A Brief History of TCP/IP

The very first *Request for Comments (RFC)* was published in April 1969, which paved the way for today's Internet and its protocols. Each of these protocols is specified in the multitude of RFCs, which are observed, maintained, sanctioned, filed, and stored by the *Internet Engineering Task Force (IETF)*.

TCP first came on the scene in 1974. In 1978, it was divided into two distinct protocols, **TCP** and **IP**, and finally documented into an RFC in 1980. Then, in 1983, TCP/IP replaced the *Network Control Protocol (NCP)* and was authorized as the official means of data transport for anything connecting to *ARPAnet*. ARPAnet was the Internet's ancestor, created by *ARPA*, the *DoD's Advanced Research Projects Agency*, again, way back in 1969 in reaction to the Soviets' launching of Sputnik. ARPA was soon redubbed DARPA, and it was divided into ARPAnet and MILNET (also in 1983); both were finally dissolved in 1990. But contrary to what you might think, most of the development work on **TCP/IP** happened at *UC Berkeley* in Northern California, where a group of scientists was simultaneously working on the *Berkeley version of Unix*, which soon became known as the *BSD*, or the *Berkeley Software Distribution series of Unix versions*. Of course, because TCP/ IP worked so well, it was packaged into subsequent releases of BSD Unix and offered to other universities and institutions if they bought the distribution tape. Basically, BSD Unix bundled with TCP/IP began as shareware in the world of academia and, as a result, became the basis of the huge success and exponential growth of today's Internet as well as smaller, private, and corporate intranets.

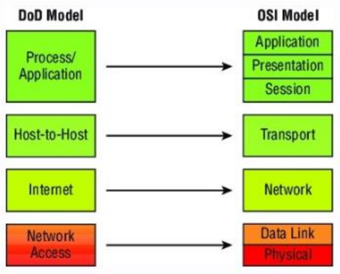
> Q: What is TCP/IP?
> A: TCP/IP is short for Transmission Control Protocol / Internet Protocol. This is a set of protocol layers that is designed to make data exchange possible on different types of computer networks, also known as heterogeneous network.
>
> - Interview Q&A

TCP/IP and the DoD Model

The Department of Defense created TCP/IP to ensure and preserve data integrity. The DoD model is a condensed version of the OSI model and only has four layers that are:

- Process/Application layer
- Host-to-Host layer
- Internet layer
- Network Access Layer

The DoD and OSI models are identical in design and concept and have similar functions in similar layers.



The DoD and OSI Models

Tip:

- When the different protocols in the IP stack are discussed, two layers of the OSI and DoD models are interchangeable. In other words, the Internet layer and the Network layer describe the same thing, as do the Host-to-Host layer and the Transport layer. The other two layers of the DoD model, Process/Application and Network Access are composed of multiple layers of the OSI model.

1. Process/Application layer

The Application layer of the DoD model is equivalent to the upper three layers of the OSI model, i.e., Session layer, Presentation layer, and Application layer. The Process/Application layer of the DoD model provides the following capabilities:

- Enable applications to communicate with each other.
- Provides access to the services that operate at the lower layers of the DoD model.
- It contains a protocol that implements user-level functions such as mail delivery, file transfer, and remote login.

2. Host-to-Host layer

A host-to-host layer of the DoD model performs the same functions as the Transport Layer of the OSI reference model. It handles issues such as flow control, reliable end-to-end communication, and ensuring error-free delivery of the data. Protocols that operate on the Host-to-Host layer are TCP and UDP.

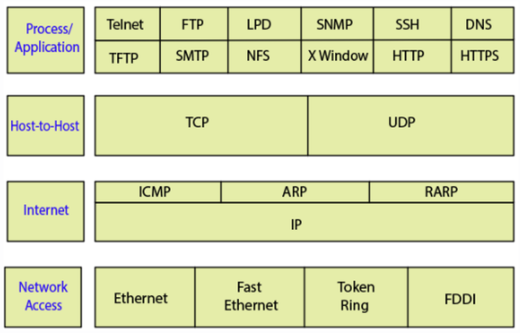
3. Internet layer

Internet layer of the DoD model performs the same functions as the Network layer of the OSI reference model. It handles the packaging, addressing, and routing of packets among multiple networks. This layer also establishes a connection between two computers to exchange the data.

