Networking

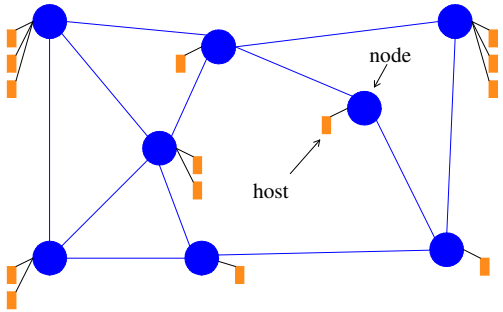
- Computer Networks
- Packet Switching
- Flow Control
- Saltzer's end-to-end-argument
- Data Communication
- Open Systems
- OSI Model
- Local Networks
- Logical views
- Programming Constructs for Communication

Different Topological Solutions

- general *mesh networks*
- ring *loop networks*
- common (shared] medium *bus networks*

Mesh Networks

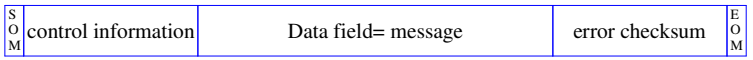
- *packet-switched*
  - asynchronous (statistic) multiplexing



Packet Switching

What to be transported can be seen as a file of data, information about the file and its data which actually consists of a number of 1:s and 0:s.

- The file data is divided by the sender into one or more pieces called messages which normally have a maximal allowed size.
- Each message is wrapped into a packet — control information + data + error checksum
- Each packet is sent individually to the receiver.
- The receiver retrieves the messages from the packets and assembles them into the original file



Data packet

Packet Switching (cont.)

- *Mesh Networks*
  - *Store-and-Forward*

    Each packet is received wholly before it is sent further.
- Local Area Networks

  Some local area networks use
  - *Check-and-Forward*

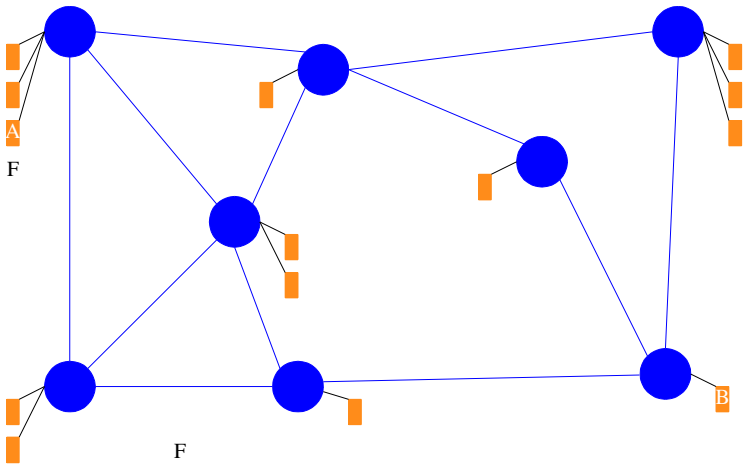
    Each packet bit is sent further as soon as possible.
    This gives much higher speed.

Terminology

- Node Computer
  Belongs to the network.
- Host
  Computer using the network.
- Routing
  How to find the way to the receiver host.
  - Routing tables
    - static
    - dynamic
- Flow Control
  To assure that all messages is actual received without errors by the receiver and that the receiver is not overloaded.
- Protocol
  Rules for how to achieve Flow Control and Routing.

