2

Objective

- understand TCP/IP and networking terminology
  - layered model
  - transport protocol
  - TCP; UDP; IP; MAC
  - IP addresses, MAC addresses,machine names, DNS, NetBIOS names, port numbers
  - routers, bridges, servers
  - client server architecture
  - HTTP, FTP, SMTP
  - multiplexing and switching
  - protocol
  - connection
- basic elements of transmission

References

- useful complement to these notes

  [Stevens] "TCP/IP illustrated volume I", The protocols, W. Richard Stevens, Addison Wesley

  Very detailed, experimental hands-on description of TCP/IP. Also volume III for HTTP

  [STA] "High Speed Networks (TCP/IP and ATM Design Principles", William Stallings, Prentice Hall, 1997

  Advanced topics

  [Keshav] "An Engineering Approach to Computer Networking", S. Keshav, Addison Wesley, 1998

  Advanced topics

  [Kurose, Ross] "Computer Networking", J.F. Kurose and K.W. Ross, Addison Wesley, 2000

  Top down approach, applications

- other references

  [Walrand and Varaiya] High Performance Communication Networks, Jean Walrand and Pravin Varayia, Morgan Kaufman, `http//www.mkp.com`

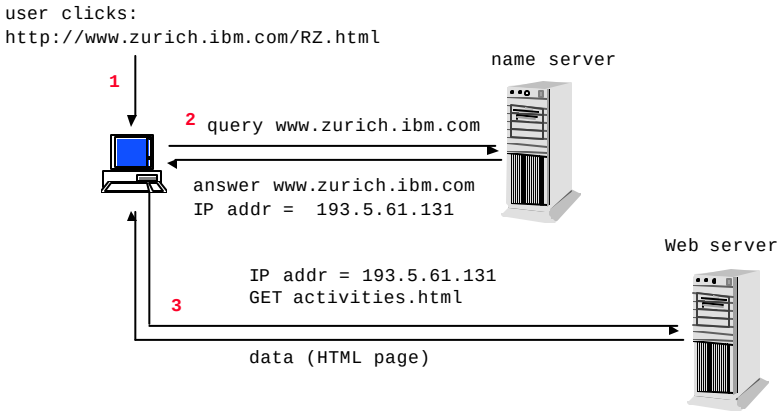
  [BG] "Data Networks", Dimitri Bertsekas and Robert Gallager, Prentice Hall

  Theoretical approach, fundamentals ideas behind basic building b locks of a computer network. Little description of real systems

4

Network Services

- network services examples:
  - distributed database, Web **(3)** , file transfer, remote login, email, news, talk, remote processing, resource sharing (file servers, printers, modems), network time, name service **(2)** ……

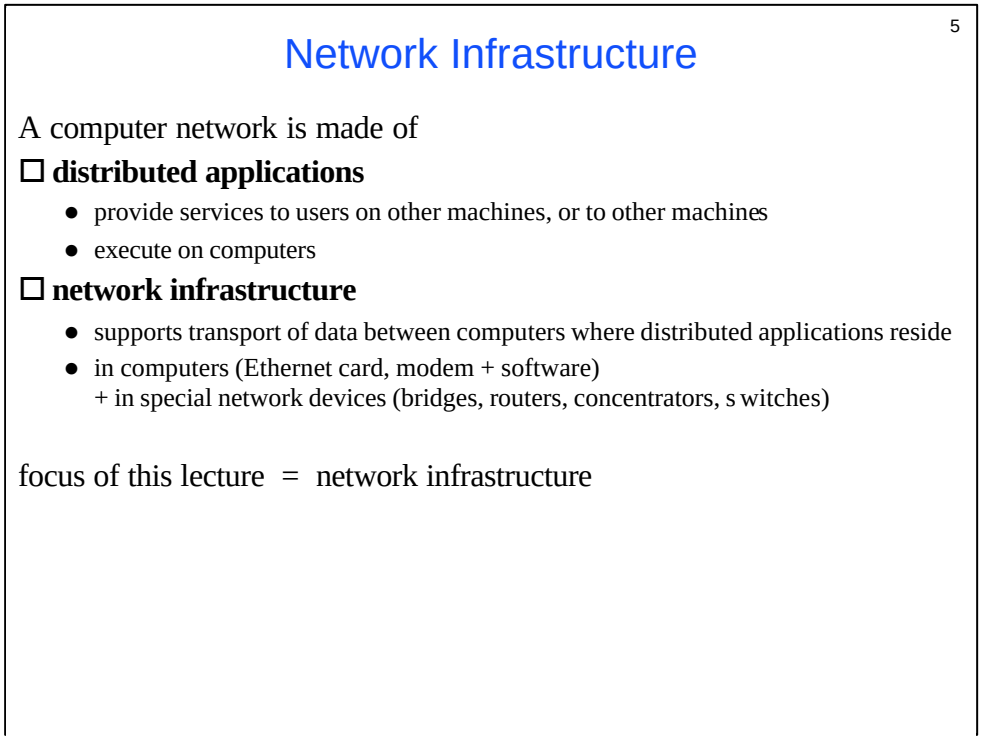


In this lecture we study computer networks.

We use a top-down approach, starting with applications. We present an overall picture, which will enable you to understand the layered model of networking software. Then in the following chapters, we will study the various components (called “layers”), one by one.

What are computer networks used for ?

Computer networks allow people and machines to communicate, using a number of services. The slide shows a small subset of services.



A computer network is made of two distinct subsets of components

- **distributed applications** are programs running on interconnected computers; a web server, a remote login server, an e-mail exchanger are examples. This is the visible part of what people call “the Internet”. In this lecture we will study the simplest aspects of distributed applications. More sophisticated aspects are the object of lectures called “Distributed Systems” and “Information Systems”.

- the **network infrastructure** is the collection of systems which are required for the interconnection of computers running the distributed applications. It is the main focus of this lecture.

