Plan

1. Computer Networks and the Internet
   - What is the Internet?
   - The network edge
   - The network core
   - Network access and physical media
   - ISPs and Internet backbones
   - Delay and loss in packet-switched networks
   - **Protocol layers and their service models**

Layered architecture

Until now we have been presenting the different components (switches, hosts, links), their relationships, different switching policies and different protocols, or services.

We need now a model to describe the **architecture** of the services provided by the Internet.

A useful analogy is an airline system.

One way to guess a structure of it, is to describe the actions taken to make a trip: first we purchase a flight ticket, then we go to airport, we give our luggage to be checked, then we board the plane as the baggage is loaded, then we take off, the plane travels to destination, lands, we get off the plane, our baggage is unloaded, we claim it and we complain to the ticketing service if the trip was bad.

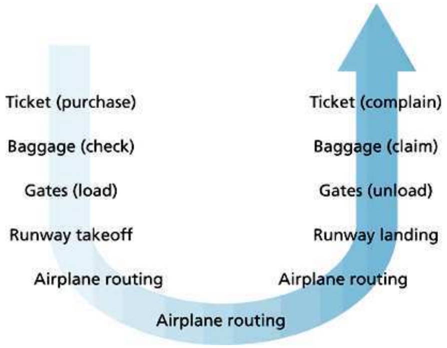
Layered architecture (cont)

The figure in the next slide summarises the series of actions undertaken to make an airplane travel.

It appears that each kind of activity, based on a functionality, in the departing airport has a corresponding counterpart in the arrival airport.

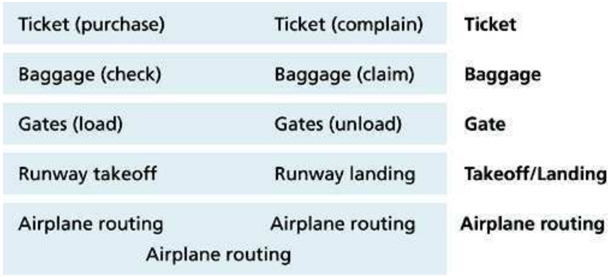
This suggests that we can look at the services in a *horizontal manner*, i.e., as belonging to same levels.

Layered architecture (cont)



Layered architecture (cont)

The following picture changes the focus according to the horizontal point of view on the service usages we described so far. The structure is divided into **service layers**.



Ticket (purchase)	Ticket (complain)	**Ticket**
Baggage (check)	Baggage (claim)	**Baggage**
Gates (load)	Gates (unload)	**Gate**
Runway takeoff	Runway landing	**Takeoff/Landing**
Airplane routing	Airplane routing	**Airplane routing**
Airplane routing		

Layered architecture (cont)

This layered architecture is of great value because it allows to reason on a specific part of the system, just by considering the upper layer (to which the current layer is a provider) and the lower layer (to which the current layer is a customer).

For instance, this model allows to change the service of one layer in a way that is not noticed by the surrounding layers.

The way a service is achieved internally can be changed without changing its observable behaviour (like reorganising the workflow of the staff at the airport gates: this does not change anything outside the service).

It is even possible sometimes to change some properties of the service that are not noticed by the surrounding layers, like imposing that boarding should be done depending on the height of the passengers: this is not noticed by the baggage check and the runway takeoff services.

Layered architecture (cont)

Coming back to networks services, these are also structured into layers. The input and output behaviour of one layer (service) is the **protocol**. The way this behaviour is achieved internally to the layer is the **protocol implementation**.

As we just saw, these are different notions: you can change the protocol (as imposing more constraints on the output, like sorting by height the passengers), or independently change its implementation (as reorganising the staff at the gates).

Notice also that a protocol in a given layer is implemented by remote entities of the network, just as the services in the airplane system were *distributed* between the departing and arriving airports.

Layered architecture (cont)

In a network, the protocols can either be implemented by hardware or software, or even by a mixture of hardware and software.

Each implementation of layer n communicates with its peer at the same level n by exchanging messages whose structure is specific to this level, called n-**layer protocol data units** or, in short, n-**PDUs**.

On one entity, all the protocols taken as a whole is called the **protocol stack**.

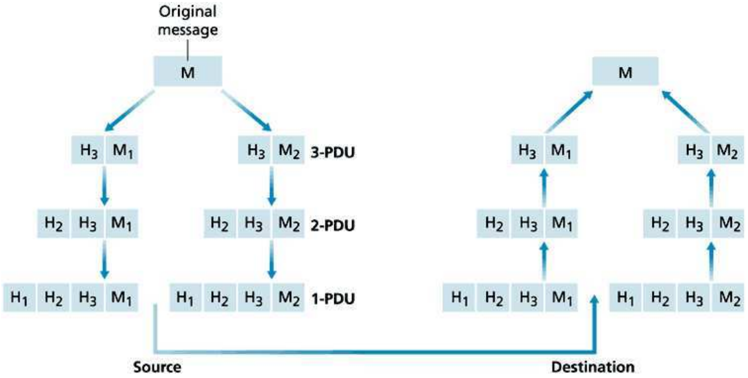
Layered architecture (cont)

When the layer n of host A sends an n-PDU to layer n of host B, it passes the PDU to the $n-1$ layer of host A and lets it deliver the n-PDU to the layer n of host B. Thus layer n is said to *rely* on layer $n-1$ for achieving the delivering, and, in turn, layer $n-1$ *offers services* to layer n.