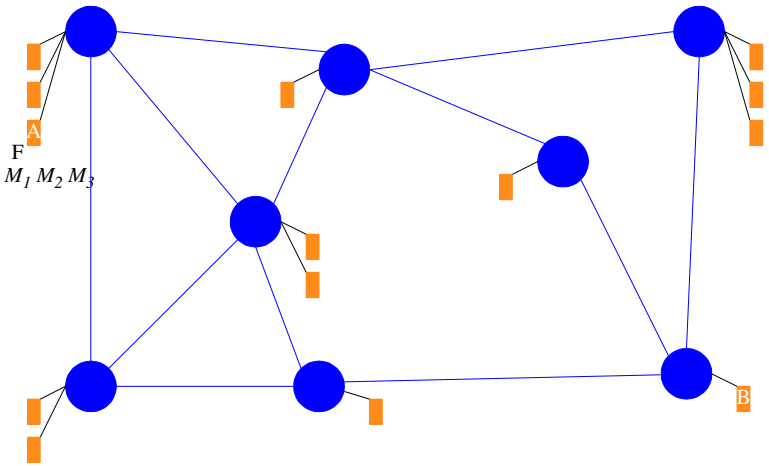
Mesh Network example

An example that shows most of the typical behavior of computer communication.



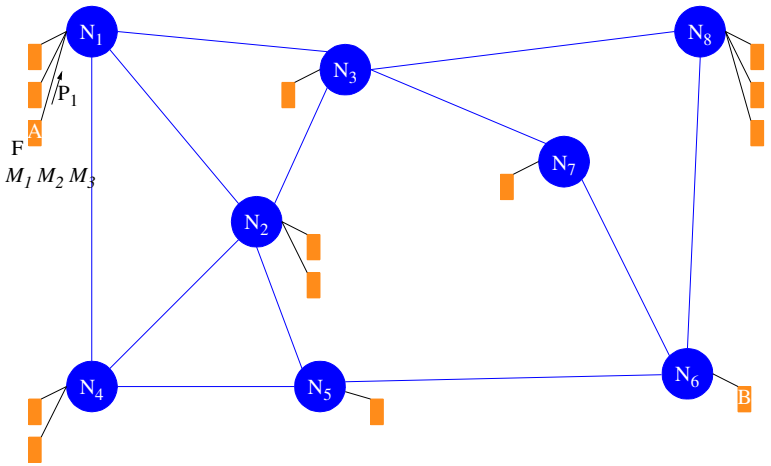
F

File that we want to copy from A to B



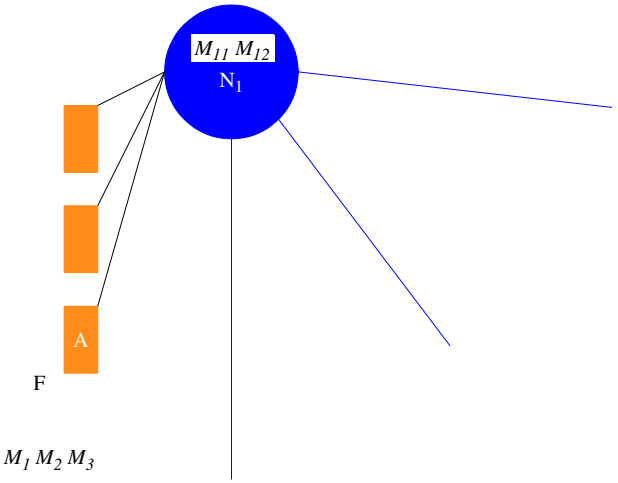
F = $M_1 + M_2 + M_3$

the file is divided into messages by A



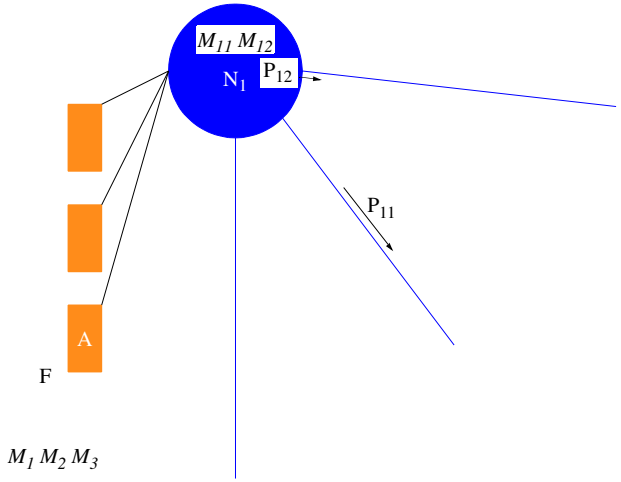
P_1 = < control, M_1, Check>

Each message is wrapped into a packet and is sent to A's node in the network.