4. Network Access Layer

The Network Access layer of the DoD model is equivalent to the lower two layers (Data Link, and Physical) of the OSI model. The hardware connected to Network access layer are:

- Network medium: Cables like coaxial, twisted pair. Today, mostly, we use a wireless medium such as Bluetooth, WI-FI.
- Network Interface Card (NIC) has two types of addresses.
  - MAC Address- It is a 48 bits physical address.
  - IP Address – It is a 32 bits logical address.



The TCP/IP protocol suite

The Process/Application Layer Protocols

Telnet (TCP 23): Telnet allows a user on a remote client machine, called the *Telnet Client*, to access the resources of another machine. The drawback of Telnet is that there are no encryption techniques available within the Telnet Protocols, so everything must be sent in the cleartext.

FTP (TCP 20, 21): File Transfer Protocol lets us transfer files, and it can accomplish this between any two machines using it. FTP allows access to both directories and files and can accomplish certain types of directory operations, such as relocating into different ones. FTP functions are limited to listing and manipulating directories, typing file contents, and copying files between hosts.

SFTP (TCP 22): Secure File Transfer Protocol is used when you need to transfer files over an encrypted connection. It uses an SSH session, which encrypts the connection, and SSH uses port 22, hence the port 22 is used for SFTP. Apart from the secure part, it's used just as FTP is.

TFTP (UDP 69): Trivial File Transfer Protocol (TFTP) is the stripped-down, stock version of FTP. TFTP is fast and so easy to use. TFTP doesn't offer the abundance of functions that FTP does because it has no directory-browsing abilities, meaning that it can only send and receive files. There's no authentication as with FTP, so it's insecure.

SMTP (TCP 25): Simple Mail Transfer Protocol detects the spooled mails and proceeds to deliver them to their destination. SMTP is used to send mail; POP3 is used to receive mail.

POP (TCP 110): Post Office Protocol gives us a storage facility for incoming mail, and the latest version is called POP3. How this protocol works is when a client device connects to a POP3 server, messages addressed to that client are released for download. It doesn't allow messages to be downloaded selectively, but once they are, the client-server interaction ends, and you can delete and tweak your messages locally at will. A newer standard, IMAP, is being used more and more in place of POP3.

IMAP (TCP 143): Because Internet Message Access Protocol (IMAP) makes it so you get control over how you download your mail, with it, you also gain some much-needed security. It lets you peek at the message header or download just a part of a message. With it, you can choose to store messages on the email server hierarchically and link to documents and user groups, too. IMAP even gives you search commands to use to hunt for messages based on their subject, header, or content. As you can imagine, it has some serious authentication features—it supports the *Kerberos* authentication scheme that MIT developed. IMAP4 is the current version.

RDP (TCP 3389): Remote Desktop Protocol is a proprietary protocol developed by Microsoft. It allows you to connect to another computer and run programs. RDP operates somewhat like *Telnet*, except instead of getting a command-line prompt as you do with Telnet, you get the actual graphical user interface (GUI) of the remote computer. Clients exist for most versions of Windows, and Macs now come with a preinstalled RDP client. Microsoft currently calls its official RDP server software *Remote Desktop Services*. Microsoft's official client software is currently referred to as *Remote Desktop Connection*. RDP is an excellent tool for remote clients, allowing them to connect to their work computer from home, for example, and get their email or perform work on other applications without running or installing any of the software on their home computer.

TLS/SSL (TCP 995/465): Both Transport Layer Security and its forerunner, Secure Sockets Layer, are cryptographic protocols that are useful for enabling secure online data-transfer activities like browsing the Web, instant messaging, Internet faxing, and so on. They're so similar. They both use X.509 certificates and asymmetric cryptography to authenticate to the host they are communicating with and to exchange a key. This key is then used to encrypt data flowing between the hosts. This allows for data/message confidentiality, message integrity, and message authentication.

SIP (VoIP) (TCP or UDP 5060/TCP 5061): Session Initiation Protocol is a hugely popular signaling protocol used to construct and deconstruct multimedia communication sessions for many things like voice and video calls, videoconferencing, streaming multimedia distribution, instant messaging, presence information, and online games over the Internet.

RTP (VoIP) (UDP 5004/TCP 5005): Real-time Transport Protocol describes a packet-formatting standard for delivering audio and video over the Internet. It's commonly employed for streaming media, videoconferencing, and push-to-talk systems—all things that make it a de facto standard in Voice over IP (VoIP) industries.