The network infrastructure problem has itself two aspects:

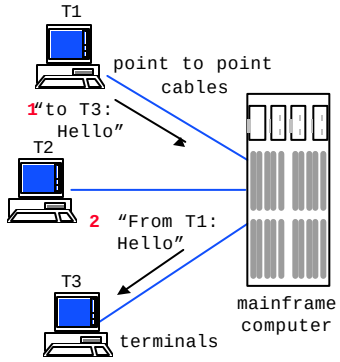
- **distance**: interconnect remote systems that are too far apart for a direct cable connection

- **meshing**: interconnect systems together; even in the case of systems close to each other, it is not possible in non-trivial cases to put cables from all systems to all systems (combinatorial explosion, cable *salad* management problems etc.).

The distance problem is solved by using a network, such as the telephone network with modems (see later). The meshing problem was originally solved easily because the terminals were not able to communicate with each other, but always has to go through a main computer. The mesh in such cases is reduced to a star network. Today this is solved by a complex set of bridges and routers.

6

Physical Layer
Data Link Layer



- ☐ physical transmission = **Physical** function
  - bits <-> electrical / optical signals
  - transmit individual bits over the cable: modulation, encoding, synchronization
- ☐ frame transmission = **Data Link** function
  - bits <-> frames
  - frame boundaries
  - addressing
  - bit error detection: CRC
- ☐ Modems, Ethernets

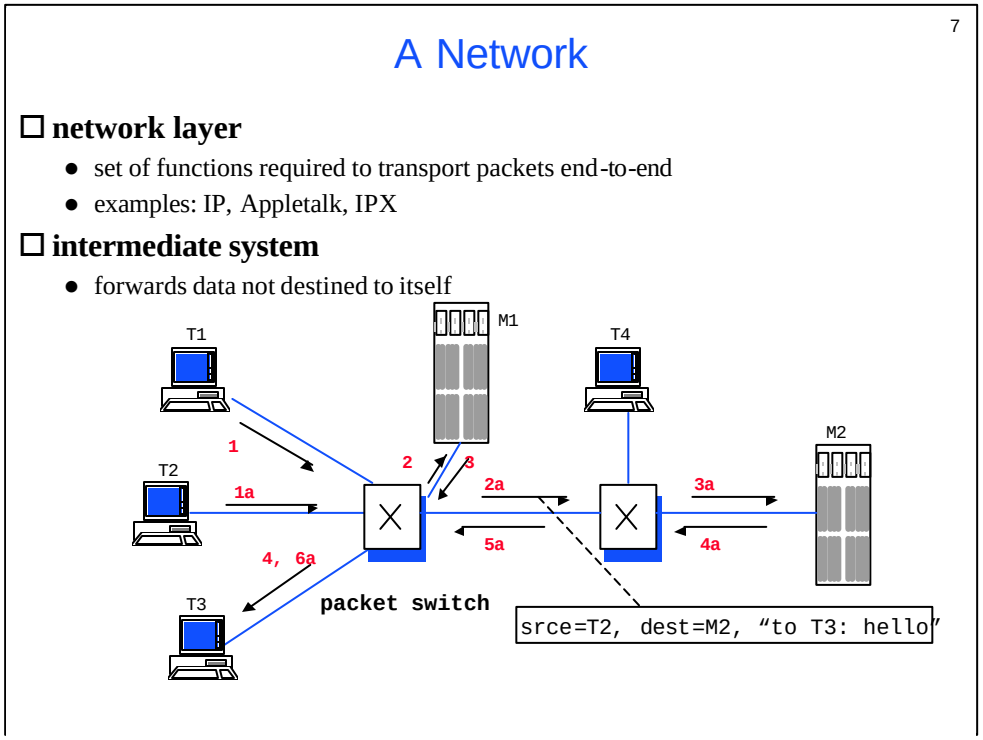
The objective of this and the following slides is to introduce the concept of layers. Like any complex computer system, a network is made of components. This decomposition is, to a large extent, stable: computer networking people have agreed on a reasonable way to divide the set of functions into what is called “layers”.

We use the term layer because the decomposition always assumes that different components can be ordered such that one component interfaces only with two adjacent components. We call “layers” the components.

We start with the simplest, and the oldest network example: it is a mainframe connected to terminals. In this case, there are mainly two functions

- physical layer: translates bits into electromagnetic waves;
- data link layer: translates frames into bits.

These two functions are implemented on cables or on radio links. The physical layer has to do with signal processing and coding; it is the object of the lecture called “Telecommunications”. The data link layer has to do with bits and bytes; we will study the data link layer in this lecture.



Modern networks have more than the physical and the data link layers. The **network layer** is a set of mechanisms that can be used to send packets from one computer to another in the world. There are two types of networks:

With **packet switching,** data packets can be carried together on the same link. They are differentiated by addressing information. Packet switching is the basis for all data networks today, including the Internet, public data networks such as Frame Relay or X.25, and even ATM. Packet switches have queues.

Circuit switching is the way telephone networks operate. A circuit emulates the physical signals of a direct end-to-end cable. When computers are connected by a circuit switched network, they establish a direct data link over the circuit. This is used today for modem access to a data network.

Modern circuit switches are based on byte multiplexing and are thus similar to packet switches, with the main difference that they perform non-statistical multiplexing (see later).

A network has **intermediate systems** (ISs): those are systems that send data to next ISs or to the destination. Using interconnected ISs saves cable and bandwidth. Intermediate systems are known under various terms depending on the context: routers (TCP/IP, AppleTalk,…), switches (X.25, Frame Relay, ATM, telephone), communication controllers (SNA), network nodes (APPN)

8

Transport Layer

- why a transport layer?
  - **transport layer** = makes network services available to programs
  - is end-to-end only, not in intermediate systems such as routers
  - may add additional functions to network services (reliability, ordering, congestion control, multiplexing)
- in TCP/IP there are two main transport protocols (there is also RTP)
  - UDP (User Datagram Protocol)
    - unreliable
    - offers a datagram service to the application (unit of information is a message)
  - TCP (Transmisssion Control Protocol)
    - reliable
    - offers a stream service (unit of information is a byte)
- application may use UDP or TCP depending on requirements
- programming interface
  - **socket** API: a library of C functions
  - socket is similar to a file descriptor; controls a communication endpoint
    - is associated with an IP address, a port number

Physical, data link and network layers are sufficient to build a packet transport system between computers. However, this is not enough for the programmer.

