv4 address allocated to the computer is 192.168.11.4 which falls in the private address space of Class C. Sub-net mask is 255.255.255.0 and the bit value=1 refers to a network identifier. The outcome of XOR between the IPv4 address (192.168.11.4) and Sub-net mask (255.255.255.0) is 192.168.11.0 and this value works as the network identifier and 4 (difference against 192.168.11.0) works as the host identifier for the network.



<Figure 8> Details of network connection within Windows environment

There is a more flexible way of address allocation compared to the existing "class-based" address allocation. It is called **CIDR (Classless Inter-Domain Routing)** and pronounced either as /saidə(r)/ or /sidə(r)/. In this method, it is possible to have more flexible segmentation of IP address range compared to the class-based method. The method uses "IP address/prefix size". To be more specific 192.168.11.4 (Subnet mask 255.255.255.0) can be expressed as 192.168.11.4/24. The existing Class A, B, and C have a prefix length of 8, 16, 24 bits respectively, so this method is useful in describing the network with different prefix size. <Figure 9> clearly shows how CIDR works without classes which were described in RFC 4632.

notation	addrs/block	# blocks	
n.n.n.n/32	1	4294967296	"host route"
n.n.n.n/31	2	2147483648	"p2p link"
n.n.n.n/30	4	1073741824	
n.n.n.n/29	8	536870912	
n.n.n.n/28	16	268435456	
n.n.n.n/27	32	134217728	
n.n.n.n/26	64	67108864	
n.n.n.n/25	128	33554432	
n.n.n.0/24	256	16777216	legacy "Class C"
n.n.x.0/23	512	8388608	
n.n.x.0/22	1024	4194304	
n.n.x.0/21	2048	2097152	
n.n.x.0/20	4096	1048576	
n.n.x.0/19	8192	524288	
n.n.x.0/18	16384	262144	
n.n.x.0/17	32768	131072	
n.n.0.0/16	65536	65536	legacy "Class B"
n.x.0.0/15	131072	32768	
n.x.0.0/14	262144	16384	
n.x.0.0/13	524288	8192	
n.x.0.0/12	1048576	4096	
n.x.0.0/11	2097152	2048	
n.x.0.0/10	4194304	1024	
n.x.0.0/9	8388608	512	
n.0.0.0/8	16777216	256	legacy "Class A"
x.0.0.0/7	33554432	128	
x.0.0.0/6	67108864	64	
x.0.0.0/5	134217728	32	
x.0.0.0/4	268435456	16	
x.0.0.0/3	536870912	8	
x.0.0.0/2	1073741824	4	
x.0.0.0/1	2147483648	2	
0.0.0.0/0	4294967296	1	"default route"

<Figure 9> CIDR address format [14]

### IPv6 Address Structure

The number of hosts and terminals connected to the IPv4 network increased significantly and the IPv4 space was running out and Internet security vulnerability issue emerged. To address these problems, the IPv6 protocol was developed in the mid 1990s and fundamentally resolved the space issue. As opposed to the existing IPv4 (32 bits), the IPv6 can be characterized with bigger space (128 bits) and can accommodate 3.4 x 10<sup>32</sup> addresses. In addition, the header field, which was not used often in the IPv4, was removed to make the header format simpler and thereby providing more effective QoS.

IPv4 did not have any security measures within itself and additionally required IPsec (Internet Protocol Security) [17] [18] for security features. However, the IPv6 provided end-to-end encryption feature for better security and privacy protection.

IPv6 has 128 bits, which is 4 times longer than the IPv4 address, and expressed in a hexadecimal number. The new address structure has 'Format Prefix' which is used to describe the form of address and 'Address Field' which literally describes the actual address. To make the long address easy to read, the address is divided by a colon in every 16 bits and each field is expressed in hexadecimal number. The prefix of IPv6 has a similar structure with the CIDR of IPv4; i.e. IPv6 address/length of network prefix in a decimal number. <Table 6> shows the typical example of the IPv6 prefix.

<Table 6> IPv6 format prefix

FP	Description
0000 001	Reserved for NSAP (Network Service Access Point)
0000 010	Reserved for IPX (Internetwork Packet Exchange)
001	Aggregatable global unicast address
1111 110 10	Link local address

FP	Description
1111 1110 11	Site local address
1111 1111	Multicast address

IPv6 addresses can be categorized into **link-local**, **uni-cast**, **multi-cast**, and **any-cast addresses** and (Table 7) summarizes how they are used and how they are presented.

(Table 7) IPv6 address structure

Category	Details
Link-local addresses	<ul style="list-style-type: none"> <li>The address that can be used only within an internal network.</li> <li>Address that can be utilized within a single network; used for identification only within the network.</li> </ul> <div style="border: 1px solid black; padding: 5px; margin: 10px 0;"> <b>FE80:0000:0000:0000:2CDA:D834:EC83:CBA2</b> </div> <p style="margin-left: 40px;">↓ Fill with 0 up to 64 bits      ↓ 10bit Prefix                                  EUI-64ID</p>
Global uni-cast addresses	<ul style="list-style-type: none"> <li>While link-local address can be used within a single network, global uni-cast address can be used to external networks as well</li> </ul> <div style="border: 1px solid black; padding: 5px; margin: 10px 0;"> <b>2001:DB8:131F:0000:0000:070D:126A:140B / 64</b> </div> <p style="margin-left: 40px;">↓                                  ↓                                  ↓ ② Network Address      ③ Interface      ① Prefix</p>
Multi-cast addresses	<ul style="list-style-type: none"> <li>To transport the packet to all the interfaces that are registered in the multi-cast group</li> <li>Format prefix is 8 bits and FF, and followed by 4-bit flag, 4-bit scope, and 112-bit group ID field.</li> <li>The first 3 bits of the 4-bit flag are not used. If the final bit is '0', it is well-known multi-cast address. If the value is "1", it is a temporarily used multi-cast address.</li> </ul>
Any-cast addresses	<ul style="list-style-type: none"> <li>It is similar to multi-cast address in that multiple interfaces are selected, but the packet is not transported to all the hosts within the group but the packet is transported only to the nearest interface.</li> <li>Presented with n-bit FP and (128-n) bit 0</li> </ul>

## 04 Internet Standards

IETF (Internet Engineering Task Force) is one of the main institutes in charge of the internet related standards. The

institute defines: basic transport protocols such as IP, TCP, UDP; and most widely used internet application layer protocols such as HTTP, SSH, FTP, SMTP, POP3, IMAP. In addition to the IETF, there are other international institutes working to manage the internet related standards as shown in (Table 8).

(Table 8) Institute for internet standardization

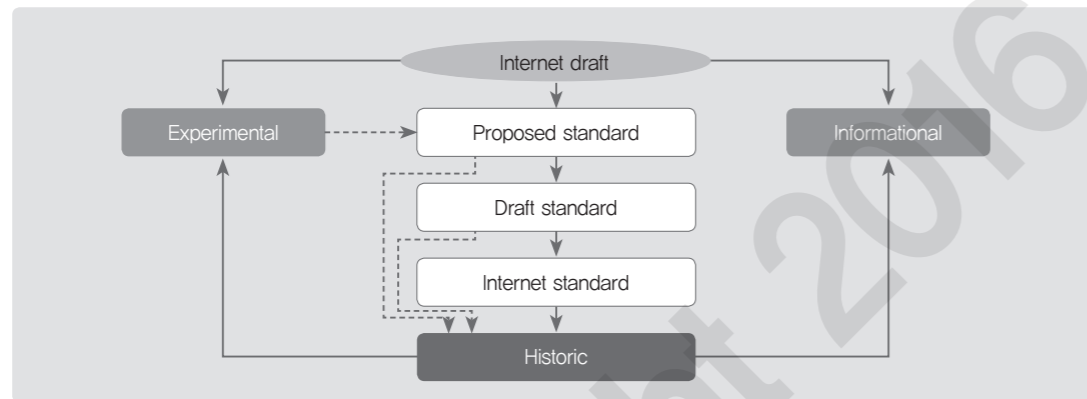
Standardization institute	Details	Standards (Ex)
ISO	<ul style="list-style-type: none"> <li>International Organization for Standardization</li> <li>An international body established in 1946 for global cooperation in intellectual activities and in the fields of science, technology, and economy.</li> </ul>	OSI reference model
ITU-T	<ul style="list-style-type: none"> <li>International Telecommunications Union-Telecommunication Standards Sector</li> <li>A part of ITU specialized for telecommunication standards</li> </ul>	ITU-T G.711
IEEE	<ul style="list-style-type: none"> <li>Institute of Electrical and Electronics Engineers</li> <li>A specialized institute authorized to develop American national standards by American National Standards Institute</li> </ul>	IEEE 802.3 IEEE 802.11 IEEE 802.15
EIA	<ul style="list-style-type: none"> <li>Electronic Industries Association</li> <li>An association planning for a uniform standard in size, measurement method, and marking in electronic devices.</li> </ul>	TIA/EIA-568-B (T568B)
W3C	<ul style="list-style-type: none"> <li>World Wide Web Consortium</li> <li>An international consortium for the Internet that created the web standard: established in 1994 and working for long-term development of the web.</li> </ul>	HTML 4.01/5 CSS 2/3
OMA	<ul style="list-style-type: none"> <li>Open Mobile Alliance</li> <li>A forum for the technological standard development and validation of interoperability under the purpose of boosting global mobile data service</li> </ul>	Push-to-talk over Cellular (PoC)

In addition, the IETF published RFC (Request for Comments) as a part of the Internet standard, the maturity level of RFC is defined as in (Table 9). This maturity level is usually written at the very top of the first page of RFC document. Just as shown in (Figure 10), if a standard is used by many people, it can be initiated with the Draft standard level and sometimes becomes the Internet standard or sometimes goes away after a period of experimental usage.

(Table 9) RFC document category

Maturity level	Details
Proposed standard	<ul style="list-style-type: none"> <li>A stabilized standard that has been made with a lot of effort and gone through enough discussion in the Internet community</li> </ul>
Draft standard	<ul style="list-style-type: none"> <li>More than two independent success and interoperability validated.</li> <li>Issues to be corrected continuously</li> </ul>
Internet standard	<ul style="list-style-type: none"> <li>A standard which was fully implemented</li> </ul>
Historic	<ul style="list-style-type: none"> <li>A standard that has yet to pass a process to be the Internet standard</li> <li>Has important meaning from a historic perspective</li> </ul>

Maturity level	Details
Experimental	<ul style="list-style-type: none"> <li>• An experimental working standard that does not have any impact on the internet operation.</li> <li>• Might not be implemented as a part of the Internet service</li> </ul>
Informational	<ul style="list-style-type: none"> <li>• General and historic tutorial information related to the Internet</li> </ul>



〈Figure 10〉 RFC standardization maturity level

## Example Question

### Question type

Descriptive question

### Question

Please write down three 'address' or 'number' structure used for sending and receiving data among the hosts on the internet and describe their purpose/usage

### Intent of the question

In order to understand the end-to-end data flow on the internet, it is necessary to comprehend the addresses and numbers used in each layer.

### Answer and explanation

- ① MAC address: physical address used to transport frame of the data link layer
- ② IP address: logical address used to transmit packets in between networks in the network layer.
- ③ Port number: a number structure used to deliver messages between two processes (or number structure used to transport segment of the transport layer)

## Related E-learning Contents

- Lecture 1 Basic Concept of Network

## Data Link Layer and Physical Layer

### ▶▶▶ Latest Trends and Key Issues

Communications, by definition, means various data are sent and received through the physical medium using an electric or optical signals. The physical layer is located at the bottom of the OSI seven layers. The purpose of this module is: to learn about the types of media and protocols working specifically for the physical layer and eventually understand the mechanism of action happening at the bottom of a network; and to understand the data link layer that works as a bridge between the medium and protocols of the physical layer and the upper layers; and to learn about diverse methods used in medium access control and logical link control.

### ▶▶▶ Study Objectives

- \* Able to explain about the standards of the physical layer
- \* Able to explain about the sub-layer of the data link layer (data link control, medium access control)
- \* Able to explain about the data link layer error detection and error correction technique.

### ▶▶▶ Practical Importance Medium

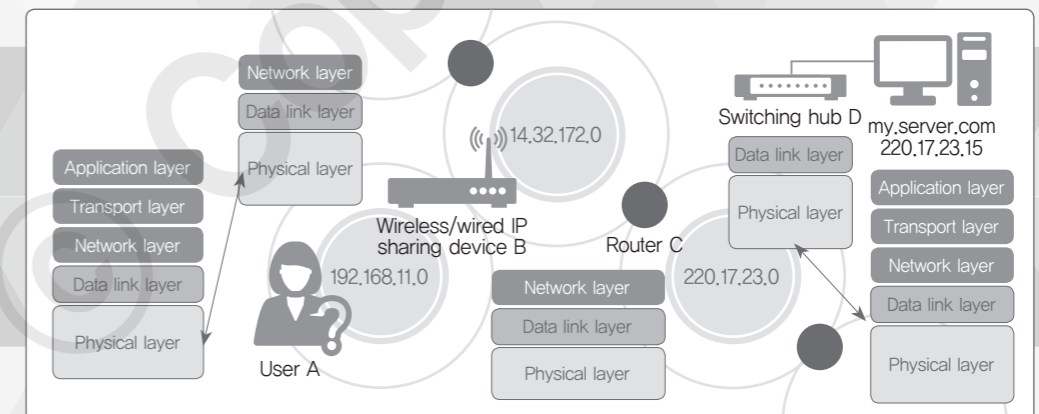
### ▶▶▶ Keywords

LLC, topology, CSMA/CD, CSMA/CA, ARP, RARP, Forward error correction, Backward error correction, Turbo code, Convolution code, BCH, Hamming code, Reed–Solomon, Parity test, Block sum check, Cyclic redundancy check, Stop–and–wait ARQ, Go–back–N, Selective–repeat, Adaptive ARQ, H–ARQ, VRC, LRC, Optical cable, IEEE 802.11, DSSS, OFDM, MIMO, 256–QAM, FHSS, Beam forming, UWB, IEEE 802.15, WPAN, ZigBee, Bluetooth, UWB

### Practical tips Mechanism of action in the data link layer

[Step 3–3] A packet was generated, including the information for inter–network routing. However, this packet cannot move from one network to another and needs some more preparation within the network. **User A's** computer has the IPv4 address of 192.168.11.5, which cannot be directly delivered to my.server.com. Hence, the computer tries to deliver the address to its gateway: wireless/ wired **IP sharing device B** (192.168.11.1). However, IP addresses cannot be used for transmission within the network, so it is necessary to get the MAC address information which is designed for transmission within the network. The MAC address matched to the gateway's IP address (192.168.11.1) should be found out by conducting broadcasting within the 192.168.11.0 network, and the packet is sent to **IP sharing device B**.

[Step 3–5] Router C can recognize that the packet is directed to the network 220.17.23.0 where it belongs and tries to send the packet to the web server's IP address 220.17.23.15. Just as in the case of [Step 3–3], the packet movement within the network requires a physical address of my.server.com or MAC address. Hence, the broadcasting is conducted within the network to acquire the MAC address corresponding to the IP address of **my.server.com** (220.17.23.15) and the packet can be delivered to the address.



⟨Figure 11⟩ Mechanism of action within data link layer



## 01 Basic Idea about Data Link Layer

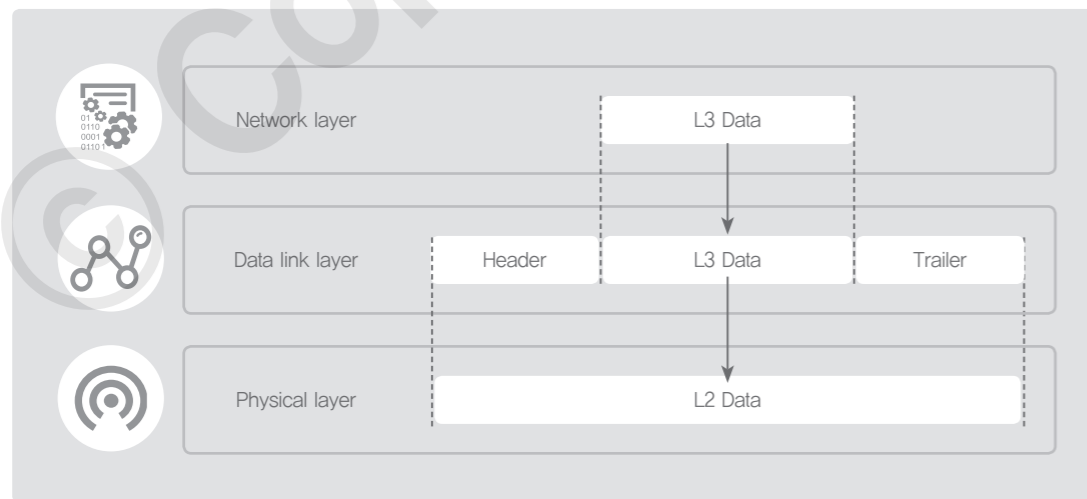
The data link layer is responsible for transferring data between the peripheral devices on the network using the Physical layer that is responsible for transporting a signal between devices. This layer has two main features: address allocation and error detection. The address allocation feature allows signals received from the Physical layer to be properly delivered securely to the devices on the network and the error detection feature detects whether there is any error in the process of signal delivery.

It does not matter a lot even though an application software developer does not fully understand the data link layer and the Physical layer. However, in the case of network related products, sometimes it is necessary for a developer to understand the standard protocols because of the following reasons: a developer may have a case that requires debugging where network access is not working or there is a data error; or a case that requires various MAC layer involvements. As for the embedded software development, sometimes even a driver level development should be done by developers. Once embedded software is ported in the test environment prepared by a hardware developer, it is difficult to find out whether any issue is directly coming from hardware side or from the ported program. In that case, JTAG debugging may be used. In the network environment, it may be needed to validate whether there is any issue from physically delivered bit or the error is from the data link layer. As more short distance communications technologies such as Bluetooth and ZigBee are accepted, it has become more important to understand the data link layer.

## 02 Data Link Layer Encapsulation

### Data Link Layer Encapsulation

As shown below, a frame of the data link layer is composed of a **Header** and a **Trailer** added onto the packet of the network layer. This kind of process is called **Encapsulation** of the data link layer. On the receiving end, a reversal process is performed and called **Decapsulation**.



(Figure 12) Encapsulation for data link layer

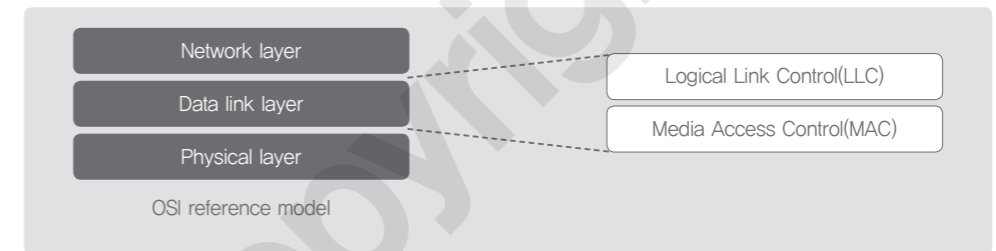
## Frame Header and Trailer

A frame header is composed of: a **Preamble** which is for bit synchronization among hosts; **SFD** (Start of Frame Delimiter) which is a field that indicates the start of a frame; and physical addresses such as the destination address and source address. The physical address has 48bit address structure and each of the devices has unique physical address as opposed to logical IP address. To identify the location of the devices in the layer 2 and below, the IP address of the layer 3 cannot be used and physical address should be used. A trailer has FCS (Frame Check Sequence) which is used to check the error in the process of transportation.

## 03 Structure of Data Link Layer

### Sub-layer of Data Link Layer

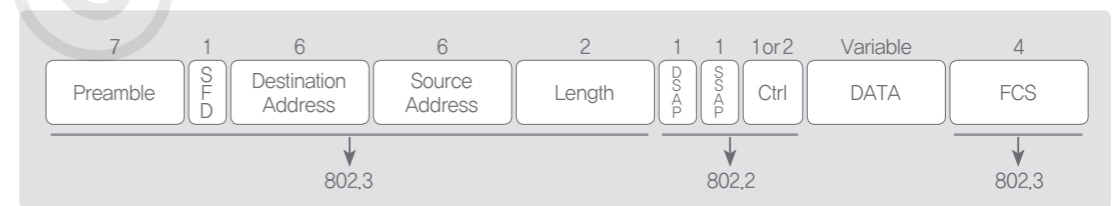
The data link layer is composed of two sub-layers: **LLC** (Logical Link Control) and **MAC** (Media Access Control). The LLC sub-layer is responsible for the connection between MAC sub-layer and the network layer (Layer 3). The MAC sub-layer is responsible for controlling mechanism, considering the topology and other characteristics of the Physical layer.



(Figure 13) Sub-layer of data link layer

### Logical Link Control

① Basic concept of logical link control



(Figure 14) IEEE 802.3 frame structure

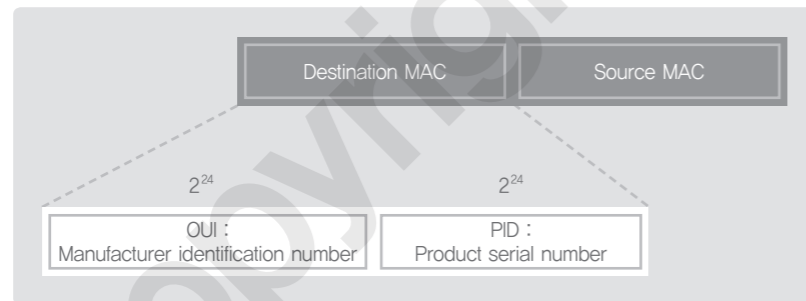
This sub-layer is the upper part of the two sub-layers of the data link layer and is also called the IEEE 802.2. This sub-layer is responsible for data transport between two neighboring nodes of the data link layer. In this case, the neighboring node has DSAP and SSAP address within the LLC sub-layer. LLC sub-layer allows diverse protocols of the MAC sub-layer to be utilized so that topology agnostic communications can be made available.

② Logical link service control option

There are three types of options in logical link service control. Type 1 is called an on-acknowledgement datagram service or called a connectionless service because the service does not require delivery acknowledgement messages. Type 2 is using virtual circuit access, just like TCP service, and a virtual session is made for data transfer. Type 3 is called acknowledgment datagram service and the service provides point-to-point datagram with delivery acknowledgement.

MAC

MAC is a layer that is responsible how data should be transferred out through physical medium. MAC, as shown below, includes the MAC address for transmitting and receiving systems. MAC addresses can be divided:  $2^{24}$  for OUI (Organizationally Unique Identifier), an identification code for the manufacturer; and  $2^{24}$  for NIC serial number from a specific manufacturer.



<Figure 15> MAC address structure

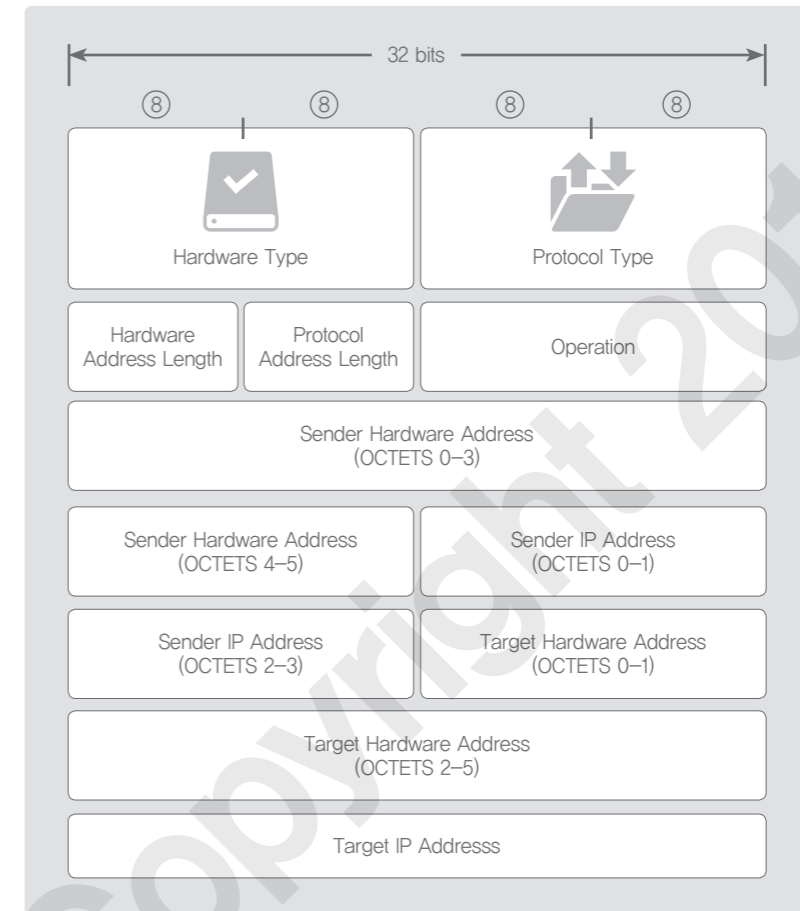
There are standardized MAC protocols: IEEE 802.3 or CSMA/CD, IEEE 802.4 or token bus, IEEE 802.5 or token ring for wired LAN; IEEE 802.11 MAC sub-layer is based on CSMA/CA for wireless LAN.

04 MAC Address Search

IP Address and MAC Address Resolution Protocol

In order to send a packet to another host, it is necessary to acquire the MAC address of the host. This job can be done by ARP (Address Resolution Protocol). There are two types of protocols used in IP address and MAC address

resolution; ARP and RARP (Reverse Address Resolution Protocol). Address resolution protocol is used to learn the MAC address by using the IP address, and RARP literally works in a reverse way as the protocol is used to learn the IP address via MAC address and the like.



<Figure 16> ARP packet structure

ARP packet is composed of: 6-byte destination MAC address, 6-byte source MAC address, 2-byte Ethernet protocol type, 2-byte hardware type, 2-byte protocol type, 1-byte hardware address length, 1-byte protocol address length, 2-byte operation code, 6-byte sender hardware address, 4-byte sender protocol address, 6-byte target hardware address, 4-byte target protocol address, and 18-byte padding.

MAC Address Search Scenario

In <Figure 11>, User A's computer and Router C are on the same network segment, it is necessary to acquire the MAC address to make communications possible. In this case, User A's computer performs broadcast to make ARP requests to all the systems within the network segment. The frame delivered during the ARP request contains an

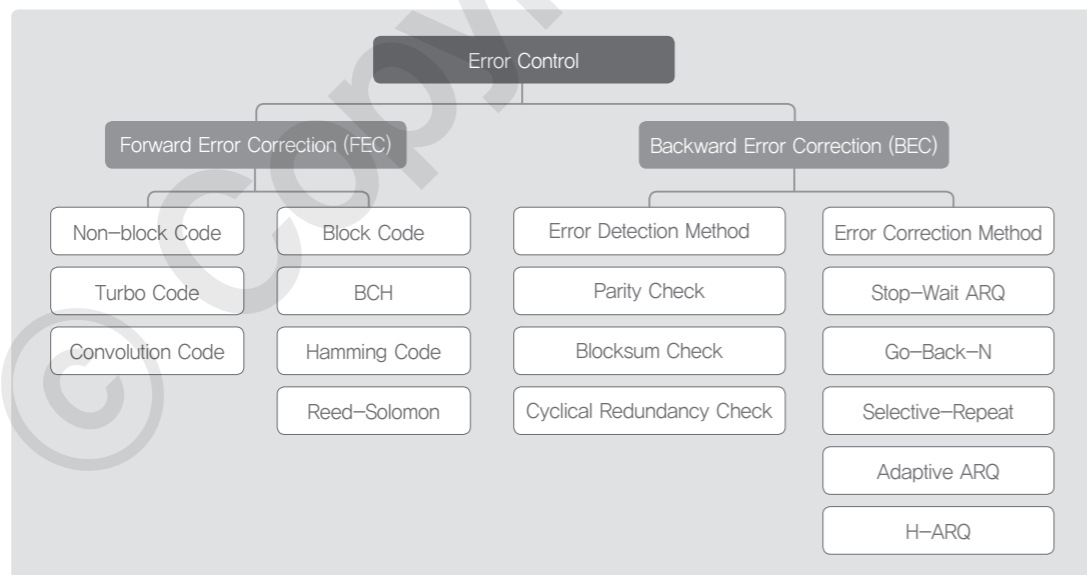
ARP packet, as shown in (Figure 16) in the data portion of the frame. The Router C, which receives a request, puts its MAC address into ARP packet and delivers the information to User A's computer via uni-cast to acquire MAC addresses of both sides and the MAC addresses are stored in the cache memory of each system. During this process, wired/wireless IP sharing device B manages the port-MAC address table for the systems connected to the port onto its cache memory.

## 05 Error Detection and Correction in Data Link Layer

### Definition of Error Control

Various errors might come about when a frame is delivered through the data link layer depending on how good the network condition is or how the transmitting and receiving devices are working. The types of errors can be divided into: Single-bit error where only one bit gets changed in the data portion, multi-bit error where two or more non-consecutive bits get changed in the data portion; and burst error where two or more bits get changed in a consecutive way in the data portion.

The definition of error control: to detect and correct errors when a transmitted data is not received or when there is an error during the transmission. Such error control methods can be divided in two: 1) Forward Error Correction (FEC) which allows the receiving end to detect and correct the error and the sender adds redundant data for error correction before sending the data; 2) Backward Error Correction (BEC) which means to give notice to the transmitting end when there is an error in the transported data and the transmitting end sends the data again for recovery.



(Figure 17) Type of error control

### Error Detection and Error Correction

#### ① Error detection

Error detection refers to a method to add redundant data to detect error on the receiving end. There are various ways of error detection such as VRC (Vertical Redundancy Check), LRC (Longitudinal Redundancy Check), CRC (Cyclic Redundancy Check), and checksum. The details are explained in the table below.

(Table 10) Error detection method

Method	Details
VRC	<ul style="list-style-type: none"> <li>Vertical Redundancy Check</li> <li>A parity check and most widely used for error detection</li> </ul>
LRC	<ul style="list-style-type: none"> <li>Longitudinal Redundancy Check</li> <li>Collect all the even number parity of all the bytes to create a data unit and add it at the end of the data block</li> </ul>
CRC	<ul style="list-style-type: none"> <li>Cyclic redundancy check</li> <li>A detection method using binary division</li> </ul>
Checksum	<ul style="list-style-type: none"> <li>Used in the protocols for the upper layer and basically founded on redundancy (VRC, LRC, CRC, etc.)</li> </ul>

#### ② Error correction

A receiving end can ask a sender to send the whole data again, or a receiver can correct the error by mechanically using an error correction code.

(Table 11) Error correction method

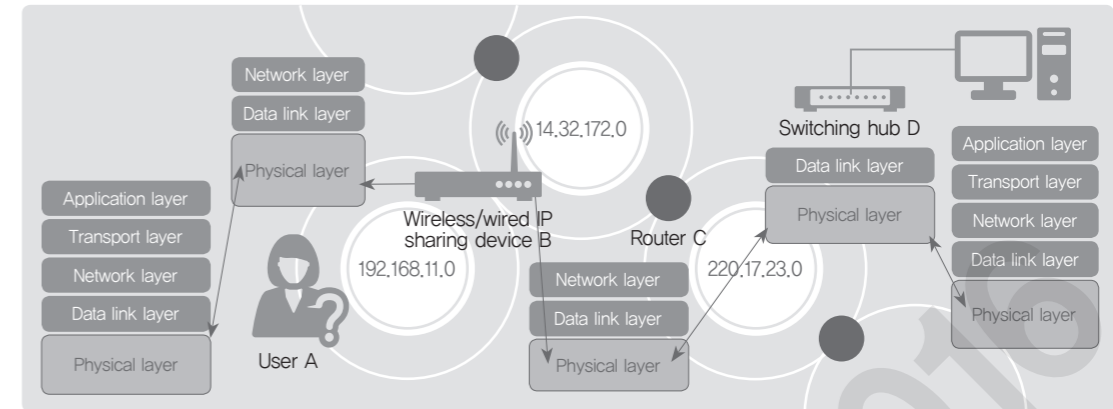
Method	Details
Single bit error correction	<ul style="list-style-type: none"> <li>To identify the location of wrong bit (parity bit)</li> <li>ASCII code needs 3-bit redundant code</li> </ul>
Redundant bit error correction	<ul style="list-style-type: none"> <li>To calculate the number of redundant bits in order to correct the number of bits in the given data by validating the relationship between the number of data bits and redundant bits</li> </ul>
Hamming code	<ul style="list-style-type: none"> <li>To identify the location of redundant bit using hamming code</li> </ul>
Multi-bit error correction	<ul style="list-style-type: none"> <li>The calculated outcome of redundant bit is used to correct multi-bit error.</li> </ul>

#### ③ ARQ: Automatic Repeat Request

ARQ means: a receiver notifies to a sender about the error and the sender shall resend the frame relevant to the error in order to resolve the issue. Typical ARQ types are stop and wait, Go-back-N ARQ, and adaptive ARQ. The details for each of the repeat request algorithms are explained below.

<Table 12> Types of ARQ algorithm

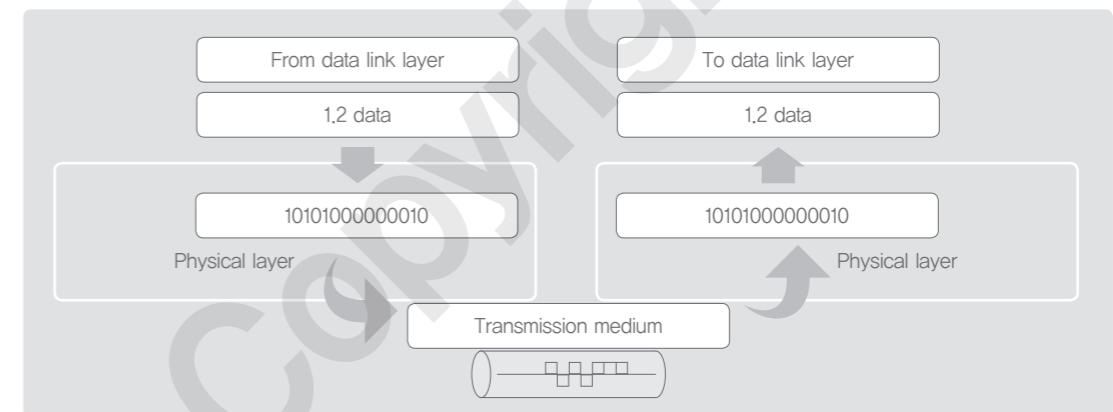
Algorithm	Details
Stop and wait ARQ	<ul style="list-style-type: none"> <li>A sender transmits one frame to a receiver. The receiver sends out either ACK or NACK signal in accordance with the presence or absence of the error in the frame.</li> </ul>
Go-back-N ARQ	<ul style="list-style-type: none"> <li>A method to send frames continuously, to address the delivery inefficiency of stop and wait ARQ method. A sender allocates a frame sequence number that is as big as the window size and sends them out continuously with a certain size.</li> </ul>
Selective-repeat ARQ	<ul style="list-style-type: none"> <li>Similar to Go-back-N ARQ, but only resends the error-specific frame</li> <li>Buffer for sending and receiving should be large enough.</li> </ul>
Adaptive ARQ	<ul style="list-style-type: none"> <li>To detect the error rate on the communications line and dynamically modifies the optimal length of the frame</li> </ul>
Hybrid-ARQ	<ul style="list-style-type: none"> <li>Compromise between forward error correction and backward error correction</li> <li>In a normal situation, the network efficiency is secured by using FEC. However, BEC is used in case of an error in order to enhance reliability.</li> </ul>



<Figure 18> Mechanism of action in Physical layer

### Signal Delivery Method

The signal delivery method of the Physical layer can be explained as follows: frames from the data link layer (upper layer of the Physical layer) are transformed into digital signals of 1 and 0, and the transformed signals are delivered to the receiving end via the transmission medium.



<Figure 19> Signal delivery of Physical layer

## 06 Physical Layer

### Operation Scenario

The Physical layer, as shown below, is intended to deliver electric signals among the devices such as User A's computer, wired/wireless IP sharing device, router switching hub, and server (my.server.com). The frames of the data link layer are transformed into digital signals of 1 and 0 in the Physical layer and the digital signals are transformed into electric signals in the transmission medium before delivery is made.

The transmission medium is a physical pathway between transmission and receiving devices. Typical examples of transmission medium are twisted-pair cable, coaxial cable, radio link, fiber optical cable, and the like. Usually, UTP is used in building LAN and fiber optical cable is used for backbone network provided by the ISP. It is a trend to use fiber optical cable more so that gigabyte level high speed delivery can be made.

In our scenario, UTP cable can cover from User A's computer to wired/wireless IP sharing device B, and from wired/wireless IP sharing device B to Router C. If the scenario wants to cover a wider area than shown in the example, fiber optical cable connection can be used along with a transmission device.

## 07 Classification of Physical Layer Medium

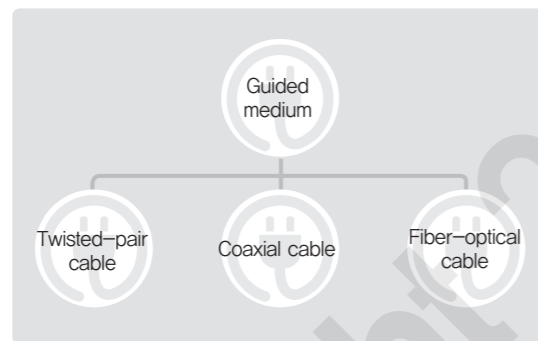
### Transmission Medium

The transmission medium is a physical path between a receiver and a transmitter. The transmission medium can be divided into Guided Media and Unguided Media and the typical examples are twisted pair cables, coaxial cables, radio links, and fiber optical cables.

## Guided Medium

### ① Types of guided medium

As for the guided medium: coaxial cables have been used a lot until the early 1990s; twisted-pair cables have been used a lot for the node connection and local area network; fiber optical cables have been used a lot for the backbone network to cover a long distance since the mid 1990s. The guided medium can be categorized into twisted pair cables, coaxial cables, and fiber optical cables as shown below.



<Figure 20> Types of guided medium

### ② Twisted-pair cable

Generally, twisted-pair cables are used a lot for the LAN. As shown below, two cables are twisted with each other to minimize the mutual interference. Every pair has different twist rates on different segments (in every inch) in order to minimize the electromagnetic interference.



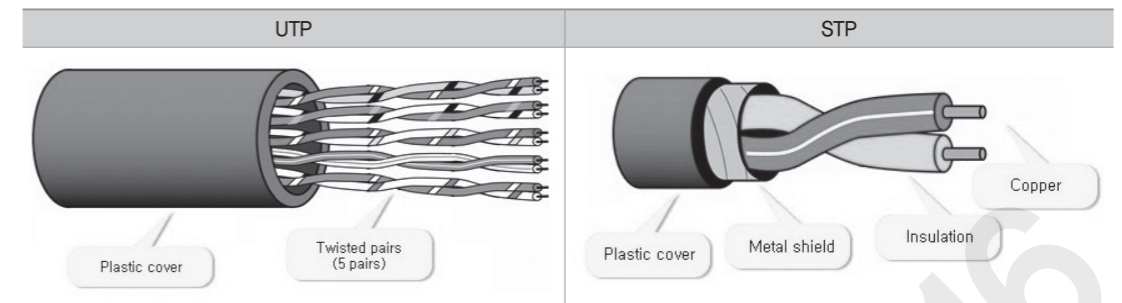
<Figure 21> Twisted-pair cable

Twisted pair cables can be divided into UTP (Unshielded Twisted Pair) and STP (Shield Twisted Pair). UTP means a stranded wire not surrounded by conductive materials and is usually used as a transmission medium for an internal telephone line or an information network.

On the other hand, STP (Shield Twisted Pair) means a stranded wire surrounded by conductive materials. It is tolerant against electrical noises and is used as a transmission medium for an internal information network using the IEEE 802.5 token-ring method.

