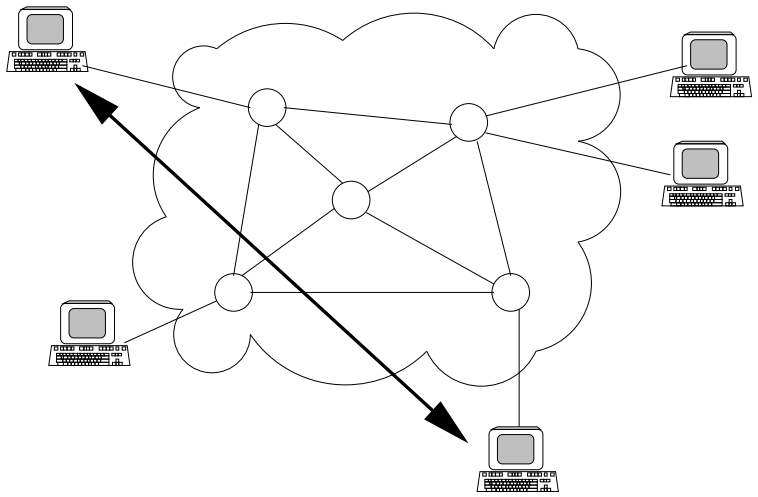
Network Layers and Their Functions

1 How to send a message over a network?

In essence, the reason that we construct networks is to send messages. So firstly, let us discuss how to send a message over a network?



In order to send a message over a network, we should do a series of jobs, such as:

- **Addressing**, finds end systems' addresses in a network
- **Routing**, chooses outgoing communication links when a message arrives at a router
- **Congestion Control**, avoids excessive data transmission which may cause a network jam
- **Reliable Data Transfer**, ensures a message received at the receiver is the same as it is at the sender

We use the human analogy of 'Postal Service' to help us understand this issue. Consider what will be done when you want to send a letter to your friend.

First, you should know where to send this letter, i.e. you should know your friend's address and postal code and write them and your address (we will discuss your address later) on the envelope. In a network, a message to be sent should also contains the address information of the sender and receiver which is known as **addressing**.

Second, suppose the postman has gotten your mail and is on his way to your friend's home according to the address written on the envelope, then he reaches an intersection of several streets. Unfortunately there is more than one street that can lead to your friend's

home. The postman has to choose one street. If he is smart enough, he can choose the shortest way to your friend's home. In a network, end systems are not directly connected to each other via a single communication link. Instead, they are indirectly connected to each other through intermediate switching devices known as *routers*. When a message arrives at a router, the router should choose one of its outgoing communication links and forward the message to the link. This procedure is known as **routing**.

Third, if there are more than 1000 letters to be sent to the same address at the same time and all the postmen choose the same way which is the shortest, this may cause a traffic jam. The post office should have some mechanism to avoid such a congestion, e.g. reduce the number of postmen working at the same time, or direct some postmen to travel through another way. In a network, in some cases, this kind of congestion can also happen. So we should have a mechanism to avoid such kind of a congestion in a network. This mechanism is kown as **congestion control**.

Fourth, when you send a packet to your friend, you may want to make sure that your friend has received it and there are no missing items. So you attache a list of all the items in the packet. After your friend has received the packet, he checks the list to find if there are any items missing, then signs a receipt. The postman then returns the receipt to you using your address on the packet thus ensuring your friend has received it correctly. In a network, sometimes, the receiver should check if the packet has an error (*error detection*) and *acknowledge* that the packet was received correctly. These two techniques are known as **reliable data transfer**.

2 How to master the complexity of a network?

As we have discussed, computer networks are very complex. Besides the isssues we have metioned above, there are many other issues we should consider when constructing a network, such as:

- **connection setup**, in some cases, end systems which are going to exchange information will setup a connection before sending the real data. End systems verify the identities of the sender or the receiver by connection setup.
- **message segmentation**, some packets transferred in a network are so large that we have to divide them into small segments in order to make the transfer efficient (we will discuss why we do in this way in other lectures).
- **multiplexing**, sometimes a single link will provide services to different users at the same time. So we use the technique named *multiplexing* to satisfy the services.

In general, a network can be an extremely complicated system. There are many pieces to the network implementation: numerious applications and protocols, various types of end systems and connections between end systems, routers, and various types of link-level media. How can we organize the network architecture with such an enormous complexity? In software engineering, we always divide a huge project into small **modules** which have their own

functions and can communicate with other modules according to predefined protocols. We want to use a similar method to construct a complicated network.