When you write a low-level program that uses the network (as we will do in this lecture), you do not handle packets, but data. The primary goal of the transport layer is to provide the programmer with an interface to the network.

Second, the transport layer uses the concept of **port**. A port is an address that is used locally (on one machine) and identifies the source and the destination of the packet **inside one machine**. We will come back to the concept of ports later in this chapter.

The transport layer exists in two varieties: unreliable and reliable. The unreliable variety simply sends packets, and does not attempt to guarantee any delivery. The reliable variety, in contrast, makes sure that data does reach the destination, even if some packets may be lost from time to time.

9

Protocol, service and other fancy definitions

- Peer entities
  - two (or more) instances of the same layer
- Protocol and a PDU:
  - the rules of the operation followed by peer entities
  - the data exchanged is called PDU (Protocol Data Unit)
  - there is one protocol (or more) at every layer
- Service and a SDU
  - the interface between a layer and the layer above - SAP (Service Access Point)
  - the interface data is called a SDU (Service Data Unit)
- Connection
  - a protocol is connection oriented if the peer entity must be synchronized before exchanging useful data (connection set up); otherwise it is connectionless.

A **protocol** is the formal definition of external behaviour for communicating entities. It defines:

- format of PDUs
- rules of operation (PDU sent, data delivered, abort)

Examples of protocols are:

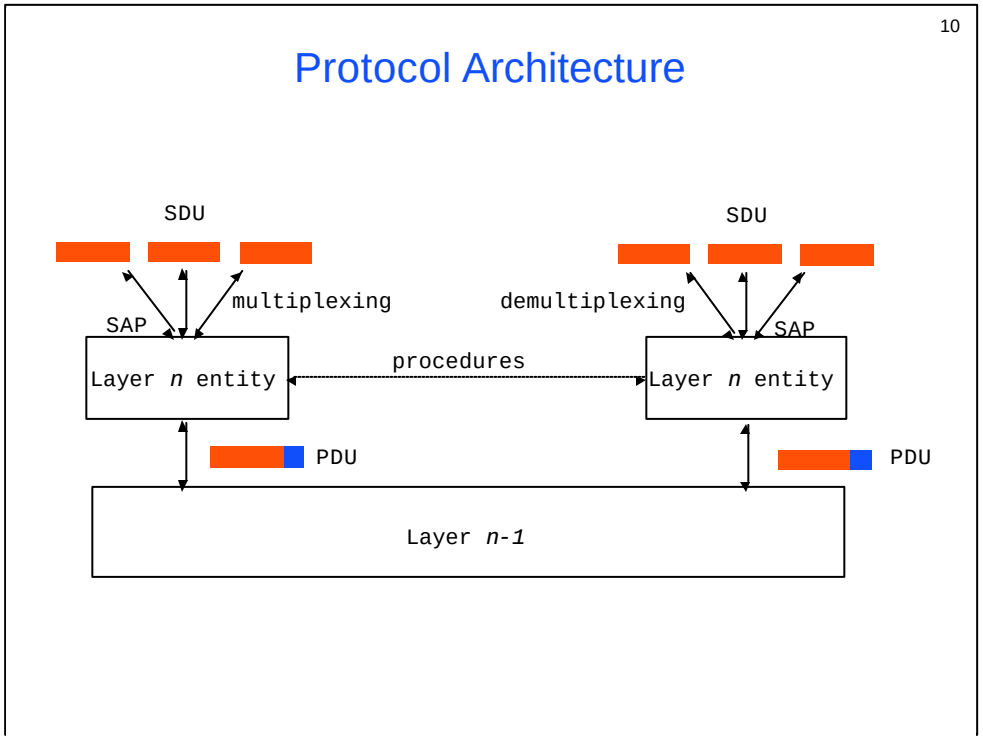
TCP

UDP

IP

Ethernet

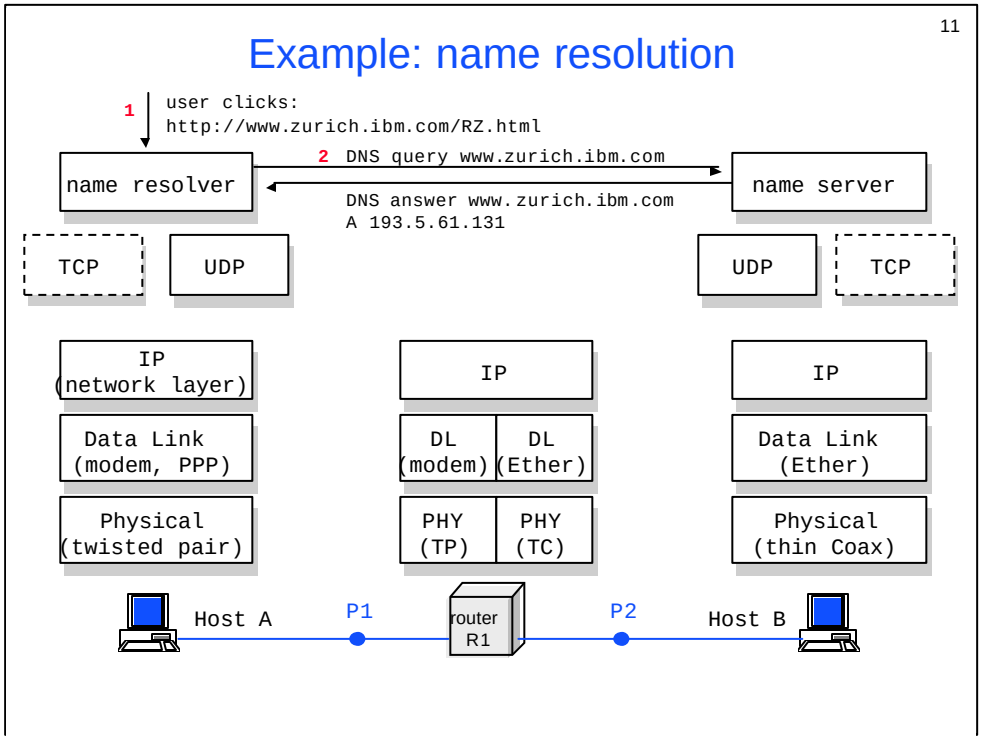
Protocols are connection oriented or connectionless. A connection exists if the communication requires some synchronization of all involved parties before communication can take place. The telephone system is connection oriented: before A can send some information to B, A has to call B (or vice versa) and say “hello”. The postal (mail) system is connectionless. If A wants to send some information to B, A can write a letter and mail it, even if B is not ready to read it.