Presence of conductive materials can make a significant difference in the two types of twisted-pair cables.

<Table 13> Structure of twisted-pair cables



UTP cable is typically categorized into CAT3, CAT5, CAT5e, CAT6 and each category can support different speed and frequency levels.

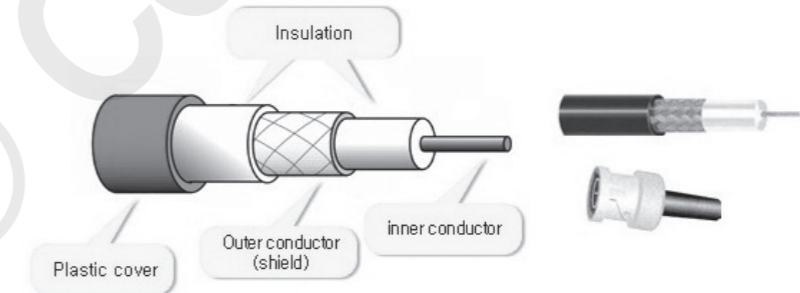
CAT3 can provide 10Mbps level signal transmission. It can be used for data transmission for a network with the minimal specifications, but usually used for delivering voice signals on the phone. CAT5 can provide 100Mbps level signal transmission, and is generally used for transmitting the data and voice signals in the Ethernet. CAT6 allows 1Gbps level signal transmission and can be used for high-speed Ethernet and gigabit Ethernet data transmission.

<Table 14> Types of UTP cables

Category	Speed and frequency	Details
CAT3	10Mbps, 16MHz	Voice signal for brand-new telephones, data transmission for the network with minimal specifications
CAT5	100Mbps, 100MHz	10~100Mbps data and voice transmission in the Ethernet
CAT6	1Gbps, 250MHz	High-speed Ethernet and gigabit Ethernet data transmission

### ② Coaxial cable

The structure of coaxial cables is composed of a single wire inside and the round conductors covering the wire outside.



<Figure 22> Structure of coaxial cables

Coaxial cables can be categorized into a thin cable and a thick cable, considering the difference in data transfer specifications.

The thin cable is categorized as an RG-5 cable which supports up to 10Mbps data transfer rate and covers up

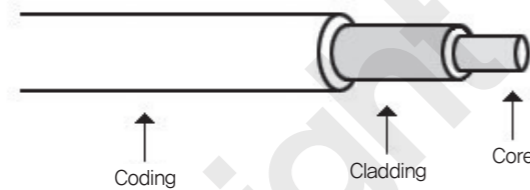
to 185m. The thick cable is categorized as an RG-8 cable which supports up to 100Mbps data transfer rate and covers up to 500m.

<Table 15> Types of coaxial cables

Type	RG designations	Details
Thin Cable	RG-5	Data transfer rate 10Mbps, Maximum coverage 185m
Thick Cable	RG-8	Data transfer rate 10Mbps, Maximum coverage 500m

④ Optical fiber cable


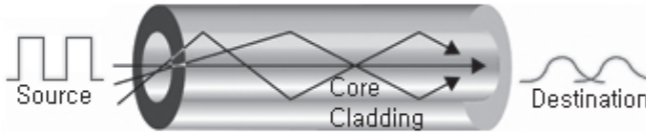
An optical fiber cable is composed of a core, cladding, and coating as shown in (Figure 23). The core is a transparent wrapper that has a high refractive index where the light passes through; the cladding is a transparent wrapper surrounding the core and has less refractive index compared to the core; and the coating is a synthetic resin wrapper protecting the core and the cladding.

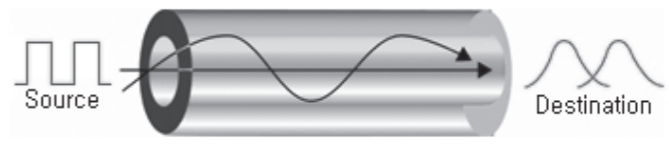


<Figure 23> Structure of optical-fiber cables

Optical fiber cables can be divided as follows: single mode where a beam from the source is delivered through the core in a single pathway; and multimode step index and multimode graded index where multiple beams from the source are delivered through the core in various pathways.

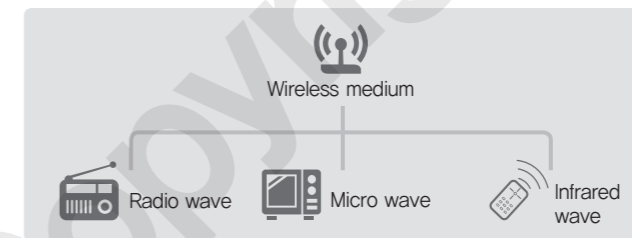
<Table 16> Types of optical-fiber cables

Type	Details
Single mode	The beam from the source is delivered through the core in a single pathway. 
Multimode step index	Beams from the source are delivered through the core in various pathways. 

Type	Details
Multimode graded index	Beams from the source are delivered through the core in various pathways. 

Wireless Medium

The wireless medium can be categorized into radio wave, microwave, and infrared wave. The radio wave, by definition, means the electromagnetic wave between 3 KHz and 1 GHz. In most cases, the radio wave radiates into all directions. This wave form is suitable for the multicasting which has only one transmitter but with multiple receivers. The microwave, by definition, means the electromagnetic wave between 1 GHz and 300 GHz. As this wave form moves into a single direction, it is possible to send the wave in a single direction with a specific focus. The microwave can be used in the uni-cast communications such as mobile phones, satellite communications, and the wireless LAN. The infrared ray, by definition, means the electromagnetic wave between 300 GHz and 400 TGHz (with 1nm-770nm wavelength) and is used for the short distance communications. The infrared ray, due to its high frequency, cannot penetrate through a wall and can be used for shorter distance communications within a closed space by using the line-of-sight propagation.

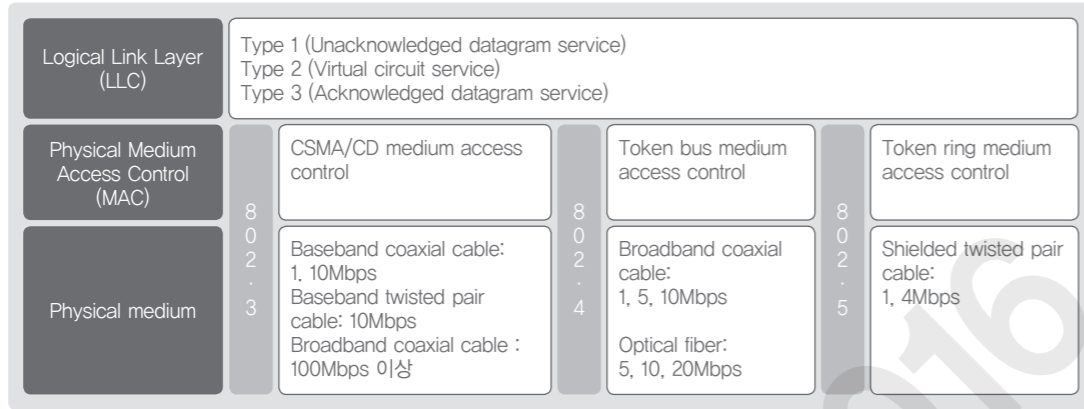


<Figure 24> Classification of wireless medium

08 IEEE 802 Standard

Basic Concept of IEEE 802

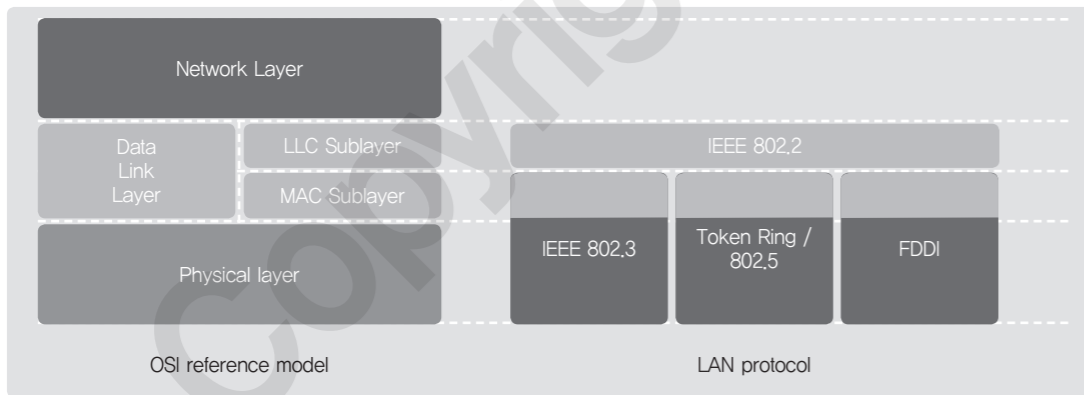
<Figure 25> shows services on each layer which are defined by the IEEE Committee. Let's look at the services of each layer. The Logical Link Control (LCC) layer provides services: Type 1 (Unacknowledged datagram service), Type 2 (Virtual circuit service) and Type 3 (Acknowledged datagram service). The Medium Access Control (MAC) layer offers the IEEE 802.3 CDMA/CD medium access control, the IEEE 802.4 token bus medium access control, and the IEEE 802.5 token ring medium access control.



〈Figure 25〉 Services on each layer

### IEEE 802.3 Standard

In the IEEE 802.3 protocol stack which is used most among the IEEE 802 standard, there are: the IEEE 802.2 protocol which is specific to the LLC sub-layer of the data link layer; and the IEEE 802.3, the IEEE 802.5 token ring, the FDDI and the like which cover the MAC sub-layer and the Physical layer.



〈Figure 26〉 IEEE 802.3 protocol stack

### IEEE 802.11 Standard

The IEEE 802.11 is a working group, responsible for managing wireless local area network (also known as wireless LAN or Wi-Fi) standards. It aims to minimize costs required for wiring and maintenance, so as to overcome the limitations of the wired LAN type Ethernet. Some common specifications in the 802.11 family include the following: the IEEE 802.11b which is capable of transmissions of up to 11 Mbps and utilizes the DSSS; the IEEE 802.11a which is capable of transmissions of up to 54 Mbps and operates in the 5 GHz band using the OFDM; and the 802.11g

which provides data transfer rate of up to 54 Mbps and operates in the 2.4 GHz band. The IEEE 802.11n, another expansion from the 802.11 standard, is a standard which operates in the 2.4 GHz and 5 GHz band and utilizes the MIMO (Multiple-Input Multiple-Output) to improve the data transfer speeds up to 600 Mbps. Wireless AP products that can support Gbps level are being launched these days. The IEEE 802.11ac achieves a theoretical maximum rate of 6.93 Gbps in the 80/160MHz band, using the multi-user MIMO/multiple MIMO spatial streams, 256-QAM, and beam forming. The IEEE 802.11ad operates in the 60 GHz frequency band and offers Gbps rate of transmissions, using various technologies. The development work for the 'Gigabit wireless LAN' standards is still going on to make it possible to transmit the UHD images and explosively growing wireless data at a higher speed.

〈Table 17〉 IEEE 802.11 protocol standards

802.11 Protocol	Frequency (GHz)	Bandwidth (MHz)	Stream data rate (Mbps)	Allowable MIMO streams	Modulation
-	• 2.4	• 20	• 1,2	• 1	• DSSS, • FHSS
a	• 5	• 20	• 6,9,12,18,24,36,48,54	• 1	• OFDM
b	• 2.4	• 20	• 1,2,5,5,11	• 1	• DSSS
g	• 2.4	• 20	• 6,9,12,18,24,36,48,54	• 1	• OFDM, • DSSS
n	• 2.4 • 5	• 20	• 7,2,14,4,21,7,28,9,43,3,57,8,65,72,2	• 4	• OFDM
		• 40	• 15,30,45,60,90,120,135,150		
ac	• 5	• 20	• 87.6	• 8	• OFDM
		• 40	• 200		
		• 80	• 433.3		
		• 160	• 866.7		

### IEEE 802.15 Standard

The IEEE 802.15, derived from the IEEE 802.11 committee for wireless LAN standards, is a working group which specifies the wireless personal area network (WPAN) standards. It is usually related to building a wireless network at home for mobile communications terminals, PCs and other peripheral devices. The IEEE 802.15 consists of several study groups, such as the WPAN Study Group (Data rates of less than 1 Mbps), the WPAN High Rate Study Group (Data rates of up to 20 Mbps) and others.


The IEEE 802.15.1 **Bluetooth**, the IEEE 802.15.3 **UWB**, the IEEE 802.15.4 **ZigBee** are the most widely used WPAN technologies.

The IEEE 802.15.1 Bluetooth is a wireless technology standard for exchanging data over a short distance by using mobile phones, laptops, and other mobile devices. The IEEE 802.15.3 UWB refers to a wireless technology that aims to transmit a large amount of digital data within a short range with low power through a wide spectrum frequency band. The IEEE 802.15.4 ZigBee is a standard technology which is designed for a low-rate home automation and data network.

<Table 18> IEEE 802.15 protocols

WPAN	Bluetooth	UWB	ZigBee
Standard	IEEE 802.15.1	IEEE 802.15.3	IEEE 802.15.4
Frequency	2.4GHz	3.1~10.6GHz	868MHz
Data transfer rate	Varying on each version	480Mbps	20/40/250Kbps
Transmission range	10~100m	10m	10~75m
Application	Transmission of voice and file, etc.	Multimedia, etc.	Sensor communications, etc.

Based on the understanding of the specifications for ZigBee, Bluetooth and other technologies, an embedded software developer is required to know which chipset should be used, which communications can be used for controlling the chipset, and which development tools can support the chipset, and ultimately, is required to draw a whole picture about the chipset. The following is an example of the specifications for a ZigBee chipset. It shows a list of applications where the chipset can be used, its key features, and the eligible ZigBee standards.



**CC2520 DATASHEET**  
2.4 GHZ IEEE 802.15.4/ZIGBEE® RF TRANSCEIVER  
SWRS068 – DECEMBER 2007

**APPLICATIONS**

- IEEE 802.15.4 systems
- ZigBee® systems
- Industrial monitoring and control
- Home and building automation
- Automatic Meter Reading
- Low-power wireless sensor networks
- Set-top boxes and remote controls
- Consumer electronics

**KEY FEATURES**

- State-of-the-art selectivity/co-existence
- Adjacent channel rejection: 49 dB
- Alternate channel rejection: 54 dB
- Excellent link budget (103dB)
- 400 m Line-of-sight range
- Extended temp range (-40 to +125°C)
- Wide supply range: 1.8 V – 3.8 V
- Extensive IEEE 802.15.4 MAC hardware support to offload the microcontroller
- AES-128 security module
- CC2420 interface compatibility mode

**Low Power**

- RX (receiving frame, -50 dBm) 18.5 mA
- TX 33.6 mA @ +5 dBm
- TX 25.8 mA @ 0 dBm
- <1µA in power down

**General**

- Clock output for single crystal systems
- RoHS compliant 5 x 5 mm QFN28 (RHD) package

**Radio**

- IEEE 802.15.4 compliant DSSS baseband modem with 250 kbps data rate
- Excellent receiver sensitivity (-98 dBm)
- Programmable output power up to +5 dBm
- RF frequency range 2394-2507 MHz
- Suitable for systems targeting compliance with worldwide radio frequency regulations: ETSI EN 300 328 and EN 300 440 class 2 (Europe), FCC CFR47 Part 15 (US) and ARIB STD-T66 (Japan)

**Microcontroller Support**

- Digital RSSI/LQI support
- Automatic clear channel assessment for CSMA/CA
- Automatic CRC
- 768 bytes RAM for flexible buffering and security processing
- Fully supported MAC security
- 4 wire SPI
- 6 configurable IO pins
- Interrupt generator
- Frame filtering and processing engine
- Random number generator

**Development Tools**

- Reference design
- IEEE 802.15.4 MAC software
- ZigBee® stack software
- Fully equipped development kit
- Packet sniffer support in hardware

<Figure 27> Example of ZigBee specifications [7]

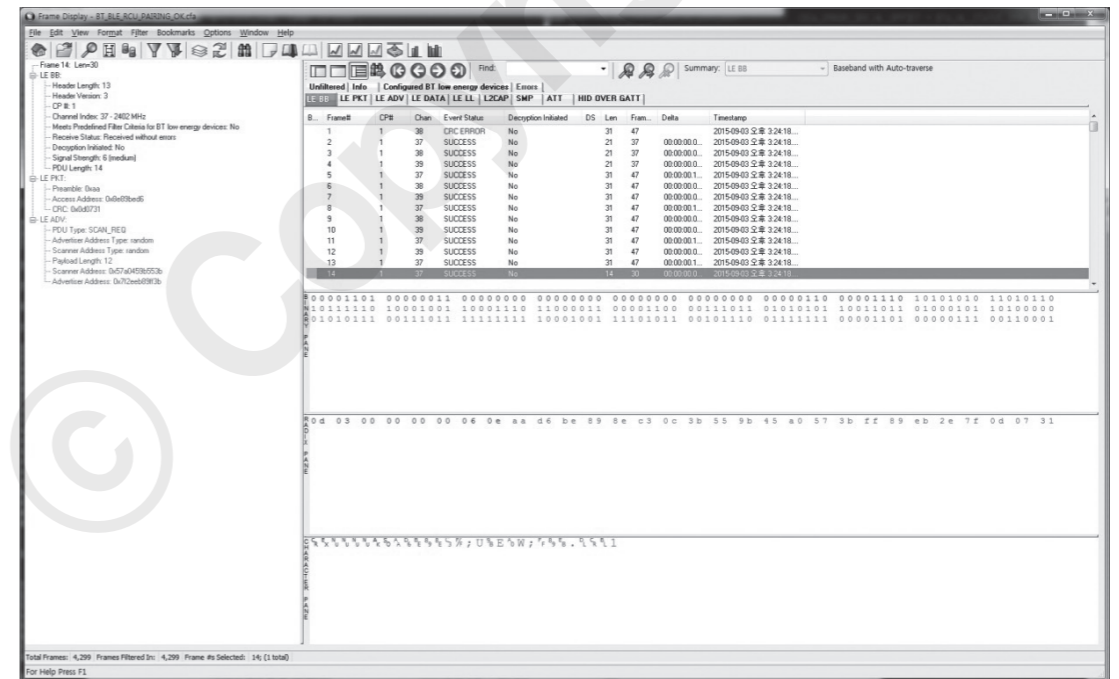
The following shows what software developers need to understand in chipset specifications

- Block diagram: the understanding of the entire chipset structure and flow in the chipset
- Communications type: SPI, I2C, etc., chipset control methods
- Memory: RAM, ROM, available size of memory, address type, etc.

- GPIO: types of the ports for controlling the external signal input/output and port configuration methods
- Development tool: tools required to develop the chipset

Considering the application area where an embedded product needs to be developed, a hardware developer is required to investigate an initial chipset and figure out how to generate an entire circuit configuration using this chipset. The hardware developer needs to review the chipset based on some criteria pertaining to a hardware perspective, such as electrical characteristics, reliability, complexity circuit configuration, chipset package appropriate for each device design, and costs. Meanwhile, a software developer needs to review various aspects of the chipset – its compliance with the applicable standards, ease of control methods, appropriateness of memory size, whether the reuse of the existing control method is possible, whether there are enough ports to control input/output of a target product, whether it is better to separate the chipset from other chipsets, and whether it is better to have one chipset equipped with various functions. An active communication process with hardware developers should come after the previously mentioned considerations in order to select the most suitable chipset.

<Figure 28> shows a screenshot taken by the Frontline's Frame Display packet capture program which displays data transmitted via Bluetooth. A CRC error is detected in the Frame 1, as seen in the figure. The embedded software developer should investigate whether the data is transmitted as expected with the program written by him/herself. A certain data is hardly likely to be transmitted at once without any errors right after porting the program on the test board. In general, the developer is required to: debug the self-made program and then; look into whether the data transmission and reception works fine by using the tools such as a packet capture program. If the aforementioned step is not available, the developer needs to directly validate data transmission at the Physical layer level by using a spectrum analyzer or the like. As more and more products are being launched with various wireless media and protocols, there is a rising need for understanding the data link layer and physical layer.



<Figure 28> Frame Display's screenshot: data transmitted via Bluetooth



## Example Question

## Question type

Descriptive question

## Question

Suppose you write a product development plan for a smart wrist band for healthcare. You need to select a right method for the near distance communications and to select a right chipset. According to the market survey, Bluetooth, ZigBee, UWB (Ultra Wide Band), and NFC (Near Field Communication) are widely used technologies. Because of the price for the product, the product shall support only one technology for communications rather than many of them.

- 1) Select just one communications method to fulfill the conditions mentioned below. (15 points)
- 2) Describe why the method was selected, especially about its characteristics and features. (15 points)

Condition 1. The ISM band should be used  
 Condition 2. Need to transport the smart phone voice data and personal healthcare record  
 Condition 3. The transfer rate should be higher than 1Mbps

## Intent of the question

Need to understand WPAN communications technologies thoroughly rather than just memorizing them.

## Answer and explanation

- 1) Bluetooth
- 2) Condition 1: Generally using 2.4GHz (ISM band)  
 Condition 2: Bluetooth is used for file and voice communications.  
 Condition 3: Theoretically, Bluetooth 1.0 can support 1 Mbps, Bluetooth 2.0 can support 2Mbps, and Bluetooth 3.0 can support up to 24Mbps.

ZigBee is usually used for small data volume such as sensor-based communications.

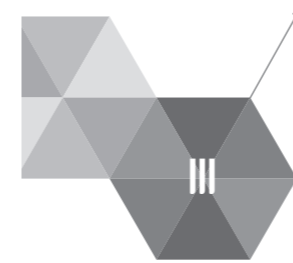
NFC requires physical distance closer than 10cm: not suitable

UWB can support up to 500Mbps, but cannot meet Condition 1.

ZigBee can support 250Kbps.

## Related E-learning Contents

- Lecture 2 Physical layer and Data Link Layer



## Understanding Routing Protocols and the Ipv4 Address System and to Utilize the Same

### Latest Trends and Key Issues

The network layer is the one which is most closely engaged with the data transmission and plays important roles for efficient data transmission. This chapter will cover: how packets are generated based on IPs; how packets are effectively delivered on the Internet network; and eventually how data is moved and processed on the network. This chapter will explain about a router, which is a device used in the network layer, its basic structure, the working mechanism for packet routing, the routing algorithm, and the routing protocol that is actually implemented.

### Study Objectives

- \* To be able to understand protocols and devices of the network layer and to be able to explain about the same
- \* To be able to explain about the basic ideas of the routing protocol, its type and algorithm
- \* To be able to understand the IPv4 addressing structure and to work on subnetting

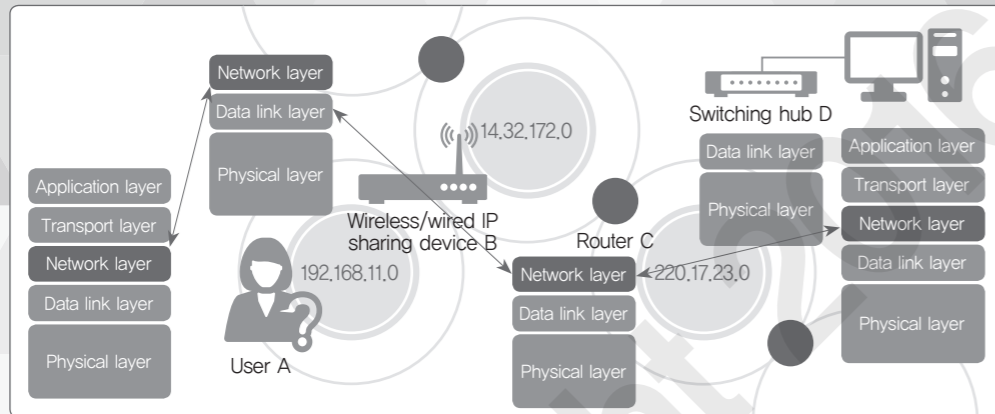
### Practical Importance High

### Keywords

Router, Routing table, Packet, PDU, Datagram method, Virtual circuit, APIPA, Segment, Metrics, MTU, STP, ARP, RARP, ICMP, IGMP, QoS, Bandwidth, IntServ, RSVP, DiffServ, Routing algorithm, Routing protocol, EGP, IGP, Distance Vector, Link State, BGP, RIP, IGRP, OSPF, IPv4, Subnetting, Supernetting, APIPA, CIDR, DHCP, NAT

Practical tips Network Layer and Routing Protocol

Mechanism of action of the network layer



(Figure 29) Conceptual diagram of router

Various scenarios have been already explained regarding the network layer as shown in (Figure 29).

[Step 3–2] The information created in the TCP protocol goes through a number of hops and is encapsulated so that it can be routed to the network (220.17.23.0) where my.server.com is residing.

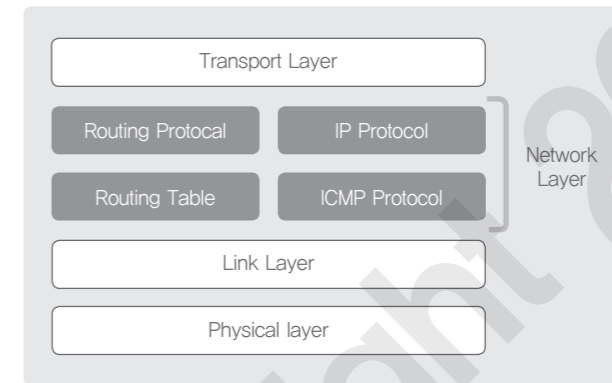
[Step 3–4] The wireless/wired IP sharing device B receives the packet from User A’s computer and looks into the final destination of the packet because the packet is not directed to the IP sharing device. The IP header contains the final destination (my.server.com) and the IP sharing device B can recognize that the final destination is not for 192.168.11.0 or 14.32.172.0 where the IP sharing device belongs. The IP sharing devices passes the packet to Router C where it belongs.

[Step 3–6] My.server.com receives the packet and looks into the IP address to identify whether the packet is directed to the right address. After that, the server removes the header information and sends the remaining data to a higher protocol layer.

## 01 Outline of Network Layer and Device

### What is Network Layer?

The network layer is the third layer in the OSI 7 model and TCP/IP, which is responsible for packet transport from the transmitting end to the receiving end. The network layer receives the Segment from the transport layer and Encapsulates the segment to deliver it to the data link layer.



(Figure 30) Network layer in TCP/IP layer

### Function of Network Layer

The network layer encapsulates a payload on the transmitting end and decapsulates the payload on the receiving end. In addition, the layer has various functions: **Packetization** so that the payload will not be changed during the data delivery; **Routing** that finds a path for the packet delivery; and **Forwarding** that is run by a router when the packet is delivered to one of the interfaces of the router. A routing rule is applied in the process of forwarding in order to create a table for decision making. The decision making table is also called a forwarding or **Routing Table**.

(Table 19) Function of network layer

Function	Details
Packetization	<ul style="list-style-type: none"> <li>• Payload encapsulation on the transmitting end and payload decapsulation on the receiving end</li> <li>• Responsible for the payload delivery without changes about the payload from the transmitting end to the receiving end</li> </ul>
Routing	<ul style="list-style-type: none"> <li>• Path finding for a packet from the transmitting end to the receiving end</li> </ul>
Forwarding	<ul style="list-style-type: none"> <li>• A function performed by a router when a packet arrives into one of the interfaces of the router</li> <li>• Routing rules are applied to create a decision making table for a router.</li> <li>※ The decision making table is called a routing table.</li> </ul>

## Internetworking Device

Internetworking means a connection between a network and another network. Typical internetworking devices are **Repeater, Bridge, Switch, and Router**

<Table 20> Internetworking devices

Device	Details
Repeater	Strengthening signals between the connection points (amplification, signal regeneration)
Bridge	Bridging two LANs and performing translation and format transformation to make the two different factors as one
Switch	A MAC address-based network separator that works like a multi-port bridge
Router	Transmitting data after finding out the optimal communications path between heterogeneous networks

Let's take a look at the role of internetworking devices in the world of the seven layers of the OSI and the four layers of the TCP/IP. A router performs packet routing using the IP address in the network layer. A bridge and a switch are responsible for frame transportation in the data link layer, while a hub and a repeater simply deliver physical signals on the Physical layer. The function of each of the devices is listed for comparison in <Table 21>.

<Table 21> Protocol layers and device for each layer

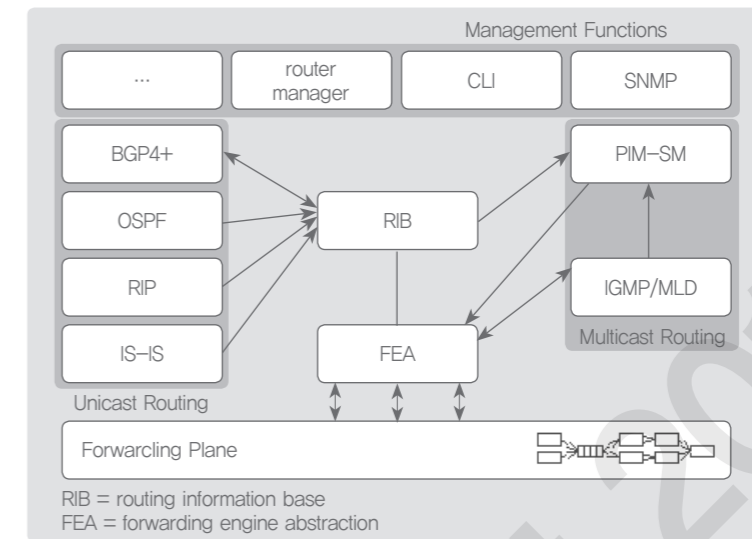
OSI 7 layers	TCP/IP 4 layers	Device
Application	Application	Gateway
Presentation		
Session		
Transport	Transport	Router
Network	Internet	
Data Link	Network Access	Bridge, Switch
Physical		Hub, Repeater

## What is Router?

A router uses more than one metric for network traffic forwarding and defines an optimal pathway. In other words, the device forwards packets from one network to another based on the information of the network layer.

### ① Structure of router

A router is composed of a **Router Control Plane** and a **Forwarding Plane**. The router control plane, which is implemented with software, is composed of processes of determining where the packets, coming through the router, should be forwarded and tables that are required for the process. The forward plane performs actual packet transmission following the requirements made in the router control plane.



<Figure 31> Conceptual diagram of router [1]

A router looks like as shown in <Table 22>. In a small network, Cisco 2500 can be used and Cisco 7300 can be used in a medium size or bigger networks.

<Table 22> Routers [2]

Cisco 7300	Cisco 2500
	

### ② Router metrics

It means a set of data collected within a given timeframe for a certain routing path. The types of router metrics are as follows:

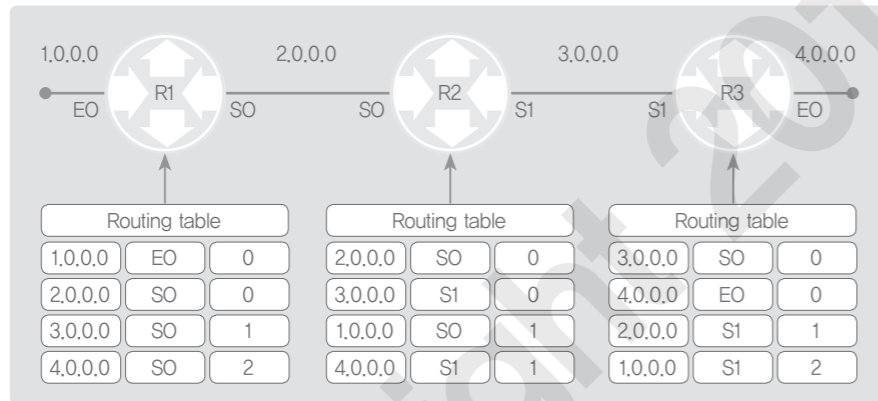
<Table 23> Routing metrics [3]

Metric	Details
Number of hops	The total number of hops between the starting point and the final destination; the smaller the number is, the faster the processing is.
MTU	Maximum Transmission Unit; literally means the maximum data a protocol can take up.

Metric	Details
Cost	The cost is determined by various factors such as transmission time, link reliability, and characteristics of a band. The higher the cost is, the lower the efficiency is.
Latency	To determine the bottleneck and to manage packet delay records between routers

③ Routing Table

A **Routing Table** refers to a database where output interfaces for each of the destination networks and IP addresses of the next hops are stored.



<Figure 32> Routing table

### What is Switch?

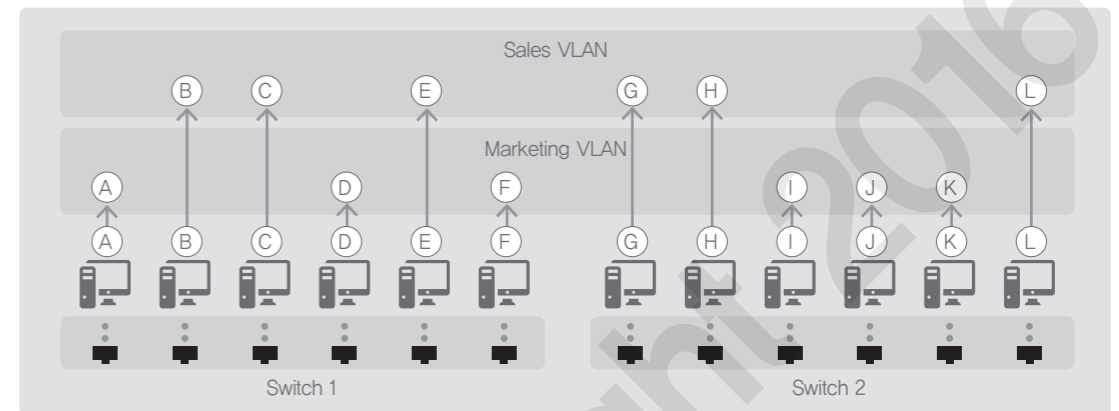
A **Switch** is a device that connects network segments and provides various functions such as **MAC Address Learning**, **Loop Prevention**, and **Filtering**. In addition, the switch has **port mirroring** and **VLAN function**). Port Mirroring (also called a port monitoring) means that a set of data transported to a certain port is replicated and the copied data is sent to a mirrored port. Port mirroring is important in that an administrator can use the mirrored port to monitor network traffic.

<Table 24> Main functions of switches

Function	Details
Address learning	• To learn all the MAC addresses of all the systems connected to the port.
Filtering	• To perform port-specific data traffic filtering using the MAC address
Loop prevention	• To avoid a network loop when multiple paths are generated to the destination system • STP (Spanning Tree Protocol) is used in the system to prevent a network loop

### VLAN (Virtual Local Area Network)

**VLAN** is aimed to overcome the physical and geographical limitations of the existing network and to build a logical network that can meet users' demands. In other words, a network is not categorized by the factors such as geographic or spatial location but other factors are used to build a logical network such as IPs, protocols, MAC addresses, and ports. There are VLAN protocols relevant to a switch such as the **ISL**, **802.1Q**, and **VTP**. The ISL and 802.1Q are VLAN tagging protocols, while the VTP is a VLAN management protocol.



<Figure 33> VLAN structure [4]

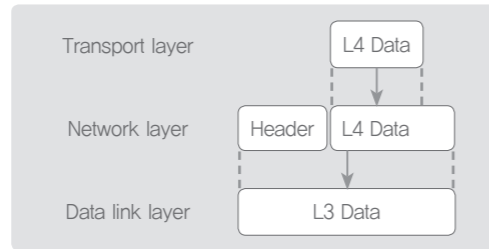
<Table 25> Switch Vs. Router

Comparison target	Switch	Router
Reference table	MAC address table	Routing table
Reference PDU	Ethernet frame	IP packet
Reference field	Destination MAC address	Destination IP address
Frame used	Ethernet	Ethernet, frame relay, PPP, etc.
Layer 2 header	No change	Replaced with new header

## 02 Encapsulation of Network Layer

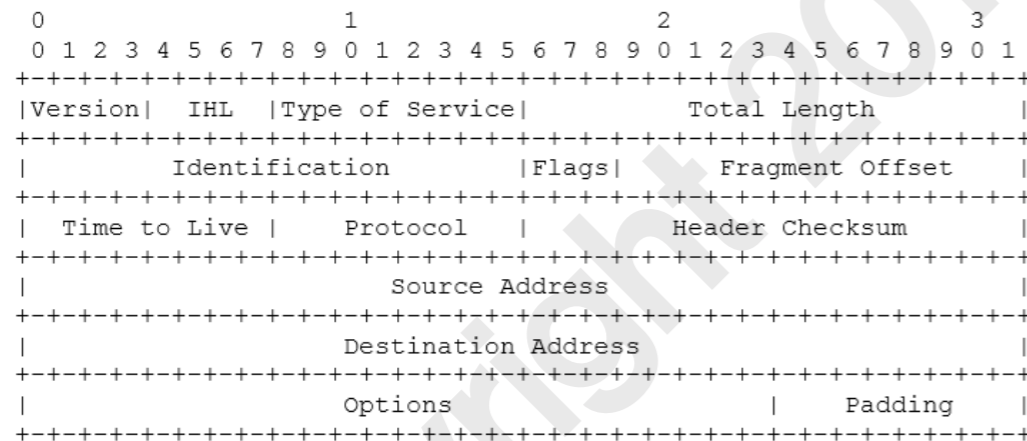
### Encapsulation of Network Layer

As shown in <Figure 34>, a packet of the network layer is made by adding a header on the segment of the transport layer. This kind of process is called **Encapsulation** and a reversal process, called **Decapsulation**, is carried out on the receiving end.



<Figure 34> Encapsulation of network layer

IPv4 Header



<Figure 35> Network packet structure [5]

In <Figure 35>, 'Version' indicates the version of the protocol where the datagram belongs, 'IHL' is the length of the header, and 'Type of Service' indicates what kind of service the host wants in the subnet. In addition, 'Total length' is the sum of the length of the header and the data in the datagram, 'Time to live' means the life limit of the packet, and 'Protocol' is used to deliver the TCP/UDP information so that the recombination of the datagram can be made at the network layer. Whereas, 'Source address' and 'Destination address' literally means the sending address and destination address which sends or receives the 32-bit datagram respectively.

03 Packet Switching and Network Layer Protocol/Command

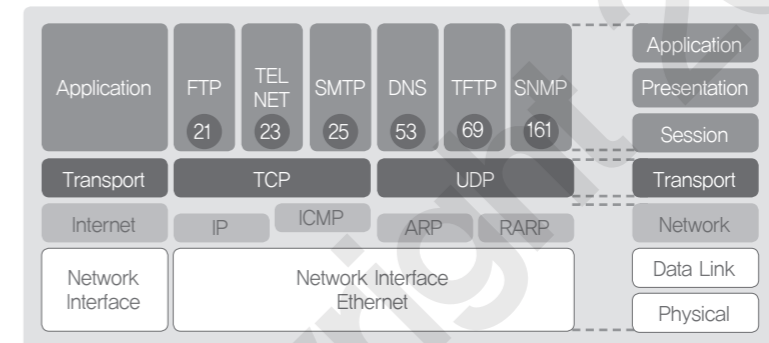
Packet Switching

There are two ways to find packet routes in the packet switching network: **Datagram Approach** and **Virtual Circuit Approach**. The datagram approach is a connectionless service which handles all the packets independently. Packets can be delivered in different routes and forwarding decision is made from the destination address of the

packet. On the other hand, the virtual circuit approach is a connection-oriented service which sets out the virtual route for the datagram even before transmission begins and the actual transmission for all the datagram is made through the same route after the connection establishment is finished. The forwarding decision is made by the **Packet Label (identifier for virtual circuit)**.

Network Layer Protocol/Command

In the network layer, there are address resolution protocols between the IP address and the MAC address: **ARP (Address Resolution Protocol)** which resolves the IP address into the MAC address; and **RARP (Reverse Address Resolution Protocol)** which resolves the MAC address into the IP address. In addition, **ICMP (Internet Control Message Protocol)** is a protocol used to send out the information of network errors, and **IGMP (Internet Group Management Protocol)** is for the IP multicast transmission.