MGCP (Multimedia) (TCP 2427/2727): Media Gateway Control Protocol is a standard protocol for handling the signaling and session management needed during a multimedia conference. The protocol defines a means of communication between a media gateway, which converts data from the format required for a circuit-switched network to that required for a packet-switched network, and the media gateway controller. MGCP can be used to set up, maintain, and terminate calls between multiple endpoints.

H.323 (Video) (TCP 1720): H.323 is a protocol that provides a standard for video on an IP network that defines how real-time audio, video, and data information is transmitted. This standard provides signaling, multimedia, and bandwidth control mechanisms. H.323 uses the RTP standard for communication.

SNMP (UDP 161/TCP 25): Simple Network Management Protocol collects and manipulates valuable network information. It gathers data by polling the devices on the network from a management station at fixed or random intervals, requiring them to disclose certain information. This protocol can also stand as a *watchdog* over the network, quickly notifying managers of any sudden turn of events. The network watchdogs are called *agents*, and when aberrations occur, agents send an alert called a trap to the management station. Besides, SNMP can help simplify the process of setting up a network as well as the administration of your entire internetwork.

SSH (TCP 22): Secure Shell Protocol sets up a secure session that's similar to Telnet over a standard TCP/IP connection and is employed for doing things like logging into systems, running programs on remote systems and moving file from one system to another system. And it does all this while maintaining an encrypted connection.

HTTP (TCP 80): Hypertext Transfer Protocol is used to manage communications between web browsers and web servers and opens the right resources when you click a link, wherever that resource may reside.

HTTPS (TCP 443): Hypertext Transfer Protocol Secure is also known as a secure HTTP. It uses the Secure Socket Layer (SSL). It is the secure version of the HTTP that provides some security tools for keeping transactions between web browsers and servers secure. It is what our browser needs to fill out forms, sign in, authenticate, and encrypt an HTTP message when we do things like making an online reservation, accessing online banking, or buying something online.

NTP (UDP 123): Network Time Protocol is used to synchronize the clocks on our computer to one standard time source. This protocol works by synchronizing devices to ensure that all the computers on a given network agree on the time.

LDAP (TCP 389): The Lightweight Directory Access Protocol, is a mature, flexible, and well supported standards-based mechanism for interacting with directory servers. It's often used for authentication and storing information about users, groups, and applications, but an LDAP directory server is a fairly general-purpose data store and can be used in a wide variety of applications.

IGMP: Internet Group Management Protocol is the TCP/IP protocol used for managing IP multicast sessions. It accomplishes this by sending out unique IGMP messages over the network to reveal the multicast-group landscape and to find out which hosts belong to which multicast group. The host machines in an IP network also use IGMP messages to become members of a group and to quit the group, too. IGMP messages come in seriously handy for tracking group memberships as well as active multicast streams. IGMP works at the Network layer and doesn't use port numbers.

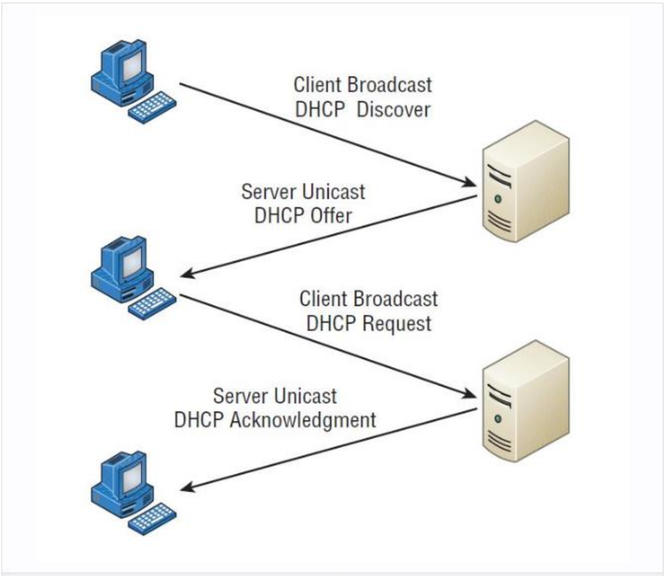
NetBIOS (TCP and UDP 137–139): Network Basic Input/Output System works only in the upper layers of the OSI model and allows for an interface on separate computers to communicate over a network. It was first created in the early 1980s to work on an IBM LAN and was proprietary. Microsoft and Novell both created a NetBIOS implementation to allow their hosts to communicate to their servers, but Microsoft's version became the de facto version.