Networking functions are structured as a layered model:

- layer *n* communicates with other layer *n* entities using layer *n* **PDUs**
- layer *n* uses the **service** of layer *n-1* and offers a service to layer *n+1*.
- entities at the same layer are said **peer entities**
- operation rules between peer entities are called **procedures**

Layering of protocol entities is reflected by the term of a **protocol stack**.



Flow 2 illustrates the **query/response protocol** of the Domain Name System (DNS). The **name resolver** and the **name server** are two application programs, probably C programs making calls to the socket library. The programs use UDP, which is the non-reliable transport protocol in the TCP/IP stack.

Let us apply the terminology on this example.

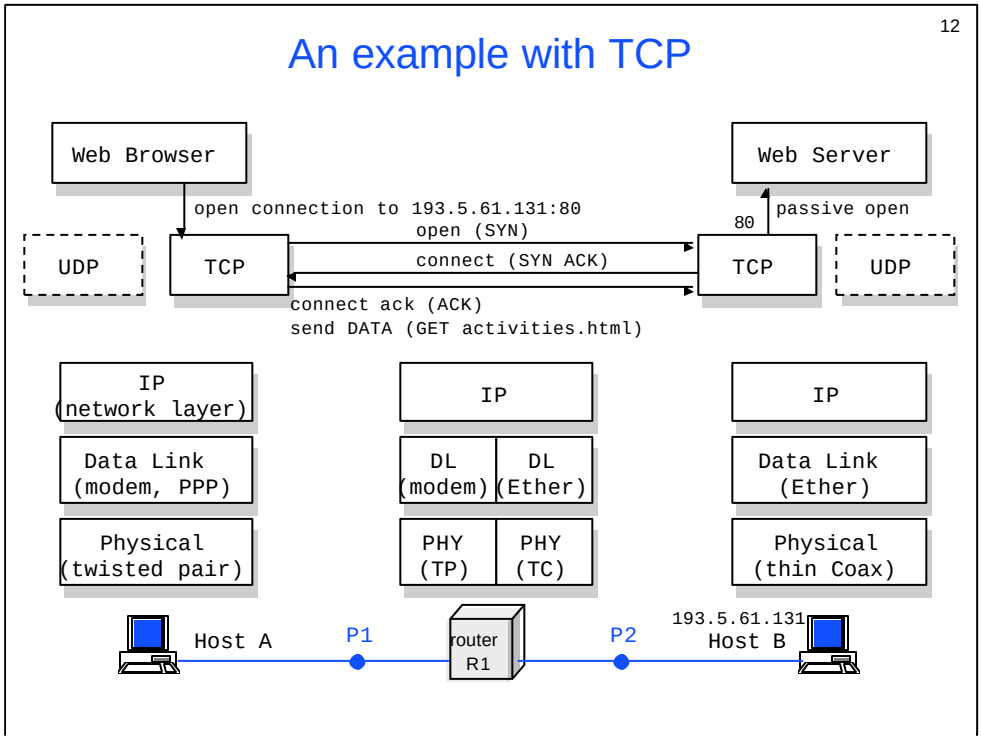
“name resolver” uses the UDP service: it creates a request to send data to “name server”. “name server” is identified by its IP address (for example: 128.178.15.7). “name resolver” also knows that “name server” can be reached by means of port 53 (a well known convention used in the Internet). The SDU is the request, with the data. The transport-PDU is called a datagram. It contains the data, the address and the port numbers. It is identified by 2 in the figure.

UDP creates a request to IP to send data to the name server machine identified by the IP address 128.178.15.7. The network-PDU is called an IP packet. It contains the UDP datagram plus the IP addressing information (and some other information, see later).

IP creates a request to send a data frame over the modem. The modem card creates a data-link PDU, called a modem “frame”. The frame contains the IP packet, maybe compressed. Then the data link layer requests transmission of the frame; the physical layer SDU is a bit. The physical layer PDU is an electromagnetic signal.

At the router

- the data frame is received, understood as an IP packet
- IP reads the IP destination address (128.178.15.7) and decides to forward it over its Ethernet interface
- IP creates a request to send the data frame over the Ethernet. An Ethernet frame is created and sent to the name server machine



Here is a second example.

A web browser always uses TCP for communication with a web server.

The web browser starts by requesting from the transport layer the opening of a connection for reliable data transport. TCP opens a connection to the peer entity at the web server machine by starting a 3-way handshake. If the connection can successfully be opened, then data can flow between the web client and server. TCP monitors missing packets and retransmits them as appropriate.

The web browser and server can thus assume that they have a reliable data pipe between them transporting data in sequence and without errors, at least as long as the TCP layer does not close the connection.

TCP is connection oriented. What is shown is the connection setup phase. TCP uses IP, which is connectionless. UDP is connectionless.

An observer at P1 or P2 would see the beginning of the message between web clients and servers only in the third data frame.