<Figure 36> Protocols of network layer

<Table 26> Protocols of network layer

Protocol	Details
ARP	<ul style="list-style-type: none"> <li>• Address Resolution Protocol (RFC 826)</li> <li>• To resolve the IP address into the MAC address</li> </ul>
RARP	<ul style="list-style-type: none"> <li>• Reverse Address Resolution Protocol (RFC 903)</li> <li>• To reversely resolve the MAC address into the IP address</li> </ul>
ICMP	<ul style="list-style-type: none"> <li>• Internet Control Message Protocol(RFC 792)</li> <li>• To send out the information about network errors</li> </ul>
IGMP	<ul style="list-style-type: none"> <li>• Internet Group Management Protocol(RFC 1112)</li> <li>• To perform IP multicast</li> </ul>

Network Layer Command

A list of commands, used to view status information or to activate a certain action, is presented in <Table 27>.

〈Table 27〉 Commands relevant to network layer

Command	Details
Ping	<ul style="list-style-type: none"> <li>To test reachability to the network layer by using the ICMP</li> <li>To use the echo request /reply message among the ICMP types</li> </ul>
Tracert/ Traceroute	<ul style="list-style-type: none"> <li>To display the route to the desired destination</li> <li>To find out where a bottleneck is (when there is an issue to get access to a certain site)</li> </ul>
Route	<ul style="list-style-type: none"> <li>To allow a manual modification of the routing table</li> </ul>
Ipconfig/ Ifconfig	<ul style="list-style-type: none"> <li>To display the TCP/IP network configuration values of a computer</li> <li>To validate and renew the DHCP and the DNS configuration values</li> </ul>
Netstat	<ul style="list-style-type: none"> <li>To check out the overall status of a network: network connection, routing table, network interface and the like</li> </ul>
Arp	<ul style="list-style-type: none"> <li>To display or modify the local ARP cache values</li> </ul>

## 04 Network Service Quality

### QoS(Quality of Service)

The term **QoS** means to guarantee a certain level of service quality and performance in various telecommunication services to meet the needs of users. The quality of services can be defined by various factors such as reliability, delay, jitter, and bandwidth as listed in 〈Table 28〉.

〈Table 26〉 Protocols of network layer

Factor	Details
Reliability	<ul style="list-style-type: none"> <li>Reliability is required for a flow to securely deliver a packet to the destination</li> </ul>
Delay	<ul style="list-style-type: none"> <li>Delay literally means the delay in the packet delivery from the transmitting end to the receiving end</li> </ul>
Jitter	<ul style="list-style-type: none"> <li>Jitter means packet delay variations within a single flow</li> </ul>
Bandwidth	<ul style="list-style-type: none"> <li>Bandwidth means the maximum transmission speed or the capability to transport information</li> </ul>

### Techniques Used to Secure Quality of Service

#### ① Scheduling

The packet processing job for the Internet is carried out by routers, which means the way scheduling is done can change how packets will move.

〈Table 29〉 Packet scheduling method

Scheduling method	Details
FIFO queuing	<ul style="list-style-type: none"> <li>A basic scheduling of the Internet, packets are delivered on a first-in-first-out basis</li> </ul>
Priority queuing	<ul style="list-style-type: none"> <li>Priority is given to the packets so that the packet in the highest priority queue can be processed first.</li> </ul>
Weighted fair queuing	<ul style="list-style-type: none"> <li>Priority is given to packets and the packets are allocated to each of the priority queues.</li> <li>A round-robin selection of the queues to deliver the packets, but the number of selected packets can increase in accordance with the weight given to a certain queue.</li> </ul>

#### ② Traffic shaping and traffic policing

Traffic shaping and traffic policing are intended to control the speed and volume of the traffic: **Traffic Shaping** is used when the traffic leaves the network, while **Traffic Policing** is used when the traffic comes into the network.

〈Table 30〉 Methods for traffic shaping and traffic policing

Traffic shaping /policing	Details
Leaky-bucket	<ul style="list-style-type: none"> <li>Packets coming under the burstiness limit can be stored in the packet bucket and go out of the bucket at normal speed.</li> <li>When the incoming packets exceed the bucket limit, the packets can be removed.</li> </ul>
Token bucket	<ul style="list-style-type: none"> <li>A basic algorithm used in traffic shaping and traffic policing</li> </ul>

#### ③ Resource reservation

A way to reserve resources necessary for a data flow of a certain service to secure service quality: buffer, bandwidth, CPU, time, and the like.

#### ④ Admission control

A process used in the controlling part of a communications network node in order to make a decision whether to accept a request for admission.

#### ⑤ Service quality model and protocol

There are service quality models such as integrated service model (IntServ) and differentiated service model (DiffServ). In addition, the protocol used for service quality is the resource reservation protocol (RSVP).

⟨Table 31⟩ Service quality model and protocol [6]

Service quality model	Details
IntServ	• To make a clear-cut reservation for the resources such as bandwidth specific to the given data flow. (per-flow basis)
RSVP	• A standardized protocol to reserve/secure the bandwidth required for application programs on the two ends to provide a certain service. (RFC 2110)
DiffServ	• The types of services are selected every time when packets are transmitted, and the differentiated service model works based on the priority given to the packets.

## 05 Routing Protocols and Algorithms

### What is Routing Protocol?

**Routing Protocol** means a set of rules that define the type of messages exchanged among routers, processes of message exchange, and activities related to receiving messages, so that a routing table can be established and updated effectively.

### What is Routing Algorithm?

**Routing Algorithm** is intended to find out the most cost-effective path between a starting router and a destination router in a network graph that showcases links along with costs. The most cost effective path is the one that has the lowest sum of cost out of all the possible paths between a starting router and a destination router. The types of routing algorithms are listed in ⟨Table 32⟩.

⟨Table 32⟩ Types of routing algorithms

Algorithm	Details
Link State Routing	• Each router has information about the overall network structure and link state in order to find out the path with the lowest cost to all the destinations.
Distance Vector Routing	• Each router keeps the table for 'the path with the lowest cost' to all destination routers.(Vector) • An initial router path-cost table keeps the cost information pertaining to the router it is connected to and its path-cost information is notified to neighboring routers.

① Dijkstra's Algorithm

**Dijkstra Algorithm** is the most well known among link state algorithms. A set of pseudo code for the algorithm is shown in ⟨Figure 37⟩.

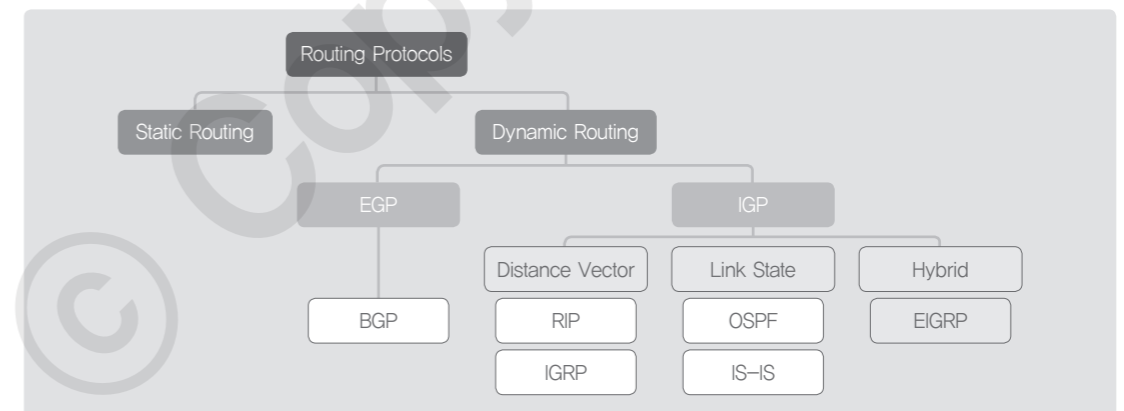
```

Dijkstra's Algorithm()
{
  //Initialization
  Tree = {root} //Tree is made only of the root
  for (y = 1 to N) //N is the number of nodes
  {
    if(y is the root)
      D[y]=0 //D[y]is shortest distance from root to node y
    else if (y is a neighbor)
      D[y] = c[root][y] // c[x][y] is cost between nodes x and y in LSDB
    else
      D[y]=∞
  }
  // Calculation
  repeat
  {
    find a node w, with D[w]
    Tree = Tree ∪ {w} //Add w to tree
    // Update distances for all neighbors of w
    for (every node x, which is a neighbor of w and not n the Tree)
    {
      D[x] = min{D[x],(D[w]+c[w][x])}
    }
  }
  } until(all nodes included in the Tree)
} // End of Dijkstra
    
```

⟨Figure 37⟩ Pseudo code of Dijkstra's algorithm

### Types of Routing Protocols

Routing protocols, as shown in ⟨Figure 38⟩, can be divided into **Static Routing** and **Dynamic Routing** based on how routing works. To drill down further, the dynamic routing can be divided into **EGP (Exterior Gateway Protocol)** and **IGP (Interior Gateway Protocol)** based on how two of them interact with AS (Autonomous System), and IGP can be categorized into **Distance Vector**, **Link State**, and **Hybrid** routing based on how the routing configuration is done.



⟨Figure 38⟩ Classification of routing protocols

Routing protocols can also be divided into the interior router protocol and the exterior router protocol when the Autonomous System (AS) is placed as a border line for classification. **AS (Autonomous System)** is a collection of networks that have the same operation policy and independent management system.

The protocol within the AS is called **IGP (Inter Gateway Protocol)**. IGP can be divided into: **RIP (Routing Information Protocol)** which uses a distance vector routing algorithm; **OSPF (Open Shortest Path First)** which uses a link state routing algorithm. The routing protocol between autonomous systems is called **EGP (Exterior Gateway Protocol)** and the most widely used EGP is **BGP (Border Gateway Protocol)** which uses a path vector routing algorithm.

### Types of Routing Protocols

Routing protocols can be categorized into: RIP which uses a distance vector algorithm and routing tables perform broadcasting in every 30 seconds for the mutual information interaction; IGRP which has updated some of RIP issues and factored in networking conditions (bandwidth, delay time, load, and the like) in the routing decision; OSPF which uses a link state routing algorithm; and BGP that is used to connect between autonomous systems. Details are explained in (Table 33).

(Table 33) Types of routing protocols

Protocol	Details
RIP (RFC 1058)	<ul style="list-style-type: none"> <li>• Uses a distance vector algorithm and routing tables performs broadcast in every 30 seconds for the mutual information interaction</li> <li>• Hop limit: Max 16 hops, VLSM not supported, load balancing not supported</li> <li>• Network condition (bandwidth, delay time, load, and the like) was not factored in.</li> </ul>
IGRP	<ul style="list-style-type: none"> <li>• Updated some of the issues of RIP</li> <li>• Networking conditions (bandwidth, delay time, load, and the like) were factored in.</li> <li>• More hops (Max 225 hops), VLSM not supported, load balancing supported</li> </ul>
OSPF (RFC 2328, RFC 1247)	<ul style="list-style-type: none"> <li>• Uses a link state routing algorithm</li> <li>• Distributes the information about changes faster than the RIP (such as user defined path, most cost effective path, multi paths, and the like).</li> <li>• All the routers maintain the same topology-related information, the VLSM and load balancing are supported.</li> </ul>
BGP (RFC 4271)	<ul style="list-style-type: none"> <li>• A kind of EGP connection between one AS to another, mutual connection for large size networks.</li> <li>• Uses TCP and based on a path vector routing</li> </ul>

## 06 Outline of IPv4

### What is IPv4 (Internet Protocol Version 4)?

**IPv4 Address** is a 32-bit address used in the IP layer, as shown in (Figure 39), and makes the Internet connection of a router or a host universal but unique at the same time.

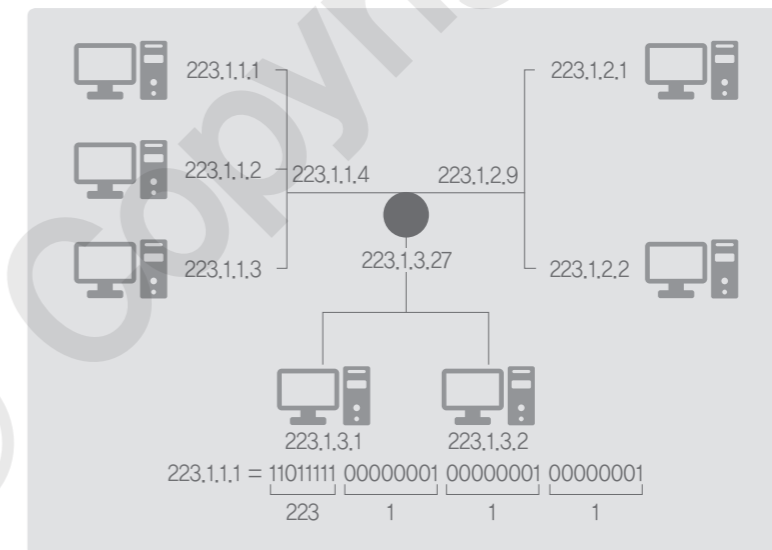


(Figure 39) IPv4 address structure

IPv4 address is composed of: an IP address (a unique identifier given to all the communications networks connected to the Internet and to the computers connected to the communications network); a network ID (used to identify the TCP/IP hosts located on the same physical network); a host ID (used to differentiate the TCP/IP hosts on the network); and a subnet mask (made with 32-bit size to define the network name for an IP address from the host name.)

There are various ways of describing IP addresses, one way is **CIDR (Classless Inter-Domain Routing, RFC 1519)** which is a way of IP address allocation where the existing 8-bit network portion and host portion are not divided. There are subnetting and supernetting as well, which does not divide the network ID of the IPv4 address with fixed 8-bit unit, but the network and host IP can be flexibly presented in accordance with the needs from a network. **Subnetting** means to divide a given IP address into small subnets considering the networking environment, while **Supernetting** means a number of network IDs are aggregated into a single network ID.

### An Example of a Network Using the IPv4

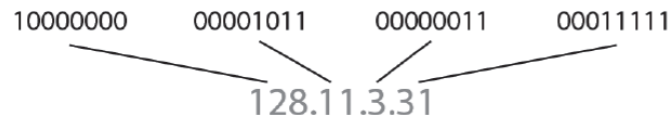


(Figure 40) Network using IPv4



## 07 IPv4 Addressing and Subnetting

### IPv4 Expression



- Binary notation: 01110101 10010101 00011101 00000010
- Dotted decimal notation

### IPv4 Addressing Structure

Class A	Network	Host	Host	Host
	1.0.0.0 ~ 126.255.255.255			
Class B	Network	Network	Host	Host
	128.0.0.0 ~ 191.255.255.255			
Class C	Network	Network	Network	Host
	192.0.0.0 ~ 223.255.255.255			
Class D	Multicast			
	224.0.0.0 ~ 239.255.255.255			
Class E	Research and Development			
	240.0.0.0 ~ 255.255.255.255			

(Figure 41) IPv4 addressing structure

IPv4 addressing structure is composed of: a network address which was given by an institute in charge of the Internet address resource management; and a host address given by a network administrator to identify individual hosts on the network.

Considering the size of a network and the number of hosts, networks can be classified into Class A, B, C, D, or E. Class A, B, and C are given to general users to build a network, Class D is for the multicasting and Class E is reserved for the future use.

(Table 34) IPv4 addressing structure [7]

Class	Address	Details
A	1.0.0.0 ~ 126.0.0.0	<ul style="list-style-type: none"> <li>• Network ID: first octet</li> <li>• Host ID: last three octets</li> <li>• Default subnet mask: 255.0.0.0</li> </ul>
B	128.0.0.0 ~ 191.255.0.0	<ul style="list-style-type: none"> <li>• Network ID: first two octets</li> <li>• Host ID: last two octets</li> <li>• Default subnet mask: 255.255.0.0</li> </ul>
C	192.0.0.0 ~ 223.255.255.0	<ul style="list-style-type: none"> <li>• Network ID: last three octets</li> <li>• Host ID: last octet</li> <li>• Default subnet mask: 255.255.255.0</li> </ul>
D	224.0.0.0 ~ 239.0.0.0	• Multicast address
E	240.0.0.0 ~ 255.0.0.0	• Experimental

In the IPv4 addressing, **Subnetting** means to divide one network ID into many network IDs, while **Supernetting** means to aggregate a number of small networks into one large network.

The 169.254.0.0 range is reserved for the Automatic Private IP Addressing, and DHCP can automatically work on this kind addressing to set up an IPv4 address, in the versions higher than the Windows 2000. However, this address cannot be routed to the Internet. Such addressing system is called Automatic Private IP Addressing or APIPA.

### Special IPv4 Address

In addition to the generally used classing systems (Class A, B, and C), there are special addresses which are called special IPv4 addresses and they are listed in (Table 35).

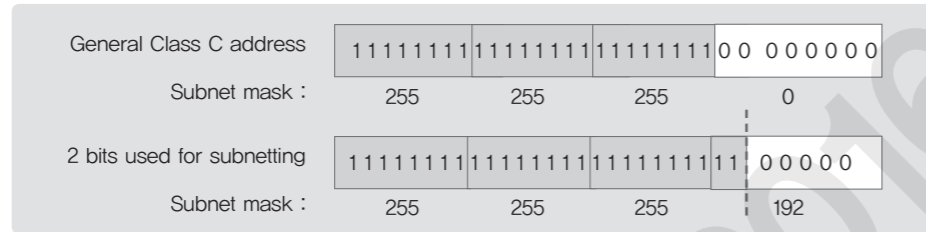
(Table 35) Special IPv4 address [7]

Network ID	Host ID	Address name	Details
specific	All 0	Network address	• Means a network address
specific	All 1	Net-directed broadcast to netID	• Used to broadcast to all the terminals within a certain network
All 0	specific	Specific host on this network	<ul style="list-style-type: none"> <li>• Specifies a terminal within the network</li> <li>• Cannot pass through a router</li> </ul>
127.X.XX		Local loopback address	• Used as a destination address looped back within a system
255.255.255.255		Limited broadcast	<ul style="list-style-type: none"> <li>• Used to broadcast to all the terminals within a router's network</li> <li>• Cannot pass through a router</li> </ul>
0.0.0.0		This host on this network	<ul style="list-style-type: none"> <li>• Used to present a terminal, when a terminal does not know its IP address</li> <li>• Cannot pass through a router</li> </ul>
10. ~	Any	Class A private address (10.0.0.0 ~ 10.255.255.255)	• A private address range that can be used without getting an authorization as a public IP.
172.16~172.31	Any	Class B private address (172.16.0.0 ~ 172.31.255.255)	
192.168.0 ~ 192.168.255	Any	Class C private address (192.168.0.0 ~ 192.168.255.255)	

### Subnetting

① Concept of subnetting

**Subnetting** means to divide a single network address into various small networks. Subnet mask refers to a mask used to identify the network address portion out of the IP address. [8].



<Figure 42> IP address using subnetting

② How subnet is used?

When a sub-network is added, a middle level layer is generated in the IP address and the IP address can be divided into three levels such as site, subnet, and host.



<Figure 43> Address structure when subnetting was used and not used

③ How subnet is applied? (Subnetting outcome using 2 or 3 bits in the Class C IP address)

<Table 36> Outcome of subnetting

Classification	Number of bits used for subnetting		
	(Not divided)	Using 2 bits	Using 3 bits
Subnet mask	255.255.255.0	255.255.255.192	255.255.255.254
Number of subnets	1	4	8
Number of hosts within a subnet	$2^8 - 2 = 254$	$2^6 - 2 = 62$	$2^5 - 2 = 30$
Number of IP addresses that can be potentially assigned	254	$4 \times 62 = 248$	$8 \times 30 = 240$

### CIDR (Classless Inter-Domain Routing)

① What is CIDR?

CIDR is intended to aggregate Class C addresses into a single network. This method can be helpful in reducing

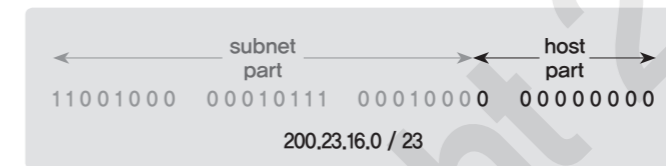
the size of a routing table and in using IP addresses with various subnet types.

② Role of CIDR

- The IP address is not classified by Class A, B, or C, but the network identifier can be flexibly assigned to make the IP address operation more flexible.
- As there is flexibility in assigning network addresses, CIDR can prevent IP addresses from being wasted and can help build a more efficient network.
- It is possible to effectively manage IP addresses that are used for routing among domains.

③ How CIDR is expressed?

The subnet portion of the IP address has a random length, and is expressed in a.b.c.d/x. Here, 'x' is the subnet portion of the address.



<Figure 44> CIDR expression

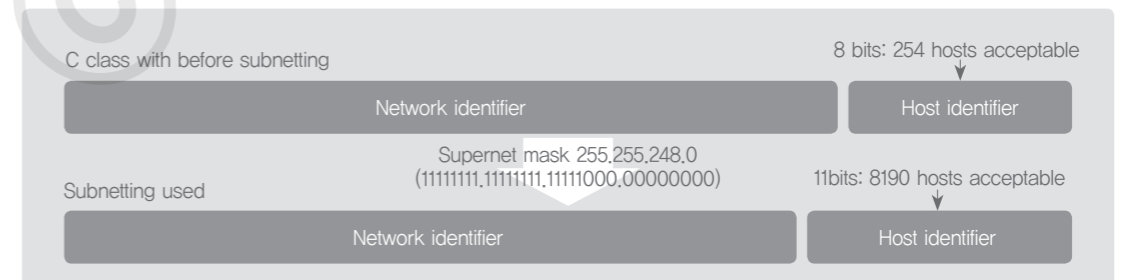
### Supernetting

① What is supernetting?

**Supernetting**, which literally works in a reverse way as subnetting does, aggregates multiple small networks into a single large network. A series of class address blocks are assigned to a single organization, so that those blocks can be regarded as one block from outside and the routing can be made to the block [9].

② How supernetting works?

- Building a single network by aggregating a lot of Class C addresses
- A part of network identifier is used as a host identifier.
- The number of Class C addresses to be aggregated should be '2 powered by N' and the addresses should be in a consecutive order.



<Figure 45> Supernetting mechanism

### How IPv4 Address is Assigned?

① DHCP(Dynamic Host Configuration Protocol)

**DHCP** means a protocol that provides a way to dynamically assign IP addresses and other detailed information to the DHCP client by using a DHCP server.

A DHCP server can assign the IP address and subnet mask and can additionally assign the default gateway, DNS server address, local router, lease time, domain name, and the like with the DHCP option.

The IPv4 DHCP lease process is conducted through the exchanges of four UDP messages. DHCPDISCOVER, DHCPOFFER, DHCPREQUEST, and DHCPACK are the four UDP messages and the details about them are listed in <Table 37>.

<Table 37> DHCP message [10]

Message	Details
DHCPDISCOVER	The IP address sent from a client, corresponding to 255.255.255.255
DHCPOFFER	The IP address, subnet mask, default gateway, lease limit and other information are delivered from the DHCP server
DHCPREQUEST	This message is sent from the client to the server. (Including that the client selected lease related information)
DHCPACK	The DHCP server sends the DHCPACK message to the client in order to allow the use of the IP address.

② NAT(Network Address Translation)

**NAT** refers to a function which can allow terminals on the private network to have communications with the public network such as the Internet. It is well defined in RFC 3022 and RFC 2663.

The reason to use NAT is: to resolve the lack of IP addresses; and to hide the internal IP from external attackers.

**NAPT (Network Address Port Translation)** is an expanded version of the basic NAT and intended to translate the source port number in the packet. NAPT is used because it allows a single IP address to gain access to multiple internal hosts. Theoretically, NAPT can handle up to 65,535 internal host communications only with a single public IP address.

<Table 38> Types of NAT [11]

Type	Details
Basic NAT	• One to one mapping for the Internet communications between 'terminals with private IP address' and 'public IP' of the Internet
NAPT	• One to many mapping for the Internet communications between 'many terminals with private IP' and 'single public IP' in order to save the number of public IP addresses

### Example Question

**Question type**

Descriptive question

**Question**

The diagram below shows an ISP (Internet Service Provider) network covering the Seoul area. A node of each area has a routing table to send the data to other nodes at the lowest cost. Fill out the routing table of the node in Seodaemoun District.

[Seodaemoun District node routing table]

Destination	Next Node
Kangseo District	
Eunpyeong District	
Kwanack District	
Dongdaemoun District	
Kangdong District	

**Intent of the question**

To understand how the routing protocol works and how to make a routing table

**Answer and explanation**

Following is the path with the lowest cost from Seodaemoun District to other five nodes

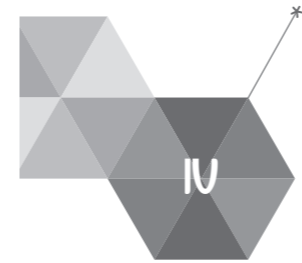
- Kangseo District: Seodaemoun District – Eunpyeong District – Kangseo District
- Eunpyeong District: Seodaemoun District – Eunpyeong District
- Kwanack District: Seodaemoun District – Kwanack District
- Dongdaemoun District: Seodaemoun District – Eunpyeong District – Kangseo District – Kangdong District – Dongdaemoun District
- Kangdong District: Seodaemoun District – Eunpyeong District – Kangseo District – Kangdong District

Therefore, the data whose destination is Kangseo District, Eunpyeong District, Kwanack District, Dongdaemoun District, Kangdong District moves: from Seodaemoun District to Eunpyeong District, Eunpyeong District, Kwanack District, Eunpyeong District, Eunpyeong District respectively.

### Related E-learning Contents

- Lecture 3 Network Layer and Routing Protocol

- **Lecture 4** IP Address Structure and Mobile IP
- **[Advanced]** **Lecture 1** Network Layer and the IPv4 Address Structure  
**Lecture 2** IPv4 Sub-netting/super-netting and Address Allocation  
**Lecture 3** Routing Algorithm and Routing Protocol



## Transport Layer Protocol

### ▶▶▶ Latest Trends and Key Issues

The rise of the 4G wireless communications services enables voice communications to be delivered as a part of data communications in the form of VoIP. It has become a trend that many applications are run on the Internet. The transport layer is a protocol layer which is directly linked with these applications. The network can be utilized efficiently only when an appropriate transport layer protocol is used in accordance with the features of the application.

### ▶▶▶ Study Objectives

- \* To be able to understand the purpose and functions of the transport layer and utilize the same
- \* To be able to understand the concept of TCP, related services, and control methods and to utilize the same
- \* To be able to understand the concept of UDP, related services, and control method and to utilize the same
- \* To be able to explain the concept of SCTP and how it works

### ▶▶▶ Practical Importance High

### ▶▶▶ Keywords

TCP, UDP, SCTP, Flow control, Error control, Congestion control, Slow Start, SNMP, Multicasting, NTP, DHCP, Kerberos, Multi-homing

### Practical tips Mechanism of Action of the Transport Layer Protocol

The transport layer protocol works end-to-end, which means it provides the data delivery service between User A's computer and the web server (my.server.com) to which the request for information is delivered. Data packets go through a number of hops between the client and the server. In the end-to-end connection, intermediate nodes, such as wired/wireless IP sharing devices or routers, are not involved in the operation of the transport layer protocol. TCP[1], UDP[2] and SCTP[3] are regarded as most widely used transport layer protocols.

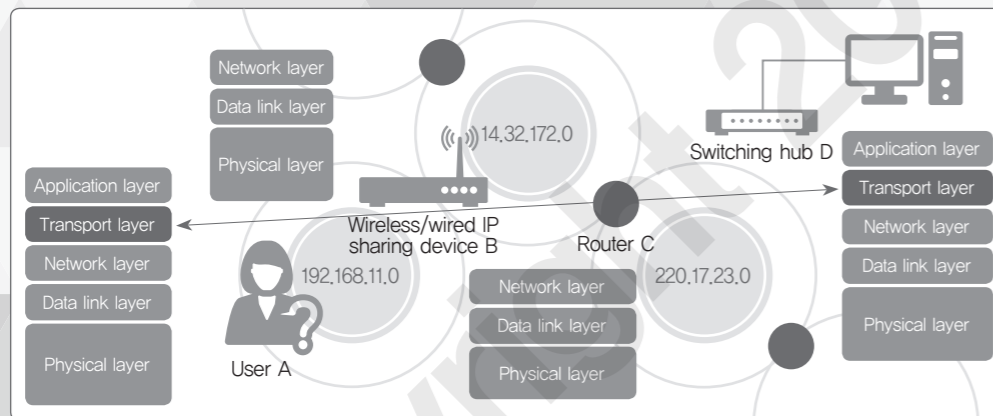


Figure 46 Mechanism of actions of transport layer protocol

## 01 Concept of Transport Layer Protocol

The transport layer protocol is responsible for transmitting the application data from one endpoint host to the other. The server application, which runs on the server, receives and processes the request from the client and then, sends back the response to the client. On the client side, the user selects an application which can deliver the service he/she needs and sends a request to the server. The response to that request is sent back to the client and shown to the user. Applications running on the network usually operate in the client-server model and the transport layer protocol is the lowest layer protocol which connects the client application to the server application.

TCP (Transmission Control Protocol) [1] and UDP (User Datagram Protocol) [2] have long been widely used among the transport layer protocols. SCTP (Stream Control Transmission Protocol) [3] has newly emerged since 2000, combining the strengths from both TCP and UDP. In this chapter, we only look at TCP and UDP which are most commonly used.

UDP is usually known as a protocol which offers a connectionless service. It does not mean that the service is provided without a connection. It means that data is transferred without a prior arrangement (without establishing a connection). On the contrary, TCP is a protocol which provides a connection-oriented service. The connection-oriented service requires that a connection be established before the data is sent or received. In this connection-oriented protocol, the connection and the service between the client and the server are terminated after the data transfer is done.

Reliability is a key feature which always comes along with the theory of the connection-oriented/connectionless services. UDP is usually known as an Unreliable Protocol, while TCP is regarded as a protocol which offers Reliable inter-process communications. TCP utilizes the response acknowledgement technique, a way of using sequence numbers and acknowledge numbers to validate that the transmitted data was received successfully.

For an application program developer, it is not a must to understand all the details on the transport layer. Rather, the developer is generally required: to know which protocol, TCP or UDP, should be selected and used, based on the understanding about their strengths and weaknesses; and to understand the service ports. In reality, however, the program development is mostly a group work, such as a project where at least two companies are working together, and a program is developed based on the linkage with a variety of other programs. Accordingly, most of programs, developed by the developers, send and receive data through the network. Hence, the developer is often required to validate that the data was sent successfully to the destination when the program is not working even after the program was fully developed to meet the service requirement and protocols defined between the application programs. In such a case, a network debugging tool, such as Wireshark or tcpdump, can be used to inspect the transmission of packets and the developer can work on debugging. There are a number of reasons for the packet transmission failures: wrong encoding method, wrong order (between big-endian and little-endian), incorrect packet processing order, wrong data format, or incorrect data parsing. The problem is that if the software developer only debugs a self-written application program (unit level), it might be time-consuming to both of the receiving end and the transmitting end. Therefore, the developer needs to understand how the transport layer works and how the data is transferred to the application layer. It is recommended to inspect the packets being transferred on the network so as to reduce the time required for debugging.

## 02 TCP

### Characteristics of TCP

TCP lies between the application layer and the network layer of the TCP/IP model and offers inter-process communications between two application layers. TCP is a stream-based protocol and has a sending buffer and a receiving buffer used for data transfer. The IP layer, residing at the bottom of the TCP protocol, offers packet-based services, not stream-based services in the data transfer. To this end, a single data is broken into a number of Segments by TCP. TCP defines the flow control, the error control, and the congestion control techniques to guarantee the reliable delivery of data. TCP assigns a number to each byte transmitted, and then, the sequence number is given to each segment. Maximum value should be in place so as to include the sequence number of each packet in the header. If  $m$  bits are reserved for the sequence number in the packet header, then the sequence numbers shall range from 0 to  $2^m - 1$ . In other words, the sequence numbers are modulo  $2^m$ .

① Flow control

Flow control coordinates the amount of packet flows to avoid packet losses, which can take place when too many data packets come in beyond the receiving capacity. The sending/receiving buffers are used for the flow control, which can store packets on the transmitting end and receiving end. The size of the sending/receiving buffer is specified, in the case of Linux, in /proc/sys/net/core/wmem\_max and /proc/sys/net/core/rmem\_max files respectively.

② Error control

Error control is a technique of detecting errors occurring during the transmission to ensure that the correct information is recovered. Error control detects and discards corrupted packets, keeps track of lost or removed packet, retransmits the lost or removed packets, and checks and discards packets received redundantly.

③ Congestion control

Congestion occurs when the network load (which refers to the number of packets being transmitted per unit time through the network) exceeds the network capacity (which means the number of packets that can be processed per unit time through the network). TCP uses this technique to control the congestion end-to-end and keeps the load level below the network bandwidth.

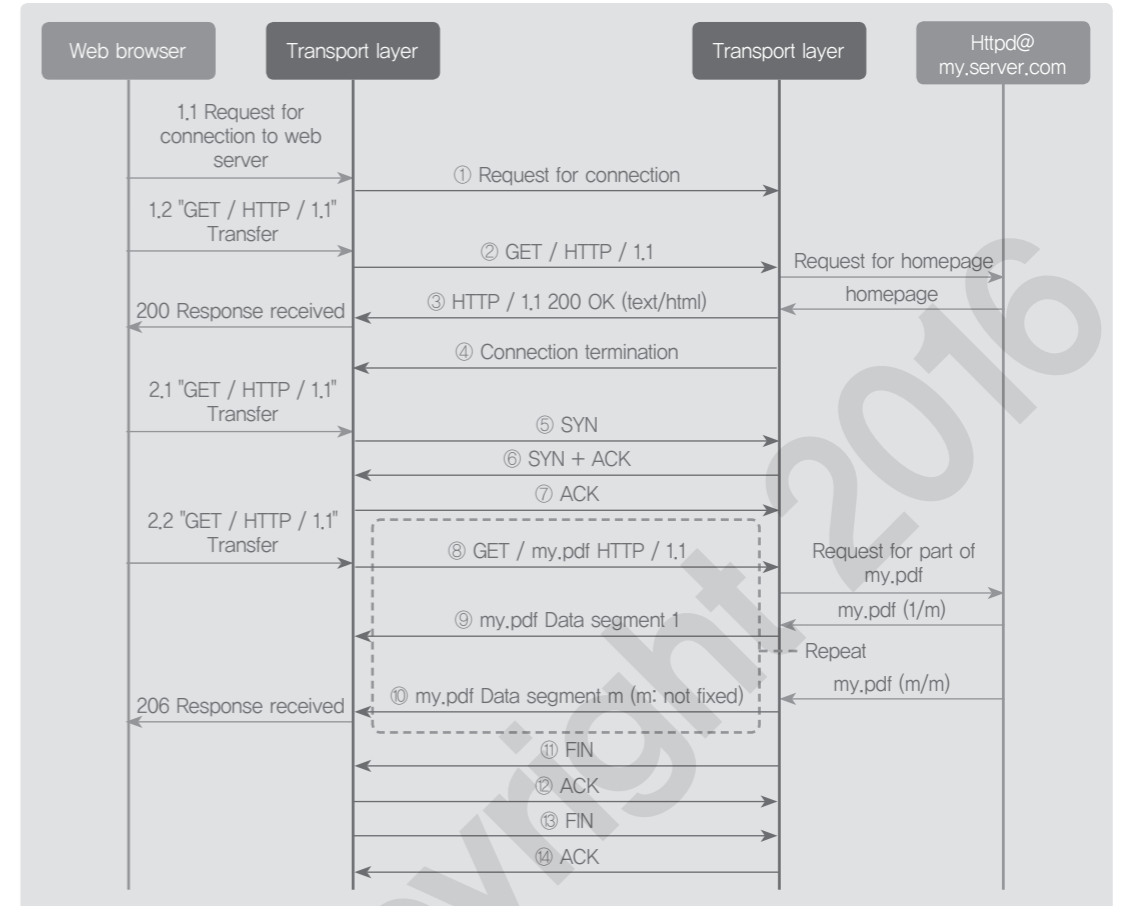
The following table presents the well-known ports which are employed by the widely used TCP services. FTP (file transfer protocol), SSH (secure remote login protocol), SMTP, POP3 and IMAP4 (protocols for mail transfer), HTTP (a protocol for web service), and many other widely used internet services are the examples of the well-known ports.

<Table 39> Well-known TCP ports

Service	TCP port	Service	TCP port
FTP [4]	21 (control), 20 (data)	DNS [8]	53 (UDP as well)
SSH [5]	22	HTTP [9]	TCP 80
Telnet [6]	23	POP3 [10]	110
SMTP [7]	25	IMAP4 [11]	143

Scenario for TCP Operation

The following figure shows how data is sent and received on the TCP protocol when User A downloads a file (my.pdf) from a homepage (my.server.com). Requests and responses, between the application layer program (web browser) and the web server (Httpd), are transferred via TCP which is a transport layer protocol. This chapter will mainly focus on the operation of the TCP protocol.



<Figure 47> Scenario of TCP operation

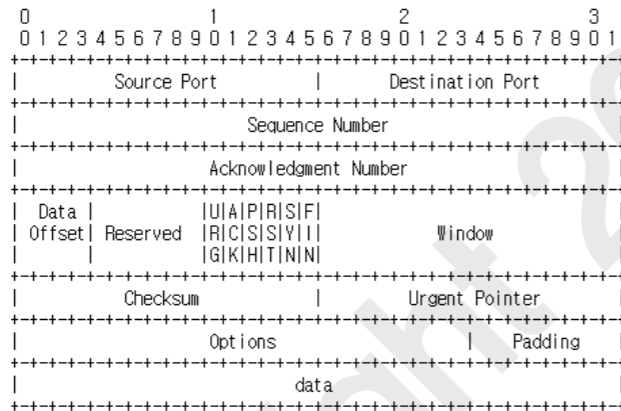
The request for a connection (Step ①) is a simple summary drawing of Step ⑤~⑦, while The connection termination (Step ④) is a summary drawing of Step ⑪~⑭. Step ②~③ and ⑧~⑩ are the application layer data which comes right after the TCP header. This data is generated by the web browser of User A's computer and the web server program (my.server.com) in accordance with the HTTP protocol. The details of the TCP protocol and the TCP operation steps will be explained in the following paragraphs.

IP packets may be delivered via many different network paths when a number of data, such as PDF files, are transferred at once, as shown in Step ⑧~⑩. In this regard, the flow control by TCP serves a significant role. Error control is also one of the important techniques of TCP, as it can be useful in addressing transmission errors caused by hosts on the network or on the Internet lines. TCP performance is also highly affected by congestion control, a method of keeping the amount of packets transmitted on the network below the network capacity (the number of packets that can be processed per unit time). Therefore, it is important to understand these techniques which are critical to the reliable operation of TCP.

## TCP Protocol

The header format used in the TCP protocol is shown in the following [1].

When adding options, the padding bytes should be added in a multiple of 4 bytes. A TCP header is 20 bytes if there is no option, which is equivalent to the size of the IP header. A TCP/IP header is 40 bytes in total and should be sent even when a single byte of user data is sent. Therefore, the amount of data transferred to the transport layer should reach a certain level so as to allow the application layer program to transfer the data more efficiently on the network.