Useful method for dealing with complexity in a network is using "**modularity**", which means modularize a network by doing the following:

- break complex problem into simpler sub-problems
- use "black box" (input/output) abstraction for sub-problems

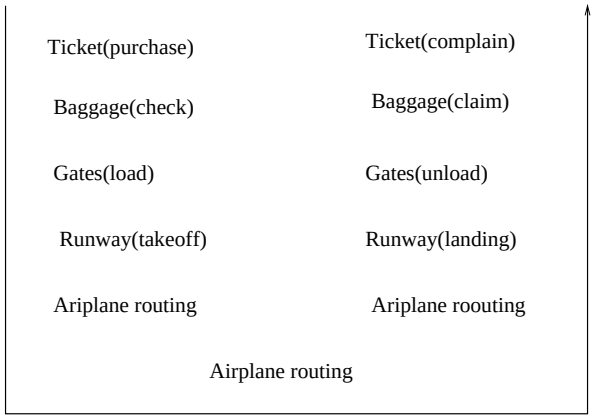
Let us look at an example of SMTP (Simple Mail Transfer Protocol)

S: 220 sf.com
C: HELO toronto.edu
S: 250 Hello toronto.edu, pleased to meet you
C: MAIL FROM: <alice@toronto.edu>
S: 250 alice@toronto.edu... Sender ok
C: RCPT TO: <bob@sf.com>
S: 250 bob@sf.com ... Recipient ok
C: DATA
S: 354 Enter mail, end with "." on a line by itself
C: How are you?
C: See you soon.
C: .
S: 250 Message accepted for delivery
C: QUIT
S: 221 sf.com closing connection

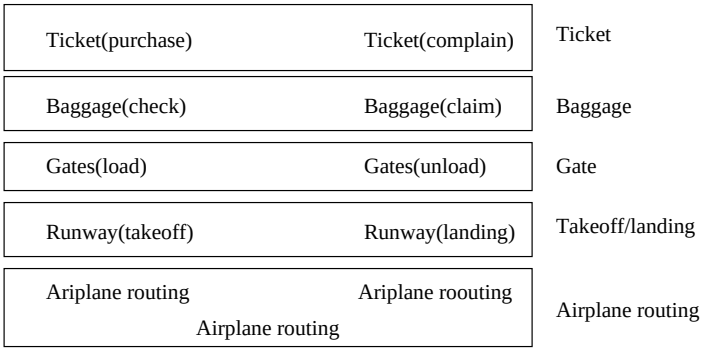
We can regard SMTP as one of the *modules* in a network. It performs some functions, such as exchanging information between a client and a server, and connection seup. But it does not perform other important functions in a network such as routing, and congestion control. Fortunately, some other "modules" perform those functions. In this example, SMTP at the sennder just passes a message to its lower "module" and does not care how the message will be transmitted; SMTP at the receiver just gets the message from its lower "module" and also does not care how the message arrives. Thus by *modularity*, we simplified a complex problem to simple sub-problems.

3 Network Layers

Before attempting to **'modularize'** the network, let us look at a human analogy-*air system*(we always use analogies to understand networks). How can you describe this complex system? One way to describe this system is to describe a series of actions you take when you fly on an ariline. You purchase your tickets, check your bags, go to the gate, and eventually get loaded onto the plane. The plane takes off and is routed to its destination. After your plane lands, you de-plane at the gate and claim your bags. If the trip was bad, you complain about the flight to the ticket agent. This scenario is shown in the following figure:



We are looking for some structure in the above figure. We note that there is a ticketing function at each end; there is also a baggage function for already-ticketed passengers, and a gate function for already-ticketed and already-baggage-checked passengers. For passengers who have made it through the gate(that is, passengers who are already ticketed, baggage-checked, and through the gate), there is a takeoff and landing function, and while in flight, there is an airplane routing function. This suggests that we can look at the functionality in the following figure:



This figure has divided the airline functionality into layers.We note that each layer, combined with the layers below it, implement some functionality, some *service*. For example, at the ticketing layer and below, airline-counter-to-airline-counter transfer of a person is accomplished. These services can only be provided in an order, which means the gate layer at the takeoff end can not provide service to the passengers before they have been provided services by the baggage layer.

As noted above, a layered architecture allows us to discuss a complex system. Let us now turn our attention to network architecture. The type of functional modularity used for computer networks is **hierarchical layering**. In each layer, we use **protocols** to implement the functionality of that layer. With a layered protocol architecture, each protocol belongs to one of the layers. It's important to realize that a protocol in layer n is distributed among the network entities (including end systems and packet switches) that implement that protocol.

The Internet stack consists of five layers: **Physical, Data link, Network, Transport, and Application** layers. The Internet layers are illustrated in the following figure.