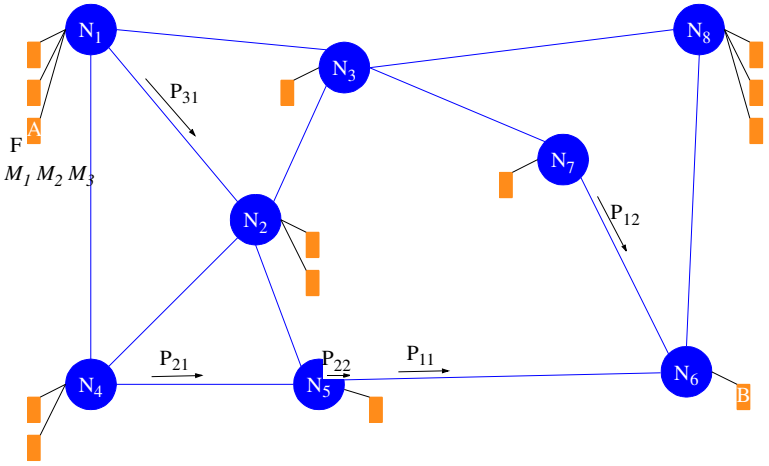
P_1 = M_{11} + M_{12}

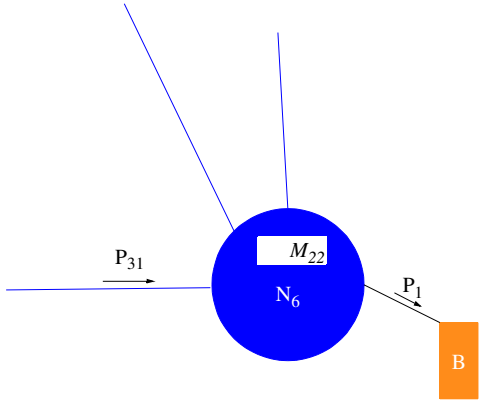
Each packet arriving to N_1 is divided into sub messages according to network internal max-length



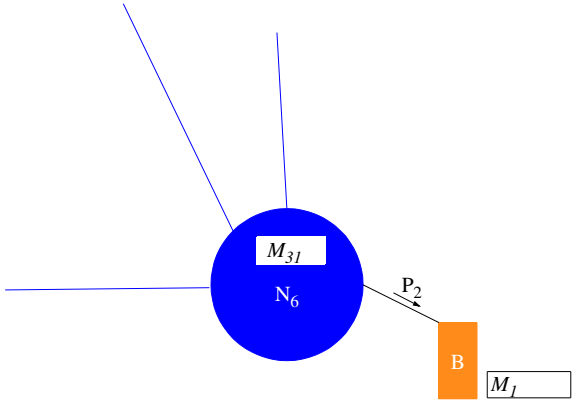
P_{11} = < control, M_{11}, Check>

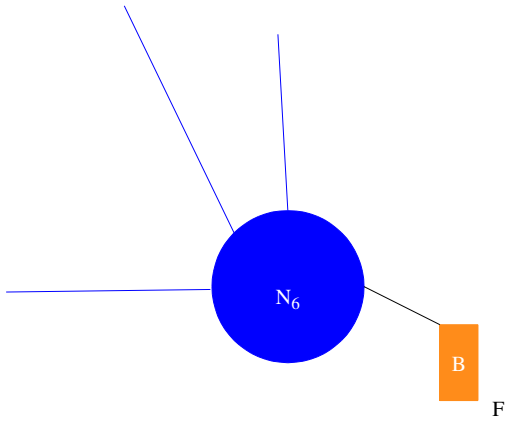
The sub messages are put into packets which are sent to B's network node, N_6.



