CS 425: Computer Networks **Jan-Apr 2020**

Lecture 3: Computer Networks – January 10, 2020

Lecturer: Swaprava Nath *Scribes: Bharat Jindal, Deepak Kumar, Divya Chauhan, Pragya Jain*

Disclaimer: *These notes aggregate content from several texts and have not been subjected to the usual scrutiny deserved by formal publications. If you find errors, please bring to the notice of the Instructor.*

3.1 Interfacing

An interface defines which primitive operations and services one component makes available to the other. It is analogous to function abstraction. In computer networks, there are two main types of interfaces:

1. Application-Network Interface
2. Network-Network Interface

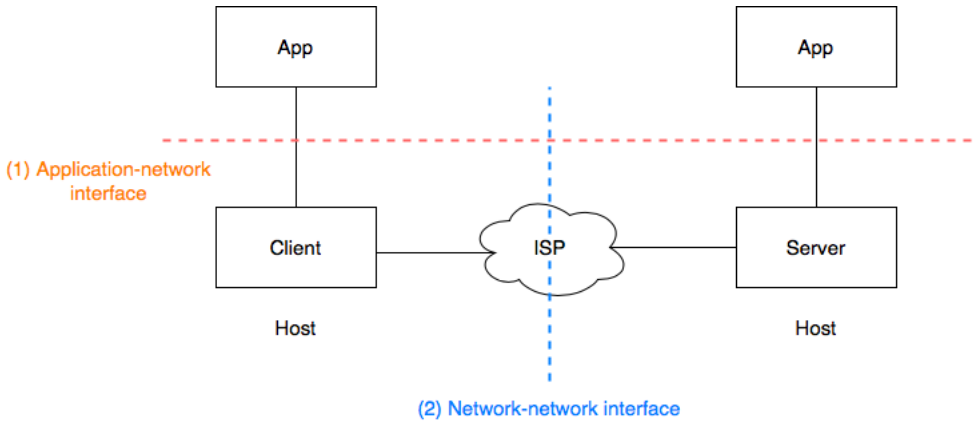


Figure 3.1: Two types of Interfaces

3.2 Application-Network Interface

An application-network interface is an interface that connects the network with applications. It allows applications *(on the local and remote host)* to communicate with each other while hiding details of the network.

Example: Socket API. It provides communication through sockets. A socket is an OS abstraction similar to file handles, representing a connection.
Figure 3.2 shows the typical flow of events for a connection-oriented socket session.

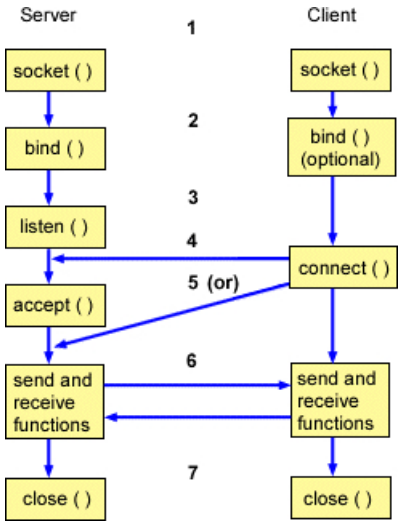


Figure 3.2: Time diagram of the client-server model

The steps involved in establishing a socket on the client side are as follows:

1. Create a socket with the socket() system call.
2. Connect the socket to the address of the server using connect().
3. Send and receive data.

The steps involved in establishing a socket on the server-side are as follows:

1. Create a socket with the socket() system call.
2. Bind/associate the socket to an address and port number using bind().
3. Listen for connections after the socket is activated using listen().
4. Accept a connection with accept(). This function blocks until a client connects with the server.
5. Send and receive data.

When a server or client wants to stop operations, it uses **close()** system call to release any system resources acquired by the socket.

3.3 Network-Network Interface

A network-network interface connects two or more networks.

3.3.1 Traceroute

A command-line tool to find the path from localhost to the remote host.

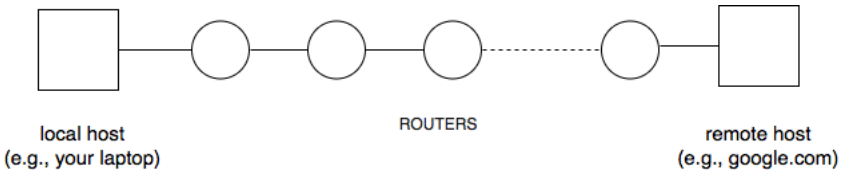


Figure 3.3:

Figure 3.4 shows the output of the Traceroute program, where the route from local computer to aol.com was traced.