SMB (TCP 445): Server Message Block is used for sharing access to files and printers and other communications between hosts on a Microsoft Windows network. SMB can run on UDP port 137 and 138 and TCP port 137 and 139 using NetBIOS.

DNS (TCP and UDP 53): Domain Name Service resolves hostnames, internet names such as www.clarusway.com (Fully Qualified Domain Name). A *Fully Qualified Domain Name (FQDN)* is a hierarchy that can logically locate a system based on its domain identifier. An IP address identifies hosts on a network and the Internet as well, but DNS was designed to make our lives easier. Think about this: What would happen if you wanted to move your web page to a different service provider? The IP address would change and no one would know what the new one was. DNS allows you to use a domain name to specify an IP address. You can change the IP address as often as you want and no one will know the difference.

DHCP (UDP 67/68): Dynamic Host Configuration Protocol assigns IP Address to hosts. It allows for easier administration and works well in small to very large network environments. Many types of hardware can be used as a DHCP Server, including a Cisco Router. There is a lot of information a DHCP server can provide to a host when the host is requesting an IP address from DHCP Server, here is the list of some common types of information a DHCP server can provide:

- IP Address
- Subnet Mask
- Domain Name
- Default Gateways
- DNS Server Address
- WINS Server Address



The DORA process

The following is the four-step process (sometimes known as the DORA process) a client takes to receive an IP address from a DHCP server:

1. The DHCP client broadcasts a DHCP Discover message looking for a DHCP server (port 67).
2. The DHCP server that received the DHCP Discover message sends a unicast DHCP Offer message back to the host.
3. The client then broadcasts to the server a DHCP Request message asking for the offered IP address and possibly other information.
4. The server finalizes the exchange with a unicast DHCP Acknowledgment message.

The Host-to-Host Layer Protocols - TCP

The main purpose of the **Host-to-Host layer** is to *shield the upper-layer applications* from the complexities of the network. This layer says to the upper layer, "Just give me your data stream, with any instructions, and I'll begin the process of getting your information ready to send."

The following sections describe the two protocols at this layer:

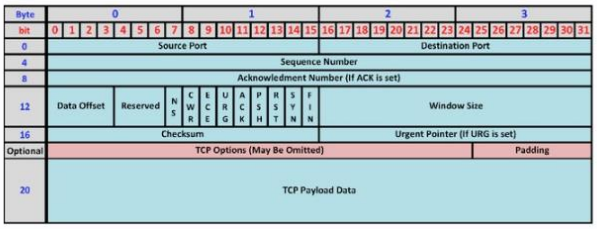
- Transmission Control Protocol (TCP)
- User Datagram Protocol (UDP)

Transmission Control Protocol (TCP) takes large blocks of information from an application and breaks them into segments. It numbers and sequences each segment so that the destination's TCP process can put the segments back into the order the application intended. After these segments are sent, TCP (on the transmitting host) waits for an acknowledgment from the receiving end's TCP process, retransmitting those segments that aren't acknowledged.

Remember that in reliable transport operation, a device that wants to transmit sets up a connection-oriented communication with a remote device by creating a session. The transmitting device first establishes a connection-oriented session with its peer system; that session is called a **call setup** or a **three-way handshake**. Data is then transferred, and when the transfer is complete, a call termination takes place to tear down the virtual circuit.

TCP is a *full-duplex, connection-oriented, reliable*, and *accurate protocol*, but establishing all these terms and conditions, in addition to error checking, is no small task. TCP is very complicated and costly in terms of network overhead.

When the Internet layer receives the data stream, it routes the segments as packets through an internetwork. The segments are handed to the receiving host's Host-to-Host layer protocol, which rebuilds the data stream to hand to the upper-layer protocols. The below figure shows the TCP segment format.



The TCP Segment

The Host-to-Host Layer Protocols - UDP

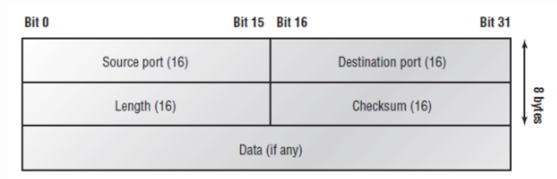
If to compare **User Datagram Protocol (UDP)** with TCP, the former is basically the scaled-down economy model that's sometimes referred to as a thin protocol. A thin protocol doesn't take up much bandwidth on a network. UDP doesn't offer all the advantages of TCP either, but it does do a fabulous job of transporting information that doesn't require reliable delivery—and it does so using far fewer network resources.