(Figure 48) TCP header format [1]

(Table 40) Components of TCP protocol header

Classification	Details
Source port address	<ul style="list-style-type: none"> <li>Defines the port number of the host application program which sends the segment (16 bits)</li> <li>Offers the same function as the source port number in the UDP header</li> </ul>
Destination port address	<ul style="list-style-type: none"> <li>Defines the port number of the host application program which receives the segment (16 bits)</li> <li>Offers the same function as the destination port number in the UDP header</li> </ul>
Sequence number	<ul style="list-style-type: none"> <li>The number assigned to the first byte of the data included in the segment (32 bits)</li> </ul>
Acknowledgement number (ACK number)	<ul style="list-style-type: none"> <li>The byte number that the receiver of the segment expects to receive (32 bits)</li> </ul>
Header length	<ul style="list-style-type: none"> <li>The length of the TCP header, measured in the units of 4 bytes (4 bits)</li> <li>(Header length: 20 ~ 60 bytes)</li> </ul>
Control	<ul style="list-style-type: none"> <li>Defines 6 different control fields or flag bits</li> </ul>
Window size	<ul style="list-style-type: none"> <li>Window size which should be kept by the opposing party, in the units of bytes</li> <li>Field length: 16 bits, max. Window size: 65,536 bytes</li> <li>The size of the data which can be sent to the segment, used in the window mechanism</li> <li>The size of buffers</li> </ul>
Source port address	<ul style="list-style-type: none"> <li>Checksum, which detects errors in the entire segment (16 bits)</li> </ul>
Destination port address	<ul style="list-style-type: none"> <li>Valid only when the urgent flag is set (16 bits)</li> <li>Utilized when the segment includes the urgent data</li> </ul>

The Sequence number, SEQ, increments by one in each 8-bits octet and is based on unsigned modulo 232 arithmetic. Therefore, the SEQ is changed as follows: 0 → 232 - 1 → 0 → ...

Six flags which can be used for the controlling purpose are explained below. It is useful to understand SYN, ACK and FIN control flags which are widely used in the TCP protocol.

- **URG:** urgent pointer field is valid
- **ACK:** acknowledgement field is valid
- **PSH:** push function
- **RST:** reset the connection
- **SYN:** synchronize sequence numbers
- **FIN:** no more data from the sender

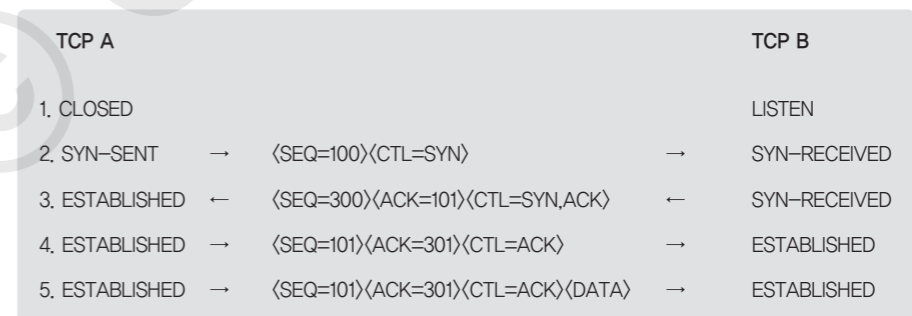
### ① TCP connection establishment

The three-way handshake is a basic procedure for establishing a TCP connection. As shown in the Step 2 of (Figure 49), this procedure starts by sending a SYN segment (SEQ=100) from one TCP endpoint (TCP A) to the other (TCP B). The SYN segment (SEQ=100) indicates the SYN control flag for this segment is set to 1. An ACK segment indicates that the ACK control flag is set to 1. A SYN+ACK segment means that both SYN and ACK control flags are set to 1.

In the Step 3 of (Figure 49), TCP B, after receiving the SYN segment, replies back with an ACK segment (ACK=101) and sends a SYN segment of its own for the connection establishment by setting a SYN control flag (SEQ=300). The aforementioned actions are carried out in a single segment called "SYN+ACK segment". TCP A receives the ACK control flag and ACK=101 which indicates that the request for connection was successfully delivered.

From the SYN control flag set, TCP A also acknowledges that the request for the connection was sent by TCP B. Then, as the final step of the three-way handshake procedure, TCP A replies with the ACK segment (ACK=301) for TCP B. At this point, a two way connection is established between TCP A and TCP B as the three-way handshake was completed.

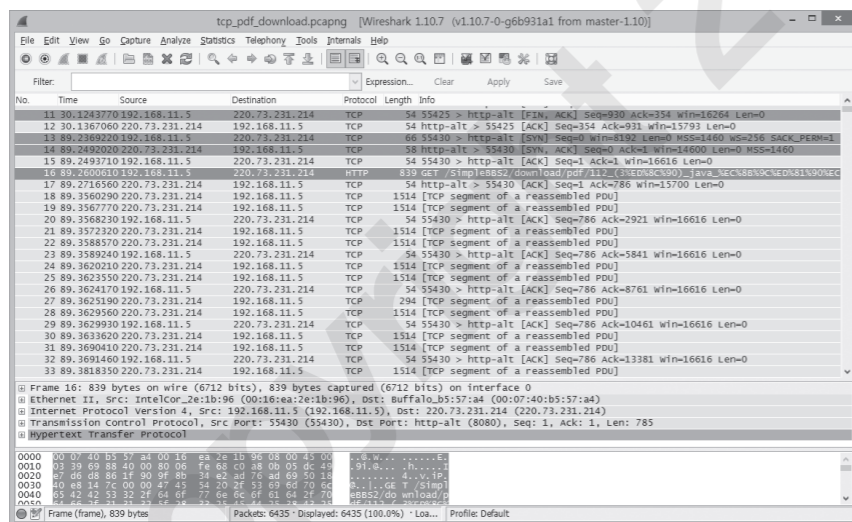
In the three-way handshake procedure, which was used to establish a TCP connection, SYN segment, SYN+ACK segment, and ACK segment are transferred between the two TCP endpoints. The SEQs, which are sent/received between the two points, serve a significant role in the flow control and the error control— a set of techniques to ensure that data streams to be sent/received are successfully transferred in the units of bytes.



(Figure 49) Three-way handshake procedure for connection establishment [1]

After receiving the SYN segment for establishing a TCP connection, TCB (Transmission Control Block), a system resource, is allocated to manage the TCP connection. However, it can be maliciously used in a way that a large number of SYN segments are sent in a short period of time to the TCP terminal host, so that there is no TCB left to be allocated in the host and then, the service becomes unavailable. This attack is called SYN flooding, a type of Denial-of-Service (DoS) attack.

In order to capture and analyze packets by using the computer's NIC (Network Interface Card) when TCP is in operation, as shown in (Figure 47), a network protocol analyzer, such as Wireshark, can be used. Programs, such as tcpdump, can offer a similar service on the Linux environment. The following figure shows the screenshot of packets captured by Wireshark on the Windows, which explains what happens when the user downloads and stores a PDF file on the web. Let's suppose that User A's host IP address is 192.168.11.4 and the web server's IP address is 220.73.233.214. Even though HTTP uses TCP, it is connectionless on the application layer. Therefore, in such a case, a connection is newly created, requested, and terminated after the request for the connection is processed, whenever there is an attempt to gain access to the web server.



(Figure 50) Wireshark (Network protocol analyzer): screenshot on TCP connection procedures

“Request for connection”, shown in (Figure 47), is a step in which User A makes a request for a TCP connection between the web browser and the web server (my.server.com) to see the homepage of the web server (my.server.com) on the web browser. In TCP, this can be done by using three-way handshake procedures. Packet No. 13, shown in (Figure 50), is the SYN segment sent to the web server by User A and its SEQ number is set to 0. Accordingly, the ACK number in the SYN+ACK segment (No. 14) sent by the web server is set to 1. The SYN control flag is also set to 1 in order to establish a TCP connection. User A receives the SYN segment from the web server which is set to synchronize the SEQ number. Then, User A sends the ACK segment (No. 15) with ACK=1 and the TCP connection is finally established.

② TCP connection termination

TCP connection termination is achieved in the following steps. As in the Step 2, TCP A sends a FIN segment (SEQ=100, ACK=300) to TCP B in order to terminate the connection. Along with the FIN segment, the ACK control

flag is set and sent to acknowledge the previously-delivered segment. After receiving them, TCP B replies with the ACK segment (SEQ=300, ACK=101) to acknowledge, like in the Step 3. The ACK number of this segment is 101 (calculated by adding 1 to the SEQ number 100 of the FIN segment) because it is sent to acknowledge the receipt of the FIN segment (SEQ=100).

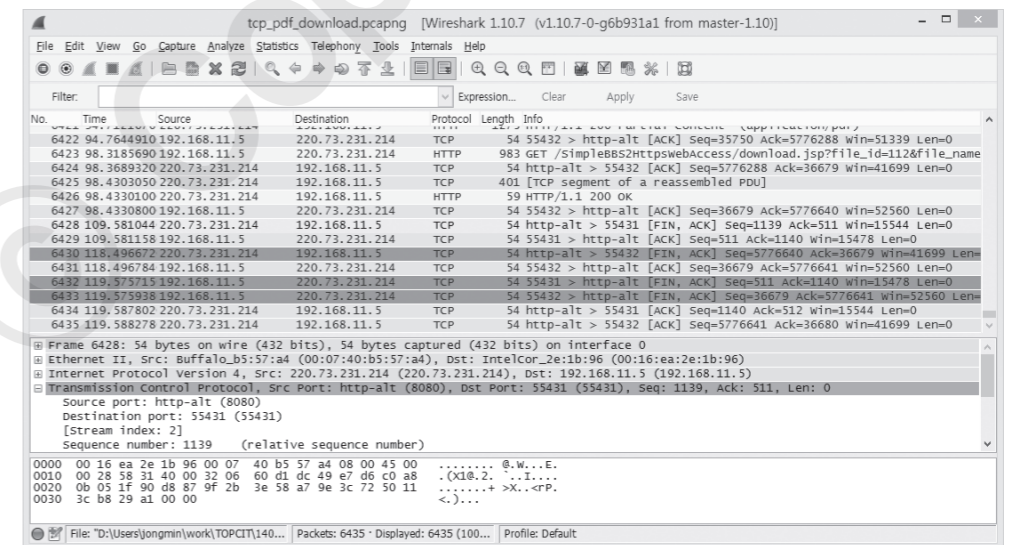
Next, TCP B sends the FIN segment (SEQ=300, ACK=101) to TCP A in order to terminate the connection just as in the Step 4. As was done in the Step 2, the ACK is sent together to acknowledge the receipt. Then, just as in the Step 5, TCP A sends back the ACK segment (SEQ=101, ACK=301) and the TCP connection is closed.



(Figure 51) General procedure for connection termination [1]

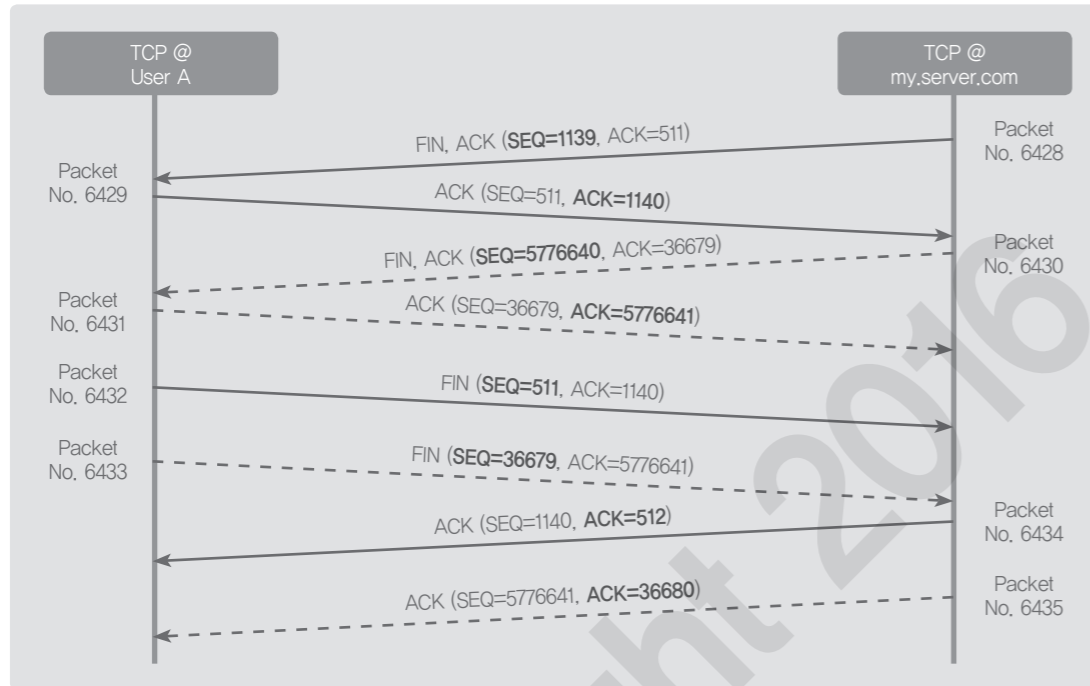
In the scenario of the TCP operation shown in (Figure 47), the steps of ④, ⑪~⑭ constitute the TCP connection termination.

In the packet capture screenshot shown below, packets from No. 6428 to No. 6435 represent the processes for TCP connection termination. Steps for TCP connection termination simplified in (Figure 53) may seem different from the steps illustrated in (Figure 52), because (Figure 53) shows the termination procedures of two different TCP connections in the same figure. However, if you distinguish the transfer of the packets drawn in a solid line from the transfer of the packets drawn in a dotted line, you can realize that the steps illustrated in the both figures are identical.



(Figure 52) Wireshark (Network protocol analyzer): screenshot on TCP connection termination procedures



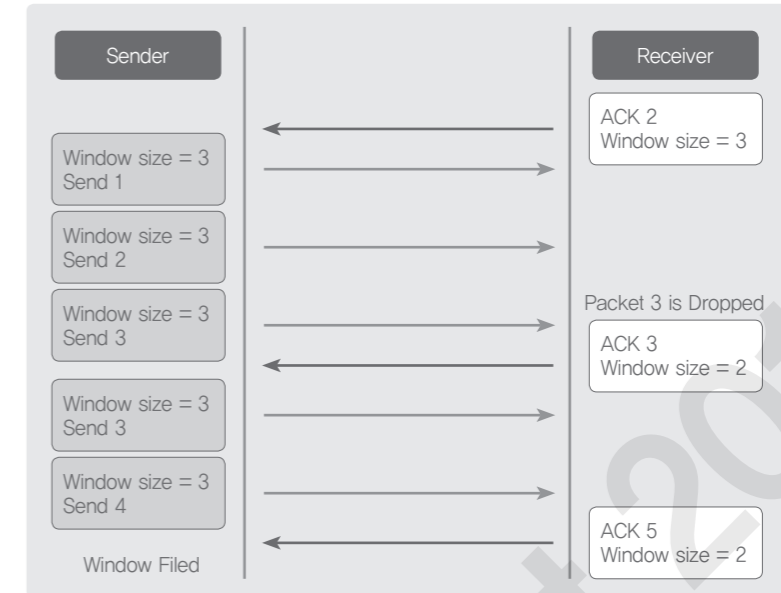


(Figure 53) Packet transfer when terminating TCP connection

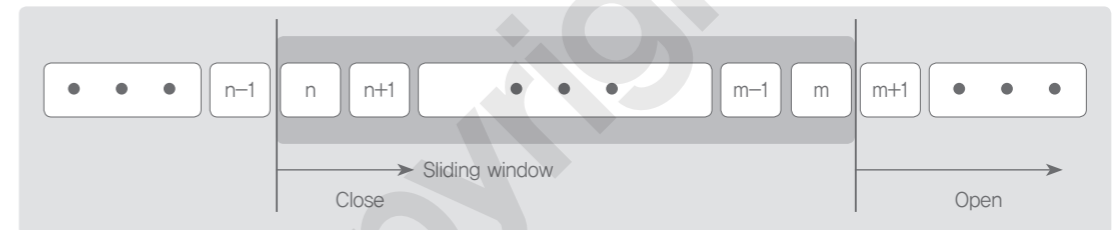
(Figure 53) shows the termination procedures for two different TCP connections, each drawn in a solid line and dotted line respectively. However, we will only look at the packet transfer illustrated in a solid line as the termination procedures for the two connections are achieved in the same manner. The web server sends an FIN+ACK segment (SEQ=1139, ACK=511, Packet No. 6429) to User A in order to terminate the TCP connection. Then, User A replies with an ACK segment (SEQ=511, ACK=1140, Packet No. 6429) for the acknowledgement of the receipt. The ACK number of the ACK segment sent by User A is 1140, calculated by adding 1 to 1139 which is the SEQ number of the FIN+ACK segment. User A, after sending the ACK segment, sends a FIN segment (SEQ=511, ACK=1140, Packet No. 6432) to the web server in order to terminate the connection. After receiving the FIN segment, the web server replies with the ACK segment (SEQ=1140, ACK=512) for the acknowledgement. In the same way the ACK number of the ACK segment (Packet No. 6429) sent by User A is calculated, the ACK number of the ACK segment sent by the web server is 512, calculated by adding 1 to the SEQ number of the FIN segment.

③ Flow control

Flow control is a mechanism for speed-matching – matching the rate at which data is generated to the rate at which data is used. For the flow control, TCP uses a sliding window protocol. The receiving TCP sets the number of octets (1 byte) and advertises it to the sender TCP. The sending TCP refers to the number in order to adjust the sliding window size. The length of this field is 16 bits, so the maximum window size is 65,535 bytes.



(Figure 54) Example of packet transmission in sliding window protocol



(Figure 55) Concept of sliding window protocol

(Table 41) Window open/close of the sliding window protocol

Action	Details
Window open	<ul style="list-style-type: none"> <li>The right edge of the window moves to the right when an ACK arrives from the receiver.</li> <li>Data is allowed to be transmitted as much as the window moves.</li> </ul>
Window close	<ul style="list-style-type: none"> <li>The left edge of the window moves to the right when data (bytes) transfer is acknowledged.</li> <li>The sender doesn't have to pay attentions to the data.</li> </ul>

Receiver window (rwnd) and congestion window (cwnd) are utilized. The size of the window is determined by the lesser of the two values: rwnd or cwnd.

<Table 42> Receiver window (rwnd) and congestion window (cwnd)

Classification	Details
Receiver window, rwnd	<ul style="list-style-type: none"> <li>The size of packets permitted to be transmitted without any data losses, when sending data from the sender to the receiver</li> <li>Advertised by using the segment which includes an ACK</li> </ul>
Congestion window, cwnd	<ul style="list-style-type: none"> <li>The size of packets permitted to be transmitted at once, in accordance with the congestion level on the network</li> <li>When the level of congestion goes up, the congestion window decreases, in order to avoid data losses. When the level of congestion goes down, the congestion window increases.</li> </ul>

④ Error control

Error control is a mechanism for detecting and handling lost, corrupted, out-of-order or duplicated segments.

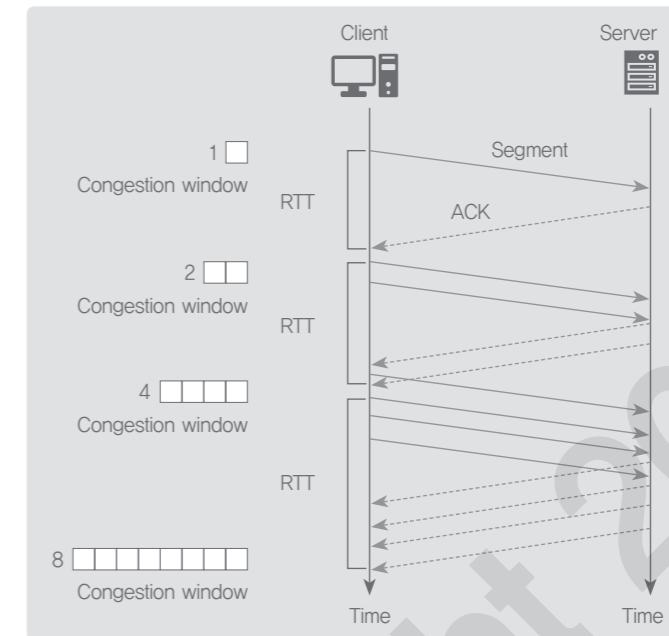
<Table 43> Tools used for TCP error control

Tools	Details
Checksum	<ul style="list-style-type: none"> <li>Each segment includes a checksum field, used to check for corrupted segments</li> </ul>
Acknowledgement	<ul style="list-style-type: none"> <li>Used to confirm the receipt of data segments</li> </ul>
Retransmission	<ul style="list-style-type: none"> <li>A key tool used for the error control mechanism. Segments are stored in the buffer until an acknowledgement is received.</li> </ul>

⑤ Congestion control

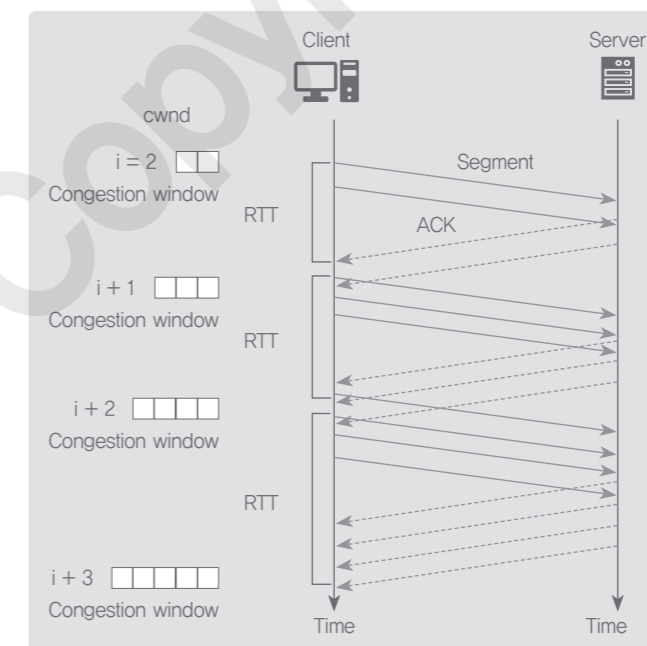
Congestion control is a mechanism (or technique) for keeping the amount of traffic, sent by the user on the network, below the capacity of the network. TCP's congestion control can be achieved by two key algorithms: **Slow Start** and **Congestion Avoidance**.

Slow start algorithm is a strategy used for congestion control by increasing the congestion window exponentially to its threshold.



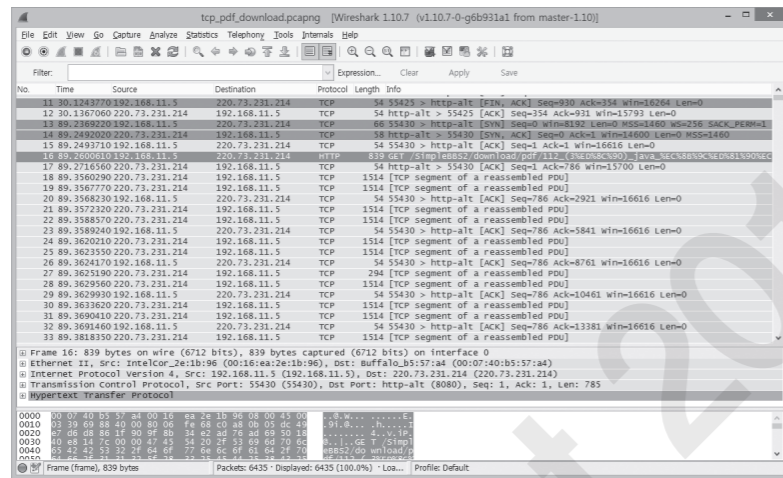
<Figure 56> Concept of slow start

Congestion avoidance algorithm is a strategy to increase the congestion window additively until congestion is detected.



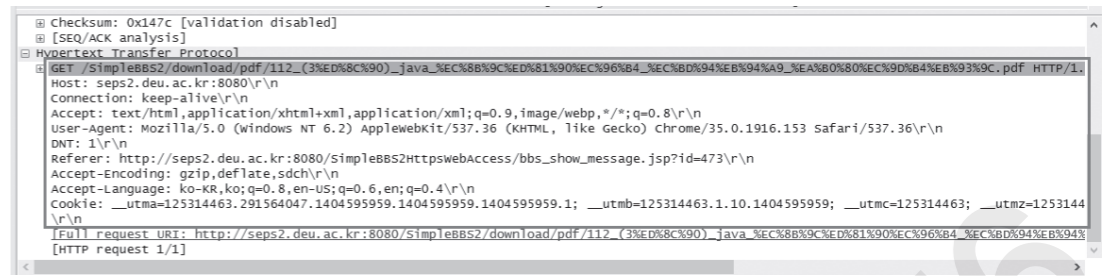
<Figure 57> Concept of congestion avoidance algorithm

Let's take an example which shows how the actual data is transmitted by the TCP flow control, error control, and congestion control on the screenshot of packets captured. It is identical to the screenshot in (Figure 50), but we will look at the packets which are received/transmitted after the TCP connection establishment.



(Figure 58) Screenshot on the packet transmission captured by Wireshark (network protocol analyzer)

(Figure 59) is presented for your better understanding. In this figure, Packet No. 16 sends a request for downloading a file via the HTTP protocol. This packet actually has user data, as shown in (Figure 60), but it will not be explained in detail. The size of user data is 785 bytes, so the ACK number of the ACK segment (Packet No. 17) for the request is set to 786. This indicates that user data corresponding to 'HTTP GET' is successfully received by the web server. This request was sent for downloading a file. Therefore, the web server sends the requested files in units of segments, as shown in Packet No. 18 and 19. A total of 2920 bytes of user data is received (1460 bytes for each segment). The TCP end-host of User A sends an ACK segment (ACK=2921) to confirm that the two TCP segments were successfully received. All processes explained above shows what happens when transmitting user data by using the TCP flow control and error control mechanisms.



(Figure 60) Example of HTTP GET

© TCP timer

TCP implementation usually depends on four timers to make the TCP operation smooth.

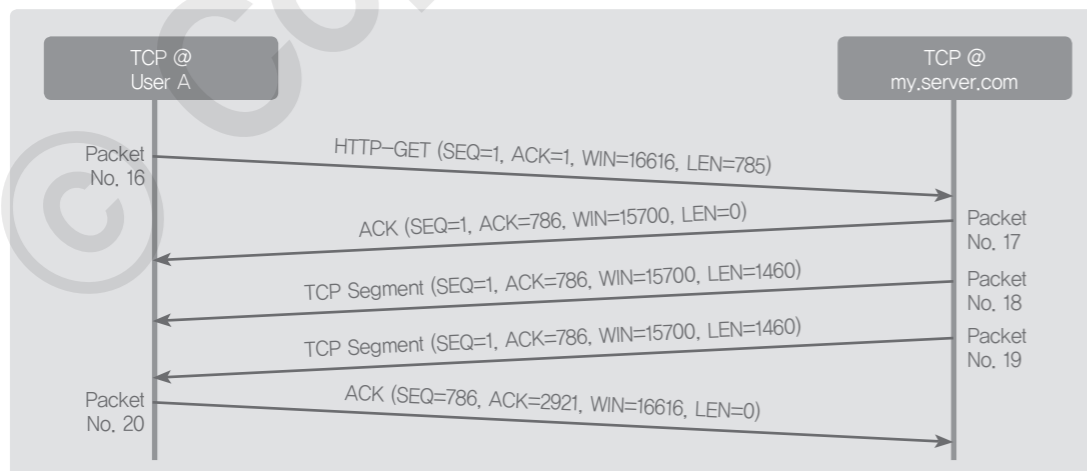
(Table 44) Types of TCP timers

Type of timer	Details
Retransmission	• Timer used to retransmit a lost segment. It works based on the RTO (retransmission time-out) which indicates the duration of waiting for the acknowledgement of the segment.
Persistence	• Used in every TCP connection to prevent a deadlock between the two TCP ends.
Keepalive	• Used to prevent a long idle TCP connection
Time-Wait	• Used during the connection termination and initiated when the last ACK is being sent. (To prevent potential failure to make a new connection.)

### 03 UDP (User Datagram Protocol)

#### Characteristics of UDP

User Datagram Protocol (UDP) is a protocol which provides connectionless service without prior arrangement. As opposed to TCP, UDP offers less reliable transport layer service. Each user datagram sent by UDP is an independent datagram and is not numbered, as opposed to TCP. UDP is so simple protocol that it doesn't have a flow control mechanism and window mechanism, which may cause packet overflowing. The checksum is the only tool for the error control mechanism in UDP. When the receiver detects an



(Figure 59) Exchange of packets for user data transmission

error through Checksum, the user datagram is removed and congestion control is not provided. This feature of the UDP transport protocol often makes the application layer directly define and implement the mechanisms of the flow control and the error control, if necessary. UDP has no functions for acknowledgement and retransmission, which makes the continuous transmission possible at a minimum transmission rate even though some packets are lost. For this reason, it provides the best-effort-service which is suitable for video playing. In order to play motion images transmitted in real time, the sender is required to include the order of the image frames in the data. Meanwhile, the receiver is required to store the images in different buffers in accordance with the frame order and process them in sequence. When a video is streamed, as opposed to downloading, the transmission might be delayed or only B and P frames are successfully received while I frame was not. In such a case, the application layer needs to come into service in order to destroy data and maintain the real-time service.

The UDP protocol is generally used for the following purposes.

- A process which needs a simple request-response communications and does not need the flow and the error control
- A process which has a mechanism for the flow and the error control in itself
- Multicasting transmission technique
- A management process, such as SNMP
- Path update protocol, such as Routing Information Protocol (RIP)

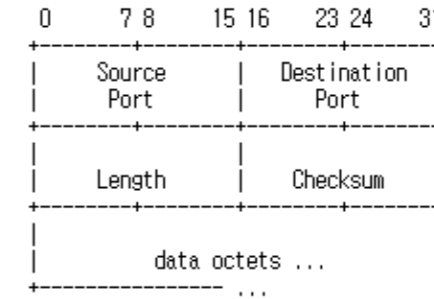
The following table presents well-known ports which are employed by the widely used UDP services. NTP service which sets the host time correctly through the network, BOOTP and DHCP which are used for the network IP management, TFTP which is a simplified version of FTP, RIP and OLSR which are used as a routing protocol, and Kerberos which is a computer network authentication protocol. These are some examples of the well-known ports of UDP.

<Table 45> Well-known UDP ports

Service	UDP port	Service	UDP port
NTP [12,13]	123	Syslog	514
BOOTP Server, DHCP Server [14,15]	67	RIP [18]	520
BOOTP Client, DHCP Client [14,15]	68	RIPng	521
TFTP [16]	69	Timed (Time Server)	525
Kerberos [7]	88	OLSR [19]	698

### UDP Protocol

A UDP packet, which is also called a user datagram, has an 8-byte header (fixed size) which consists of 4 fields (2 bytes or 16 bits for each). Each field of the UDP header is used for the purposes illustrated in <Table 46>.

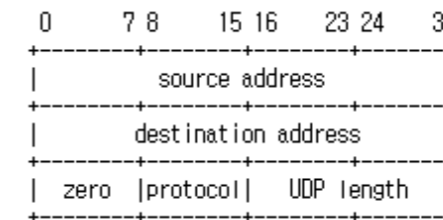


<Figure 61> Format of user datagram header [2]

<Table 46> Fields of user datagram header

Field name	Length	Details
Source Port	2Bytes	Port number of source port packets, ranging from 0 to 65,535
Destination Port	2Bytes	Port number of destination port packets
Length	2Bytes	The total length of UDP header and data field
Checksum	2Bytes	Used to detect data errors
Data octets	Variable	Save data to be transmitted

Source Port indicates the port used by the sending process, while Destination Port means the port number used by the process running on the destination host. The total length is defined by UDP header size (the minimum value of the length is eight.), plus data size. The checksum is the 16-bit one's complement of the one's complement sum of a Pseudo Header of information from the IP header, the UDP header, and the data. If not calculated, the checksum field is filled with 0s. <Figure 62> indicates the pseudo-IP header comprising some IP header fields which are necessary to calculate the checksum of UDP.



<Figure 62> Pseudo-IP header [2]

As mentioned above, the implementation of UDP is simpler compared to that of TCP: it receives data from the application layer and sends the data to the IP layer; receives a UDP user datagram from the IP layer and sends it to an appropriate application program based on the analysis of the datagram.

## Use Case of Multicast Sockets via UDP

As there is growing needs for multimedia data transfer, such as the sending and receiving video traffic on the network, multicasting starts to be widely employed to reduce the overload on the network. UDP should be used for transmission of multicast packets

〈Table 47〉 Source code used on sender's side: sending multicast data [20]

1	int main(int argc, char **argv)
2	{
3	/* Initialization and declaration */
4	...SKIP...
5	/* Making UDP Socket for Multicast */
6	sender_sock = socket(PF_INET, SOCK_DGRAM, 0);
7	/* Setting Multicast IP */
8	memset(&multi_addr, 0, sizeof(multi_addr));
9	multi_addr.sin_family = AF_INET;
10	multi_addr.sin_addr.s_addr = inet_addr("Multicast Group IP");
11	multi_addr.sin_port = htons("Multicast Port");
12	
13	state = setsockopt(send_sock, IPPROTO_IP, IP_MULTICAST_TTL, (void*)&multi_TTL, sizeof(multi_TTL));
14	...SKIP...
15	/* Sending data */
16	sendto(send_sock, buf, strlen(buf), 0, (struct sockaddr*)&multi_addr, sizeof(multi_addr));
17	...SKIP...
18	}

〈Table 47〉 shows a part of source codes which are employed on the sender's side to transfer data over a multicast connection. As seen in 'Row 6', if the socket type is set to 'SOCK\_DGRAM', it works as UDP socket, while if the type is set to 'SOCK\_STREAM', it works as a TCP socket. In 'Row 10', the IP address of the multicast group, to be used on the sender's side, is selected. In 'Row 11', the port to be used for this connection is selected. In 'Row 13', you can select "IP\_MULTICAST\_TTL" in socket option. As the default value is 1 and it should be changed to an appropriate value in order to enable data to be forwarded beyond the router. If the value is too big, data can overflow into the network which you don't intend to use. 'Row 16' shows the source code which is used for actual data transmission.

〈Table 48〉 Source code used on receiver's side: for receiving multicast data [20]

1	int main(int argc, char **argv)
2	{
3	/* Initialization and declaration */
4	...SKIP...
5	/* Making UDP Socket for Multicast */
6	recv_sock=socket(PF_INET, SOCK_DGRAM, 0);
7	memset(&addr, 0, sizeof(addr));
8	addr.sin_family = AF_INET;
9	addr.sin_addr.s_addr = htonl("Receiver IP");
10	multi_addr.sin_port = htons("Multicast Port"); 입력
11	/* Binding */
12	...SKIP...
13	/* Join the Multicast Group */
14	join_addr.imr_multiaddr.s_addr = inet_addr("Multicast Group IP");
15	join_addr.imr_interface.s_addr = htonl("Receiver IP");
16	state = setsockopt(recv_sock, IPPROTO_IP, IP_ADD_MEMBERSHIP, (void*)&join_addr, sizeof(join_addr));
17	...SKIP...
18	/* Receiving data */
19	str_len = recvfrom(recv_sock, buf, BUFSIZE-1, 0, NULL, 0);
20	...SKIP...
21	}

〈Table 48〉 presents a part of source codes which are used by the receiver to receive multicast data. In 'Row 6', like the case for the source code on the sender's side, the socket is configured as UDP socket. In 'Row 14', the IP address of multicast group where the receiver wants to join is selected. In 'Row 16', preparation to receive the data is completed (i.e. the data to be sent to the multicast group selected by "IP\_ADD\_MEMBERSHIP" option).

A socket created to utilize the multicast service should be a UDP socket. Right after joining the multicast group, the receiver can receive data. When testing whether a written program works well, the following points should be considered. The connection between the multicast server and the client is generally established and tested with a hub or L2 switch. In such a case, however, the multicast packets are sent to all the ports of the hub, so it is difficult to know that the delivery of the packets was made through a multicast or unicast. Therefore, the client and the server should be connected in the L3 switch or higher. Without any specific configuration works, an L3 switch should work like an L2 switch. Therefore, VLANs should be separated between the sender and the receiver. IP address ranges should be also assigned to each separately, so that the packets can be routed. The IGMP protocol should be established so that multicast packets can be routed and fully transmitted. Even though the sender and receiver get ready for the transmission of the multicast packets in the real Internet environment, if routers and endpoint hubs are not configured to support multicast service, the multicast packets cannot be transported.

## 04 SCTP(Stream Control Transmission Protocol)

### Features of SCTP

Stream control transmission protocol, or SCTP, is a newly-developed transport layer protocol which combines the strengths of UDP and TCP for multimedia communications [3]. The SCTP protocol provides the services shown in the following table. The SCTP association is a broader concept than the end-to-end TCP connection. In the SCTP association, a single SCTP endpoint is allowed to have multiple IP addresses. This is called **Multi-homing**. It is designed to allow the network-level fault tolerance. To emphasize the use and meaning of the multi-homing, a connection is called an association in the SCTP context. Each association in the SCTP context can support multiple streams, whereas TCP covers a single stream in an end-to-end connection.

- Association startup and takedown
- Sequenced delivery within streams
- User data fragmentation
- Acknowledgement
- Congestion avoidance
- Chunk bundling
- Packet validation
- Path management

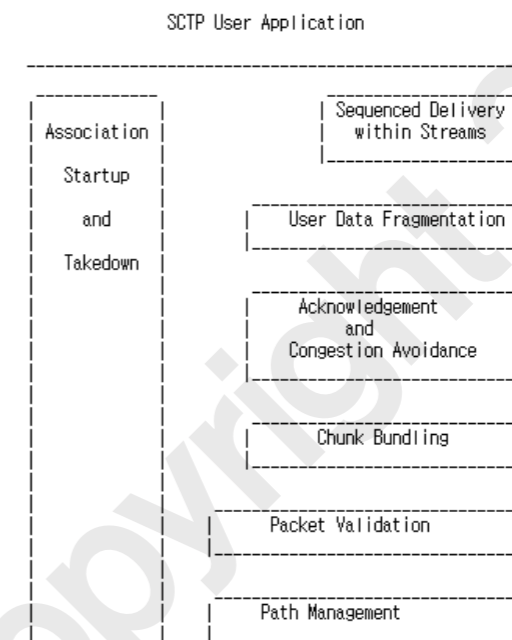
<Table 49> Features of SCTP service

Feature	Details
Process-to-process communications	• Offers a process-to-process communications
Multiple streams	• Delivers a multiple stream service in each connection, which is called an association in the SCTP context
Multi-homing	• The sending/receiving host defines multiple IP addresses for an association in each of the ends. • Among those IP addresses, one address is used as a primary address and others are secondary.
Full-duplex communications	• A full-duplex service in which data can be delivered to both directions at the same time
Connection-oriented	• A connection-oriented protocol. In the SCTP context, a connection is called an association.
Reliability	• A reliable transport protocol which uses an acknowledgement to check whether data is successfully delivered.

TSN (Transmission Sequence Number), SI (Stream Identifier) and **SSN (Stream Sequence Number)** are the numbers (identifiers) used for SCTP. While, a byte is the unit of data in TCP, **Data Chunk** is the unit of data in SCTP. Role of **Transmission Sequence Number (TSN)** in SCTP is similar to that of the sequence number in TCP. In SCTP, there may be several streams in each association. Each stream needs to be identified by using a 16-bit stream identifier (SI) which starts from '0'. SSN (Stream Sequence Number) is used to define each data chunk within the same stream.

### SCTP Protocol

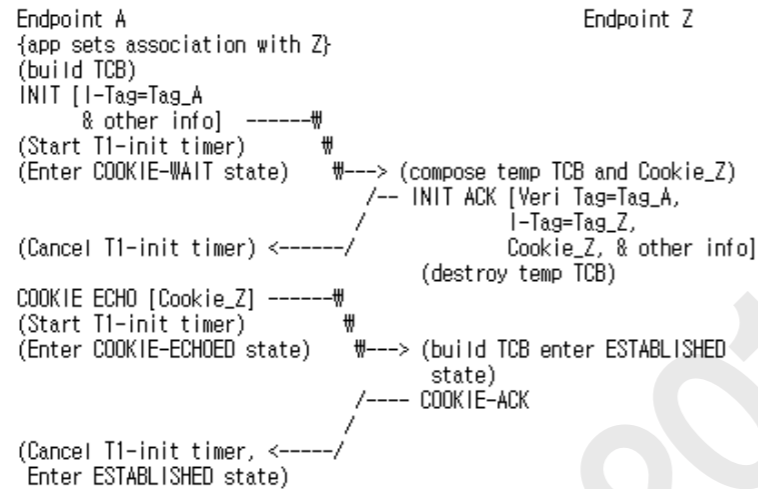
This chapter will briefly explain about the functions of the SCTP services, shown in the following figure, instead of the detailed explanation of SCTP. There are eight SCTP functions as shown below.