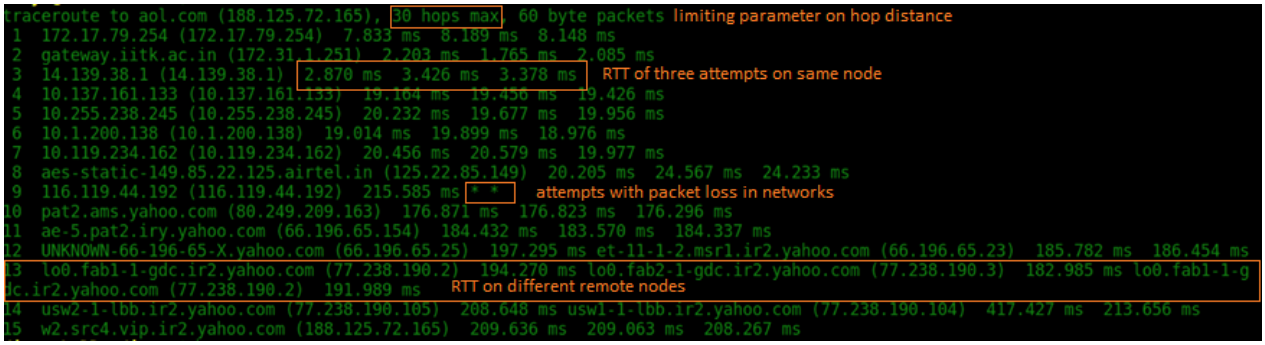


Figure 3.4: Traceroute on aol.com

Following are the columns in the above output:

1. Number of the router along the route
2. Name of the router
3. Address of the router
4. Last three columns are the Round Trip Time (RTT) or delays for three different hop attempts.

Round-trip time (RTT): The duration (in milliseconds) it takes for a network request to go from a starting point to a destination and back again to the starting point.

Hop Count: It refers to the number of intermediate network devices, such as routers, through which data must pass between source and destination.
Hop count can be determined from the last router number. In the above example (Figure 3.4), the hop count is 15.

3.4 Protocols and Layering

3.4.1 Protocol

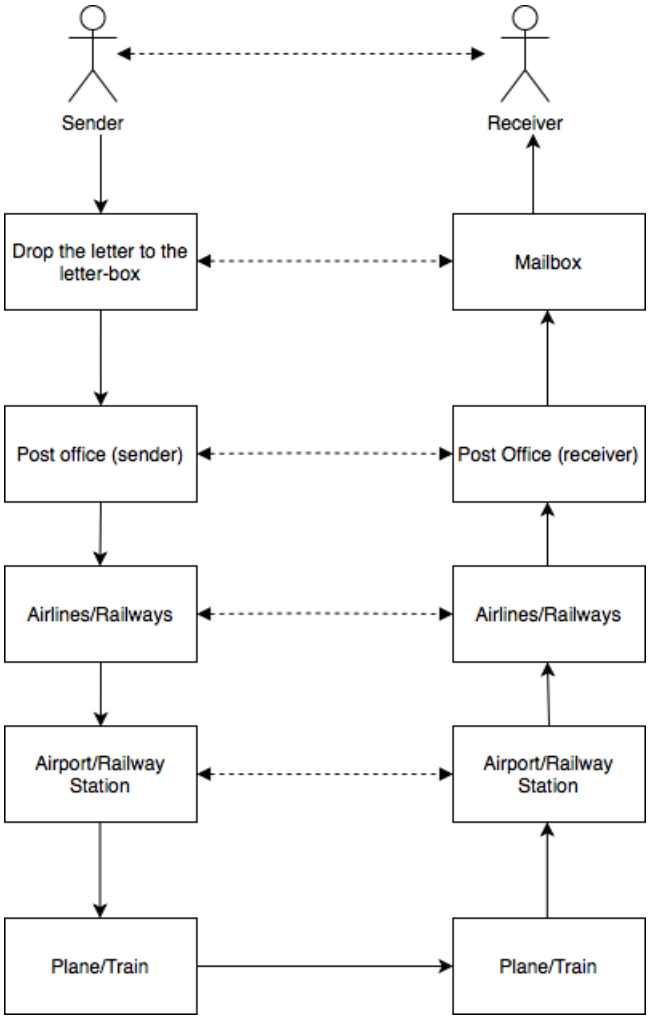
A protocol is a standard agreement/set of rules between communicating devices on how to communicate with each other. Some well-known protocols are: TCP/IP, UDP, FTP, ethernet, 802.11.
Each instance of a protocol:

- talks **virtually** to its peers using the protocol.
- can use the services of the lower layer only.