There are some situations in which it would definitely be wise for developers to opt for UDP rather than TCP. Remember the watchdog SNMP up there at the Process/Application layer. SNMP monitors the network, sending intermittent messages and a fairly steady flow of status updates and alerts, especially when running on a large network. The cost in overhead to establish, maintain, and close a TCP connection for each one of those little messages would reduce what would be an otherwise healthy, efficient network.

Another circumstance calling for UDP over TCP is when reliability is already handled at the Process/Application layer. DNS handles its own reliability issues, making the use of TCP both impractical and redundant. But ultimately, it's up to the *application developer* to decide whether to use UDP or TCP, *not the user* who wants to transfer data faster.

UDP does not sequence the segments and doesn't care in which order the segments arrive at the destination. But after that, UDP sends the segments off and forgets about them. It doesn't follow through, check up on them, or even allow for an acknowledgment of safe arrival—complete abandonment. Because of this, it's referred to as an *unreliable* protocol. This doesn't mean that UDP is ineffective, only that it doesn't handle issues of reliability. Because UDP assumes that the application will use its own reliability method, it doesn't use any. This gives an application developer a choice when running the IP stack: **TCP for reliability** or **UDP for faster transfers**.

Further, UDP doesn't create a virtual circuit, nor does it contact the destination before delivering information to it. Because of this, it's also considered a **connectionless** protocol.



The UDP Segment

Key features of TCP and UDP

The below table highlights some of the key concepts that you should keep in mind regarding these two protocols.

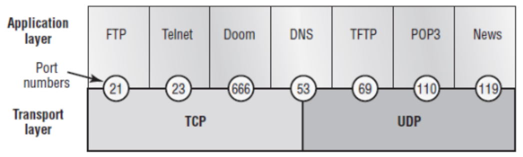
TCP	UDP
Sequenced	Unsequenced
Secure	Unsecure
Connection-oriented	Connectionless
Slow	Fast
Guaranteed transmission	No guarantee
Flow control	No flow control
Reliable	Unreliable
Virtual circuit	No virtual circuit
High overhead	Low overhead
Acknowledgment	No acknowledgment
Windowing flow control	No windowing or flow control
20 bytes header	8 bytes header

A telephone analogy could really help you understand how TCP works. Most of us know that before you speak to someone on a phone, you must first establish a connection with that person—wherever they are. This is like a virtual circuit with TCP. If you were giving someone important information during your conversation, you might say, "You know?" or ask, "Did you get that?" Saying something like this is a lot like a TCP acknowledgment—it's designed to get your verification. From time to time (especially on cell phones), people also ask, "Are you still there?" They end their conversations with a "Good-bye" of some kind, putting closure on the phone call. TCP also performs these types of functions.

Alternatively, using UDP is like sending a postcard. To do that, you don't need to contact the other party first. You simply write your message, address the postcard, and mail it. This is analogous to UDP's connectionless orientation. Because the message on the postcard is probably not a matter of life or death, you don't need an acknowledgment of its receipt. Similarly, UDP doesn't involve acknowledgments.

TCP and UDP must use **port numbers** to communicate with the upper layers because they're what keep track of different simultaneous conversations originated by or accepted by the localhost. Originating source port numbers are dynamically assigned by the source host and will usually have a value of 1024 or higher. Virtual circuits that don't use an application with a well-known port number are assigned port numbers randomly from a specific range instead. These port numbers identify the source and destination application or process in the TCP segment.

The below figure illustrates how both TCP and UDP use port numbers.



Port numbers for TCP and UDP

You just need to remember that numbers below 1024 are considered the well-known port numbers and are defined in RFC 3232. Numbers 1024 and above are used by the upper layers to set up sessions with other hosts and by TCP as the source and destination identifiers in the TCP segment.

The below table gives you a list of the typical applications used in the TCP/IP suite, their well-known port numbers, and the Transport layer protocols used by each application or process.