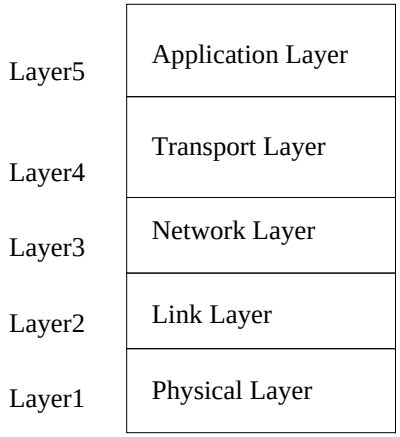


Figure 1: The Internet protocol stack

There are several ways to define a layered network architecture. In this course, we consider the 5 Internet layers. Another model consists of the 7 OSI layers.

3.1 Layer Functions

In a computer network, each layer may perform one or more of the following generic sets of tasks:

- *Error Control*: Making the logical channel between the layers in two peer network elements more reliable
- *Flow Control*: Avoiding overwhelming a slower peer with protocol data units
- *Segmentation and Reassembly*: Dividing large data chunks into smaller pieces at the sending side and reassembling the smaller pieces into the original large chunk at the receiving side
- *Multiplexing*: Allowing several higher-level sessions to share a single lower-level connection
- *Connection Setup*: Providing handshaking with a peer

3.2 Application Layer

- **Service:**
  The application layer is responsible for supporting network applications, such as Web, E-mail, and file transfer

- **Functions:**

  – Connection Setup

  – Flow Control

  – Error Control

  One function of which the application layer performs is Connection Setup. Let us go back to the former example of SMTP to see how the application layer setup a connection.

  S: 220 sf.com
  C: HELO toronto.edu
  S: 250 Hello toronto.edu, pleased to meet you
  C: MAIL FROM: <alice@toronto.edu>
  S: 250 alice@toronto.edu... Sender ok
  C: RCPT TO: <bob@sf.com>
  S: 250 bob@sf.com ... Recipient ok
  .
  .
  .
  C: QUIT
  S: 221 sf.com closing connection

  First, the client says 'Hello' to the server; after verifying the identity of the client, the server returns 'Hello'. Second, the client requests to send email from $alice@toronto.edu$; after verifying, the server says 'sender's address is OK'. Third, the client request to send email to $bob@sf.com$;after verifying, the sever says 'the recipient's address is OK'. So far, the client has setup a connection to the server and can send data to the server. After the client finishes sending data, it requests to close the connection; the server accepts the request and then closes the connection.

- **Protocols:**
  The application layer includes many protocols, such as HTTP to support the Web, SMTP to support electronic mail, and FTP to support file transfer.

- **Location:**
  The application layer is located in end systems and hosts.

3.3 Transport Layer

- **Service:**
  A transport layer protocol provides for **logical communication** between application processes running on different hosts. At the transport layer, a large packet is always divided into small segments. It is more efficient to transmit small segments than a whole large packet. We will discuss it in later courses.

- **Functions:**
  - Message Fragmentation and Reassembly
  - Flow Control
  - Error Control
  - Congestion Control
  - Connection Setup

- **Protocols:**
  In the Internet there are two transport protocols, TCP and UDP. TCP provides a reliable, connection-oriented service to the invoking application. UDP provides an unreliable, connectionless service to the invoking application. TCP also segments long messages into shorter segments and provides a congestion control mechanism

- **Location:**
  The Transport layer is located in end systems and hosts.

3.4 Network Layer

- **Service:**
  The network layer is responsible for routing datagrams from one host to another

- **Functions:**
  - Routing
  - Addressing
  - Congestion Control

- **Protocols:**
  The Internet's network layer has two principle components. One is IP protocol which defines the fields in the IP datagram as well as how the end systems and routers act on these fields. The other contains numerous routing protocols

- **Location:**
  The network layer is located in end systems, hosts, and routers

3.5 Data Link Layer

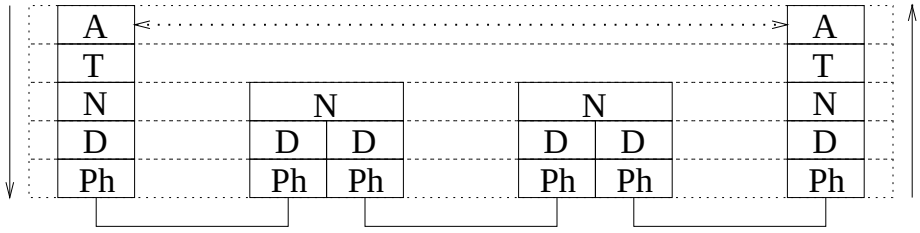
- **Service:**
  The data link-layer is used to move a datagram over an individual link
- **Functions:**
  - Framing
  - Error Control
  - Retransmissions
- **Protocols:**
  The data link-layer protocol defines the format of the units of data exchanged between the nodes at the ends of the link, as well as the actions taken by these nodes when sending and receiving these data units. ARQ (Automatic Repeat Request), CSMA/CD for Ethernet and Wave LAN
- **Location:**
  The data link layer is located in end systems, hosts, and routers