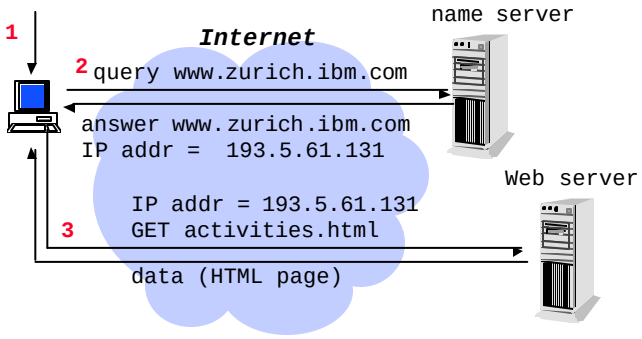
13

What is the Client-Server model?

distributed applications use the client-server model

- **server** = program that awaits data (requests) to be sent to it
  - interprets a request and send a response
- **clients** send data (requests) to servers
  - wait for a response

```
user clicks:
http://www.zurich.ibm.com/activities.html
```

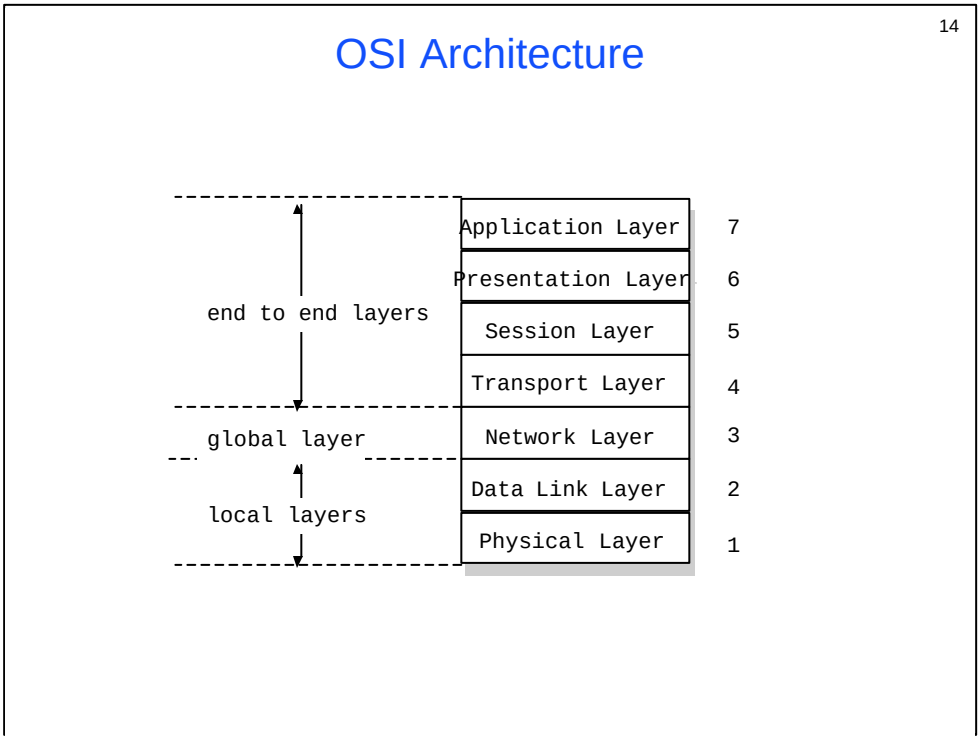


We use the terms “client” and “server” in the following sense.

When two entities say A and B, want to communicate, there is a boostrap problem: how can you initialize both A and B such that the communication can take place. One solution is to manually start A, then B, but this defeats the purpose of networking. The only way we have found so far is to request that one of the two, say B, is started and immediately puts itself in a **listening** position. We say that B is a server. A system, such as A, which talks to B, is said to be a client.

Being a server or a client is relative to a given protocol. For example, consider the application level protocol called FTP (file transfer protocol). The FTP server is a machine that waits for other machines to send requests for logging in. When an FTP client has contacted an FTP server, then after an initial navigation phase, the FTP client has to wait for the FTP server to open a connection back to the client (try it !). In that interaction, the FTP client is a TCP server, namely, a machine which waits for some other machine to open a TCP connection.

In everyday’s life, most people use the term “server” to designate a machine whose main function is to be a server for some protocol: a name server, a file server, a news server ...



An **architecture** is a set of external behaviour specifications for a complete communication system. It describes protocols, but not how to implement them.

The **OSI (Open Systems Interconnection) architecture** defines protocols and service specifications.

It is an official standard, similar to the TCP/IP architecture, but is not much implemented. However, the OSI *model* is used most frequently to describe all systems, including TCP/IP

Architectures do not interoperate by themselves at the protocol level. For example, the OSI transport protocols are not compatible with TCP or UDP. Worse, there is no compatibility at the service level, so it is not possible to use layer *n* of one architecture and put it on top of layer *n-1* of some other architectures.
There are fortunately exceptions to this statement. Layer interfaces where service compatibility is often implemented are:

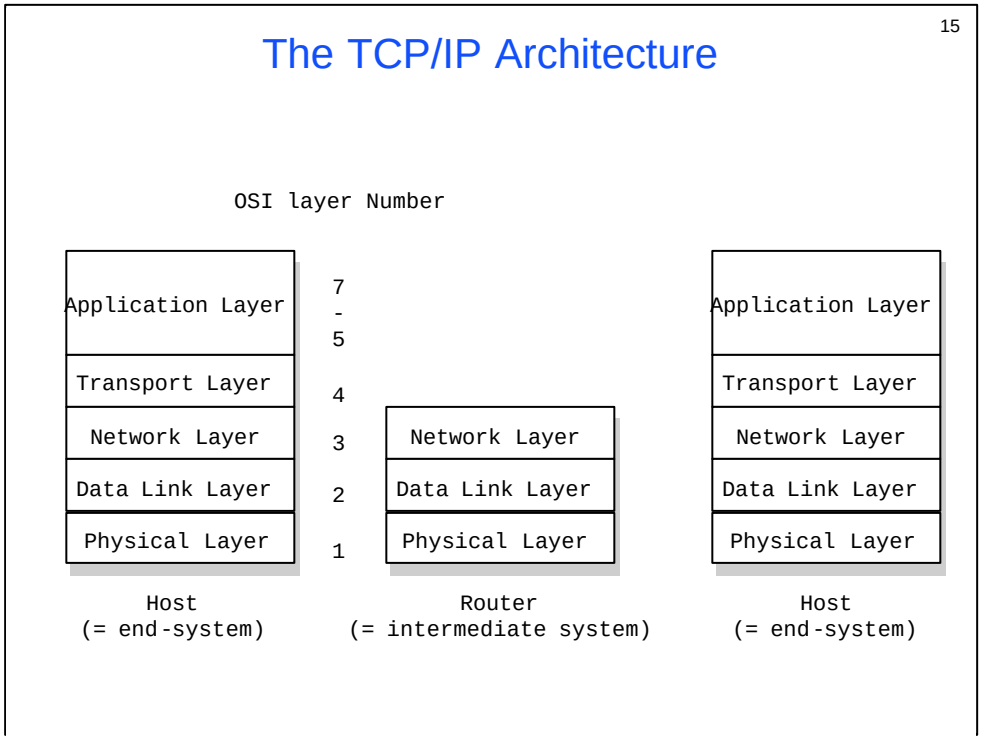
the data link layer

the transport layer.