TCP	UDP
Telnet 23	SNMPv1/2 161
SMTP 25	TFTP 69
HTTP 80	DNS 53
FTP 20, 21	BOOTPS/DHCP 67,68
SFTP 22	NTP 123
DNS 53	
HTTPS 443	
SSH 22	
SMB 445	
POP3 110	
IMAP4 143	
RDP 3389	
SNMPv3 161	

The Internet Layer Protocols

In the DoD model, there are two main reasons for the Internet layer's existence: **routing** and **providing** *a single network interface* to the upper layers.

None of the other upper- or lower-layer protocols have any functions relating to routing—that complex and important task belongs entirely to the Internet layer. The Internet layer's second duty is to provide a single network interface to the upper-layer protocols. Without this layer, application programmers would need to write what are called hooks into every one of their applications for each different Network Access protocol. This would lead to different versions of each application—one for Ethernet, another one for Token Ring, and so on. To prevent this, IP provides one single network interface for the upper-layer protocols.

All network depends on IP. And all the other protocols at this layer, as well as all those at the upper layers, use it. Never forget that. All paths through the DoD model go through IP. The following sections describe the protocols at the Internet layer:

- Internet Protocol (IP)
- Internet Control Message Protocol (ICMP)
- Address Resolution Protocol (ARP)
- Reverse Address Resolution Protocol (RARP)

Internet Protocol

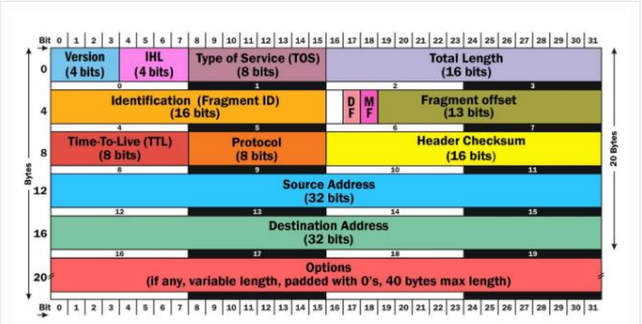
Internet Protocol (IP) is essentially the *Internet layer*. The other protocols at this layer simply exist to support it.

IP looks at each packet's destination address. Then, using a routing table, it decides where a packet is to be sent next, choosing the best path. The protocols of the *Network Access layer* at the bottom of the *DoD model* don't possess IP's enlightened scope of the entire network; they deal only with physical links (local networks).

Identifying devices on networks requires answering these two questions: Which network is it on? And what is its ID on that network? The answer to the first question is the *software address*, or *logical address*. The answer to the second question is the hardware address. All hosts on a network have a **logical ID** called an **IP address**. This is the software -or logical- address and contains valuable encoded information, simplifying the complex task of routing.

IP receives segments from the *Host-to-Host layer* and fragments them into packets if necessary. IP then reassembles packets back into segments on the receiving side. Each packet is assigned the IP address of the sender and of the recipient. Each router (Layer 3 device) that receives a packet makes routing decisions based on the packet's destination IP address.

The below figure is an IPv4 header.


Version (4 bits)
IHL (4 bits)
Type of Service (TOS) (8 bits)
Total Length (16 bits)
Identification (Fragment ID) (16 bits)
D F
M F
Fragment offset (13 bits)
Time-To-Live (TTL) (8 bits)
Protocol (8 bits)
Header Checksum (16 bits)
Source Address (32 bits)
Destination Address (32 bits)
Options (if any, variable length, padded with 0's, 40 bytes max length)
Bytes
20 Bytes
Bit

IPv4 header

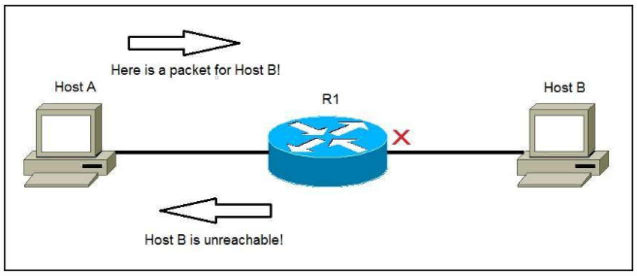
Internet Control Message Protocol

Internet Control Message Protocol (ICMP) works at the *Network layer* and is used by IP for many different services. ICMP is a management protocol and messaging service provider for IP. Its messages are carried as IP packets. ICMP packets have the following characteristics:

- They can provide hosts with information about network problems.
- They are encapsulated within IP datagrams.

The following are some common events and messages that ICMP relates to and the two most popular programs that use ICMP:

- **Destination Unreachable** - If a router can't send an IP datagram any further, it uses ICMP to send a message back to the sender, advising it of the situation. For example, the below figure shows that the Ethernet interface of the Lab B router is down.