<Figure 63> View on functions of SCTP transmission service [3]

① Association startup and takedown

An association can be initiated upon the SCTP user's request. The four-way handshake technique, shown in <Figure 64>, is employed in order to prevent the synchronization attack (SYN attack). Endpoint A sends an INIT chunk to the Endpoint Z to initiate an association. After receiving the INIT chunk, the Endpoint Z replies with an INIT-ACK chunk which contains the cookie information. On receipt of this INIT-ACK chunk, the Endpoint A sends a COOKIE-ECHO chunk, which echoes the cookie, to acknowledge the receipt of the INIT-ACK chunk. The Endpoint Z then sends a COOKIE-ACK chunk, which is the final step of the four-way handshake process. In the SCTP association, TCB (transmission control block), a key system resources, can be allocated only when the Endpoint Z receives the authentic COOKIE-ECHO chunk, which helps eliminate the risk of DoS attack that was caused by the SYN attack when TCB was generated upon receiving the SYN segment in the TCP connection.



(Figure 64) General procedure for association setup [3]

② Sequenced delivery step within the stream

A stream in the SCTP context refers to the sequence of user messages. At the time of association setup, a user can specify the number of streams that can be supported. Each stream is identified by a stream identifier (SI). In addition to SI, user messages within the stream will be assigned with the stream sequence number (SSN) to ensure that data chunks are delivered in an ordered fashion. Even when one stream is blocked to deliver the next user message, other streams can still offer transmission services.

③ User data fragmentation

Like in the TCP context, SCTP's user message has a certain limitation in MTU (path Maximum Transmission Unit), hence, SCTP is providing the Fragmentation feature to overcome this issue. The fragments delivered individually are reassembled into one user message on the SCTP layer.

④ Acknowledgement and congestion avoidance

Regardless of streams, a transmission sequence number (TSN) is assigned to each data chunk when SCTP is used. In addition to sequenced delivery, a reliable data delivery can be guaranteed as the receiver sends an ACK to every TSN it received. The ACK and congestion control are relevant to triggering retransmission of the packet in case when an ACK is not received within the given time. In other words, packet retransmission is carried out in accordance with the procedures for the congestion avoidance, which is similar to the congestion control of the TCP context.

⑤ Chunk bundling

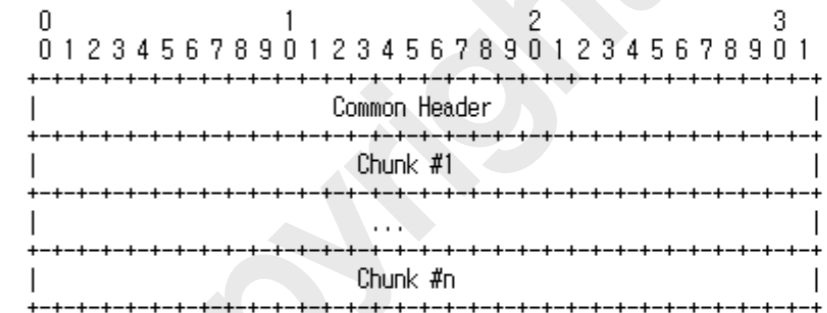
An SCTP packet is composed of a common header, followed by several user data and chunks which contain the SCTP control information. An SCTP user can request the assembly of multiple user data into one SCTP packet. This is called a chunk bundling, which is responsible for assembling multiple chunks into one SCTP packet and subsequently disassembling the packet into multiple chunks.

⑥ Packet validation

Each SCTP common header has a Verification Tag and a 32-bit checksum field. A verification tag, VT, means the value which is selected by each endpoint at the time of association startup. All packets are sent with the same verification tag during the lifetime of the association. If, during the lifetime of the association, a packet is received with an unexpected verification tag, the packet is discarded. In addition, the CRC-32 checksum can be set for the SCTP packet transmission in order to provide better protection against the data corruption.

⑦ Path management

SCTP offers the Path Management mechanism in which the sender, an SCTP user, can manipulate the Transport Address (SCTP port number and IP address) used at the destination point of the SCTP packet. At the time of the association establishment, the Primary Path is defined for each SCTP endpoint and is used for the transmission of the SCTP packets. In case the primary path goes down, another available transport address or one of the transport addresses defined by Multi-Homing, can be selected and used. The Path management and the Packet Verification (a mechanism achieved by using verification tags and checksum) are performed at the same time. SCTP packets contain blocks which are composed of a common header and chunks. Chunks are divided into: control chunks; and data chunks. The control chunk is delivered prior to the data chunk.



(Figure 65) Format of SCTP packet [3]

A general header defines the endpoints of each association to which the packet belongs, guarantees that the packet belongs to a particular association, and preserves the integrity of the contents of the packet including header itself.

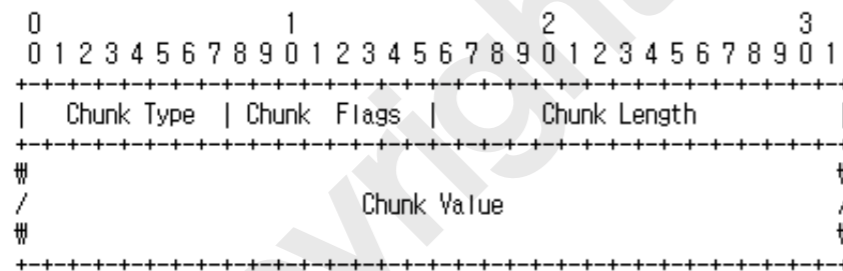


(Figure 66) Format of SCTP common header [3]

<Table 50> Components of SCTP common header

Classification	Details
Source port number	• Identical to that of TCP and UDP
Destination port number	• Identical to that of TCP and UDP
Verification Tag	• Used as an association identifier and repeated for all packets during the time of the association
Checksum	• The size of checksum is 32 bits to allow the use of the CRC-32 checksum (The checksum in UDP, TCP and IP is 16 bits)

Control information and user data are carried in chunks. The first three fields – type, flag and length – are common to all chunks. The information field refers to the type of chunk and the type field is limited to 256 bytes of the chunk. Only some fields, however, are defined and the remaining fields have yet to be defined for the future. The flag field defines a specific flag which may require a specific chunk.



<Figure 67> Chunk format [3]

Chunks are classified into the data chunk and the control chunk, depending on its type. The table shown below presents some of the major chunk types.

<Table 51> Types of chunk

ID value	Chunk type	ID value	Chunk type
0	DATA	6	ABORT
1	INIT	7	SHUTDOWN
2	INIT ACK	8	SHUTDOWN ACK
3	SACK	9	ERROR
4	HEARTBEAT	10	COOKIE ECHO
5	HEARTBEAT ACK	11	COOKIE ACK

Example Question

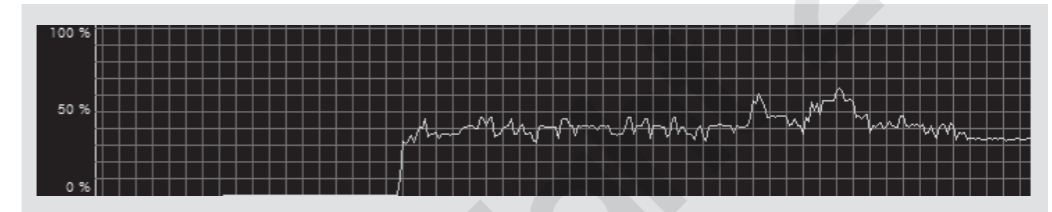
Question type

Short-answer question

Question

The figure below shows the overload on the network when downloading a file by using TCP. A sawtooth wave is generated during the data transfer because of the feature of TCP, even though its network capacity is enough. In TCP, congestion control is achieved by using the (①) algorithm, which slowly increases the bandwidth of the network used for the data transfer from its initial value, not using all available bandwidths at once. If it reaches the limit, the congestion control is achieved by using the (②) algorithm, which additively increases the congestion window.

Please fill in the blanks.



Intent of the question

To evaluate whether a learner understands the key features of TCP and the mechanisms used for the congestion control

Answer and explanation

- ① Slow Start
- ② Congestion Avoidance

Related E-learning Contents

- **Lecture 5** Understanding of Transport Layer and UDP Protocol
- **Lecture 6** TCP and SCTP Protocols
- **[Advanced]** IGP and EGP



## Application Layer Technologies, Including Web Applications

### ▶▶▶ Latest Trends and Key Issues

Since the mid-2000s, with the rise of mobile platforms, such as Android and iPhone, the society has entered into the Mobile Age. In this society, many application programs, useful for business and leisure activities, have been launched. Among them, some programs use the standard application layer protocols, such as SMTP, POP3, IMAP, and FTP. On the other hand, SNS apps, including KakaoTalk, LINE, and Facebook, are run on the self-defined application layer protocol. In order to provide a convenient and great application programs to users, a developer is required to understand the concept of the application layer protocol which lies on the transport layer and how it works.

### ▶▶▶ Study Objectives

- \* To be able to understand the meaning of the application layer protocol, and design and utilize the same
- \* To be able to understand and utilize the HTTP protocol which is used for the data transfer between the client and the web server
- \* To be able to understand the FTP protocol which is used for the file exchange with the server, and utilize the same to develop programs. To be able to enhance the productivity in the development of application programs by using open source libraries
- \* To be able to understand a JSP-based server programming technology and the client programming technology, both used for the development of web application programs, and utilize them in developing software

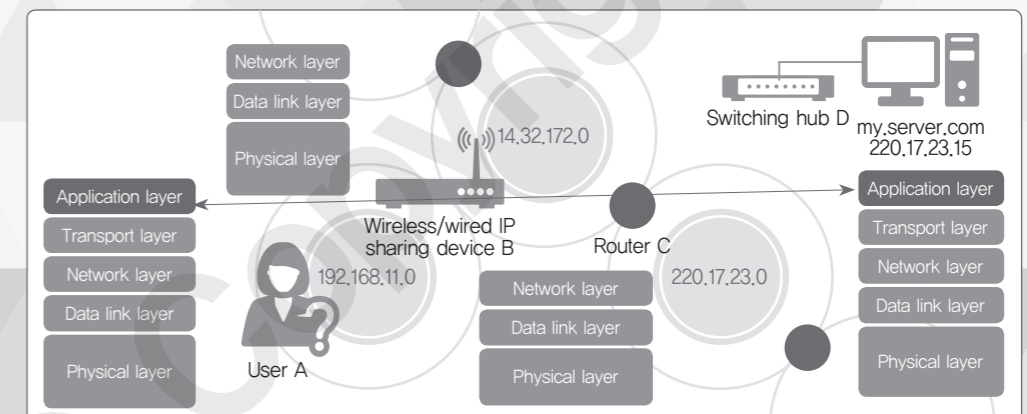
### ▶▶▶ Practical Importance High

### ▶▶▶ Keywords

Application Layer Protocol, HTTP, GET Method, POST Method, FTP, Control Connection, Data Connection, PORT mode, PASV Mode, Apache 2.0 License, Web Application Program, Server Programming Technology, Client Programming Technology, JSP, HTML, JavaScript, AJAX, Scriptlet Expression, Scriptlet, Directives, Standard Action, Expression Language, Tag Extension

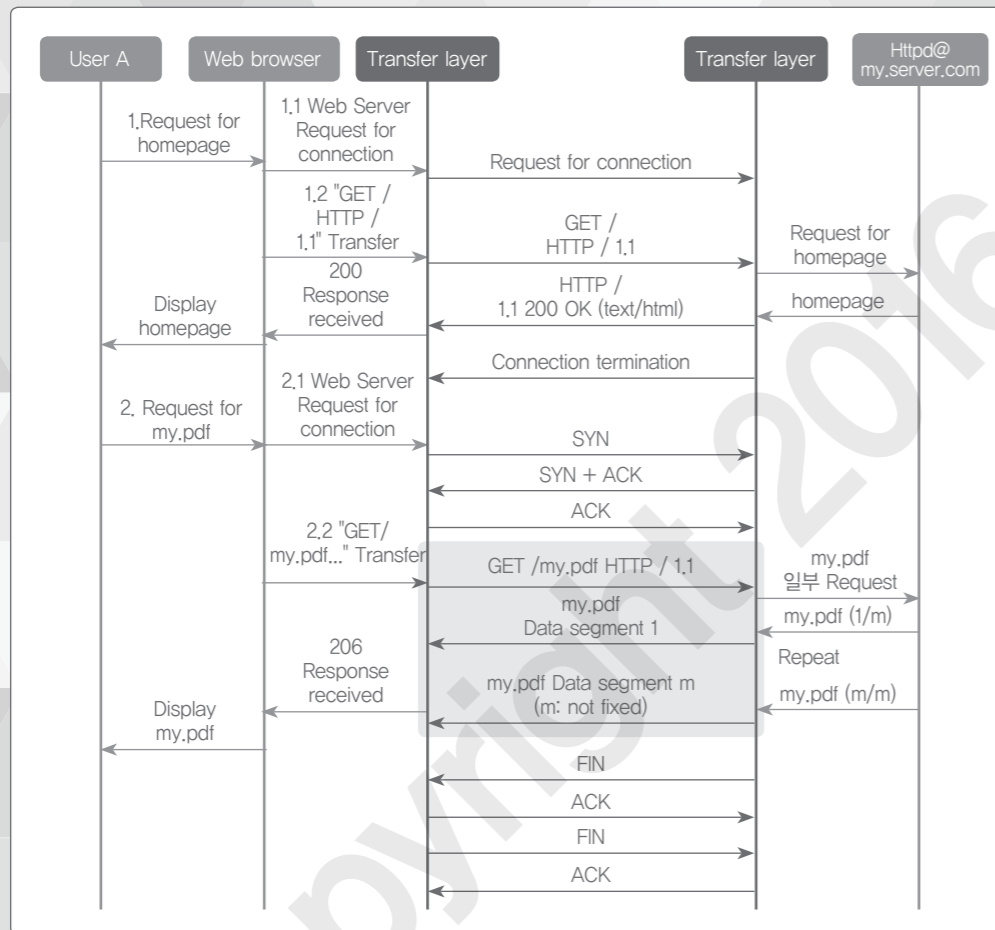
### + Practical tips: Operation of the application layer protocol

〈Figure 68〉 displays how the application program operates on the Internet. Along with the transport layer, the application layer provides an end-to-end communications service from one application program to another. It means that, just like in the operation of the transport layer protocol, such as TCP, the intermediate nodes, including wired/wireless IP sharing devices or routers, are not involved in the operation of the application layer protocol. The data can be generated and transferred in a way that users want; however, widely-used application programs usually employ globally-recognized standard application layer protocols to minimize incompatibility issues.



〈Figure 69〉 Scenario on application layer operation

〈Figure 69〉 illustrates how the application layer protocol and transport layer protocol interacts with one another under the scenario of User A's web usage described in the Chapter 1: Introduction to Network. It shows what happens when User A downloads a file (my.pdf) on the homepage of the web server (my.server.com). This chapter will explain how HTTP, an application layer protocol, works. In addition, how HTTP interacts with the transport layer protocol will be explained to help you understand better about the protocol.



(Figure 69) Scenario on application layer operation

This chapter explains how the HTTP protocol works in [Step 1] and [Step 3–4] from the scenario ① on User A's web usage, which was described in the Chapter 1. Introduction to Network. The web browser, an application program, is connected with httpd, a web server program, through the TCP connection. Then, it brings the web contents requested by User A and displays the contents in the readable form. In this process, the web browser utilizes the HTTP/1.1 protocol for the communications with the web server. This application layer protocol, the HTTP/1.1 protocol, is text-based [8]. The text data that is generated in accordance with the HTTP/1.1 protocol is delivered to the application layer program on the opposite side over the transport layer protocol.

Transmission and receipt of the user data between application layer programs over the transport layer is achieved by using the **Socket Programming**. The socket programming is a programming mechanism required to implement the application layer protocols. Detailed explanation, however, will not be provided as it goes beyond the scope of this chapter. Please refer to other books for further details.

## 01 What is Application Layer Protocol?

You might encounter application layer technologies many times when you operate the network-related systems or develop network application programs. An **Application Layer Protocol**, which is applied to the application layer technologies, serves as a protocol used by application programs to send/receive required information. There are several application layer protocols which are widely used: FTP[1] (used for file transfer), TELNET[2] (used for terminal connection), SMTP[3], POP3[4], and IMAP[5] (all of these three are used for e-mail access and transmission), The DNS[6] (used for host name and IP address mapping), and SNMP[7] (used for network management). HTTP[8] is also one example of the application layer protocol used for the data transfer between the web client and the web server. Another example of the application layer protocol is BitTorrent that is widely recognized among the P2P protocols (a method for terminal-to-terminal communications, not for conventional communications with the network infrastructure).

## 02 HTTP

### Characteristics of HTTP

HTTP (Hypertext Transfer Protocol) is a protocol for the application layer for hypermedia information systems that can link texts with video clips or voice files. HTTP has been used to transport contents of the World Wide Web that was started to be used in the early 1990s. The first version, HTTP/0.9, was used just for low level data transmission through the Internet. However, HTTP/1.0 (RFC 1945) and above has been using messages defined for the MIME. HTTP/1.1 is the currently used version, which is a little bit more reliable than HTTP/1.0.

Just like other application-level protocols, HTTP is built based on the "Request/response protocol" model in the client-server environment. Microsoft's Internet Explorer, Google's Chrome browser, Mozilla's Firefox web browser and the web servers such as Apache HTTP Server are utilized to make the HTTP protocol working, so the end users can use the web without any difficulty even they do not know how the HTTP protocol is implemented. However, if you, as a developer, can refer to the text-based application layer protocols such as HTTP in the process of building an Internet application program or new application-level protocols, it will help you with the design and implementation.

### HTTP Protocol

HTTP works based on the request/response model. The client sends the **Request Message** to the server, including the request method, URI, protocol version, and MIME information message. The server sends back the response to the client, including the **Status Line**: protocol version, success or fail code, MIME server information, and the like. HTTP/1.1 (the up-to-date version) is widely used in this process.

```

Frame 4: 983 bytes on wire (7864 bits), 983 bytes captured (7864 bits) on interface 0
Ethernet II, Src: IntelCor_2e:1b:96 (00:16:ea:2e:1b:96), Dst: Buffalo_b5:57:a4 (00:07:40:b5:57:a4)
Internet Protocol Version 4, Src: 192.168.11.5 (192.168.11.5), Dst: 220.73.231.214 (220.73.231.214)
Transmission Control Protocol, Src Port: 55425 (55425), Dst Port: http-alt (8080), Seq: 1, Ack: 1, Len: 929
Hypertext Transfer Protocol
GET /SimpleBBS2HttpswebAccess/download.jsp?file_id=112&file_name=%283%ED%8C%90%29_JAVA_%EC%8B%9C%ED%81%90%EC%96%B4_%EC%BD%94%EB%94%A9_%EA%B0%80%EA%B3%B5%EC%A7%80%EC%82%AC%ED%95%AD; user_agent=chrome/35.0.1916.153 safari/537.36\r\n
Host: seps2.deu.ac.kr:8080\r\n
Connection: keep-alive\r\n
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8\r\n
User-Agent: Mozilla/5.0 (Windows NT 6.2) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/35.0.1916.153 Safari/537.36\r\n
DNT: 1\r\n
Referer: http://seps2.deu.ac.kr:8080/SimpleBBS2HttpswebAccess/bbs_show_message.jsp?id=473\r\n
Accept-Encoding: gzip,deflate,sdch\r\n
Accept-Language: ko-KR,ko;q=0.8,en-US;q=0.6,en;q=0.4\r\n
[truncated] Cookie: JSESSIONID=IC157AA6F52A78AC9F1F53606D7006EB; title=%EA%B3%B5%EC%A7%80%EC%82%AC%ED%95%AD; user_agent=chrome/35.0.1916.153 Safari/537.36\r\n
[Full request URI: http://seps2.deu.ac.kr:8080/SimpleBBS2HttpswebAccess/download.jsp?file_id=112&file_name=%283%ED%8C%90%29_JAVA_%EC%8B%9C%ED%81%90%EC%96%B4_%EC%BD%94%EB%94%A9_%EA%B0%80%EA%B3%B5%EC%A7%80%EC%82%AC%ED%95%AD; user_agent=chrome/35.0.1916.153 Safari/537.36\r\n
[HTTP request 1/1]
[Response in frame: 7]
    
```

<Figure 70> Example of the HTTP request message

<Figure 70> shows an example of the HTTP request message. The request message is clearly defined in RFC 2616[8] as shown below. Some of them, that may need further explanations, are listed in <Table 52>. To find out more, you can refer to RFC 2616. If you take another look at <Figure 70> which shows an example of an HTTP request after understanding the request message and definition, it will be helpful for you to understand better.

<Table 52> HTTP request message and definition

Request	= Request-Line *(( general-header   request-header   entity-header ) CRLF) CRLF [message-body]
Request-Line	= Method SP Request-URI SP HTTP-Version CRLF
Method	= "OPTIONS"   "GET"   "HEAD"   "POST"   "PUT"   "DELETE"   "TRACE"   "CONNECT"   extension-method
Request-URI	= "*"   absoluteURI   abs_path   authority
HTTP-Version	= "HTTP/" 1*DIGIT "." 1*DIGIT
SP	= <US-ASCII SP, space (32)>
CRLF	= CR LF
CR	= <US-ASCII CR, carriage return (13)>
LF	= <US-ASCII LF, linefeed (10)>

The method selected in the Request-Line of the HTTP request message is used to define how HTTP protocol will work. RFC 2616 contains definitions for eight methods, but expansion is also possible. Among them, GET and POST are widely used. **GET Method** is usually used when searching for the contents in the server when having access to a homepage. **POST Method** is usually used to deliver the information, a user typed-in, to the web server (just like a form).

In the HTTP request message, the method selected in the Request-Line is important, and there is another important information called a request-header shown in <Table 53>. The detailed definition about the request - header is well defined in RFC 2616, so we will skip the definition in this text book.

<Table 53> Request-header definition for HTTP request message

Request-header	= Accept   Accept-Charset   Accept-Encoding   Accept-Language   Authorization   Expect   From   Host   If-Match   If-Modified-Since   If-None-Match   If-Range   Max-Forwards   Range   Referer
----------------	---

```

Frame 7: 59 bytes on wire (472 bits), 59 bytes captured (472 bits) on interface 0
Ethernet II, Src: Buffalo_b5:57:a4 (00:07:40:b5:57:a4), Dst: IntelCor_2e:1b:96 (00:16:ea:2e:1b:96)
Internet Protocol Version 4, Src: 220.73.231.214 (220.73.231.214), Dst: 192.168.11.5 (192.168.11.5)
Transmission Control Protocol, Src Port: http-alt (8080), Dst Port: 55425 (55425), Seq: 348, Ack: 930, Len: 5
[2 Reassembled TCP Segments (352 bytes): #6(347), #7(5)]
Hypertext Transfer Protocol
HTTP/1.1 200 OK\r\n
Server: Apache-Coyote/1.1\r\n
Content-Transfer-Encoding: binary\r\n
Content-Disposition: attachment;filename="%283%ED%8C%90%29_JAVA_%EC%8B%9C%ED%81%90%EC%96%B4_%EC%BD%94%EB%94%A9_%EA%B0%80%EA%B3%B5%EC%A7%80%EC%82%AC%ED%95%AD; user_agent=chrome/35.0.1916.153 safari/537.36\r\n
Transfer-Encoding: chunked\r\n
Date: Sat, 05 Jul 2014 22:02:13 GMT\r\n
\r\n
[HTTP response 1/1]
[Time since request: 0.108459000 seconds]
[Request in frame: 4]
[HTTP chunked response]
    
```

<Figure 71> Example of HTTP response message

<Figure 71> is an example of an HTTP response message. Just like in the case of the request message, only some of them, which may need further explanation, are included in this textbook. You can refer to RFC 2616 for more details. When you take another look into the example after studying the message and the definition, it will be easier to understand <Figure 71>. The Status-Code information of Status-Line tells us how the HTTP request message was processed. When the code starts with "2", just like 200 (OK) or 206 (Partial Content), it means the processing was successful. When the code starts with "4" or "5", it indicates "client error" or "server error" respectively.

<Table 54> HTTP response message definition

Response	= Status-Line *(( general-header   response-header   entity-header ) CRLF) CRLF [ message-body ]
Status-Line	= HTTP-Version SP Status-Code SP Reason-Phrase CRLF
Status-Code	= "100" ; Section 10.1.1: Continue "101" ; Section 10.1.2: Switching Protocols "200" ; Section 10.2.1: OK "201" ; Section 10.2.2: Created ... SKIP ... "206" ; Section 10.2.7: Partial Content "300" ; Section 10.3.1: Multiple Choices ... SKIP ... "403" ; Section 10.4.4: Forbidden "404" ; Section 10.4.5: Not Found "405" ; Section 10.4.6: Method Not Allowed "406" ; Section 10.4.7: Not Acceptable

In the HTTP response message, the Status-Code selected in the Status-Line is important, and there is another important information called a response-header shown in (Table 55). The detailed definition about response-header can be found in RFC 2616.

(Table 55) HTTP response message's response-header definition

response-header = Accept-Range   Age   ETag   Location   Retry-After   Server   Vary   WWW-Authenticate   Proxt-Authenticate
---

(Table 56) shows more details, especially focusing on the HTTP header, about what is sent and received in a scenario of the application layer which was shown in (Figure 69). 'C' and 'S' means a client and a server respectively, which means the client and server correspond to the web browser User A is using and the web server (my.server.com) respectively. The file size of my.pdf<sup>3</sup> was too big in the previous transmission, and the transmission was initialized. Hence, the client side already knows about the file size and started to make the preparation to bring the file again.

(Table 56) Scenario of file transfer (HTTP example)

```

C: GET /my.pdf HTTP/1.1\r\n
Host: my.server.com:8080\r\n
Connection: keep-alive\r\n
Accept: */*\r\n
Accept-Encoding: gzip, deflate, sdch\r\n
Accept-Language: ko-KR, ko;q=0.8, en-US;q=0.6, en;q=0.4\r\n
Range: bytes=0-32767\r\n
S: HTTP/1.1 206 Partial Content\r\n
ETag: W/"5763831-1385516368000"\r\n
Content-Range: bytes 0-32767/5763831\r\n
Content-Type: application/pdf\r\n
Content-Length: 32768\r\n
Date: Wed, 16 Jul 2014 07:52:18 GMT\r\n
pdf file binary data
C: GET /my.pdf HTTP/1.1\r\n
Range: bytes=32768-337615\r\n
If-Range: W/"5763831-1385516368000"\r\n
S: HTTP/1.1 206 Partial Content\r\n
ETag: W/"5763831-1385516368000"\r\n
Content-Range: bytes 32768-337615/5763831\r\n
Content-Type: application/pdf\r\n
Content-Length: 304848\r\n
Date: Wed, 16 Jul 2014 07:52:18 GMT\r\n
pdf file binary data
...
C: GET /my.pdf HTTP/1.1\r\n
Range: bytes=5187280-5731062\r\n
If-Range: W/"5763831-1385516368000"\r\n
S: HTTP/1.1 206 Partial Content\r\n
ETag: W/"5763831-1385516368000"\r\n
Content-Range: bytes 5187280-5731062/5763831\r\n
Content-Type: application/pdf\r\n
Content-Length: 543783\r\n
Date: Wed, 16 Jul 2014 07:52:27 GMT\r\n
pdf file binary data
    
```

<sup>3</sup> When downloading an actual PDF through Wireshark network protocol analyzer, another server was used (not my.server.com) and the pdf file name was not my.pdf. We used my.pdf as an example to make the explanation easier

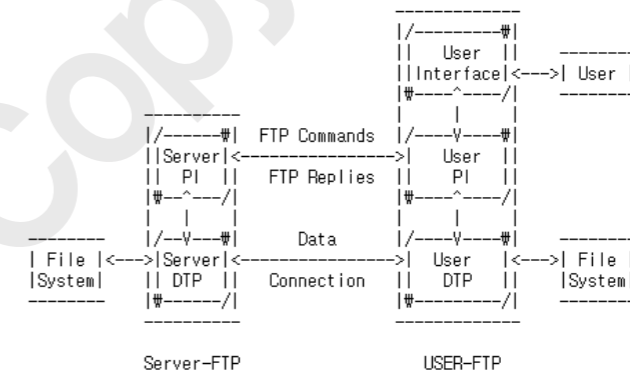
When a part of the file is selected from the client side by using the Range Header Parameter in the GET method, the server sends back the corresponding file. When you look at the **Content-Range** header parameter and the **Content-Length** parameter, you can see whether the transmission was conducted as expected or not. As such, the client can make rounds of request to bring partial files first. Later, the web browser makes the partial files into a complete file and sends the file in an appropriate format.

The server sends "HTTP/1.1 206 Partial Content\r\n ... SKIP ... pdf file binary data" to the client. However, the total size is bigger than the TCP MSS (Maximum Segment Size, default value: 1460). Therefore, the data is sent from the server to the client in multiple TCP segments. This can be found in the TCP segments transferred over the transport layer which links the web browser and the web server, as shown in (Figure 69): the scenario of the application layer. It is easy to understand this way: the transport layer receives multiple TCP segments, collects them to make the meaningful user data, and sends them to the application layer.

### 03 File Transmission Protocol

#### Characteristics of FTP

FTP is a protocol to transmit a file or a part of a file from one system to another system; it has two connections as opposed to other application layer protocols. The connections are the **Control Connection** and the **Data Connection**, using **Port 21** and **20** respectively. The control connection is sustained up until a client is terminated or an FTP session is finished. The data connection establishes a TCP connection whenever a file starts to be transmitted. Another port, other than port 20, can be used depending on how the port is used within the connection. (Figure 72) shows an FTP model which is defined by the FTP standard [1]. PI is an abbreviation for Protocol Interpreter and DTP stands for Data Transfer Process.



(Figure 72) FTP model [1]

FTP connection can be divided into two modes: active transfer and passive transfer (PASV). In the case of the **Active Transfer**, a server makes a connection request to a client; however, there is an issue when the connection is directed to an external server system through a firewall and the like. In the case of the **PASV**, on the other hand, a client asks for a connection to a server first using another channel (NOT the default data channel). As the client

is asking for an FTP connection, it is possible to pass through a firewall. This method can be applicable even when you cannot see the client IP from outside because the client network is using a proxy server.

<Table 57> Comparison between active transfer and passive transfer

Classification	Active transfer	Passive transfer
Summary	A server is connected to a certain port of a client to send the data	A client is connected to a certain port of a server to send the data
Purpose	Generally used FTP	FTP for a client which has security features
Port number	Control port: 21 Data port: 20	Control port: 21 Data port: higher than 1024

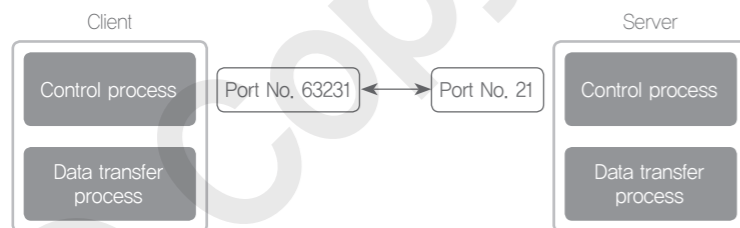
### FTP Protocol

The FTP control connection works as follows.

- ① An FTP server opens port 21 and waits for a request from a client (passive port open)

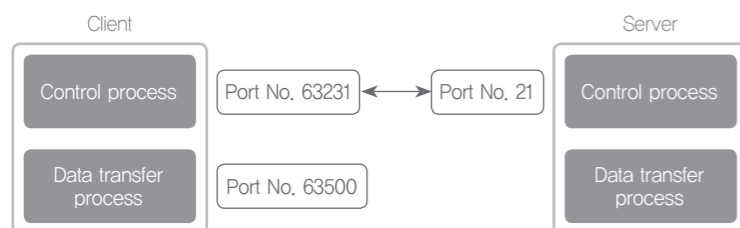


- ② The client is selecting a random port and sends out an FTP request (active port open)

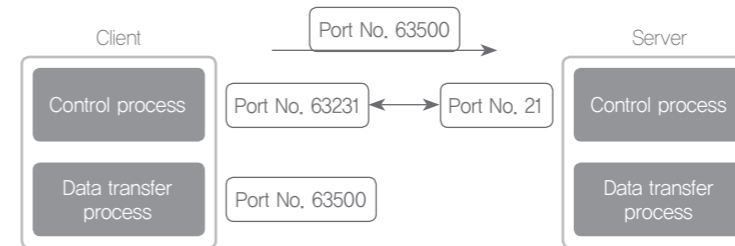


The FTP data connection works as follows.

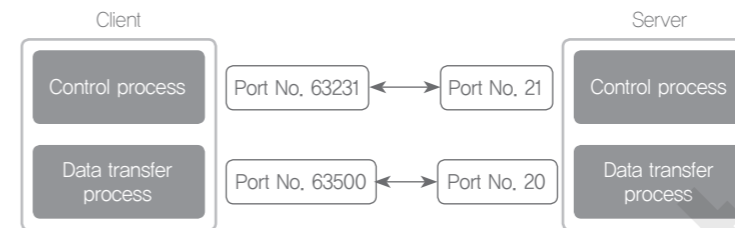
- ③ A client is opening a random port and activates the passive open status.



- ④ The client uses PORT command to deliver the open port number to the server.



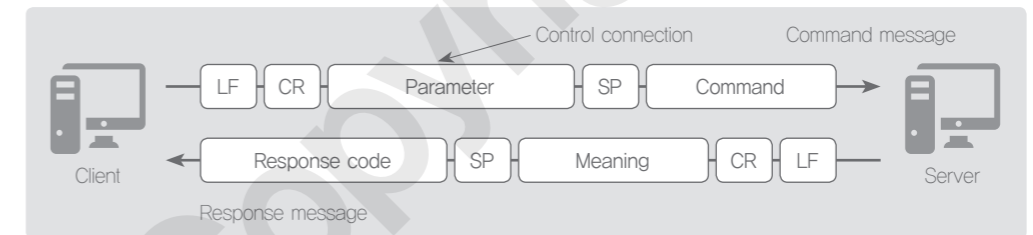
- ⑤ The server receives a random port number sent by the client, and turns port 20 into the passive port open status.



The procedure for the FTP command processing and the list of available commands are as follows.

#### FTP command processing

- A control connection is established between a server control process and a client control process in order to initiate communications.

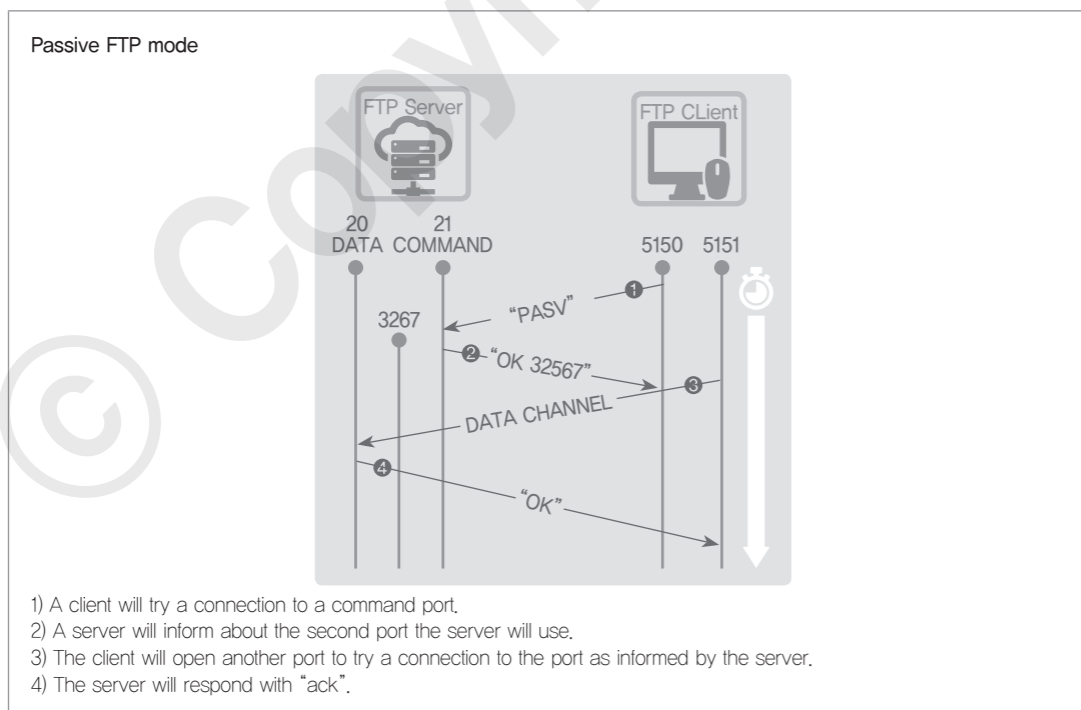
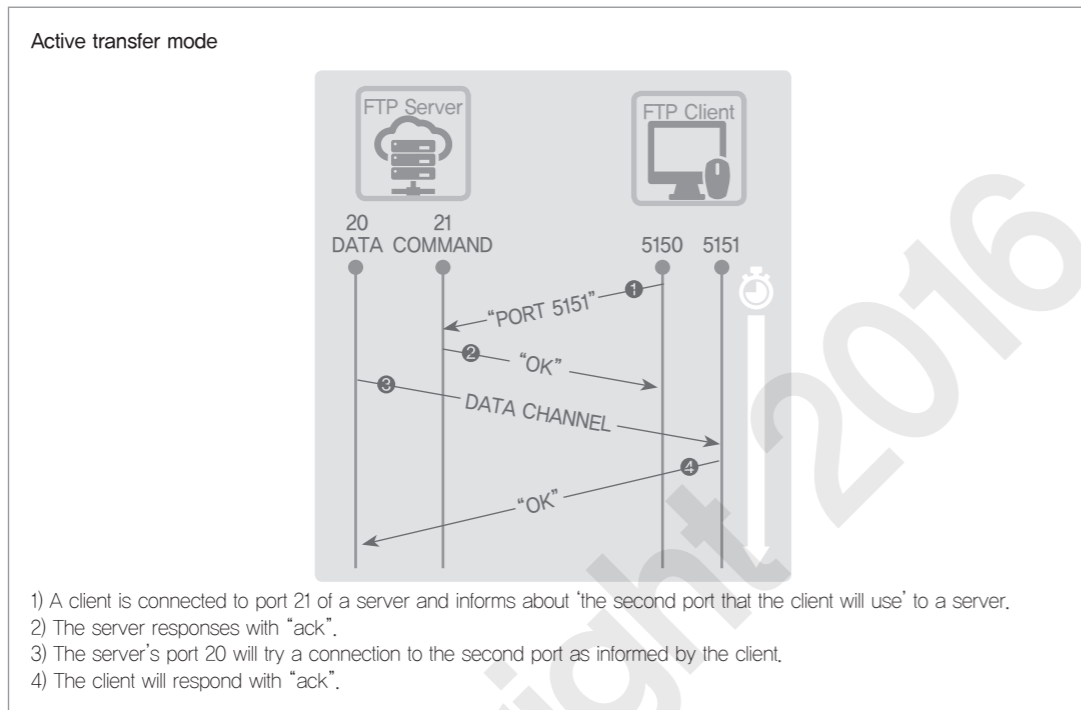


- It is a conversational processing: a user gives a command to a server and the server responds to the command.