Why protocols?
modularity of operations / tasks
e.g. in automobile industry, separate teams handle separate tasks to make a car.

3.4.2 Protocol Layering

Before understanding the actual network architecture, lets look at an example where a person sends a letter to his/her friend.



ensures end-to-end delivery

chooses the path between post offices

decides how to reliably reach from station to station

Figure 3.5: Post-Office Architecture

The above example demonstrates some analogies with computer networking. A letter is sent from sender to receiver, similar to a packet which is sent from source host to destination host on the Internet. Each *box represents a function or a* ***responsibility***. Boxes connected via *dotted lines talk* ***virtually*** whereas those connected via a *solid line show* ***physical communication***. So, the functionalities in Figure 3.5 can be viewed in a horizontal manner. The functionalities are divided into layers, providing a framework in which we can discuss the post-office architecture.

Similarly, to provide structure to the design of network protocols, protocols are organised in layers. Each protocol belongs to one of the layers, just as each function in the post office architecture (Figure 3.5) belonged to a layer. Each layer provides its service by (1) performing certain actions within that layer, and (2) using the services of the layer directly below it.

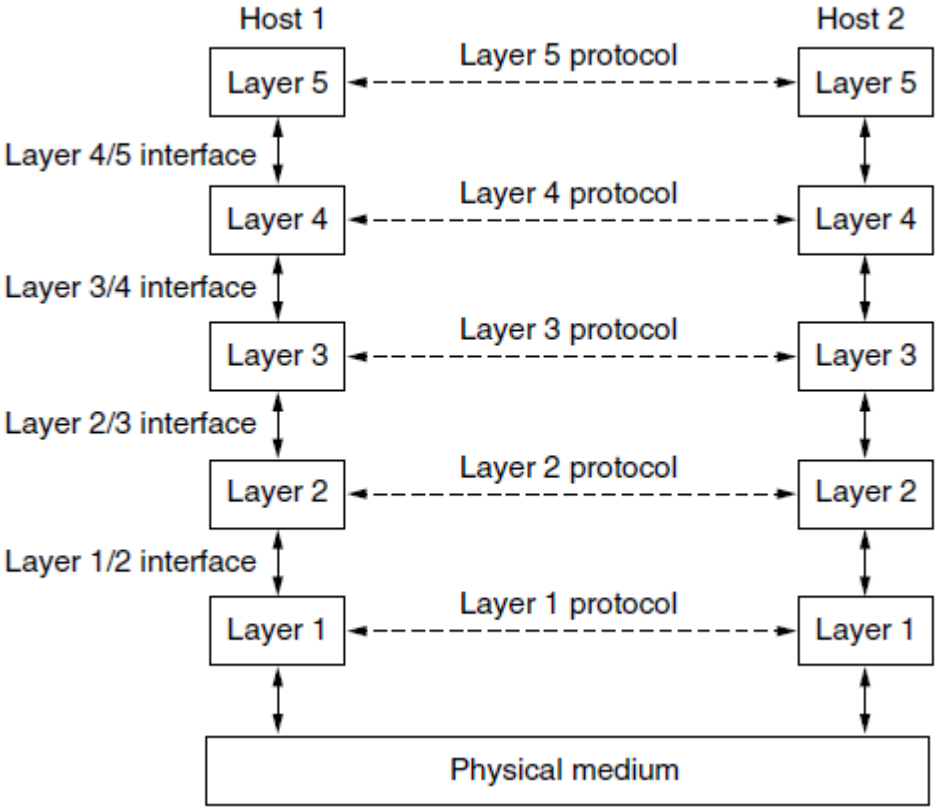


Figure 3.6: Layers, Protocols and Interfaces

Between each pair of adjacent layers is an interface. The interface defines which primitive operations and services the lower layer makes available to the upper one.
The protocol itself can change in some layer without the layers above and below it even noticing.
It is not even necessary that the interfaces on all machines in a network be the same, provided that each machine can correctly use all the protocols.

3.4.3 Purpose

A **layered architecture** provides modularity, making it much easier to change the implementation of the service provided by any particular layer. To reduce design complexity, most networks are organized as a stack of layers or levels, where each layer is built upon the one below it.

Protocol stack: Protocols of various layers taken together such that they are able to connect both vertically between the layers of the network and horizontally between the end-points of each transmission segment.