For example, it is possible to use various protocol families over the same local area network (LAN).

The OSI presentation layer is in charge of hiding specific data representation formats. It defines ASN.1, an abstract, universal means for encoding all types of data structures. ASN.1 has also become part of the TCP/IP architecture, in the application layer.

The OSI session layer synchronizes events between end-systems, in order for example to support failure recovery. It is implemented in TCP/IP over a number of application layer protocols and TCP.



The **TCP/IP Architecture**, or the **Internet Architecture** is described by a collection of Internet standards, published in documents called RFCs (Requests For Comments), available for example from ftp://ftp.switch.ch/standard.

The picture shows all the layers of the Internet Architecture. There exists, inside every layer, a number of protocols that we will discover in this course.

There exist other architectures, each of them having a different set of layers and names for layers. There are:

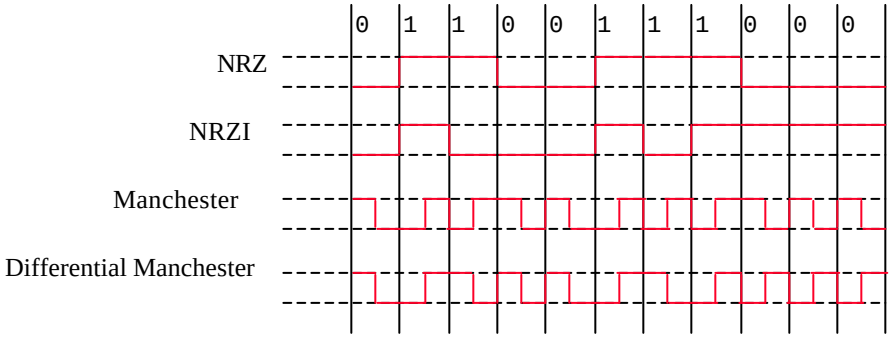
- proprietary architectures: SNA (IBM), Decnet (Digital), AppleTalk (Apple), XNS (Xerox), UUCP (Unix internal protocols), etc
- the ITU architecture defines public networks for telephony, telex, fax, data networks (X.25, Frame Relay, mail and directory services) and ATM
- the IEEE LAN architecture defines layers 1 and 2 for local area networks. We will see some details later.

Having several architectures is a nuisance; everything would be simpler if there would be only one. Today, the TCP/IP architecture has become dominant, so this is the only one we will study in detail. The ITU architecture (Frame Relay and ATM) does also play an important role and we will study it at the end of the course.

16

Physical Layer

- layer 1 function is to transmit/receive a sequence of bits on electrical or optical system
  Bits are encoded as analog signal; here are examples different channel coding
  (Ethernet uses Manchester encoding)



- modulation
  - adapt signal to fit a channel, eg. modems (telephone frequency band 300 - 3400 Hz)
  - amplitude, frequency, phase modulation, or hybrid (amplitude and phase)

We see here some rudiments of transmission. The diagram shows some very primitive channel encoding methods. They are used on short distances, for example with Ethernet or Token Ring.

17

Bit Rates

- **Bit Rate** (*débit binaire*, *Bitrate*) of a transmission system = number of bits transmitted per time unit

  units: **b/s**, kb/s = 1000 b/s, Mb/s = 10e+06 b/s, Gb/s=10e+09 b/s
- Shannon-Hartley law: $C_{max} = B \log_2 (1 + S/N)$, with B = bandwidth (Hz), S/N = signal to noise ratio (not expressed in dB)

  example: telephone circuit: B = 3 kHz, S/N = 30 dB, C_{max} - 30 kb/s
- Practical Bit Rates:
  - modem: 2.4 kb/s to 33.6 kb/s (56kb/s on reception over ISDN at server), 9.6 kb/s GSM
  - ISDN line: 124 kb/s to 2 Mb/s
  - Ethernet: 10 Mb/s, 100 Mb/s, 1Gb/s
  - Token Ring 4 Mb/s, 16 Mb/s
  - FDDI: 100 Mb/s
  - ATM: 2 Mb/s to 622 Mb/s
  - SDH: 155 Mb/s to 2.4Gb/s
- Transmission time = time to send x bits at a given bit rate
- Example: time to send 1 MB at 10 kb/s = ?

The bit rate of a channel is the number of bits per second. The bandwidth is the width of the frequency range that can be used for transmission over the channel. The bandwidth limits the maximal bit rate that can be obtained using a given channel.

In general, the information theory gives the maximum bit rate available under some modelling assumptions. The Shannon-Hartley laws gives the maximum bit rate, for a given bandwidth, assuming the channel is a white noise channel.

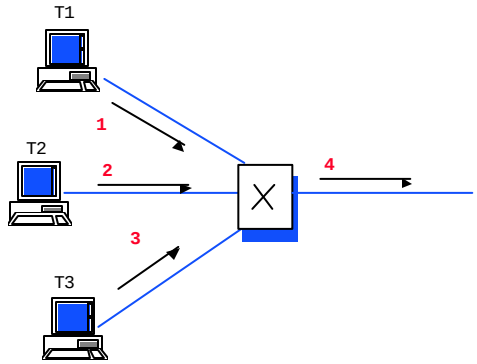
Many people confuse bandwidth and bit rate, but you should keep the distinction.

The bit rate and the number of bits to transmit determine the transmission time.

Statistical and Non-statistical Multiplexing

18

- Multiplexing
  - several sources use the same link
- Statistical Multiplexing
  - the bit rate is less than the sum of the incoming bit rates
  - may produce packet loss; requires congestion control



Multiplexing means putting several sources on the same link. The most common ways of multiplexing is by sharing time slots (temporal multiplexing) or frequency bands (frequency multiplexing). Temporal multiplexing is used in many telecommunication networks - in circuit switching each source is allocated with one time slot. When a source does not use the link during its slot, the available capacity is unused.