3.6 Physical Layer

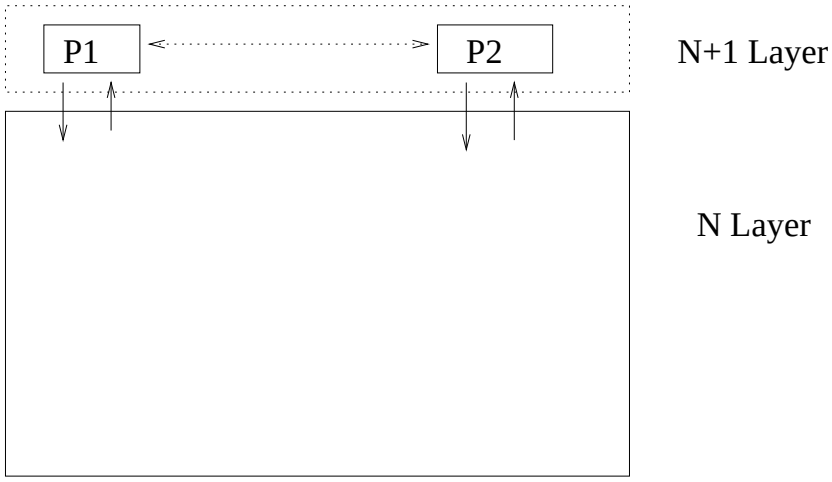
- **Service:**
  While the job of the data link layer is to move entire frames from one network element to an adjacent network element, the job of the physical layer is to move the individual bits within the frame from one node to the next
- **Functions:**
  - Modem (Modulator/Demodulator)
- **Protocols:**
  The protocols in this layer are dependent on the actual transmission medium of the link. For example, Ethernet has many physical layer protocols: one for twisted-pair copper wire, another for coaxial cable, another for fiber, and so on
- **Location:**
  The data link layer is located in end systems, hosts, and routers

3.7 The Roller-Coaster Ride of a Message and Layer dependence

From the above discussion, we have known that in order to move a message from a end system to another end system, the message should be moved through all the layers in each node on the way to the destination. Let us look at the following figure: to users, it seems that one application in a end system can communicate "directly" with another application in another end system. But actually, we can only say that these two applications can communiate logically. In fact, the journey of a message can be described as: it sets off at the sender's application layer. After it goes down through all the layers in the sender, it is transmitted through a physical link. When it arrives at a router, it goes up first and then go down through all the layers of the router. Continue travelling like this, it arrives at the physical layer of the receiver. At last, it goes up through all the layers of the receiver and eventually arrives at the receiver's application layer This kind of moving is like the Roller-Coaster Ride. We use the following figure to illustrate it.



Let us take a closer look at the interaction of adjacent layers. The N+1 layers in different hosts can not communicate with each other directly. They rely on lower layers to provide services to them



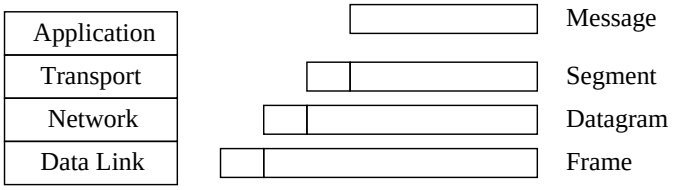
3.8 Terminology

So far, we have an overview of a network. The following are some terms related to networks

- **Peer or Peer Processes:**
  Members of the same layer at different locations
- **Protocol:**
  Set of rules for how peers interact
- **Network Architecture or Network Reference Model:**
  Set of layers used for a network

3.9 "Growing" Data Units

As a message starts its journey to go through the layers, at the send node, each layer will add some Protocol Control Information (PCI) or Header to the packet. So the packet will get 'fatter' while going through the layers. At the receive node, each layer will remove its own layer's Header and when the message arrives in the application in the receiver, it will be the same as before being sent. We give a special name **PDU**s to the data units which have been added PCIs, and we also give special names for PDUs in different layers. We use the following figure to illustrate the 'Growing' Data.



3.10 Why Layers?

There are many advantages of layering, such as:

- Simplifying a complex system
- Flexibility, easily add on new protocols in a layer with the remainder of the system unchanged
- Standardization, with layering it is easy to set the standards for each layer. Then different products from different factories can interact with each other according to the same standards.