ICMP error message is sent to the sending host from the remote router.

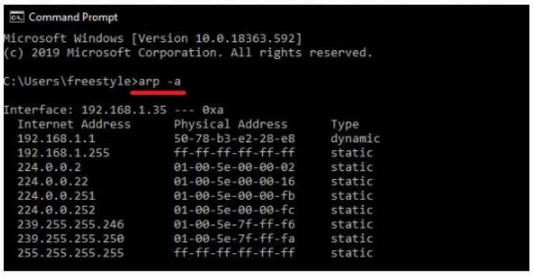
When Host A sends a packet destined for Host B, the Lab B router will send an *ICMP Destination Unreachable* message back to the sending device.

- **Buffer Full** - If a router's memory buffer for receiving incoming datagrams is full, it will use ICMP to send out this message until the congestion abates.
- **Hops** - Each IP datagram is allotted a certain number of routers, called hops, to pass through. If a datagram reaches its limit of hops before arriving at its destination, the last router to receive it deletes it. The executioner router then uses ICMP to send an obituary (mortal) message, informing the sending machine of the demise (drop) of its datagram.
- **Ping** - Ping uses ICMP echo request and reply messages to check the physical and logical connectivity of machines on an internetwork.
- **Traceroute** - Traceroute uses IP packet Time to Live time-outs to discover the path a packet takes as it traverses an internetwork.

Address Resolution Protocol (ARP)

Address Resolution Protocol (ARP) is a procedure for mapping a dynamic **Internet Protocol address (IP address)** to a permanent physical **machine address** in a local area network (LAN). The physical machine address is also known as a **Media Access Control** or **MAC address**.

The job of the ARP is essentially to translate 32-bit addresses to 48-bit addresses and vice-versa. This is necessary because in IP Version 4 (IPv4), the most common level of Internet Protocol (IP) in use today, an IP address is 32-bits long, but MAC addresses are 48-bits long.



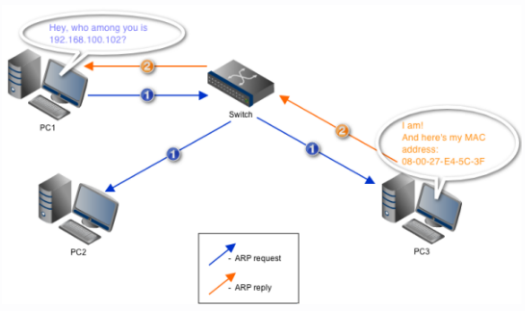
```
Command Prompt
Microsoft Windows [Version 10.0.18363.592]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\freestyle>arp -a

Interface: 192.168.1.35 --- 0xa
  Internet Address      Physical Address      Type
  192.168.1.1           50-78-b3-e2-28-e8     dynamic
  192.168.1.255         ff-ff-ff-ff-ff-ff     static
  224.0.0.2             01-00-5e-00-00-02     static
  224.0.0.22            01-00-5e-00-00-16     static
  224.0.0.251           01-00-5e-00-00-fb     static
  224.0.0.252           01-00-5e-00-00-fc     static
  239.255.255.246       01-00-5e-7f-ff-f6     static
  239.255.255.250       01-00-5e-7f-ff-fa     static
  255.255.255.255       ff-ff-ff-ff-ff-ff     static
```

ARP table

ARP works between network layers 2 and 3 of the Open Systems Interconnection model (OSI model). The MAC address exists on layer 2 of the OSI model, the data link layer, while the IP address exists on layer 3, the network layer.

In IPv6, which uses 128-bit addresses, ARP has been replaced by the **Neighbor Discovery protocol**.

When a new computer joins a LAN, it is assigned a unique IP address to use for identification and communication. When an incoming packet destined for a host machine on a particular LAN arrives at a gateway, the gateway asks the ARP program to find a MAC address that matches the IP address. A table called the ARP cache maintains a record of each IP address and its corresponding MAC address.