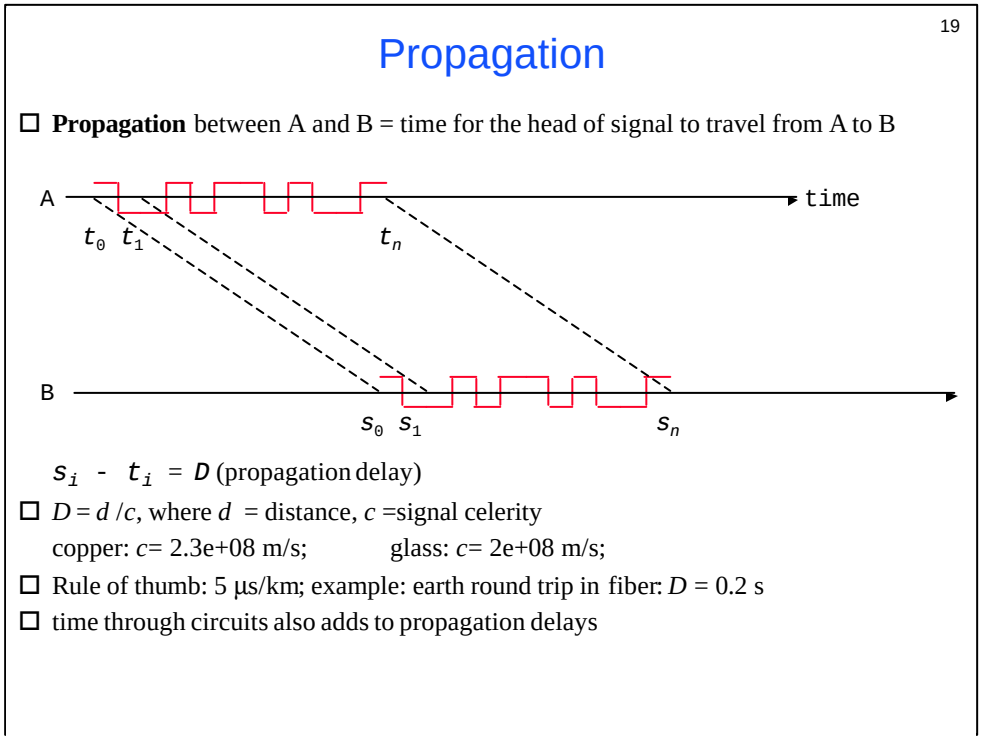
In statistical multiplexing data units are stamped with identifiers so that a source may send data at will.

On a packet switch, the bit rate of the output (4) is often less than the sum of the bit rates of all inputs (1 to 3). There is a queue at the output; if several packets arrive at the same time, then only one of them is transmitted whiles others have to wait. If nothing special is done, then once in a while, the queue may overflow and packets are lost. This happens everyday on the Internet. Special mechanisms, called congestion control, are required to avoid that packet losses happen too frequently. Congestion control is the object of the advanced lecture on networking.

In contrast, with circuit switching, the bit rate of the outgoing circuit (4 on the picture) is at least equal to the sum of the incoming circuits bit rates (1 to 3). There is no loss of data.

What is the value of statistical multiplexing ?

Well, economy. Most of the time, sources are not active, so circuit switching tends to waste bit rates.



Propagation is the time taken by the front of a signal to reach the destination. **It is independent of the bit rate.**

Propagation of an electro magnetic signal is the speed (also called celerity) of light. It depends on the wavelength and the element in which the signal is propagating.

Acoustic waves move at ca. 300 m/s. What is the propagation time if we use an acoustic phone system between two cities which are 1000 km apart ?

20

Examples

- At time 0, computer A sends a packet of size 1000 bytes to B; at what time is the packet received by B for each of the following cases ?

distance	20 km	20000 km	2 km	20 m
bit rate	10 kb/s	1 Mb/s	10 Mb/s	1 Gb/s
1-way propagation	?			
transmission	?			
reception time	?			

Compute the values for these examples and try to find scenarios where they apply. Meditate the results.

Examples

- At time 0, computer A sends a packet of size 1000 bytes to B; at what time is the packet received by B (c = 2e+08 m/s)?

distance	20 km	20000 km	2 km	20 m
bit rate	10 kb/s	1 Mb/s	10 Mb/s	1 Gb/s
1-way propagation	0.1 ms	100 ms	0.01 ms	0.1µs
transmission	800 ms	8 ms	0.8 ms	8 µs
reception time	800.1 ms	108 ms	0.81 ms	8.1 µs
	modem	*satellite*	*LAN*	*Hippi*

22

Throughput

- **Throughput** (*am* thruput, *f* débit utile, *g* Durchsatz) for a transmission system or a communication flow =

  number of useful data bits / time unit

  units:

  b/s, kb/s, Mb/s
- Example 1:

  PCM voice (8 kHz, 8 bits per sample -> 64 kb/s)

  throughput = 64 kb/s
- Example 2: Stop and Go protocol

The throughput defines how much data can be moved by time unit. It is equal to the bit rate if there is no protocol (example 1). However, in most practical cases, the throughput is less than the bit rate for two reasons:

- protocol overhead: protocols like UDP use some bytes to transmit protocol information. This reduces the throughput. If you send one-byte messages with UDP, then for every byte you create an Ethernet packet of size 1 + 8 + 20 + 26 = 53 bytes, thus the maximum throughput you could ever get at the UDP service interface if you use a 64 kb/s channel would be 1.2 kb/s.

- protocol waiting times: some protocols may force you to wait for some event, as we show on the next page.