#### Types of commands included in the FTP message

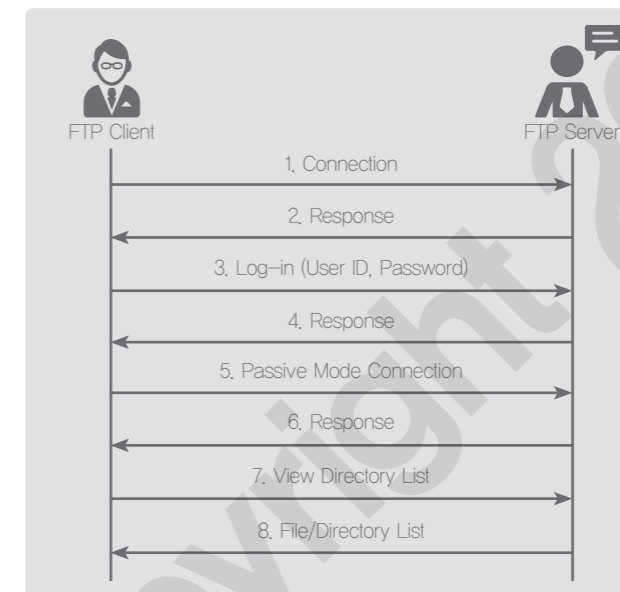
Types of command	Actual command
Access commands	USER, PASS ACCT, REIN, QUIT, ABOR
File management commands	CWD, CDUP, DELE, LIST, NLIST, MKD, PWD, RMD, RNFR, RNTD, SMNT
Data format commands	TYPE, STRU, MODE
Data port selection commands	PORT, PASV
File transfer commands	GET, PUT, MGET, MPUT, RETR, STOR, APPE, STOU, ALLO, REST, STAT
Other commands	HELP, NOOP, SITE, SYST

The active and passive transfer modes work in time sequence as shown below.



**Example of Actual FTP Protocol**

(Figure 73) shows a scenario to view file/sub-directory list located in a certain directory of a user account of an FTP server. The first example will be using the "C" language based on RFC 959[1], which defines the FTP protocol. The reason for taking this example is because you may work in a special environment like the embedded environment or work on programming based on a newly defined protocol. It is also recommended to utilize a pre-built library, if available, because it can enhance productivity. To this end, the next chapter will cover how to utilize the **Apache Commons Net™ Library** in order to meet the goal described in the scenario.



(Figure 73) Scenario to display directory list

(Table 58) ftp\_pasv\_list.c

```

1  #include <stdio.h>
2  #include <sys/socket.h>
3  #include <arpa/inet.h>
4  #include <stdlib.h>
5  #include <string.h>
6  #include <unistd.h>
7
8  #define RCVBUFFSIZE 2048
9  #define FTP_SERVER "192.168.11.20"
10 #define USER      "tester"
11 #define PASSWD     "tester"
12 /* Control connection port */
13 #define FTP_CPORT 21
    
```

```

14
15 const char *CLIENT = "client";
16
17
18 void DieWithError(char *errorMessage)
19 {
20     perror(errorMessage);
21     exit(1);
22 }
23
24 int get_data_port(char *msg)
25 {
26     int p1 = 0;
27     int p2 = 0;
28     int cnt = 0;
29     char *ptr = msg;
30     char buf[10];
31     int j;
32
33     memset(buf, 0, 10);
34     while (*ptr) {
35         if (*ptr == ',') {
36             cnt++;
37         }
38         if (cnt == 4) {
39             ptr++;
40             j = 0;
41             while ((buf[j++] = *ptr++) != ',');
42             buf[j - 1] = '\0'; // erase ','
43             p1 = atoi(buf);
44             printf("p1 = %d\n", p1);
45
46             memset(buf, 0, 10);
47             j = 0;
48             while ((buf[j++] = *ptr++) != '\0');
49             buf[j - 1] = '\0'; // erase '\0'
50             p2 = atoi(buf);
51             printf("p2 = %d\n", p2);
52             return (p1 * 256 + p2);
53         }
54         ptr++;
55     }
56
57     printf("p1 = %d, p2 = %d\n", p1, p2);

```

```

58
59     return -1; // error if reached here
60 }
61
62
63 int main(int argc, char *argv[])
64 {
65     int sock;
66     struct sockaddr_in ftpServerAddr;
67     unsigned short ftpServerPort;
68     char servIP[16];
69     char strBuffer[RCVBUFSIZE];
70     int bytesRcvd, totalBytesRcvd;
71     int data_port;
72
73     pid_t pid;
74
75     memset(servIP, 0, 16);
76     strcpy(servIP, FTP_SERVER);
77     ftpServerPort = FTP_CPORT;
78
79     if ((sock = socket(AF_INET, SOCK_STREAM, 0)) < 0)
80         DieWithError("socket() failed\n");
81
82
83     memset(&ftpServerAddr, 0, sizeof(ftpServerAddr));
84     ftpServerAddr.sin_family = AF_INET;
85     ftpServerAddr.sin_addr.s_addr = inet_addr(servIP);
86     ftpServerAddr.sin_port = htons(ftpServerPort);
87
88     if (connect(sock, (struct sockaddr *) &ftpServerAddr,
89               sizeof(ftpServerAddr)) < 0)
90         DieWithError("connect() failed\n");
91
92
93     totalBytesRcvd = 0;
94     memset(strBuffer, 0, RCVBUFSIZE);
95     bytesRcvd = recv(sock, strBuffer, RCVBUFSIZE - 1, 0);
96     printf("< %s : %s", FTP_SERVER, strBuffer);
97
98     /* USER */
99     memset(strBuffer, 0, RCVBUFSIZE);
100    sprintf(strBuffer, "USER %s\n", USER); // LJM: should be modified

```

```

101 printf("> %s : %s", CLIENT, strBuffer);
102 send(sock, strBuffer, strlen(strBuffer), 0);
103 totalBytesRcvd = 0;
104 memset(strBuffer, 0, RCVBUFSIZE);
105 bytesRcvd = recv(sock, strBuffer, RCVBUFSIZE - 1, 0);
106 printf("< %s : %s", FTP_SERVER, strBuffer);
107
108
109 /* PASS */
110 memset(strBuffer, 0, RCVBUFSIZE);
111 sprintf(strBuffer, "PASS %sWrWn", PASSWD); // LJM: should be modified
112 printf("> %s : %s", CLIENT, strBuffer);
113 send(sock, strBuffer, strlen(strBuffer), 0);
114
115 totalBytesRcvd = 0;
116 memset(strBuffer, 0, RCVBUFSIZE);
117 bytesRcvd = recv(sock, strBuffer, RCVBUFSIZE - 1, 0);
118 printf("< %s : %s", FTP_SERVER, strBuffer);
119
120
121 /* PASV */
122 memset(strBuffer, 0, RCVBUFSIZE);
123 strcpy(strBuffer, "PASVWrWn");
124 printf("> %s : %s", CLIENT, strBuffer);
125 send(sock, strBuffer, strlen(strBuffer), 0);
126
127 totalBytesRcvd = 0;
128 memset(strBuffer, 0, RCVBUFSIZE);
129 bytesRcvd = recv(sock, strBuffer, RCVBUFSIZE - 1, 0);
130 printf("< %s : %s", FTP_SERVER, strBuffer);
131
132 data_port = get_data_port(strBuffer);
133
134 /* LIST */
135 /* First, fork a process */
136
137 pid = fork();
138
139 if (pid == 0) { /* child process */
140     int childSock;
141     struct sockaddr_in serverAddr;
142     int len = sizeof(serverAddr);
143     char buf[800];

```

```

144
145     memset(&serverAddr, 0, sizeof(serverAddr));
146     serverAddr.sin_family = AF_INET;
147     serverAddr.sin_addr.s_addr = inet_addr(FTP_SERVER);
148     serverAddr.sin_port = htons(data_port);
149
150     // socket for data connection
151     if ((childSock = socket(AF_INET, SOCK_STREAM, 0)) < 0)
152         DieWithError("socket() failedWn");
153
154     if (connect(childSock, (struct sockaddr *) &serverAddr,
155               sizeof(serverAddr)) < 0) {
156         DieWithError("connect() failed.");
157     }
158
159     totalBytesRcvd = 0;
160     memset(strBuffer, 0, RCVBUFSIZE);
161     bytesRcvd = recv(childSock, strBuffer, RCVBUFSIZE - 1, 0);
162     printf("< %s : Wn%sWn", FTP_SERVER, strBuffer);
163
164     printf("WnWnchild: terminated...Wn");
165     exit(0);
166 } else {
167     /* send LIST cmd thru the control connection */
168     memset(strBuffer, 0, RCVBUFSIZE);
169     strcpy(strBuffer, "LISTWrWn");
170     printf("> %s : %s", CLIENT, strBuffer);
171     send(sock, strBuffer, strlen(strBuffer), 0);
172
173     totalBytesRcvd = 0;
174     memset(strBuffer, 0, RCVBUFSIZE);
175     bytesRcvd = recv(sock, strBuffer, RCVBUFSIZE - 1, 0);
176     printf("< %s : %sWn", FTP_SERVER, strBuffer);
177 }
178
179 close(sock);
180
181 } /* main */

```

⟨Table 58⟩ showcases an example of implementing the scenario shown in ⟨Figure 73⟩. For better understanding, we will explain the actions described in the scenario in sequence. '1: Connection' corresponds to 'Row 75~90' which is calling the connect() function by using the FTP server's IP address (FTP\_SERVER) and control port number (FTP\_



CPORT). '2: Response', to the above mentioned connection, corresponds to 'Row 93~96'. <Table 59> describes the meaning of the response codes. The FTP response codes are composed of three numbers. It is easy to analyze the message if you understand the meaning of the first and the second number.

<Table 59> Meaning of response code

Response code	Meaning
1yz	Positive Preliminary reply
2yz	Positive Completion reply
3yz	Positive Intermediate reply
4yz	Transient Negative Completion reply
5yz	Permanent Negative Completion reply
x0z	Syntax
x1z	Information
x2z	Connections
x3z	Authentication and Accounting
x4z	Unspecified yet
x5z	File system status

'3: Log-in (User ID, Password)' is a step to make a connection to a specific user account. The ID and password should match each other. 'Row 99~106' is about sending the **USER Command** to the FTP server, where USER ID is sent to the FTP server following the USER Command. 'Row 110~118' corresponds to the **PASS Command**, responding to the USER Command, which sends out the password to the FTP server for the authentication. The next step can work only when the response codes for the USER Command and the PASS Command are in normal status. <Figure 73> briefly shows '4: Response' as a single step (actually, there are two responses one for the USER Command and the other for the PASS Command).

After the user authentication, '5: Passive Mode Connection' is carried out in 'Row 122~125'. This action is intended to remove the possibility of network connection issue that might be caused by firewalls and the like. At this time, the PASV Command is delivered to the FTP server. The response for the command is received in 'Row 127~130' or '6: Response' step. When the FTP client sends the PASV Command to the FTP server in request of a passive mode connection, the FTP server replies with "227 Entering Passive Mode. A1,A2,A3,A4,a1,a2" message to the FTP client. "A1, A2, A3, A4" means the IP address of the FTP server (A1,A2,A3,A4). While, "a1, a2" means the port number opened for the data connection. The actual port number is  $a1*256 + a2$ . The contents in the response message may be different but the structure is "A1,A2,A3,A4,a1,a2", which includes the IP address of the FTP server and data connection port number. The FTP client, receives this information, should be able to process the information properly. The `get_data_port()` function of 'Row 24~60' extracts the open port number from the server's response message relevant to the PASV Command and returns the value. Then, 'Row 132' uses the `get_data_port()` function to get the open FTP server's port number.

To run the command for '7: View Directory List', a new data connection needs to be generated by using the IP address of the FTP server and the port number received in '6: Response' step which were designated for the passive mode connection. To do that, a new process is generated by using the `fork()` function as shown in 'Row

137'. The newly created sub-process receives the outcome of the LSIT Command as shown in 'Row 139~165'. The process which has control connection sends the **LIST Command** to the FTP server as shown in 'Row 168~171'. Then, the FTP server sends out a response message through the control connection and this message is received in 'Row 173~176'. The first response code for the LIST Command is 150, which tells you about the file system status (5 on the second digit) and means positive preliminary reply (1 on the first digit). When sending the LIST Command, if the directory is not selected as a parameter, the FTP server will send the directory files and directory list of the current working directory through the data connection. The data connection made for the sub-process is sustained only during the outcome is being received so that the data stream, sent from the FTP server in response to the LIST Command, can be processed. 'Row 145~157' is about making a new socket connection by using the data connection port number of the server which is a part of the PSAV Command response message. If the connection is successful, the outcome of the LIST Command, which corresponds to '8: File/Directory List', is received in 'Row 159~162'. As the data connection was completed, 'Row 165' calls the `exit()` function to terminate the sub-process.

<Table 60> Outcome of viewing FTP list

1	ubuntu:~/Projects/FTP1\$ ./ftp_pasv_test
2	< 192.168.11.20 : 220 Welcome to Prof.Lee's FTP service.
3	> client : USER tester
4	< 192.168.11.20 : 331 Please specify the password.
5	> client : PASS tester
6	< 192.168.11.20 : 230 Login successful.
7	> client : PASV
8	< 192.168.11.20 : 227 Entering Passive Mode (192,168,11,20,117,247).
9	p1 = 117
10	p2 = 247
11	> client : LIST
12	< 192.168.11.20 : 150 Here comes the directory listing.
13	
14	< 192.168.11.20 :
15	drwxr-xr-x 2 1001 1001 4096 Sep 13 13:19 Desktop
16	drwxr-xr-x 2 1001 1001 4096 Sep 13 13:19 Documents
17	drwxr-xr-x 2 1001 1001 4096 Sep 13 13:19 Downloads
18	drwxr-xr-x 2 1001 1001 4096 Sep 13 13:19 Music
19	drwxr-xr-x 2 1001 1001 4096 Sep 13 13:19 Pictures
20	drwxr-xr-x 2 1001 1001 4096 Sep 13 13:19 Public
21	drwxr-xr-x 2 1001 1001 4096 Sep 13 13:19 Templates
22	drwxr-xr-x 2 1001 1001 4096 Sep 13 13:19 Videos
23	-rw-r--r-- 1 1001 1001 8980 Oct 04 2013 examples.desktop
24	-rw-r--r-- 1 1001 1001 5241 Aug 30 20:10 ftp_pasv_test.c
25	drwxrwxr-x 2 1001 1001 4096 Aug 30 20:10 temp
26	drwxrwxr-x 2 1001 1001 4096 Aug 30 20:10 work
27	
28	

29	
30	child: terminated...
31	ubuntu:~/Projects/FTP1\$

⟨Table 60⟩ shows the outcome of the program described in ⟨Table 58⟩. As a result of '1: Connection' step, the welcome message of 'Row 2', which corresponds to '2: Response' is received from the FTP server. According to ⟨Table 59⟩, **Response Code 220** means 'Positive Completion Reply' and is related to the 'Connection'. Any other response codes can also be interpreted with the same mechanism. The outcome of '3: Log-in (User ID, Password)' and '4: Response' correspond to 'Row 3~6'. '5: Passive Mode Connection' and its pair '6: Response' correspond to 'Row 7 and 8' respectively. 'Row 9~10' is the message printed out of the console to validate whether the right port number was found in the message delivered from the `get_data_port()` function. '7: View Directory List' covers: sending the LIST Command to the FTP server which corresponds to 'Row 11'; and receiving the response, for the LIST command, coming from the control connection, and this action corresponds to 'Row 12'. As the response code is 150, it means the file system status and positive preliminary reply. The outcome was transmitted through the data connection and is printed in 'Row 15~26'. As a result of the LIST Command, which was sent to the FTP server, the files and directory information of the current working directory are displayed.

### Example of FTP Implementation with Library

The previous example was intended to give an instruction how to implement a protocol based on the standard and how to use the 'C' language in building an FTP protocol. In this chapter, we will look into how a pre-built **Open Source Library** can help us enhance productivity as opposed to building from scratch. In the case of the network program development, most of them are built based on the standard. However, FTP protocol has two implementation options: one is starting from scratch; and the other is to utilize a pre-built library for higher productivity.

**Apache Commons Net™**, which was used for FTP protocol implementation, has been used in implementing many of the widely used Internet protocols such as FTP/FTPS, FTP over HTTP, NNTP, SMTP(S), POP3(S), IMAP(S), Telnet, NTP/SNTP. [9].

⟨Table 61⟩ FtpList.java

1	/*
2	* Licensed to the Apache Software Foundation (ASF) under one or more
3	* contributor license agreements. See the NOTICE file distributed with
4	* this work for additional information regarding copyright ownership.
5	* The ASF licenses this file to You under the Apache License, Version 2.0
6	* (the "License"); you may not use this file except in compliance with
7	* the License. You may obtain a copy of the License at
8	*
9	* <a href="http://www.apache.org/licenses/LICENSE-2.0">http://www.apache.org/licenses/LICENSE-2.0</a>
10	*
11	* Unless required by applicable law or agreed to in writing, software

12	* distributed under the License is distributed on an "AS IS" BASIS,
13	* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
14	* See the License for the specific language governing permissions and
15	* limitations under the License.
16	*/
17	
18	import java.io.IOException;
19	import java.io.PrintWriter;
20	import java.util.logging.Level;
21	import java.util.logging.Logger;
22	import org.apache.commons.net.PrintCommandListener;
23	import org.apache.commons.net.ftp.FTPClient;
24	import org.apache.commons.net.ftp.FTPFile;
25	import org.apache.commons.net.ftp.FTPReply;
26	
27	/**
28	* Simplified version of FTPClientExample.java to show how to use the Apache
29	* Commons Net library
30	*    
31	* URL: <a href="https://commons.apache.org/proper/commons-net/index.html">https://commons.apache.org/proper/commons-net/index.html</a> 
32	* URL:
33	* <a href="https://commons.apache.org/proper/commons-net/javadocs/api-3.3/index.html">https://commons.apache.org/proper/commons-net/javadocs/api-3.3/index.html</a>
34	*
35	*
36	* @author Gil-Dong Hong
37	* @since August 30, 2015
38	*/
39	public class FtpList {
40	
41	private FTPClient ftpClient = new FTPClient();
42	private String hostIp = null;
43	private String userid = null;
44	private String passwd = null;
45	private int cPort = 21;
46	
47	/**
48	* Constructor
49	*
50	* @param hostIp an IP address of an FTP server, ex) "192.168.11.110"
51	* @param userid a user account

```

52  * @param passwd a password of a user account
53  * @param cPort a control port of an FTP server. ex) 21 (default)
54  */
55  public FtpList(String hostIp, String userid, String passwd, int cPort) {
56      this.hostIp = hostIp;
57      this.userid = userid;
58      this.passwd = passwd;
59      this.cPort = cPort;
60
61      ftpClient.addProtocolCommandListener(
62          new PrintCommandListener(new PrintWriter(System.out)));
63  }
64
65  /**
66   * Gets the result of an FTP LIST command
67   *
68   * @return if the list of directories or files is present. <br>
69   * null if login is not successful.
70   */
71  public String getList() {
72      StringBuilder buffer = new StringBuilder();
73
74      boolean loginSuccessful = login();
75
76      if (loginSuccessful) {
77          ftpClient.enterLocalPassiveMode();
78
79          try {
80              FTPFile[] files = ftpClient.listFiles();
81              for (FTPFile f : files) {
82                  buffer.append(f.getRawListing() + "WrWn");
83              }
84          } catch (IOException ex) {
85              Logger.getLogger(FtpList.class.getName()).log(Level.SEVERE, null, ex);
86          }
87      }
88
89      return buffer.toString();
90  }
91

```

```

92  /**
93   * Indicates whether the login is successful or not
94   *
95   * @return TRUE if login is successful
96   */
97  private boolean login() {
98      boolean status = false;
99      int reply;
100
101      try {
102          ftpClient.connect(hostIp, cPort);
103          reply = ftpClient.getReplyCode();
104
105          if (!FTPReply.isPositiveCompletion(reply)) {
106              ftpClient.disconnect();
107              System.err.println("FTP server refused connection.");
108              System.exit(1);
109          }
110
111          if (!ftpClient.login(userid, passwd) {
112              ftpClient.logout();
113          }
114          status = true;
115
116      } catch (IOException ex) {
117          Logger.getLogger(FtpList.class.getName()).log(Level.SEVERE, null, ex);
118      }
119      return status;
120  }
121

```

(Table 61) shows an example how the FTPClientExample.java, a sample from the Apache Commons Net™ Library, was modified to meet the goal described in the scenario of (Figure 5). The FTPClientExample.java program is in compliance with the Apache 2.0 License, and the modified program FtpList.java also includes the **Apache 2.0 License** in 'Row 1~16'. The instance variables to be used in the FtpList class are defined in 'Row 41~45'. While, 'Row 41' creates the FTPClient object which works as an FTP client so that it can be utilized within the class. In 'Row 56~59', string objects such as hostIp, userid, passwd, are initialized with the parameters of constructors which were delivered. The last row of the FtpList constructor, 'Row 61~62' will add a protocol command listener in order to monitor how the FtpClient object behaves and make the FTP command and responses are printed with System.out on the console. In this manner, it is easier to verify whether the FTP protocol works normally within the library. Among the steps shown in (Figure 73), '1: Connection ~ 6: Response' are conducted in the login method. '1:

**Connection** is carried out in 'Row 102', and **'2: Response'** for the Connection is carried out in 'Row 103'. In 'Row 105~109', the received code is validated and the connection is terminated if the response code is not 'positive completion reply' (code 220). If a connection is made with the FTP server, **'3: Log-in (user ID, password)' ~ '6: Response'** steps are conducted in 'Row 111'. As the FtpClient object is providing the login method, it is simpler than the previous example which was using the 'C' language. (In the previous example, it was necessary to send the USER command and the PASS command respectively and then process the responses for those commands.) Because of this benefit, a good library can make the program development easier and enhance productivity. If the log-in is made normally, 'true' is returned in order to notify that the log-in was completed normally.

The getList method in 'Row 71~90' is responsible displaying the directory list. First, the login method is run from 'Row 74' in order to cover **'1: Connection' ~ '6: Response'** steps shown in (Figure 73). 'Row 76' validates whether the log-in was successful or not, and calls the enterLocalPassiveMode method of the FtpClient object in order to make the **Passive Mode Connection** as was shown in 'Row 77'. This method makes the **PASV (or EPSV) Command** sent to the FTP server before the data connection so that **'5: Passive Mode Connection'** and **'6: Response'** can be processed.

**'7: View Directory List'** is carried out in 'Row 80'. The LIST Command is sent to the FTP server in order to view file/directory information of the selected directory. **'8: File/Directory List'** returns the file/directory information sent from the FTP server into the FtpFile object array so that file/directory list can be processed. In 'Row 81~82', the contents of the FtpFile object array should be stored at the StringBuilder object (buffer), and they are used as a return value of the getList method.

<Table 62> TestDrive.java

1	public class TestDrive {
2	public static void main(String[] args) {
3	FtpList ftpList = new FtpList("192.168.11.20", "tester", "tester", 21);
4	String listResult = ftpList.getList();
5	System.out.println("LIST result = \r\n" + listResult);
6	}
7	}

TestDrive.java of (Table 62) shows whether the FtpList class can work in accordance with the scenario shown in (Figure 73). In 'Row 3', the FtpList object is generated. In configuring the parameters, it should be set as hostIp = "192.168.11.20", userId = "tester", passwd = "tester", cPort = 21 in order to generate an object that can try access to the FTP server, just like the FTP program example of (Table 58) written in the 'C' language. Next, 'Row 4' calls the getList method of the FtpList object to get the directory list as a string object (listResult). Next, 'Row 5' prints out the returned value so that users can see the selected directory list.

(Table 63) shows the **Outcome**. When the getList method of the FtpList object is called, the login method is called

first to print out strings onto the console from 'Row 2' to 'Row 8'. The FTP-related commands and the response messages are displayed onto the console as an outcome of running the addProtocolCommandListener method of the FtpClient class. The outcome of running the enterLocalPassiveMode method of the FtpClient class is shown onto the console just like in 'Row 9~10'. The contents in 'Row 11~18' are printed onto the console by 'Row 83~88' of (Table 61). When comparing the two examples, the same directory name was printed as you can see in (Table 60). This means sending and receiving commands with the FTP server worked fine just as intended in the given scenario.

<Table 63> Outcome of View FTP List

1	ubuntu:~/NetBeansProjects/FtpClient\$ java -jar dist/FtpClient.jar
2	220 Welcome to Prof.Lee's FTP service.
3	USER tester
4	331 Please specify the password.
5	PASS tester
6	230 Login successful.
7	SYST
8	215 UNIX Type: L8
9	PASV
10	227 Entering Passive Mode (192,168,11,20,76,215).
11	LIST
12	150 Here comes the directory listing.
13	226 Directory send OK.
14	LIST result =
15	drwxr-xr-x 2 1001 1001 4096 Sep 13 13:19 Desktop
16	drwxr-xr-x 2 1001 1001 4096 Sep 13 13:19 Documents
17	drwxr-xr-x 2 1001 1001 4096 Sep 13 13:19 Downloads
18	drwxr-xr-x 2 1001 1001 4096 Sep 13 13:19 Music
19	drwxr-xr-x 2 1001 1001 4096 Sep 13 13:19 Pictures
20	drwxr-xr-x 2 1001 1001 4096 Sep 13 13:19 Public
21	drwxr-xr-x 2 1001 1001 4096 Sep 13 13:19 Templates
22	drwxr-xr-x 2 1001 1001 4096 Sep 13 13:19 Videos
23	-rw-r--r-- 1 1001 1001 8980 Oct 04 2013 examples_desktop
24	-rw-r--r-- 1 1001 1001 5241 Aug 30 20:10 ftp_pasv_test.c
25	drwxrwxr-x 2 1001 1001 4096 Aug 30 20:10 temp
26	drwxrwxr-x 2 1001 1001 4096 Aug 30 20:10 work
27	
28	ubuntu:~/NetBeansProjects/FtpClient\$

## 04 JSP Web Programming

### Characteristics of Web Programming

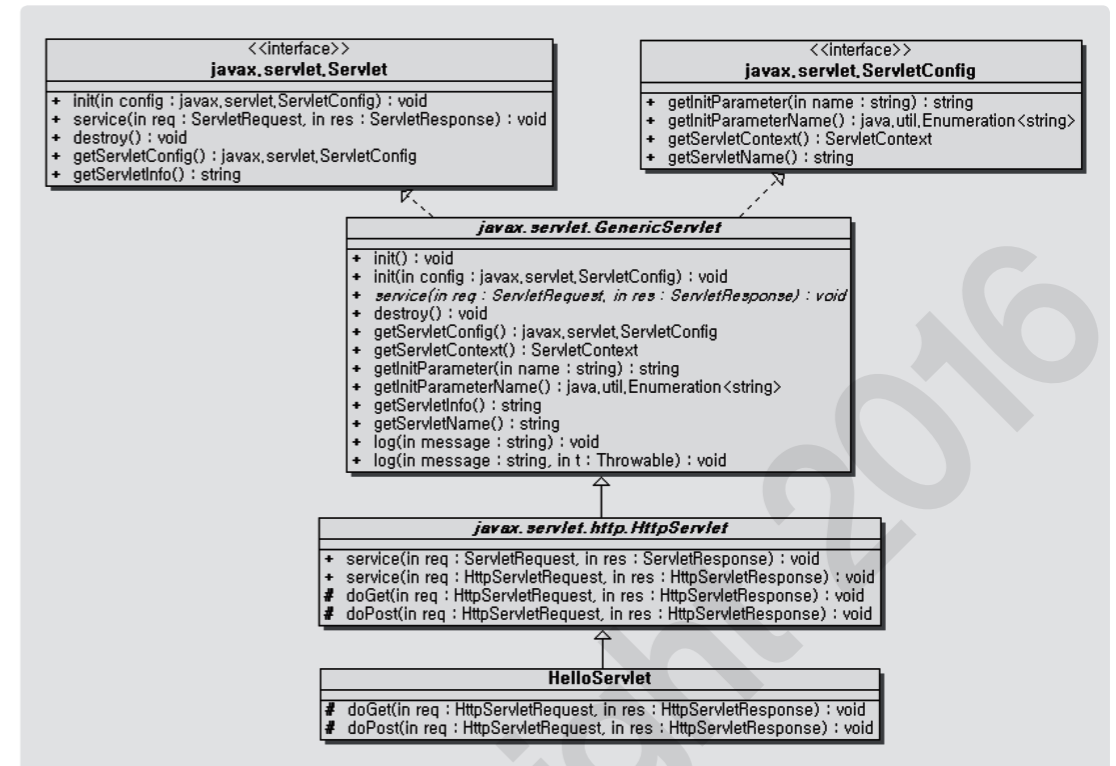
Generally used application programs are hardware/OS-dependent and should be installed on machines to run the program. However, **Web Applications** can be run in any environment as long as web browsers can work. Recently, it is not a trend to develop a set of application programs: one for the mobile environment and the other for the desktop environment. As the **Responsive Web Technology** is available, it is possible to develop a single application which can respond to various devices.

There are diverse **Server Programming Technologies** for web application program development such as JSP, ASP, PHP, Python, and the like. Also, there are **Client Programming Technologies** such as: HTML 4[11] and HTML 5[12]—description languages for web page structure; CSS2/CSS3[13]—description languages of web page presentation; and JavaScript[14]—a script language of the web client side. JavaScript was originally called ECMAScript, which was created as an object-oriented programming language used for calculation and object manipulation in the host environment. JavaScript was used in the Netscape browser and Microsoft introduced similar technology called Jscript. JavaScript was used along with XML to support asynchronous communications technology for conversational applications. Since then, AJAX (Asynchronous JavaScript and XML) technique has been used widely for the web development.

There will be a brief introduction about the JSP technology which can be used for developing web applications. In 1997, Sun Microsystems developed a Java-based technology called a servlet for dynamic content generation. The technology was not widely used as a web server technology due to its convenience in use. However, the technology is used in many areas these days because of the JSP (JavaServer Pages) technology which can allow descriptions in the XML format and translate them into the servlet.

A JSP web application program is composed of: JSP page, servlet, JavaBeans component on the server side; and static documents such as HTML/image, Java applets, JavaScript on the client side [16]. In a JSP web application program, a JSP page plays an important role and takes the responsibility for the “View” out of ‘Model–View–Controller’. However, once the program is running, JSP pages are turned into the servlet, so practically there is no difference with the servlet. However, it is more convenient for developers because they can simply describe what they want in the XML format instead of Java programming.

A servlet is composed of various components as shown in (Figure 74): the javax.servlet.Servlet interface which defines the **Servlet Life Cycle**; the javax.servlet.ServletConfig interface which takes the responsibility of **Initializing Servlet**; and the javax.servlet.GenericServlet abstract class which specifies a **Protocol-Independent Servlet** by implementing the two interfaces mentioned above. The GenericServlet abstract class is inherited to specify the javax.servlet.http.HttpServlet class that can be operated through the **HTTP protocol**. Usually, this HttpServlet class is inherited so that developers can specify the servlet with the features he/she would like to put in.

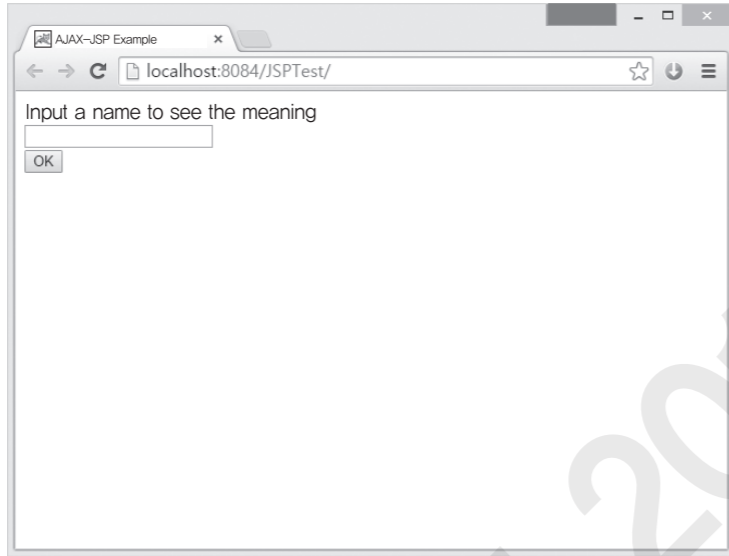


(Figure 74) How to implement a web application program using the servlet [16]

Hence, JSP pages and servlets can be used to build the same feature. In addition, features only specific to JSP pages can be used to implement a certain function easier than using servlets. For example: **Declaration, Scriptlet Expression, Scriptlet** can be used to make the Java programming language elements available in JSP pages; **Directives** which are used to select attributes in JSP pages or works as a tag to do page-level work in the XML format; **Standard Action** which can have the impact on the current ‘out stream’ and can modify or create objects [16]. In addition, there are various other features to help developers with implementation: **Expression Language** which makes it possible to write JSP pages without a script and which is differentiated from the Scriptlet in its expression and syntax; **Tag Extension** which can introduce a new action into JSP pages; and **Standard Tag Library** such as JSTL. There is a certain limitation to introduce all the JSP features in detail in the TOPCIT Essence. It is recommended to study other materials and standard documents such as JSR-245 (JSP) and JSR-315 (Servlet).

### Examples of Web Programming

There will be an introduction about the steps how a web application programs can be built by using multiple technological components of JSP. To that end, a simple web application program will be developed: which takes the English name as an input and shows its meaning as an output. To make the explanation easier, screen shots will be introduced, following the sequence of actions. This kind of screen design will be used during the web application program development/analysis/design process. (Figure 75) shows a default screen of the web application program we will build. There is a ‘Form Field’ where you can put a name in and an ‘OK’ button.



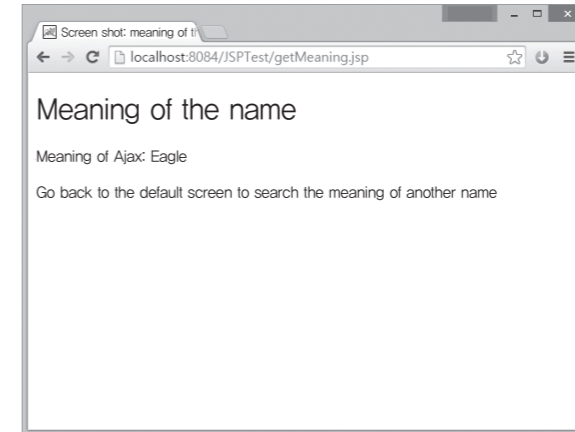
(Figure 75) Default screen

To find out the meaning of a name, a user can start to type-in a name as shown in (Figure 76). As the user cannot know what names are available on the list, there is an automatic completion feature so that the user can select one from the recommendation. The names in the gray box are the matches for 'A'.



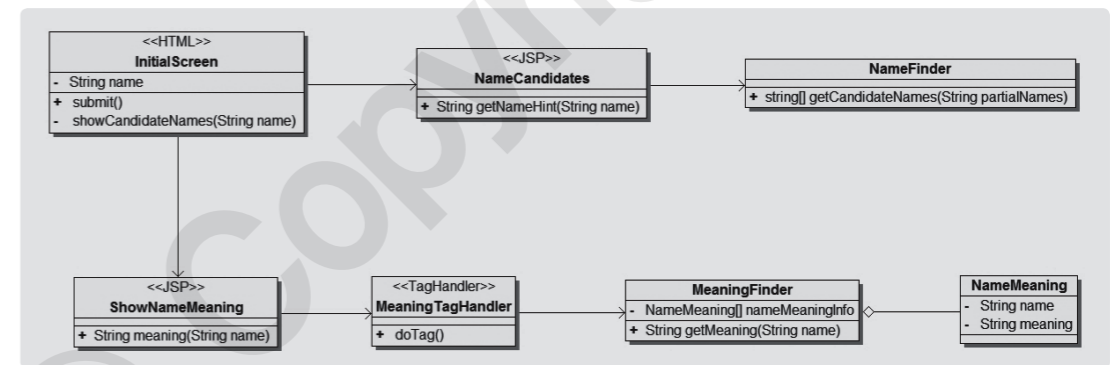
(Figure 76) Screen shot: partial input and automatic completion

Next, the user types in 'j', 'a', 'x' in sequence and click the OK button in order to move to another screen that shows the meaning of a name as shown in (Figure 77).



(Figure 77) Screen shot: meaning of the searched name

(Figure 78) is a UML class diagram that indicates what kinds of classes are necessary and what kinds of relationship should be built among them to meet the goal of our scenario. For easier understanding, attributes and operations are written in Java syntax. The default screen shown in (Figure 75) corresponds to the InitialScreen class. The automatic completion of the candidates shown in (Figure 76) is carried out by the NameCandidates class. The screen for showing the meaning of the searched name shown in (Figure 77) corresponds to the ShowNameMeaning class. Other classes are doing additional roles so that the classes mentioned above can work as intended.



(Figure 78) Class diagram: 'Finding the meaning of names'

InitialScreen class which is equivalent to the default screen of (Figure 75) is implemented in index.html file as shown in (Table 64). 'Row 1' tells you it is based on HTML 5. In 'Row 8~10', the CCS technology defines the font color and background color of the tag whose id is nameHint. 'Row 12~29' shows how the showCandidateNames function of the InitialScreen class is implemented with JavaScript. Whenever a letter is typed in, a call is made so that the getNameHint.jsp of 'Row 25' can be called in AJAX method to bring in the name candidates corresponding to the alphabet, and finally the inside of a tag, whose id is nameHint, gets changed. The getNameHint.jsp is equivalent to the NameCandidates class of (Figure 75). 'Row 35~40' presents the HTML form where the user name comes as an input. If the OK button is clicked, the getMeaning.jsp, which is equivalent to the ShowNameMeaning class, is called

as shown in 'Row 35' in order to display the meaning of the name which was typed-in, 'Row 36' shows that the showCandidateNames JavaScript function is called when a KEYUP event takes place. (A KEYUP event happens when a user pushes a keyboard with a finger and puts the finger off from the keyboard.)

〈Table 64〉 index.html

1	<!DOCTYPE html>
2	
3	<html>
4	<head>
5	<title>AJAX-JSP 예제</title>
6	<meta charset="UTF-8">
7	
8	<style>
9	#nameHint { color: aliceblue; background-color: lightslategray }
10	</style>
11	
12	<script>
13	function showCandidateNames( name ) {
14	if (name.length === 0) {
15	document.getElementById("nameHint").innerHTML = "";
16	return;
17	} else {
18	var xmlhttp = new XMLHttpRequest();
19	xmlhttp.onreadystatechange = function () {
20	if (xmlhttp.readyState === 4 && xmlhttp.status === 200) {
21	document.getElementById("nameHint").innerHTML =
22	xmlhttp.responseText;
23	}
24	}
25	xmlhttp.open("GET", "getNameHint.jsp?name=" + name, true);
26	xmlhttp.send();
27	}
28	}
29	</script>
30	
31	</head>
32	<body>
33	Type-in the name, to see the meaning of the name!
34	
35	<form name="info" action="getMeaning.jsp" method="POST"
36	onkeyup="showCandidateNames(name,value)">
37	<input type="text" name="name" value="" size="20" />

38	<div id="nameHint"> </div>
39	<input type="submit" value="OK" name="OK" />
40	</form>
41	</body>
42	</html>

The getNameHint.jsp is equivalent to the NameCandidates class of 〈Figure 78〉 and can be implemented as 〈Table 65〉 shows. 'Row 1~5' shows the **Comment** used in JSP. The comment used in JSP, as opposed to the HTML comment, is not shown to the user of the web application programs. 'Row 7' presents a **Page Directive** that defines the attributes of a page. 'Row 10' is about the **<jsp:useBean> Standard Action** which enables JavaBeans components to be used along with JSP. By using JavaBeans components, which can be reused, it is possible to use all the benefits of the Java programming language in JSP. This means user/customer requests can be met easily and flexibly as if you were using the Java language. Here, we are using the NameFinder class of the beans package. 'Row 19~28' works as a **Scriptlet** which enables users to use Java programming language easily within JSP pages. These kinds of features of the programming language can be used within JSP pages, which come with pros and cons; development work can become easier but it is hard to read at the same time. Hence, it is recommended not to use scriptlets within JSP pages, and the **Tag Extension** is provided as an alternative.