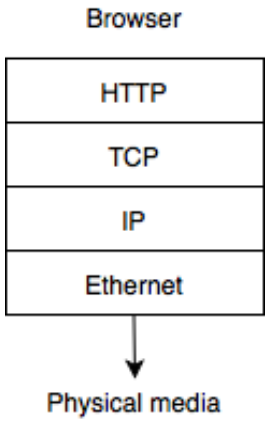


Figure 3.7: An example protocol stack for web-viewing

Communication using the protocol stack is done such that -
Each operation is done over the previous in a **top-down** order in the **sender** and a **bottom-up** order in the **receiver**.

3.4.4 "How is the request sent?" - Encapsulation

Each layer takes the message from the above layer and appends additional information that will be used by the receiver-side layers. The layer segment thus encapsulates the received message with its header information. At each layer, a packet has two types of fields: header fields and a payload field. The payload is typically a packet received from the previous layer.

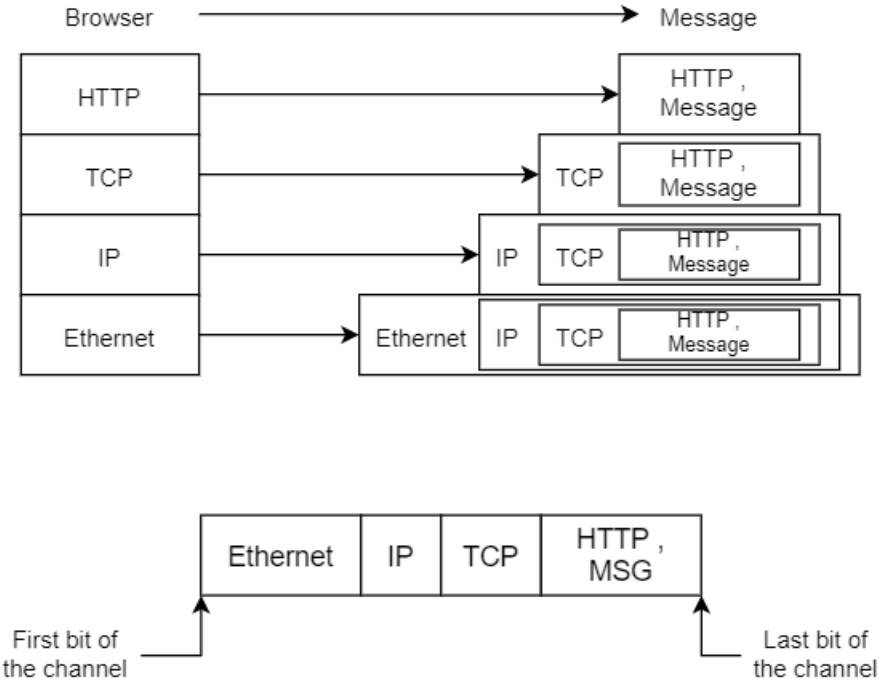


Figure 3.8: Encapsulation of a message in top-down layering

Normally encapsulation adds **bits** as headers.

In practice,

1. Encapsulation bits can also be added as trailers.
2. Encryption/compression might happen along with encapsulation.
3. Segmentation.
   - Large messages can be split into smaller fragments
   - Each fragment is encapsulated individually by TCP layer
   - These fragments may take different paths to reach the destination node.

3.4.5 "How is the message received and read?" - Demultiplexing

The received signal is a string of 0s and 1s. In order to find the path taken by a packet, we need some identification key.

DEMUX key: It contains information that allows the receiver-side layers to deliver the message up to the appropriate protocol.

Demultiplexing: At the receiving end, the layers use DEMUX key to determine the subsequent protocol. This process is called demultiplexing.

For example, in Figure 3.9, an IP packet will use the DEMUX key to determine if further, it is a TCP or UDP packet.