Layer n is an **upper layer** with respect to layer $n-1$, which is then a **lower layer**.

Layered architecture (cont)

Let us give more insight about protocol layering. Consider a network with four layers in the following figure.



Layered architecture (cont)

The application message, M, is cut into two packets, M_1 and M_2. They are both sent to the layer below, which is layer 3. This layer does not know how to interpret the contents of M_1 and M_2, and does not care about it. His role is to add a **header** H_3 to each packet, whose structure is specific to layer 3, and then passes them to the layer below.

Layer 2, similarly, does not know the structure of the incoming 3-PDU, and does not care. It just adds a header H_2 specific to layer 2. The next layer behaves similarly.

The remote protocol stack works symmetrically, as the message goes up. Each level reads the outermost header (supposed to correspond, by construction, to the current level) and removes it.

Finally, the application message is reassembled back into M.

Layered architecture (cont)

In order for one layer to **interoperate** with the layer below it, the **interfaces** between the two layers must be precisely defined.

An interface contains the format of the PDU as well as the kind of functionality a layer provides (you can think to the types of functions in programming languages).

This allows to improve the implementation of a service to be unnoticed from adjacent layers, as long as the interface remains the same (this is similar to changing the code of a function, while keeping its type).

Layered architecture (cont)

In a computer networks, each layer may perform one or more of the following general tasks:

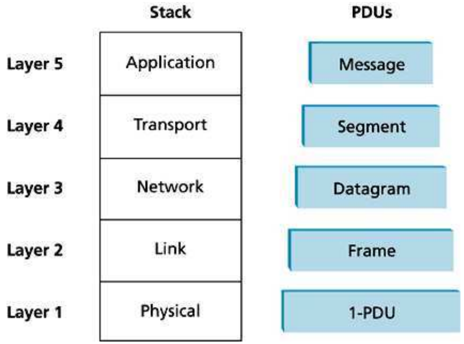
- **Error control**, which makes the logical channel (i.e., from one peer to another, at the same stack level, while ignoring the lower levels) more reliable.
- **Flow control**, which avoids overwhelming a slower peer with PDUs.
- **Segmentation and reassembly**, which divides data at the source and reassembles the pieces at the destination.
- **Multiplexing**, which allows several high-level sessions (i.e., full runs of a given service instance) to share a single lower-level channel.
- **Connection setup**, which provides handshaking with a peer.

Layered architecture (cont)

The **Internet protocol stack** is made of five layers: the physical layer, the data link layer, the network layer, the transport layer and the application layer (bottom-up).

It is common usage to give a name to the four upper layers, instead of *n*-PDU. They are, from data link layer to application layer: **frame**, **datagram**, **segment**, **message**.

Layered architecture (cont)



Layered architecture (cont)

The **application layer** supports network-oriented programs and may include many different protocols, like HTTP to support the Web, SMTP to support e-mail and FTP to support file transfer.

The **transport layer** provides the service of transporting application-layer messages between the client and the server. In the Internet, there are two protocols at this level: TCP and UDP.

The **network layer** routes datagrams from one host to the other. It has two main components. One defines the fields of the datagrams as well as how the hosts and the routers acts on these fields: this is the famous IP protocol. Another one contains routing protocols, which determines the paths between sources and destinations.

Layered architecture (cont)

The **link layer** is in charge of moving a packet from one router to another. By contrast, the network layer moves a packet along the whole path, not just two switches.

At each node, the network layer passes the datagram to the link layer, which delivers the corresponding frame to the next node. At this node, the frame is passed up to the network layer again.

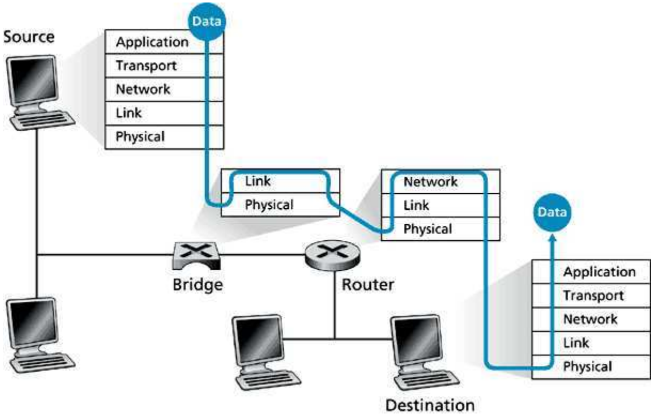
This is because the services provided by the link layer depends on the specific link protocols, which may be different for the incoming link and the out-going link (hence the frame must go up to the network layer at each node). For instance, one link can use PPP and the next one Ethernet.

Layered architecture (cont)

The **physical layer** is in charge of moving bits along a link, by propagating electromagnetic signals along a link. As with the link layer, the protocols at this layer are link-dependent and also depend on the actual transmission rate of the link.

For instance, Ethernet has many physical layer protocols: one for twisted-pair copper wire, another for coaxial cable, another for fiber optics etc. This is because in each case the way to move a bit across the link is different.

Layered architecture (cont)



Bridges are a special kind of switches (they do not support IP).