Here, the names in the form filed, which were delivered by the request object in the scriptlet, are delivered as parameters to the getCandidateNames method of the NameFinder object to bring the candidate names in a string array. Just as in 'Row 23~25', if the returned string array is not equal to null, they are printed line-by-line on the out object. These outcomes are delivered in asynchronous way to the JavaScript code of the index.html so that the candidate names can be displayed to users.

〈Table 65〉 getNameHint.jsp

1	<%--
2	Document  : getNameHint
3	Created on : Aug 30, 2015, 10:04:50 AM
4	Author    : jongmin
5	--%>
6	
7	<%@page contentType="text/html" pageEncoding="UTF-8"%>
8	<!DOCTYPE html>
9	
10	<jsp:useBean id="finder" scope="page" class="beans.NameFinder"/>
11	
12	
13	<html>
14	<head>
15	<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
16	<title>find name candidate</title>
17	</head>

18	<body>
19	<%
20	String partialName = request.getParameter("name");
21	System.out.println(partialName);
22	String[] candidates = finder.getCandidateNames(partialName);
23	if (candidates != null) {
24	for (String name : candidates) {
25	out.println(name + " ");
26	}
27	}
28	%>
29	
30	</body>
31	</html>

The NameFinder class used in the scriptlet within the getNameHint.jsp, shown in <Table 65>, is implemented just as shown in <Table 66>. 'Row 14~15' means all the names within the current working system. Usually, this type of data should have been managed in a table of a DBMS, but the data is intentionally managed in a string array type for simpler operation. 'Row 17~36' defines the getCandidateNames method. 'Row 18' is intended to declare variables so that the candidates meeting the requirement of 'Row 18' can be returned as string array type. To find how many names match to the pre-typed information, 'Row 20~24' is scanning through the string array. If the number of candidates is higher than 0, the string objects are generated as shown in 'Row 25~26', as many as the number of the candidates. The next step is to find out the names which match with the condition and to store them in the variable 'candidates'. If all the steps are completed, the final value is returned as shown in 'Row 35'.

<Table 66> NameFinder.java

1	package beans;
2	
3	/*
4	* To change this license header, choose License Headers in Project Properties.
5	* To change this template file, choose Tools   Templates
6	* and open the template in the editor.
7	*/
8	/**
9	*
10	* @author jongmin
11	*/
12	public class NameFinder {
13	
14	private String[] names =
15	{"Aaron", "Abel", "Adam", "Adel", "Ajax", "Jack", "Jacob", "James"};

16	
17	public String[] getCandidateNames(String partialName) {
18	String[] candidates = null;
19	int count = 0;
20	for (String name : names) {
21	if (name.startsWith(partialName)) {
22	count++;
23	}
24	}
25	if (count > 0) {
26	candidates = new String[count];
27	
28	int i = 0;
29	for (String name : names) {
30	if (name.startsWith(partialName)) {
31	candidates[i++] = name;
32	}
33	}
34	}
35	return candidates;
36	}
37	}

The ShowNameMeaning class of <Figure 78> is used to realize the getMeaning of index.html shown in <Table 67>. Here, 'Row 8' is using the **Taglib Directive**, which is very important in that it can connect JSP page script with existing Java programming language. In the getNameHint.jsp of <Table 65>, it was possible to use the scriptlet in order to freely use the Java codes in JSP pages. However, it does not provide good readability as codes and tags are mixed together. To overcome this kind of weakness, JSP provides the tag extension method so that the Java codes can be used in a tag structure, which means the presentation layer and the logical layer are separated to make independent development possible for each of the layers. This chapter will not cover details about the tag extension which is used to specify a tag library with the taglib directive.

The taglib directive in 'Row 8' needs to be connected to the MeaningTagHandler.java of <Table 68> so that it can be used in the getMeaning.jsp. The doTag method of the MeaningTagHandler.java is called in 'Row 19' so that the meaning for the selected name can be displayed. In this step, \$param["name"] is used as a parameter value, which indicates that the value of the name sent from the form of index.html in **Expression Language**. In the Java code terms, it has the same meaning with the request.getParameter("name"). The Expression Language can help reduce the use of the Java codes within JSP pages to make JSP pages simpler.

<Table 67> getMeaning.jsp

1	<%—
2	Document : getMeaning



3	Created on : 8 August 2015 11:35:06 p.m.
4	Author : jongmin
5	—%>
6	
7	<%@page contentType="text/html" pageEncoding="UTF-8"%>
8	<%@taglib uri="/WEB-INF/tlds/tag_library.tld" prefix="mylib" %>
9	
10	<!DOCTYPE html>
11	<html>
12	<head>
13	<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
14	<title>Display the meaning of the name</title>
15	</head>
16	<body>
17	<h1>Meaning of the name</h1>
18	
19	<mylib:meaning name="\${param['name']}" />
20	
21	</body>
22	</html>

The MeaningTagHandler.java which defines the tag library used in <code>getMeaning.jsp</code> is implemented as shown in <Table 68>. When the <code>mylib:meaning</code> tag is called, the <code>doTag</code> method of the MeaningTagHandler.java is called. The JspWriter object (out) in 'Row 19' is related to the PriterWriter object of the ServletReponse which is connected to the web browser. Therefore, when HTML data is sent to the JspWriter object, a user can see the rendered contents through the web browser. The MeaningFinder object, which can tell you about user name and the relevant meaning, is generated in 'Row 23'. While, 'Row 24' calls the <code>getMeaning</code> method of the MeaningFinder object in order to bring up the wanted information. 'Row 26~28' is about showing this information to users.

<Table 68> MeaningTagHandler.java

1	
2	package taghandler;
3	
4	import beans.MeaningFinder;
5	import javax.servlet.jsp.JspWriter;
6	import javax.servlet.jsp.JspException;
7	import javax.servlet.jsp.tagext.JspFragment;
8	import javax.servlet.jsp.tagext.SimpleTagSupport;
9	
10	/**
11	*

12	* @author jongmin
13	*/
14	public class MeaningTagHandler extends SimpleTagSupport {
15	private String name;
16	
17	@Override
18	public void doTag() throws JspException {
19	JspWriter out = getJspContext().getOut();
20	
21	try {
22	// TODO: insert code to write html before writing the body content.
23	MeaningFinder meaningFinder = new MeaningFinder();
24	String meaning = meaningFinder.getMeaning(name);
25	
26	out.println(name + "means: " + meaning);
27	out.println("   if you want to see the name of the meaning again");
28	out.println("<a href=W"/JSPTest/index.htmlW">Go back to the</a>default screen");
29	
30	JspFragment f = getJspBody();
31	if (f != null) {
32	f.invoke(out);
33	}
34	
35	} catch (java.io.IOException ex) {
36	throw new JspException("Error in MeaningTagHandler tag", ex);
37	}
38	}
39	
40	public void setName(String name) {
41	this.name = name;
42	}
43	
44	}

The MeaningFinder class is implemented as shown in <Table 69> and such information should be managed in a table structure of a DBMS just like the case of the NameFinder.java of <Table 66>. However, it is implemented in a simple array on purpose in this example. The NameMeaning object is used as an element of the array, and in <Table 70>, it is defined as a simple class which expresses the name and the corresponding meaning in strings. In 'Row 26~36', the <code>getMeaning</code> method is implemented. However, we skip the detailed explanation here as it is not so much different from general Java programming.

〈Table 69〉 MeaningFinder.java

1	package beans;
2	
3	/**
4	* cf. <a href="http://www.aussiethings.com.au/babynames/aboy.htm">http://www.aussiethings.com.au/babynames/aboy.htm</a>
5	*
6	* @author jongmin
7	*/
8	public class MeaningFinder {
9	private NameMeaning[] nameMeaningInfo = {
10	new NameMeaning("Aaron", "To sing; Messenger"),
11	new NameMeaning("Abel", "A breath, Son"),
12	new NameMeaning("Adam", "Of the red earth"),
13	new NameMeaning("Adel", "Fire, Noble"),
14	new NameMeaning("Ajax", "Eagle"),
15	new NameMeaning("Jack", "Replacer. A form of James/John"),
16	new NameMeaning("Jacob", "Held by the heel, Replacer. A form of James"),
17	new NameMeaning("James", "Supplanter, Replacer")
18	};
19	
20	/**
21	* gets the meaning of a given name
22	*
23	* @param name name to want to know the meaning
24	* @return if a corresponding name exists
25	*/
26	public String getMeaning(String name) {
27	String meaning = null;
28	
29	for (NameMeaning temp : nameMeaningInfo) {
30	if (temp.getName().equals(name)) {
31	meaning = temp.getMeaning();
32	break;
33	}
34	}
35	return meaning;
36	}
37	}

〈Table 70〉 NameMeaning.java

1	package beans;
2	
3	/**
4	*
5	* @author jongmin
6	*/
7	public class NameMeaning {
8	private String name;
9	private String meaning;
10	
11	public NameMeaning(String name, String meaning) {
12	this.name = name;
13	this.meaning = meaning;
14	}
15	
16	public String getName() {
17	return name;
18	}
19	
20	public String getMeaning() {
21	return meaning;
22	}
23	}

To meet the goals of the scenarios shown in 〈Figure 75〉~〈Figure 77〉 in the JSP web application programs, many technologies were used: client programming technologies such as HTML, CSS, JavaScript, AJAX; and JSP server programming technology. Also the JSP technical elements were used such as page, directive, 〈jsp:useBean〉, standard action, scriptlet, tag extension, and expression language. As software becomes more sophisticated, we need various technical elements as was described in our examples. Therefore, it is important to further develop programming capability based on the knowledge of these technical elements.

## Example Question

## Question type

Multiple choice question

## Question

Following are the examples of the application layer protocols.  
Choose just one protocol which has different client–server connection method.  
Briefly describe what is the feature and characteristics of the protocol.

① FTP      ② HTTP      ③ POP3      ④ SMTP      ⑤ TELNET

## Intent of the question

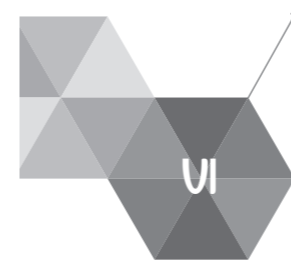
It is necessary to understand how the client–server connection works in order to build and realize application layer protocols.

## Answer and explanation

FTP: there are two types of client–server connection– Control Connection and Data Connection

## Related E–learning Contents

- Lecture 7 Application Layer Technologies



## Keep up with the Recent Trends in Network Technology

### ▶▶▶ Latest Trends and Key Issues

Cutting–edge network technologies have emerged backed by the high–speed data transmission and various convergence technologies. With the rise of technologies which can offer high quality services in traffic transmission on the Internet, services such as multimedia communications and VoIP have been initiated. More recently, technologies based on Internet of Thing (IoT) have been more widely used and the number of IoT standardization groups and institutes are growing as well. Software–based network, which has a whole different structure from the existing network, has been newly introduced. In this chapter, we will take a look at the recent network trends and key issues.

### ▶▶▶ Study Objectives

- \* To be able to understand the concept of multimedia network and VoIP and explain relevant protocols
- \* To be able to explain basic network technologies for IoT (Internet of Thing)
- \* To be able to explain the concept of software–based network

### ▶▶▶ Practical Importance Medium

### ▶▶▶ Keywords

Lossless compression, Lossy compression, QoS, SIP, H,323, RTP, RTCP, IMS, CSCF, HSS, 3GPP, SDN, OpenFlow, Control Plane, Data Plane, NFV, Virtualization, MQTT, CoAP

## 01 Multimedia Network

### Type of Image Compression

Image data makes up the majority of traffic on the multimedia network and the compression techniques of the image data can be divided into **Lossless** (reversible) compression and **Lossy** (irreversible) compression. The lossless compression, which is also called a reversible compression, is a compression algorithm that can recover all the original data without any data loss when the image data is decompressed. It provides a lower compression ratio than the lossy compression algorithm.

The lossy compression, which is also called an irreversible compression, is a compression algorithm in which some data is lost when the image data is decompressed, so the decompressed data is not perfectly identical to the original data.

#### ① Lossless compression

In the lossless compression technique, compression and decompression algorithms work exactly in the opposite way each other, so not any part of the data is lost. In this technique, every single bit of data remains the same after the data is decompressed or compressed. Some of the main lossless compression techniques in use are: **Run-length Encoding, Dictionary Coding, Huffman Coding, and Arithmetic Coding.**

〈Table 71〉 Types of lossless compression

Compression Type	Details
Run-length encoding (RLE)	• Repeated occurrences of a symbol are represented by the symbol and the number of occurrences (run-length) of the symbol.
Dictionary coding	• It scans through a message to build up a dictionary. When a string of symbols in the dictionary entry is found in the message, it is replaced by a code value (index value) assigned to that entry.
Huffman coding	• When data is coded in a binary pattern, shorter codes are assigned to the symbols that occur more frequently and longer codes are assigned to those that occur less frequently.
Arithmetic coding	• The entire message is mapped to a small interval inside [0,1] and the small interval is then encoded as a binary pattern.

#### ② Lossy compression

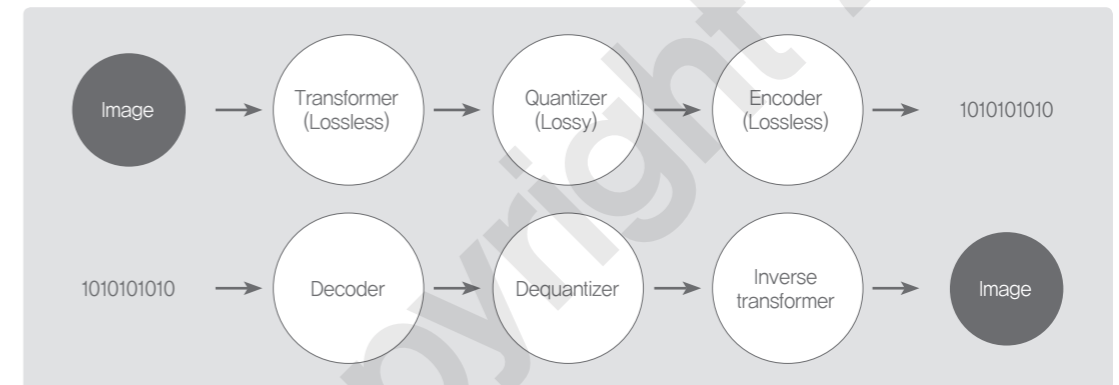
The Lossy compression refers to a data compression technique which eliminates redundant or unnecessary information in order to gain higher compression ratio at the expense of accuracy. Some of the main lossy compression techniques in use are **Predictive Coding** and **Transform Coding**. The predictive coding refers to a technique used to convert an analog signal to a digital signal. Instead of quantizing a PCM (Pulse Code Modulation) sample independently, it quantizes the difference between two adjacent samples. The difference values are smaller than the values of the original sample, so it can be coded using fewer bits. The transform coding is a technique used to transform a signal from one domain (typically, spatial/time domain) into another domain (generally, frequency domain), and then compress this signal.

### Multimedia data

Multimedia data refers to the data that consists of various media types like text, image, video and audio whereas the text comes in two formats: Plain Text and Hypertext. The text data uses Unicode as the underlying language to represent symbols. The text can be compressed using lossless compression method.

Image, which is also called a still image in the multimedia context, is the representation of a photograph, a fax page, or a frame in a moving picture. As shown in 〈Figure 79〉, an image is converted into the binary data through the transformation, quantization, and encoding procedures. This binary data can be converted to the image by going through the inversed procedures.

**DCT (Discrete Cosine Transform)** is a mechanism which is widely used in the first step of transformation processes for JPEG image compression. The decompression is achieved by an 'inverse DCT mechanism'. For the transformation and inverse transformation, the image is split into 8 x 8 blocks. The quantization refers to a process where the real numbers in the output of the DCT transformation are rounded off to the nearest integer, mostly resulting in 0. After the quantization, the image components are arranged in a zigzag order before the image is encoded. Finally, the lossless compression is made, using the run-length encoding or the arithmetic coding. [1]

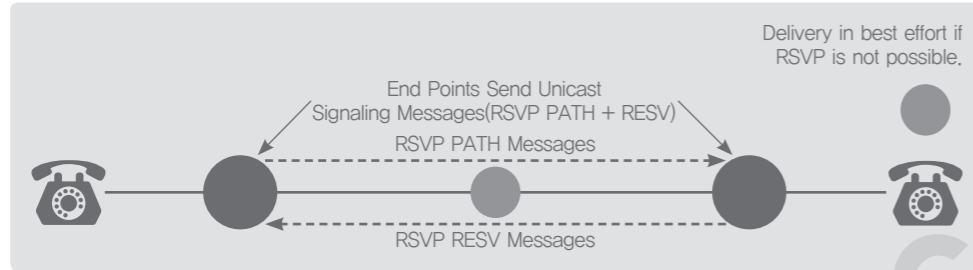


〈Figure 79〉 Image data transformation process

Video data is composed of multiple frames and each frame represents one image. This means that a video file requires a high transmission rate. The video compression is achieved by two compression methods: spatial and temporal. The spatial compression of each frame is done in the JPEG format and each frame is compressed independently. In the temporal compression, redundant frames are removed and three types of previously coded frames are used: I-frame (coded independently), B-frame and P-frame (both coded based on other I-frames). An analog audio signal is converted to a digital signal, using an Analog to Digital Converter (ADC). The ADC is composed of two main functions of sampling and quantization.

### QoS(Quality of Service)

To preserve the QoS on the multimedia network, **RSVP (Resource reservation protocol)** and the **TOS Field** are mainly used. As shown in 〈Figure 80〉, RSVP is designed to reserve the network bandwidth for an end-to-end data transmission where data is processed on a first-in-first-out basis. In the TOS field method, each packet is given with a 'TOS field class' and the processing priority is determined based on the value in the TOS field.



<Figure 80> How RSVP contributes to the Quality of Service

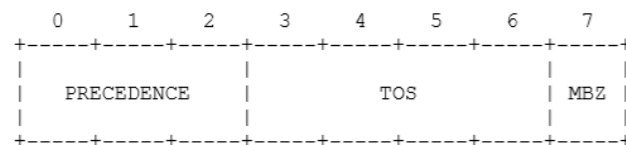
RSVP is one of the signaling protocols from the IETF where a data flow can be reserved with the static bandwidth through an end-to-end signaling. It is generally designed to reserve a queue [2]. There are two fundamental RSVP message types: Path and Resv. The path message which is initiated by the sender host travels through the network to all the receivers along the path. It stores information which is necessary for the receiver. On receipt of the path message, the receiver sends back the Resv message upstream to the sender. This Resv message reserves resources of the routers that support RSVP. When the router does not support RSVP along the way, packets are delivered through the best-effort delivery mechanism.

```
<Path Message> ::=
    <Common Header> [ <INTEGRITY> ]
    [ [ <MESSAGE_ID_ACK> | <MESSAGE_ID_NACK> ] ... ]
    [ <MESSAGE_ID> ]
    <SESSION> <RSVP_HOP>
    <TIME_VALUES>
    [ <EXPLICIT_ROUTE> ]
    <LABEL_REQUEST>
    [ <PROTECTION> ]
    [ <LABEL_SET> ... ]
    [ <SESSION_ATTRIBUTE> ]
    [ <LSP_REQUIRED_ATTRIBUTES> ... ]
    [ <LSP_ATTRIBUTES> ... ]
    [ <NOTIFY_REQUEST> ]
    [ <ADMIN_STATUS> ]
    [ <POLICY_DATA> ... ]
    <sender_descriptor>
    [ <S2L_sub-LSP_descriptor_list> ]
```

<Figure 81> Format of RSVP PATH message [3]

As shown in <Figure 82>, in the mechanism where a TOS field is used to prioritize the IP datagram, 1-byte TOS (Type of Service) field is checked and the priority is made based on the value. There are eight classes in the TOS Class, ranging from 0 to 7 and the priority goes up if the number is higher.

The first field, labeled PRECEDENCE, represents the priority or importance of the packet. The second field, labeled as TOS, denotes how the network makes tradeoffs among throughput, delay, reliability, and cost. The last field, labeled as MBZ (must be zero), is currently unused.



<Figure 82> TOS field structure [4]

TOS field is expressed as 4-bit binary numbers. Each number has a meaning, as shown in <Table 72>.

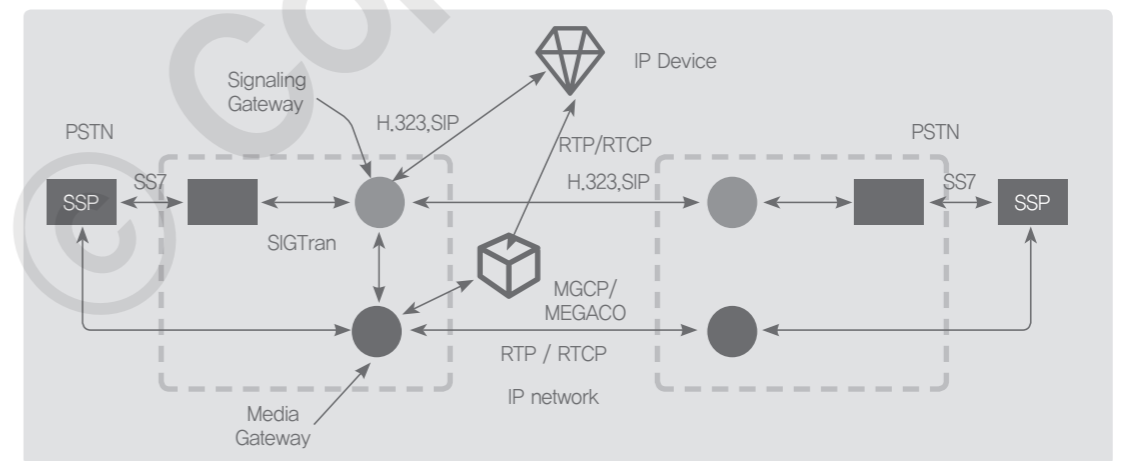
<Table 72> TOS field values [4]

Binary number	Meaning
1000	minimize delay
0100	maximize throughput
0010	maximize reliability
0001	minimize cost
0000	normal service

## 02 Basic Idea about VoIP and Call Signaling Protocol

### What is Voice Over Internet Protocol (VoIP)?

VoIP is a voice communications technology that relies on the packet data transferred over the IP network. <Figure 83> shows the VoIP system architecture. The two main blocks for the VoIP services are the **Media Gateway** and the **Signaling Gateway**. The media gateway provides support for the delivery of multimedia data. It converts the data to the relevant format and sends it to the target network. It is controlled by MGCP (Media Gateway Control Protocol). A signaling gateway is a network component responsible for call signaling by using protocols, such as H,323, SIP, MGCP and MEGACO. Its function is to convert signals on the PSTN and the IP network, so that the signals can be used between the two.



<Figure 83> VoIP system architecture

## VoIP Call Signaling Protocol

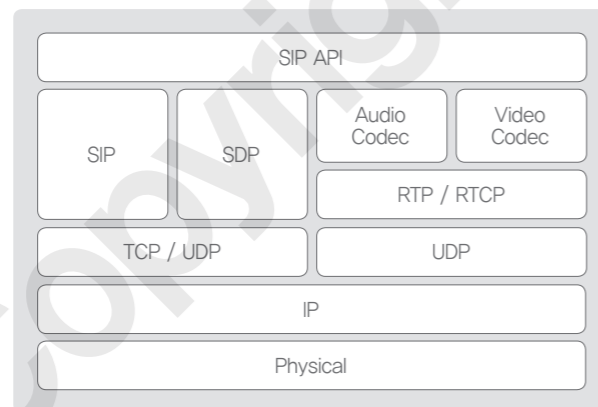
VoIP call signaling protocol is composed of **SIP (Session Initiation Protocol)** and **H.323**. SIP is an application layer signaling protocol which specifies procedures that guide intelligent terminals, which want to join a communications on the Internet, to identify each other and find their locations. It is also used to create, modify, and terminate multimedia sessions between the terminals. H.323 protocol is a standard method defined by the ITU-T to provide audio, data, video communication services on the LAN which does not provide guaranteed quality of service. Multimedia communication services can be provided by following simple procedures which do not require any modification in the existing network elements. It has been widely used by the early VoIP operators.

### ① SIP (Session Initiation Protocol)

**SIP** is an application layer signaling protocol used for setting, modifying, and terminating multimedia communication sessions. It is independent of the lower-layer transport protocol and is scalable because SIP is an HTTP text-based protocol.

SIP provides mechanisms to control sessions, which enables circuit-switched call control in the packet-switched network. It also enables multimedia applications to run on the packet network, the Internet. It is a convenient solution for the text-based addressing, such as URL and E-mail, and for the message parsing and extension.

The SIP stack is shown in (Figure 84). It is located above the transport layer protocols, TCP and UDP. (Table 73) illustrates components of the session initiation protocol.



(Figure 84) Stack of the session initiation protocol

(Table 73) Components of SIP

Classification	Details
SIP	<ul style="list-style-type: none"> <li>• RFC 3261, defines the basic set of information regarding SIP</li> </ul>
SDP	<ul style="list-style-type: none"> <li>• Session Description Protocol, RFC 4566/3264</li> <li>• Defines a multimedia session parameter</li> </ul>
Audio Codec	<ul style="list-style-type: none"> <li>• G.711A, G.723.1, G.729A</li> <li>• Used for the voice coding, Spec of audio codec varies for the compatibility with various systems.</li> </ul>

Classification	Details
Video Codec	<ul style="list-style-type: none"> <li>• H.263, MPEG-4, H.264</li> <li>• Used for the video coding. H.263 is commonly used. Up-to-date version is H.264</li> </ul>
RTP/RTCP	<ul style="list-style-type: none"> <li>• Real-time Transport (Control) Protocol, RFC 3550, RFC 3551</li> <li>• Real-time communications protocol</li> </ul>

### ② SIP message structure

In SIP, users are identified by an SIP URI (Uniform Resource Identifier), which is similar to an e-mail address. By utilizing the SIP URI, SIP can provide the IP address agnostic services. A SIP message consists of the Start Line (which specifies the method type in request and SIP URI), Header (which sets the value for controlling sessions), Body (which shows what type is set in the content-type), and CR/LF (which is a blank line between the header and the body).

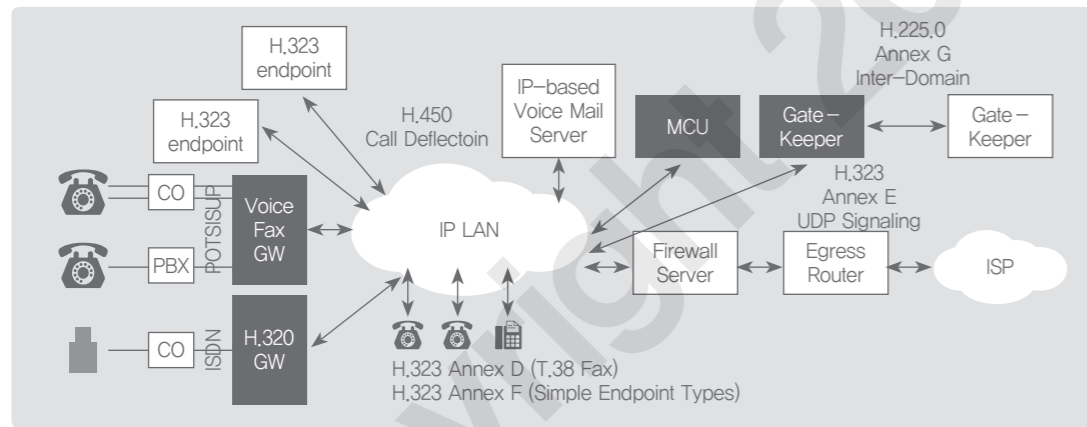
(Table 74) SIP Header [5]

Classification	Details
Request-URI	<ul style="list-style-type: none"> <li>• Set to the same value of the URI in the To header</li> <li>• When setting the Request-URI of REGISTER, it is set as a domain name without @</li> </ul>
To	<ul style="list-style-type: none"> <li>• Specify the logical recipient of the request</li> </ul>
From	<ul style="list-style-type: none"> <li>• Indicates the identity of the initiator of the request message</li> <li>• It contains the destination UA (User Agent), the end point URI (Uniform Resource Indicators), and a tag parameter.</li> </ul>
Call-ID	<ul style="list-style-type: none"> <li>• Identify the identity of the initiator of the request message</li> <li>• It contains a print-out name and a URI of a UA who sends the request.</li> <li>• It contains a tag parameter. The tag parameter serves as a general mechanism used together with the Call-ID to identify a dialogue. Its value should be unique and encrypted.</li> </ul>
CSeq	<ul style="list-style-type: none"> <li>• Serves to identify and orderly arrange transactions</li> <li>• It consists of a sequence number and the method.</li> </ul>
Max-Forwards	<ul style="list-style-type: none"> <li>• Used to limit the number of hops that the request may take before reaching the destination</li> <li>• When a request reaches 0 before reaching its destination, the 483 response (Too Many Hops_error) is returned as a reply.</li> </ul>
Via	<ul style="list-style-type: none"> <li>• It contains the transport protocol used to send the message and the SIP version.</li> </ul>
Contact(option)	<ul style="list-style-type: none"> <li>• An address designated to a UAC (User Agent Client) which sends the SIP request message</li> <li>• A contact address is used as a reference when the UAS sends a response.</li> <li>• It is the mandatory header to send an INVITE.</li> </ul>
Require(option)	<ul style="list-style-type: none"> <li>• Used by a UAC to identify an extension necessary to handle the request sent.</li> </ul>

H.323

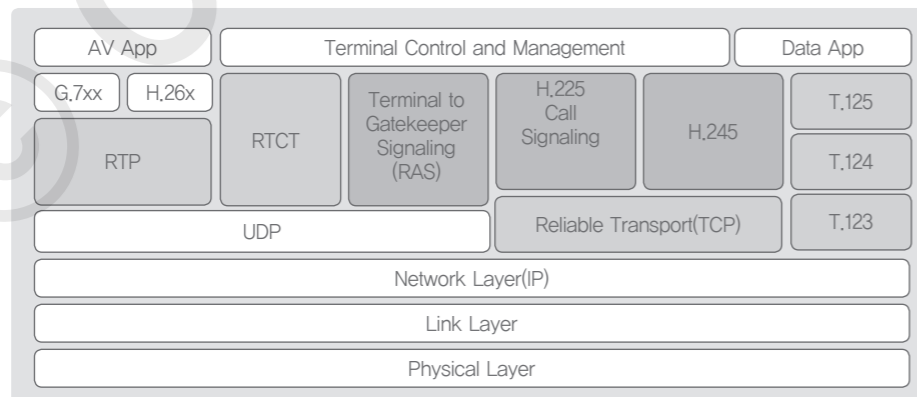
H.323 is a standard defined by the ITU-T to provide audio, data, video services on the LAN which does not offer guaranteed quality of service. Multimedia communications services are provided by following simple procedures which do not require any modification of the existing network elements. It has been widely used by the early VoIP operators.

The architecture of H.323 network is shown in (Figure 85). It consists of the following components: a terminal which is a machine used by a real user, such as general telephone devices, fax and PC equipped with multimedia devices; a gatekeeper which is used to perform the translation between the E.164 and an IP address, to perform redirection and call verification, and to manage call signaling, components, and bandwidths; and a gateway which provides a mapping function to logically connect (through encoding, protocol, and call control) different networks (IP network, PSTN, ISDN, ATM, etc.)

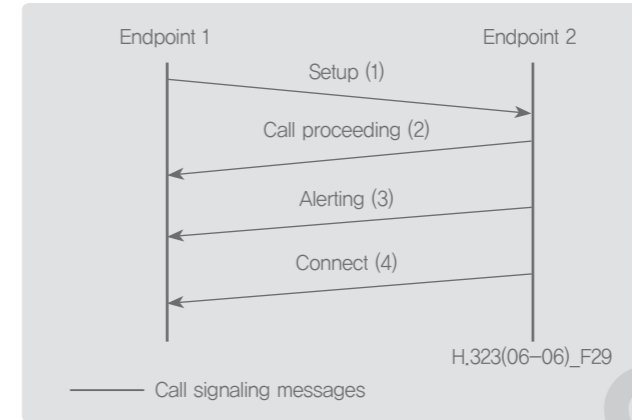


(Figure 85) H.323 network architecture

The H.323 stack is shown in (Figure 86). (Figure 87) illustrates how an end-to-end call is basically established without a gatekeeper.



(Figure 86) The H.323 protocol stack



(Figure 87) Basic call configuration (without a gatekeeper) [6]

03 Media Transport Protocol

Types of Media Transport Protocol

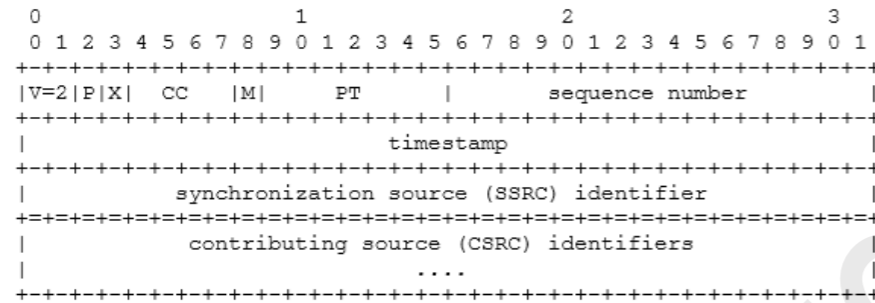
The media transport protocol comprises two parts: RTP (Real Time Transport Protocol) and RTCP (Real Time Control Protocol). RTP is designed to carry traffic real-time on the Internet. It is mainly used to transmit the video or audio data in real time on the Internet. Meanwhile, RTCP is used to control RTP that is a protocol for streaming the video or audio on the Internet. Both RTP and RTCP are specified in the IETF RFC 1889.

RTP(Real-time Transport Protocol)

RTP (Real-time Transport Protocol) is designed to address real-time traffic request on the Internet and mainly used to deliver video or audio data in real time on the Internet. RTP is mostly used upon the UDP (User Datagram Protocol) of the transport layer.

In the media file transfer over the RTP protocol, the sender packetizes media data, which is compressed with the codec, into the RTP packet and sends it to the receiver using UDP. Since it runs on the UDP, RTP doesn't guarantee timely delivery of packets, nor does it prevent packet losses. Hence, a variety of information included in the RTP packet header should be utilized properly in the video and audio applications to make sure that the processing can be completed seamlessly. RTCP can be used to maintain the QoS of RTP and to make the media streams synchronized.

The format of the RTP header is shown in (Figure 88). The RTP header consists of a fixed header (min. 12 bytes, from V to SSRC), followed by optional extension headers. The RTP payload comes after the RTP header.



<Figure 88> Structure of the RTP packet header [7]

<Table 75> Components of RTP packet header [7]

Header	Bits	Details
V (Version)	2 bits	• Indicates the version of the protocol. The current version is 2.
P (Padding)	1 bit	• Used to indicate if there are extra padding bytes. If this field is set, padding bytes are added at the end of the RTP packet. • Padding bytes are used in the encryption algorithm or used to align the packet length.
X (Extension)	1 bit	• Indicates the existence of an extension header between the fixed header and the payload.
CC (CSRC Count)	4 bits	• Indicates the number of CSRC identifiers that comes after the fixed header
M (Marker)	1 bit	• Defined by a profile, and its usage is determined by the payload type • Used to mark the frame boundary of the media file (or the like)
PT (Payload Type)	7 bits	• Indicates the format of the payload: audio or video encoding type field
Sequence Number	16 bits	• Incremented by one for each RTP packet sent, and used by the receiver to detect packet losses and to restore packet sequence • Used at the application layer
Timestamp	32 bits	• Reflects the sampling timing of the first byte of the RTP data packet • Used to enable the receiver to play back the received media at in the order of the contents transferred • It is derived from the media sampling clock of the sender. The field unit is determined in accordance with the payload type. It is determined by the RTP profile for applications.
SSRC	32 bits	• Synchronization Source • Used to identify the RTP stream source • The values in this field should be unique within the same RTP session.

<Table 76> Payload Type

Type	Application	Type	Application	Type	Application
0	PCMu Audio	7	LPC audio	15	G728 audio
1	1016	8	PCMA audio	26	Motion JPEG
2	G721 audio	9	G722 audio	31	H.261
3	GSM audio	10–11	L16 audio	32	MPEG1 video
5–6	DV14 audio	14	MPEG audio	33	MPEG2 video

### RTCP(Real-time Transport Control Protocol)

RTCP is a protocol for controlling RTP which is used for the video and voice streaming over the Internet and is defined in RFC 1889 along with RTP as a part of the IETF standard. RTCP packet types include: Sender Report Packet, Receiver Report Packet, Source Description Message, Bye Message, and Application Specific Packet. The role of each packet type is described in <Table 77>.

<Table 77> RTCP packet type [8]

Type	Details
Sender report packet	A type of packet used to report about the sending and receiving statistics of all the RTP packets that were sent by active senders within a certain session.
Receiver report Packet	A type of packet generated by passive participants; participants not sending any RTP packets. Reporting is about the service quality information directed to senders and other receivers except for the packet receiver.
Source description packet	A type of packet that is periodically sent to deliver the additional information about the source
Bye packet	A type of packet that is delivered to terminate a stream
Application specific packet	A type of pack that is used experimentally for the application programs

### IMS (IP Multimedia Subsystem)

IMS (IP Multimedia Subsystem) was first introduced by the 3GPP (3rd Generation Partnership Project) which is developing the international standards for mobile communications. The IMS is a basic platform to provide IP multimedia services and the SIP-based call control is used as a core technology for the IMS.



① Introduction to IMS

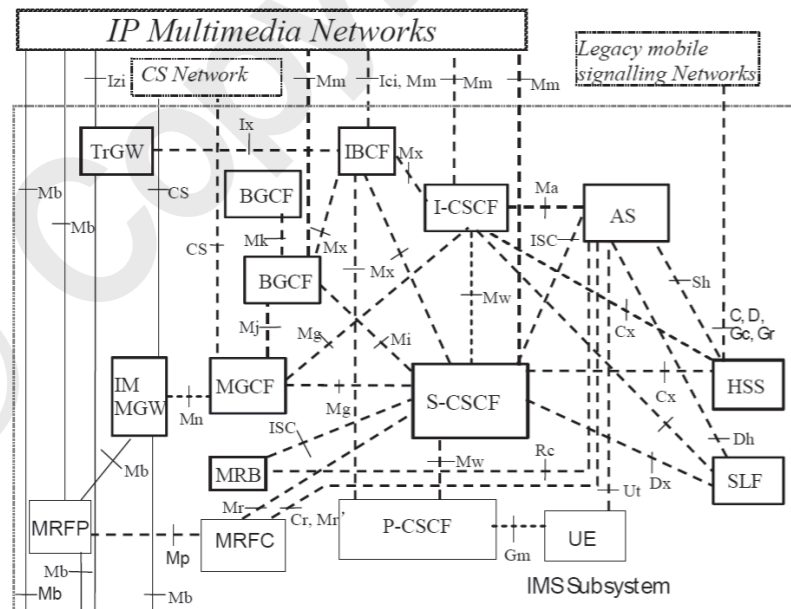
IMS is a communications platform defined by the 3GPP (an international standards institute) in order to control multimedia sessions and provide multimedia services based on SIP. The IMS can also be defined as a core network designed to provide integrated services in the wired/wireless multi access network environment.

② Goal of IMS service

IMS is intended to provide multimedia services such as voice, audio, video based on the Internet protocol, and to develop and modify the services swiftly. A set of commonly used internet-based technologies were used to make the price more affordable. In addition, an effective session management has been introduced to make the interface with various 3rd party applications easier. Hence, it is possible to expand the business further thanks to global interconnectivity among services.