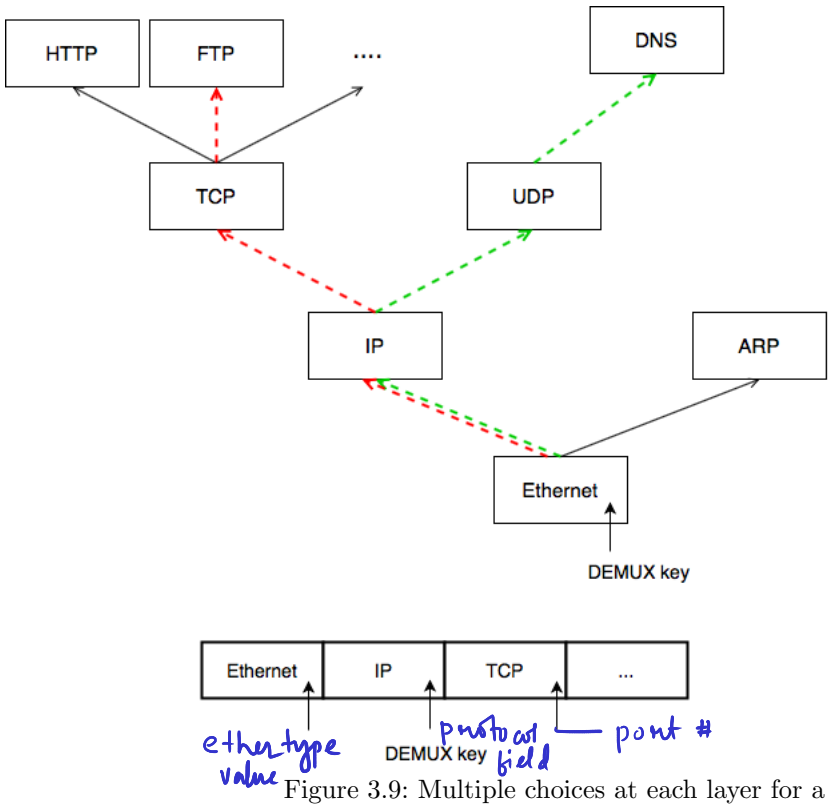


Figure 3.9: Multiple choices at each layer for a packet

DEMUX key may also include **error-detection bits** allowing the receiver to determine whether bits in the message have been changed along the route.

3.4.6 Advantages of layering

1. Provides a structured way to understand the network stack.
2. Information hiding.
   - Functionality inside a layer is self-contained; one layer doesn't need to reason about the other layers. Example: Consider the protocol stack for web-viewing (figure 3.6). Here, the browser doesn't care which physical media is used or which path is taken.
3. Modularity.
   - Can replace a layer without impacting other layers.
   - Lower layers can be reused by higher layers. Example - TCP and UDP both are layered upon IP.
4. Facilitates the connection of dissimilar devices **(Internetworking)**.

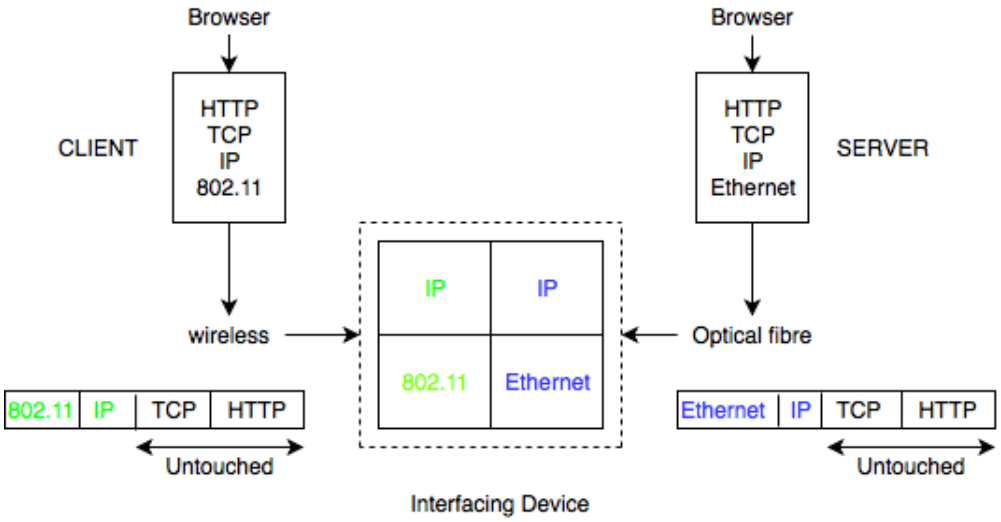


Figure 3.10: Internetworking

In Figure 3.10, the interfacing device removes the headers of the network layer (IP of the client) and link layer (802.11) of the packet received from the client. It then adds the appropriate headers for these layers (IP of server, Ethernet) so that they can be correctly received by the server.

3.4.7 Disadvantages

1. Large headers leads to inefficient throughput.
   - Exhaustion of scarce physical resources.
2. Information hiding sacrifices Quality of Service (QoS)
   - Example: VoIP packets need short latency period, therefore requiring different protocols. Not being able to classify different data packets due to information hiding effects the QoS here.
3. Message Corruption.
   - Message bits received can be damaged due to fluke electrical noise, random wireless signals, hardware flaws, software bugs. Each subsequent layer adds additional reliability threats.

3.5 References

[1] JF Kurose, KW Ross, Computer Networking: A Top-Down Approach, 5th Ed., Addison-Wesley, 2009.

[2] AS Tanenbaum, DJ Wetherall, Computer Networks, 5th Ed., Prentice-Hall, 2014