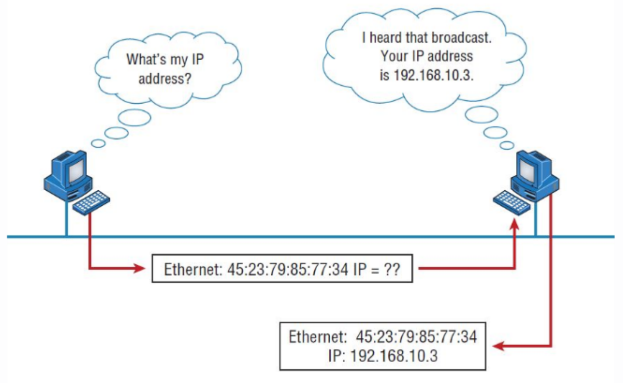

Hey, who among you is 192.168.100.102?
I am! And here's my MAC address: 08-00-27-E4-5C-3F
PC1
Switch
PC2
PC3
- ARP request
- ARP reply

ARP query

All operating systems in an IPv4 Ethernet network keep an ARP cache. Every time a host requests a MAC address in order to send a packet to another host in the LAN, it checks its ARP cache to see if the IP to MAC address translation already exists. If it does, then a new ARP request is unnecessary. If the translation does not already exist, then the request for network addresses is sent and ARP is performed.

ARP broadcasts a request packet to all the machines on the LAN and asks if any of the machines know they are using that particular IP address. When a machine recognizes the IP address as its own, it sends a reply so ARP can update the cache for future reference and proceed with the communication.

Host machines that don't know their own IP address can use the **Reverse ARP (RARP)** protocol for discovery.


What's my IP address?
I heard that broadcast. Your IP address is 192.168.10.3.
Ethernet: 45:23:79:85:77:34 IP = ??
Ethernet: 45:23:79:85:77:34 IP: 192.168.10.3

An ARP cache size is limited and is periodically cleansed of all entries to free up space; in fact, addresses tend to stay in the cache for only a few minutes. Frequent updates allow other devices in the network to see when a physical host changes their requested IP address. In the cleaning process, unused entries are deleted as well as any unsuccessful attempts to communicate with computers that are not currently powered on.

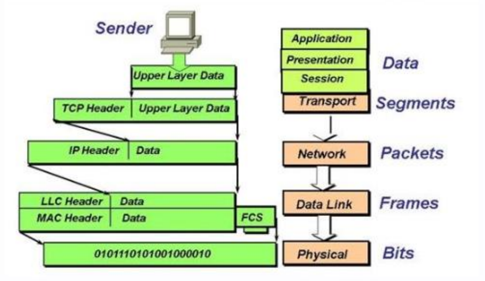
Data Encapsulation

When a host transmits data across a network to another device, the data goes through **encapsulation**: It's wrapped with protocol information at each layer of the OSI model. Each layer communicates only with its peer layer on the receiving device.

To communicate and exchange information, each layer uses **Protocol Data Units (PDUs)**. These hold the control information attached to the data at each layer of the model. They're usually attached to the header in front of the data field but can also be in the trailer, or end, of it.

Each **PDU** attaches to the data by *encapsulating* it at each layer of the OSI model, and each has a specific name depending on the information provided in each header. This PDU information is read-only by the peer layer on the receiving device. After it's read, it's stripped off, and the data is then handed to the next layer up.

The below figure shows the PDUs and how they attach control information to each layer. This figure demonstrates how the upper-layer user data is converted for transmission on the network. The data stream is then handed down to the Transport layer, which sets up a virtual circuit to the receiving device by sending over a synch packet. Next, the data stream is broken up into smaller pieces, and a Transport layer header (a PDU) is created and attached to the header of the data field; now the piece of data is called a segment. Each segment is sequenced so the data stream can be put back together on the receiving side exactly as it was transmitted.



Data Encapsulation

In summary, at a transmitting device, the data-encapsulation method works like this:

1. User information is converted to data for transmission on the network.
2. Data is converted to segments, and a reliable connection is set up between the transmitting and receiving hosts.
3. Segments are converted to packets or datagrams, and a logical address is placed in the header so each packet can be routed through an internetwork.
4. Packets or datagrams are converted to frames for transmission on the local network. Hardware (Ethernet) addresses are used to uniquely identify hosts on a local network segment.
5. Frames are converted to bits, and a digital encoding and clocking scheme is used.

> “ Q: What is data encapsulation?
> A: Data encapsulation is the process of breaking down information into smaller manageable chunks before it is transmitted across the network. These chunks are wrapped with protocol information at each layer of the OSI model. In this process, the source and destination addresses are attached into the headers, along with parity checks.
>
> \- Interview Q&A ”