③ Structure of IMS network

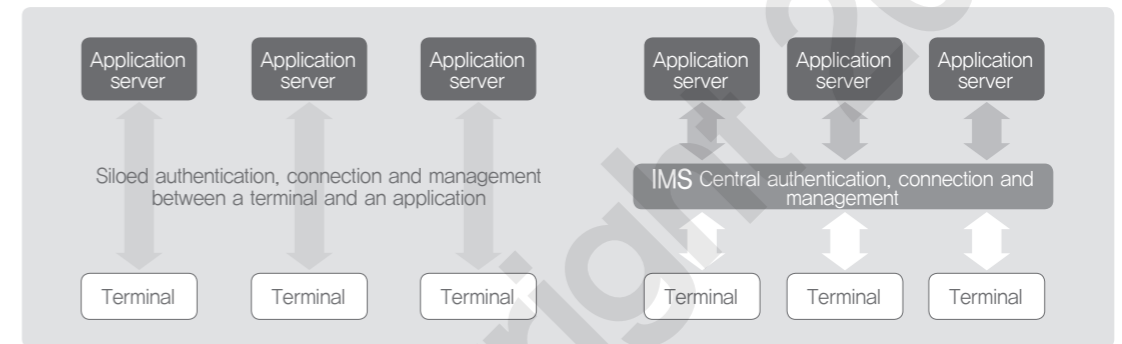
The logical structure of the all IP networks can be categorized into Radio Network Domain, the GPRS Packet Switched Service Domain, and IP Multimedia Service Domain. The GPRS packet switched service domain can be replaced with the IP network provided by packet routers, which is beyond the 3GPP scope. While, the wireless domain can be replaced by other wireless network access domains and the data network access domains such as Wibro and Mobile-LAN, which are beyond the 3GPP scope. The service domain for IMS is composed of: CSCF (Call Session Control Function) which is responsible for the registration and multimedia call processing of SIP messages; and HSS (Home Subscriber Server) which is an upgrade from the existing HLR (Home Location Register) of the legacy mobile network with the addition of the IP multimedia user mobility management and authentication.



<Figure 89> IP Multimedia: core network system reference architecture [9]

<Table 78> Main components of IMS service domain

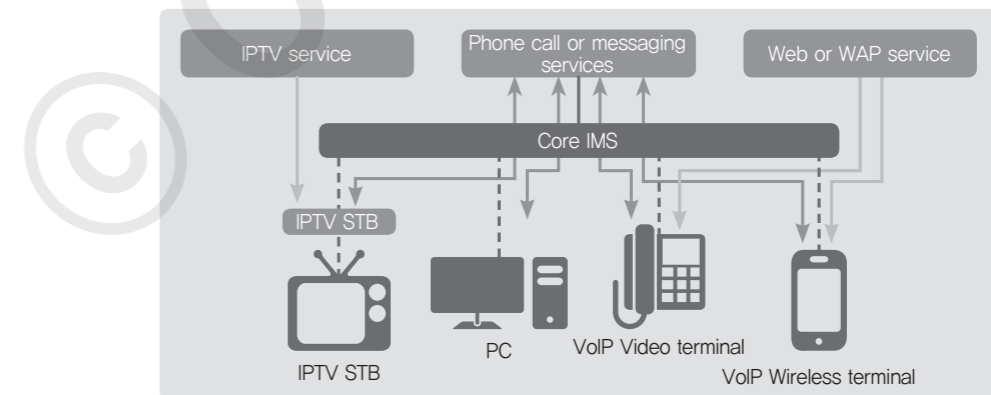
Classification	Detail
P-CSCF	• The first IMS node encountered when a user (UE, User Equipment) is trying to establish a connection.
I-CSCF	• The first IMS node encountered when the user's home network is connected to IMS.
S-CSCF	• A subsystem that controls the user's session • A registrar sending an authentication challenge to the user with the authentication vectors supplied by the HSS.
HSS	• Supports the user authentication, message integrity check, and encryption



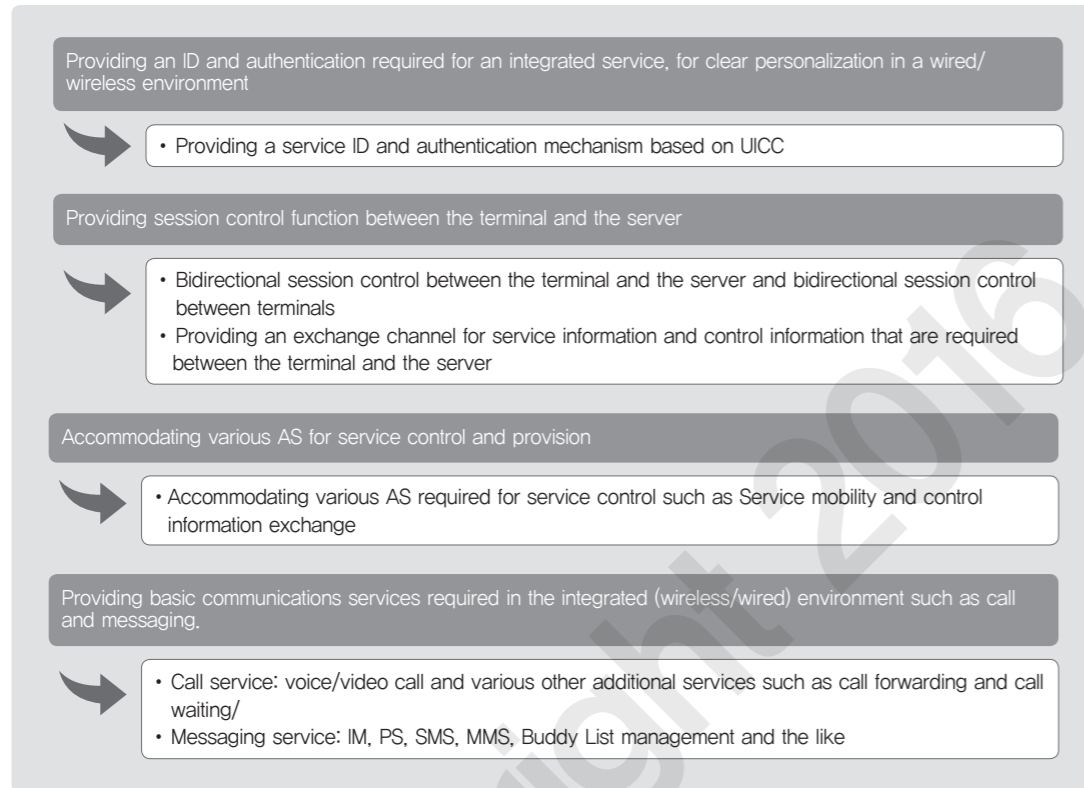
<Figure 90> IMS's centralized session control [10]

④ Role of IMS in convergence environment

- To build a structure for the ID and authentication management in the convergence environment
- To provide bidirectional channels for service control
- To make service session connections via independent service infrastructure



<Figure 91> Convergence service using IMS [10]

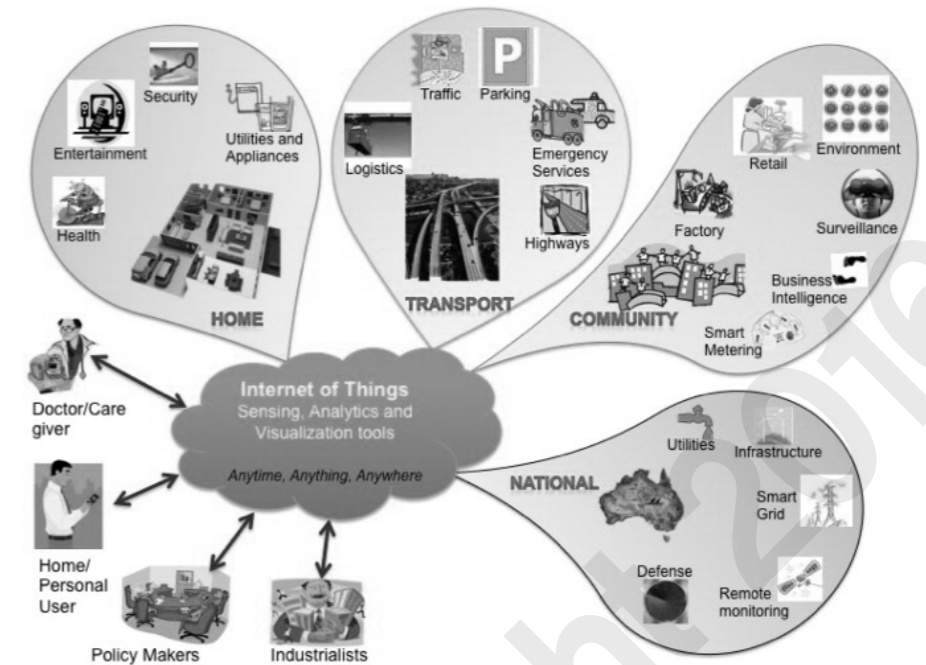


〈Figure 92〉 Role of IMS for convergence [10]

## 04 Network Technologies for IoT (Internet of Things)

### Introduction to IoT

IoT (Internet of Things) refers to equip various things with sensors and functions that can enable communications under the purpose of sensing, networking, and information processing in a mutually collaborative way without any human involvement. In addition, IoT means to allow everything, be it a human or a thing, can exchange information thanks to the evolution of technologies: ubiquitous technology that enables users to gain access to information network at any time and at any place; and the Machine To Machine technology that makes intelligent communications possible between things.



〈Figure 93〉 How IoT can be applied

### Trend of IoT Standardization

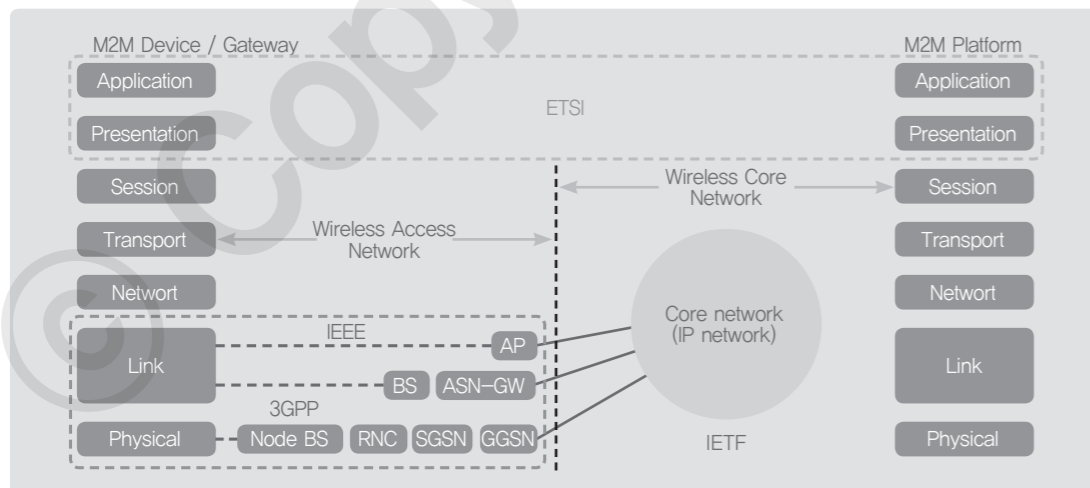
IoT standards, especially on the mobile communications, were mainly driven by the 3GPP and ETSI. However, oneM2M started to lead the standardization works from 2012. The Open Mobile Alliance (OMA), which was designed to manage a myriad of devices, created the LWM2M (Lightweight M2M) protocol and suggested a client-server based communications method. The most up-to-date LWM2M, which uses CoAP (Constrained Application Protocol) and HTTP for the data transfer, was released by oneM2M.

〈Table 79〉 Trend of IoT standardization

Standardization institute	Details
3GPP MTC	<ul style="list-style-type: none"> <li>• Started to work on standardization in 2008 to enhance the 3GPP networks and systems under the name of MTC</li> <li>• The scope of standardization covers M2M/IoT devices, wireless network sections, and mobile networks comprising core networks. There are four TSGs (Technical Specification Group) which create the standards respectively.</li> <li>• SA: Service &amp; System Aspects</li> <li>• CT: CN and Terminal</li> <li>• RAN: Radio Access Network</li> <li>• GERAN: GSM EDGE Radio Access Network</li> </ul>

<p>ETSI TC M2M</p>	<ul style="list-style-type: none"> <li>Started the standardization of machine to machine communications based on the service use cases in 2009 under the name of TC M2M</li> <li>The scope of standardization covers overall components end-to-end, including users and service providers.</li> <li>The M2M standards, related to the mobile communications, are made in cooperation with the 3GPP</li> </ul>
<p>oneM2M</p>	<ul style="list-style-type: none"> <li>A standardization of common platforms that can support smart phone, smart cars, and various other IoT services.</li> <li>Seven regional standardization institutes TTA (Korea), TTA, ATIS (US), ETSI (Europe), CCSA (China), TTC and ARIB (Japan), and major companies across the world are participating via the standardization institutes.</li> <li>The recently released Rel.1 includes CoAP and HTTP as its standard protocols.</li> </ul>
<p>ITU-T IoT-GSI/ ITU-T JCA-IoT</p>	<ul style="list-style-type: none"> <li>As a part of the ITU-T, the IoT-GSI(Internet of Things Global Standards Initiative) is responsible for IoT related standardization planning.</li> <li>The JCA-IoT(Joint Coordination Activity on Internet of Things) developed the JCA-IoT Roadmap 3.0 and is working in coordination with other standardization institutes.</li> <li>SG2(Study Group 2) is responsible for operational aspects of service provision and telecommunications management; SG11 for signaling requirements, protocols, and test specifications; SG 16 for multimedia coding, systems, and applications; SG3 for economic and policy issues; SG9 for broadband cable and TV; SG13 for future networks including cloud computing, mobile, and next-generation networks; and SG17 for security.</li> </ul>

<Figure 94> shows institute-level and protocol layer-level information regarding the scope of standardization.



<Figure 94> IoT standardization scope of each institute [10]

## Core Technology of IoT

The core technologies of IoT cover the areas of sensing, wireless/wired communications, network infrastructure, IoT service, and interface.

### ① Sensing technology

The information we get from machines and objects are collected via sensors. A wide range of sensors are embedded within devices so as to acquire the information from the environment where the devices are located, be it temperature, humidity, heat, illumination, and ultrasonic wave. Those sensors should be optimized to consume the least amount of electricity so that they can work for a long period of time. There is a development trend of sensors: physical sensors are replaced by smart sensors which have a standardized interface and information processing power. In addition, those smart sensors have so called virtual sensing function; extracting certain information from the data those sensors already acquired. As such, sensors come to have more roles to play. This means a wide range of sensors should be covered with convenient control functions and interactive communications. Hence, a market for the hardware platform, especially open source hardware, is growing.

### ② Wired/wireless communications and network infrastructure technology

Generally, wired technology such as Ethernet or PLC (Power Line Communication) can be used for machines to gain access to the Internet. However, mobile communications can be more effective, considering its easiness of installation and convenience of mobility; such as WLAN, Bluetooth, ZigBee, UWB (for near field communications) and 3G, LTE, and the like. In addition, there are new and emerging technologies for the sensor networking.

- BLE (Bluetooth Low Energy): also called Bluetooth Smart, not compatible with the existing Bluetooth, but compatible with the Bluetooth 4.0, a low power near filed wireless communication standard
- Z-Wave: Using the intelligent mesh network topology, but not having a master node, a protocol for the devices that require low power and low bandwidth

### ③ IoT service and interface technology

In order to automatically analyze and share a lot of data collected by the sensors, we need technologies such as ontology-based Semantic Web, cloud computing for large-scale distributed processing, and Open API for various service accesses.

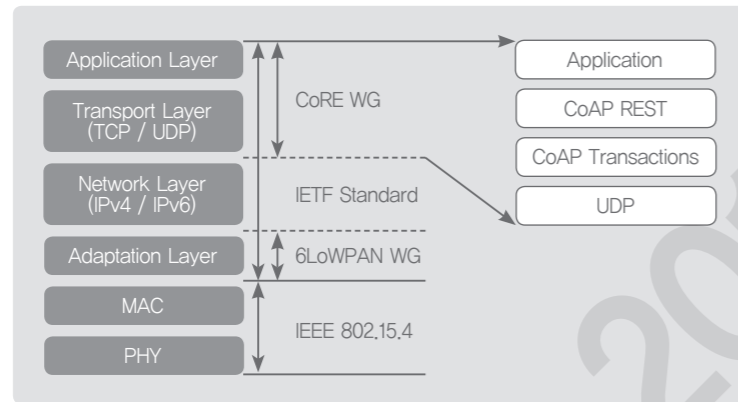
## Major Protocols for IoT

The biggest requirement for IoT protocol is 'light weight', and the generally used protocols can be favorite ones for IoT thanks to their compatibility and expandability. In particular, technologies such as CoAP and MQTT have unique characteristics that make them suitable for small device connection to the Internet for data transfer. Hence, these two are suggested as a communications protocol for various architectures.

### ① CoAP

CoAP (Constrained Application Protocol) is a light way application layer protocol that was developed by the IETF CORE working group for the machine to machine communications. Even though the protocol is supposed to use

the UDP transport layer (located above the IP layer), the design is independent from the lower layers and the protocol can be used in the network and transport layers. To reduce the load on the end point, binary encoding was employed to make the message smaller and to make the encoding and decoding easier.

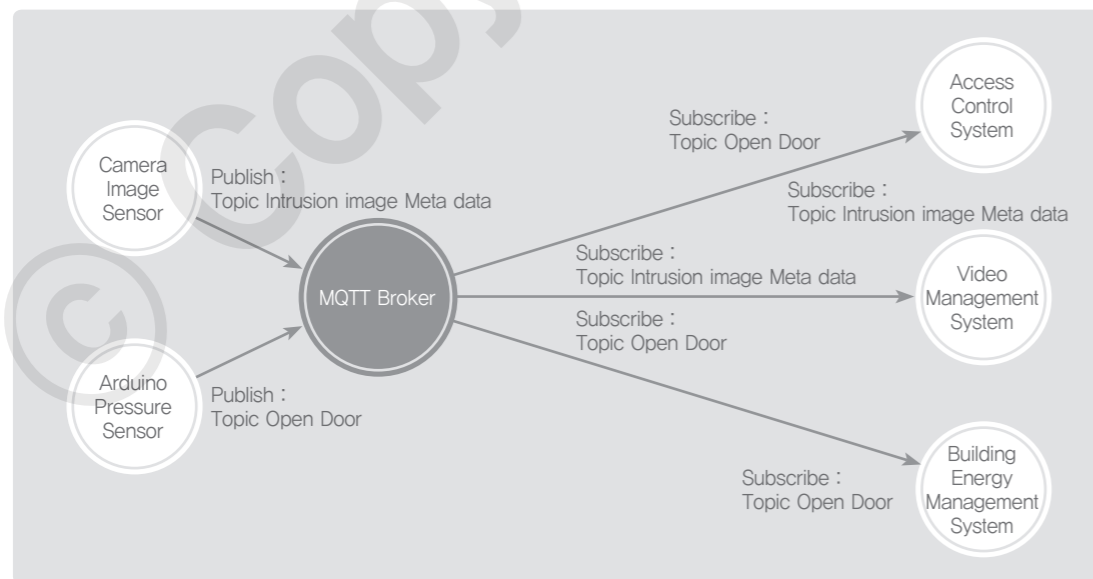


<Figure 95> Conceptual diagram for CoAP

The reason CoAP is drawing an attention is because there are increasing number of cases where ZigBee and the like are favored choices for the communications with the endpoints, rather than using fast and stable methods such as the Ethernet or Wi-Fi.

② MQTT

MQTT (Message Queue Telemetry Transport) is a "Publish-Subscribe" based light weight and low speed messaging protocol which can be used on an unreliable and high latency networks.



<Figure 96> Use case of MQTT

MQTT is a model for flexible data transmission between multiple endpoints and multiple servers, which publish and subscribe topics. <Figure 96> describes how the information acquired from Arduino's pressure sensor can be delivered by publishing the 'topic door open' to access the control system, video management system, and building energy management system and how the metadata topic generated from the camera's image sensor can be delivered to the other system by using MQTT.

Protocols that are based on CoAP and HTTP are leading the trend in many standardization institutes, but still there are architectures adopting MQTT and the REST protocol. Meanwhile, the OASIS is continuously updating the standards for MQTT.

<Table 80> MQTT vs. CoAP

Classification	MQTT	CoAP
Purpose	Messaging protocol for IoT	Messaging protocol for IoT
Topology	N:M	1:1
Model	Broker and multiple client	Server-client
Mechanism of Action	Publish and subscribe	Request and response
Information	Event	Status information
Protocol	Mainly TCP	Mainly UDP
Standards	OASIS standards	IETF CoRE standards

### 05 Software Based Network

Software Defined Network (SDN) and Network Function Virtualization (NFV) must be the two technologies that are getting the hottest attention in the networking industry. In the past, a few hardware manufacturers dominated the global market but it will be changed thanks to the openness and virtualization trend in networking. This means there will be a growing market potential for 3rd party software/hardware companies. In Korea, a task for 'research and standardization of smart internet' was launched in 2012 with a specific focus on the SDN and NFV. ETRI and many other participants are making efforts to enhance national technological capability in the SDN. TTA's PG220 and the Smart Internet Technology Standardization Council are working hard to develop a Korean Standard and to make the Korean Standard a part of the international standards.

#### Limitation of Conventional Communications Environment and Paradigm Shift

① Changes in traffic pattern

The existing applications generate a huge volume of traffic because they go through various application servers

and DBs before delivering the actual data to users. Such a limited client-server communications is now replaced with the system-to-system access environment.

② Boom of virtualization technology

The number of servers connected to the network increased significantly because of virtualization technology, which fundamentally changed the existing assumption about the physical location of hosts.

③ Increasing complexity of network

The network structure is becoming more complex as a lot of computers rely on a set of discrete protocols.

④ Difficulty in network design and management

In the initial stage of network design, the oversubscription method has been used, considering various factors such as traffic pattern and concurrency. However, the traffic pattern is dynamically changing these days, making it more difficult to predict the traffic volume.

⑤ Mounting reliance on hardware manufacturers

As the market is driven by major manufacturers, especially in the core networking, it is difficult to introduce a new service or technology.

**SDN (Software Defined Network)**

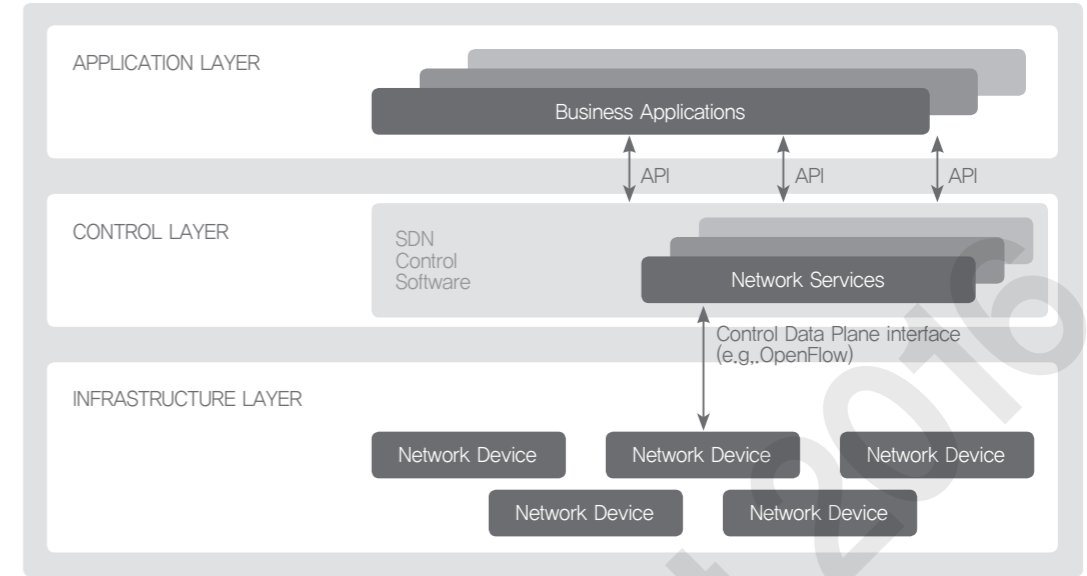
SDN is a next generation technology that can handle the route configuration/control and complex operation of networking with software programming. The SDN started to draw attention mainly because of the **OpenFlow** of ONF. In addition, other international standardization institutes such as the IETF and the ITU-T started to develop standards for the core technologies of SDN in earnest. Therefore, there is a severe competition among rival companies to secure dominance in the SDN technology standards.

① Beginning of SDN

SDN's beginning goes back to Oct of 2010 when Stanford University hosted the 'Open Network Summit', an expert conference on the SDN and OpenFlow. The SDN and OpenFlow can be defined as a technology that enables programming-level configuration and manipulation of network just like any other computer program works. This innovative communications technology is drawing a lot of attention. In March 2011, the **Open Networking Foundation (ONF)** was established to facilitate the introduction and to develop standards of the SDN and OpenFlow.

② How SDN works?

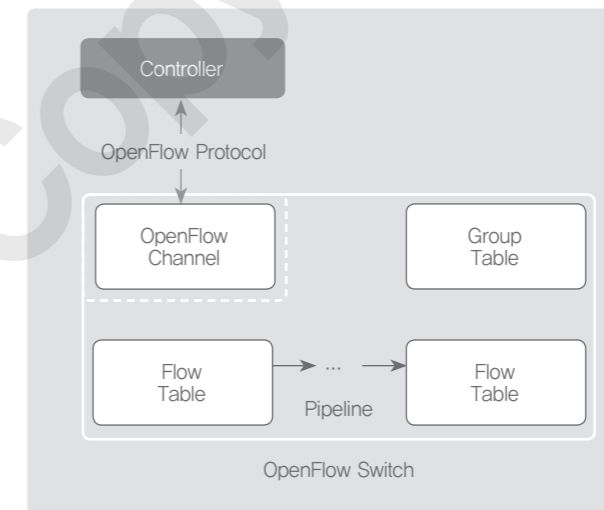
⟨Figure 97⟩ shows how network programming is possible: the **Control Plane** (to tell the network what should be moved to where) and the **Data Plane** (to send the packet to the destination) are separated. The foundation for the SDN are switching devices which use the industry standard control protocols like the OpenFlow and is programmable through the SDN controller.



⟨Figure 97⟩ SDN architecture [15]

③ OpenFlow technology

OpenFlow, as shown in ⟨Figure 98⟩, separates the packet control functions from the transfer functions, and controls networks based on programming. OpenFlow is composed of a **Controller** and a **Switch**, whereas the controller gives an order to the switch, and the switch takes care of activities related to data flow such as packet delivery or correction.

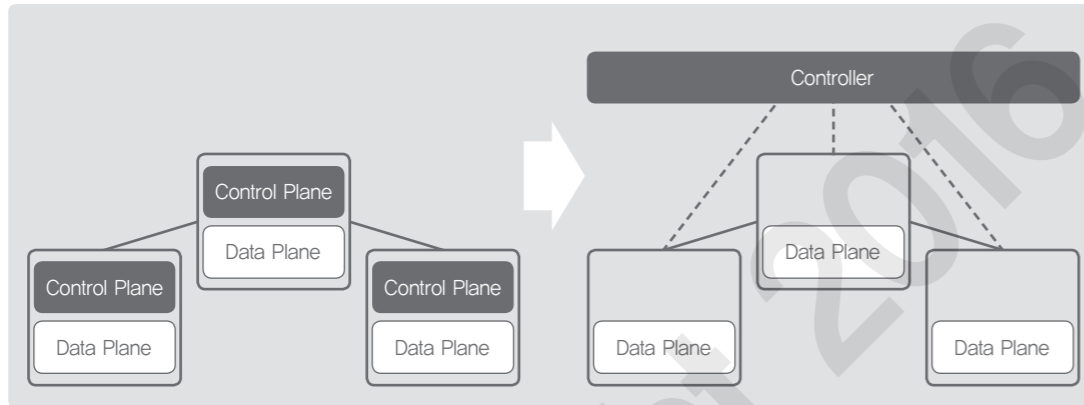


⟨Figure 98⟩ Main components of OpenFlow switch [16]

④ How SDN is applied?

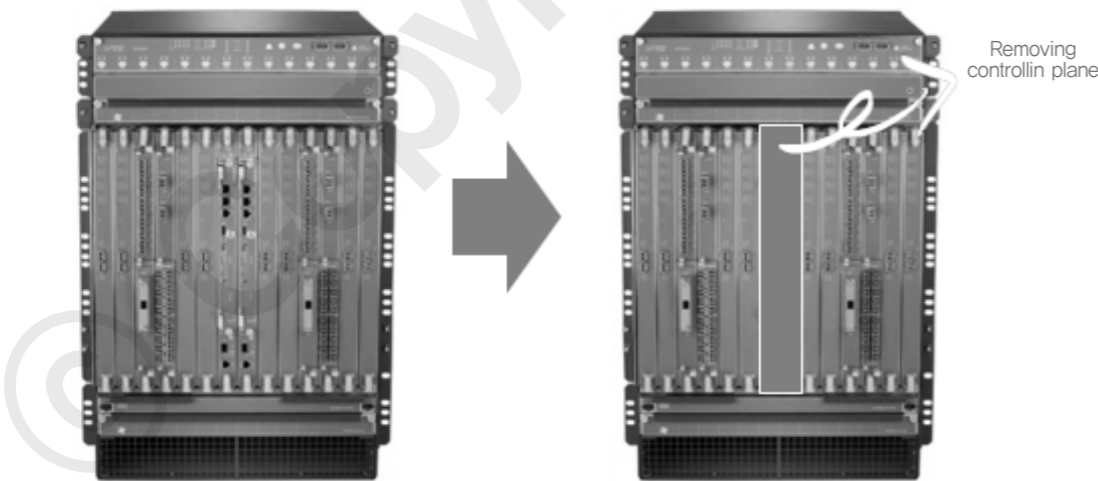
Existing network devices had control planes and data planes within a single box as shown in (Figure 99). However, a controller can work as a control plane for each of the physical devices and the physical device will have a data plane only.

Here, the OpenFlow works as a standard of the interface technology for the SDN operation,



(Figure 99) Network structure before/after SDN

It means to remove a processing module out of a mid-large size router and to leave only switching functions as shown in (Figure 100), which means the router will simply operate, only following the order from the controller.



(Figure 100) Removing controlling plane from mid-size router

NFV (Network Function Virtualization)

NFV is emerging as a buzz word thanks to the high interest from network operators. In addition, the NFV groups are preparing a set of standards for communications service providers and hardware manufacturers, and they are working on rounds of proof of concept to validate the feasibility of this technology.

① Beginning of NFV

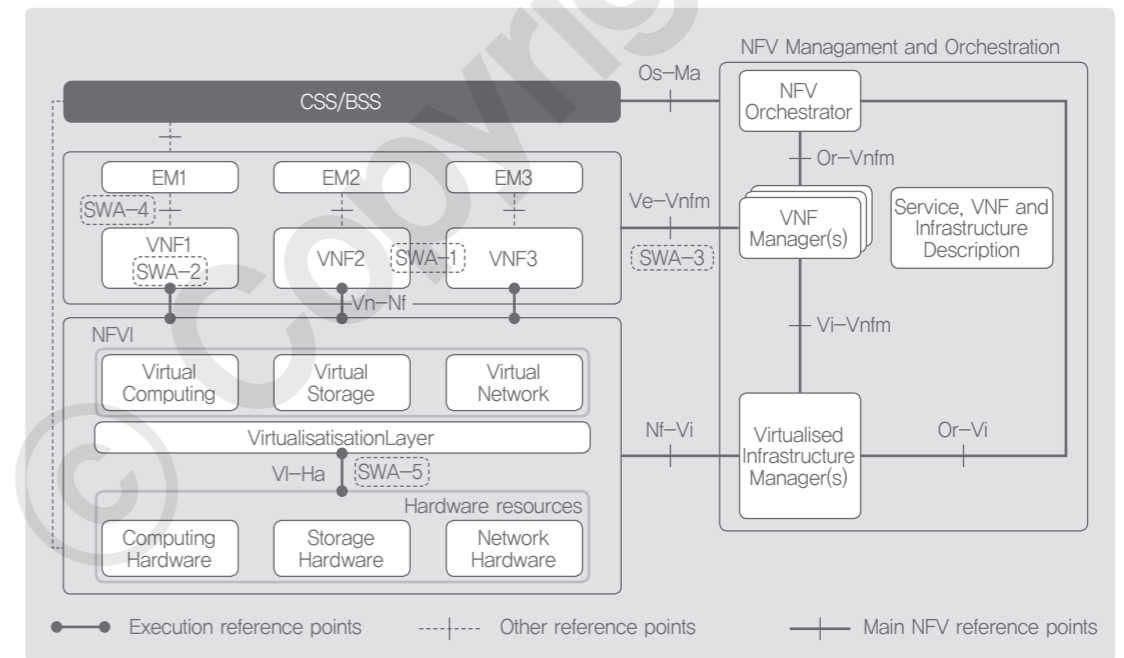
With the rise of various services and increase in the network speed, current internet service providers face serious issues about the hardware space and power supply. In addition, as the life cycle of network devices becomes shorter, it has become more difficult to achieve return on investment in a sustainable way. Therefore, the NFV was introduced to address these issues with virtualization technology.

② Basic concept of NFV

NFV is running on the high-performance x86 platforms, and a user can activate network functions if and when necessary. A VM (Virtual Machine) or Service Profile is created to realize the virtualization on the network by using x86.

③ Structure of NFV architecture framework

As seen in (Table 81), the NFV architecture framework is composed of three functional groups: VNF group, NFV infrastructure, Management and Orchestration,



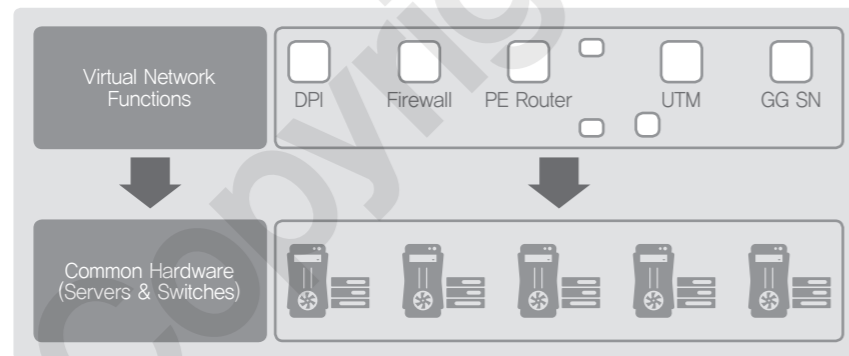
(Figure 101) NFV architecture framework (Five types of VNF interface) [17]

<Table 81> Components of the NFV architecture framework

Classification	Details
VNFs	<ul style="list-style-type: none"> <li>Virtual Network Functions</li> <li>Software implementations of network functions that can support various application programs</li> </ul>
NFVI	<ul style="list-style-type: none"> <li>NFV Infrastructure</li> <li>Providing support for virtualization and VNF operation, and providing computing, storage, physical hardware resources to support networking functions</li> </ul>
Management & Orchestration	<ul style="list-style-type: none"> <li>Hardware and software resource management, delivery, and VFN management.</li> </ul>

④ Example of NFV implementation

NFV literally means to achieve functions of networking devices through the virtualization, which also means NFV does not require a specific device. As shown in <Figure 102>, a generally used high-performance x 86 server platforms can be used for virtualizing the functions which used to be provided by networking devices.

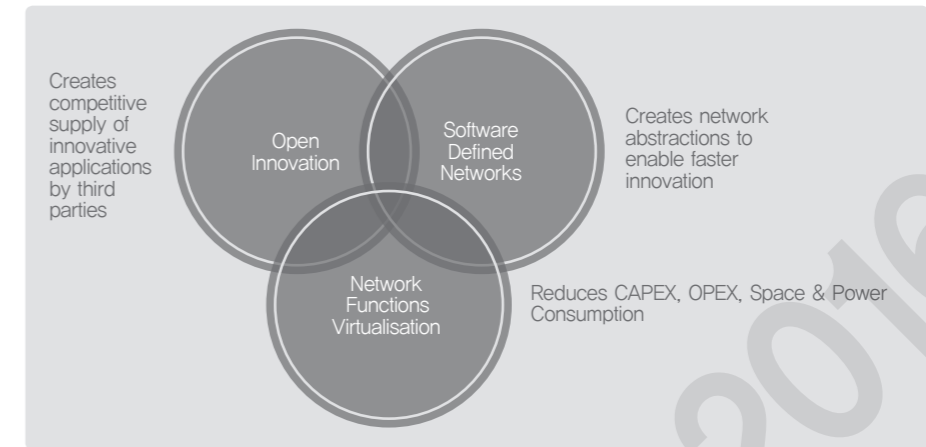


<Figure 102> NFV implementation case

SDN & NFV

At the initial stage, the SDN was developed mainly by researchers and data center architects, and NFV was developed by the ISPs (Internet Service Provider). However, SDN and NFV are mutually complementary and it is a trend that ISPs are utilizing these two technologies on their networks.

① Relationship between NFV and SDN



<Figure 103> Relationship between NFV and SDN [18]

SDN can make a network virtualized in order to facilitate faster changes in the network and the NFV can save the CAPEX (Capital expenditures), OPEX (Operating expenditures), space, and resources. In addition, the SDN and the NFV are mutually complementary but can be implemented independently as well.

② Comparison of NFV and SDN

<Table 82> Comparison of NFV and SDN

Item	SDN	NFV
Purpose	Software approach to realize network functions: separation of control and data planes and centralized network control	Using VM (Virtual Machine) and high-performance x86 servers to realize the functions used to be provided by networking devices
User Base	Started from campus, data center, and cloud, but started to be used by telecommunications service providers	Started with a specific target: network devices of telecommunications service providers.
Target	Networking devices such as mid-large size routers and switches	Networking devices such as mid-large size routers and switches
Function	Cloud orchestration and networking	Router, firewall, gateway, CDN, WAN optimization, definite SLA
Protocol	Mainly OpenFlow	None
Leading institute	Open Networking Foundation (ONF)	ETSI NFV Working Group

## Example Question

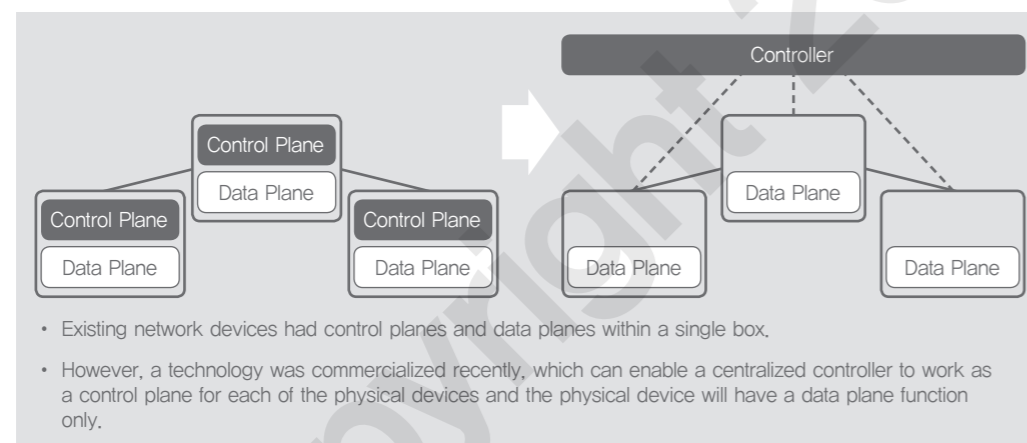
### Question type

Short-answer question

### Question

Please refer to the following images and explanations to answer the question.

1. What is the next generation networking technology that can handle the network routing configuration/control and the complex operation activities with software programming?
2. What is the ONF (Open Networking Foundation)'s control data plane interface technology or protocol? (Packet forwarding function and controller function of a networking device is separated into a standard interface to grant openness.)



### Intent of the question

To understand the up-to-date trend of networking

### Answer and explanation

1. SDN (Software Defined Network)
2. OpenFlow

## Related E-learning Contents

- Lecture 8 Multimedia Technology