23

A Simple Protocol: Stop and Go

- Packets may be lost during transmission:
  bit errors due to channel imperfections, various noises.
- Computer A sends packets to B; B returns an acknowledgement packet immediately to confirm that B has received the packet;

  A waits for acknowledgement before sending a new packet; if no acknowledgement comes after a delay *T1*, then *A* retransmits

Question: What is the maximum throughput assuming that there are no losses ?

notation:

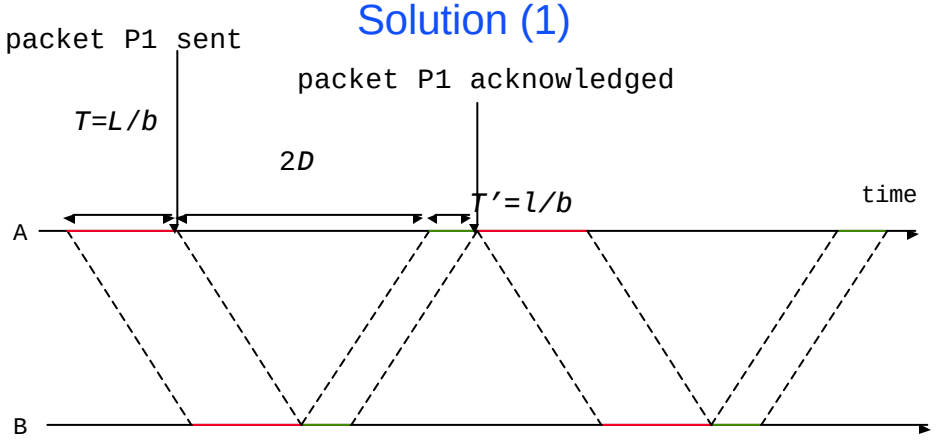
- packet length = *L*, constant (in bits);
- acknowledgement length = *l*, constant
- channel bit rate = *b*;
- propagation delay = *D*
- processing time = 0

This example is a simple protocol, often used, for repairing packet or message losses. The idea is simple

- identifiy all packets with some number or some other means
- when you send one packet, wait until you receive a confirmation
- after some time, if no confirmation arrives, consider that the packet has been lost and retransmit.

Compute the maximum throughput of this protocol, assuming the source has an infinite supply of packets to send, the destination generates the confirmation instantly, and the bit rate of the channel is constant.

Solution (1)



cycle time = $T + 2D + T'$

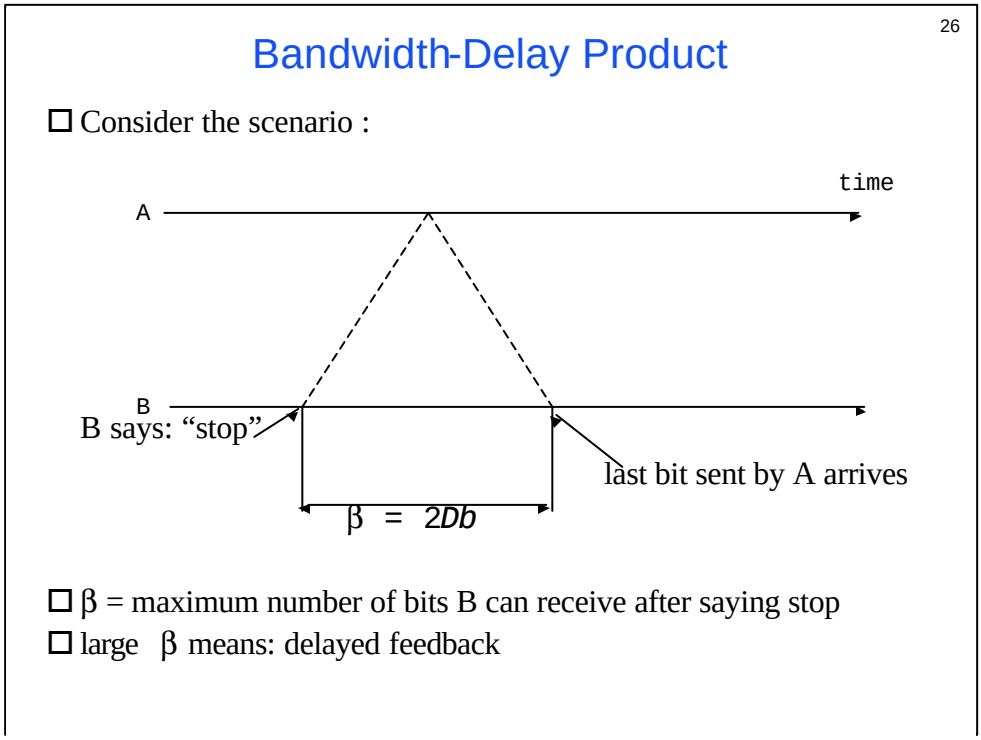
useful bits per cycle time = L

throughput = $Lb / (L + l + 2Db) = b / (\omega + \beta/L)$

with $\omega=(L+l)/L$=overhead and $\beta=2Db$=**bandwidth-delay product**

Solution (2)

distance	20 km	20000 km	2 km	20 m
bit rate	10 kb/s	1 Mb/s	10 Mb/s	1 Gb/s
propagation	0.1ms	100 ms	0.01 ms	0.1µs
transmission	800 ms	8 ms	0.8 ms	8 µs
reception time	800.1 ms	108 ms	0.81 ms	8.1 µs
	modem	*satellite*	*LAN*	*Hippi*
bw delay product	2 bits	200 000 bits	200 bits	200 bits
throughput = $b \times$	99.98%	3.8%	97.56%	97.56%



As an illustration of the effect of propagation, consider the scenario above.

The number β is called the “bandwidth”-delay product (why with quotation marks?). It expresses the latency (in terms of number of bits) of a channel. We will find it important in the rest of the lecture.

Facts to Remember (this chapter)

- Computer networks are organized using a layered model
- There is one layered model per architecture
  - ex. TCP/IP, Appletalk, Novell Netware, OSI
  - but the numbering is standard (1 to 7)
- Layers 1 and 2 correspond to cables (or wireless channels)
- Layer 3 = network layer; has mainly intermediate systems, eg. routers
- Layer 4 = transport; is in end systems only
- UDP provide the simplest access to network services
- Layer 5-7 is the application layer (web, e-mail, etc)
- Concepts you should know
  - protocol, peer entities, PDU, service
  - transmission time versus propagation time
  - bandwidth delay product