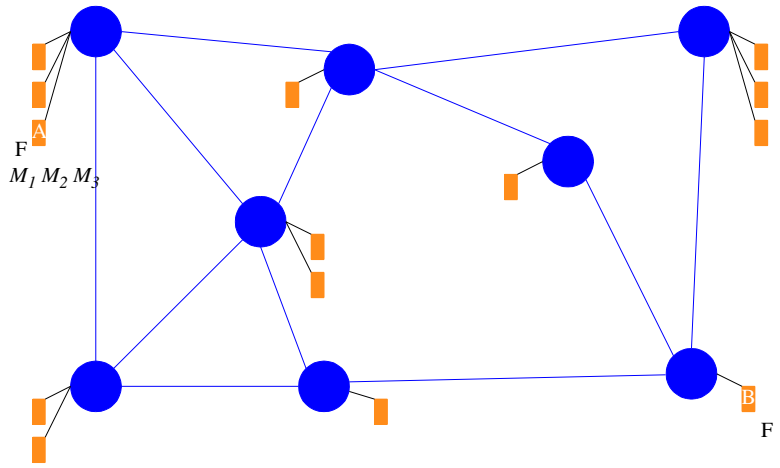


The sub messages are assembled by N_6 into the packets sent by A.
Then the packets are sent to B.





B assembles the messages into the original file F.



Fundamental Characteristics of Computer Communication

Messages might

- be divided,
- sent different ways on different underlying networks,
- have to be (partly) re-sent after errors.

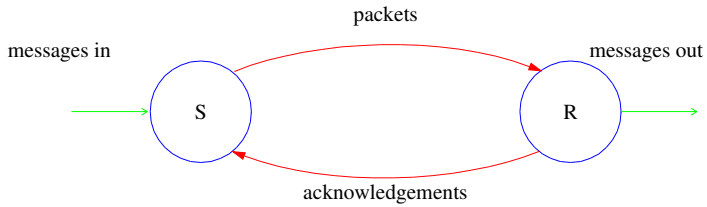
This leads to the following observation:

- Fundamental characteristics of computer communication as seen by the processes in a distributed system:
  - when sending a message the sender can't know when it is going to be received by the receiver.
  - when receiving a message the receiver can't know when it was sent by the sender.

  this is the fundamental obstacle for synchronization in a Distributed System and thus for distributed control.
- Another problem is to treat concurrent threads in a proper way:
  - provide mutual exclusion
  - avoid deadlocks
  - avoid starvation

Flow Control

- Controls the flow of packets (messages) between two units:
  - the packets should be delivered in the same order as they are sent.
  - no packet should be lost
  - no packet should be duplicated
  - the receiver should not be overloaded
  - the communication should not be allowed to be locked (hanged), *deadlock*



Two units with two way connection.

Two Transfer Principles

- Connection-oriented transfer:
  - a connection is established using initiate message(s) will delay the start of data transfer.
  - then the data packets can be sent
  - short packet addresses gives smaller packets
  - guarantee that the packets will arrive in correct order

  X.25, TCP
- Connectionless transfer:
  Datagram:
  - the data packets are sent directly without establishing a connection no delay for data transfer
  - long packet addresses gives bigger packets
  - no guarantee that the packets will arrive at the receiver or even that the receiver exists.

  IP, UDP

Salzer's end-to-end-argument

- To get a reliable data transfer it is enough **and** necessary with check sums and acknowledgements on the very highest level of the communication layers.
  - cheap protocol, no extra overhead.
  - expensive resend when something goes wrong. Everything has to start again from the beginning.
- Checking at lower levels
  - expensive, many acknowledgement messages, overhead.
  - cheap resend when something goes wrong. Only part of the data has to be resent part of the distance.
  - performance issue,
    you have to consider:
    - the probability that there will be an error
    - the cost of resend compared to checking overhead
- The check at the highest level is a reliability and safety issue and **can not** be excluded.
  This must be done by the application, there is no way to let the network take care of this!

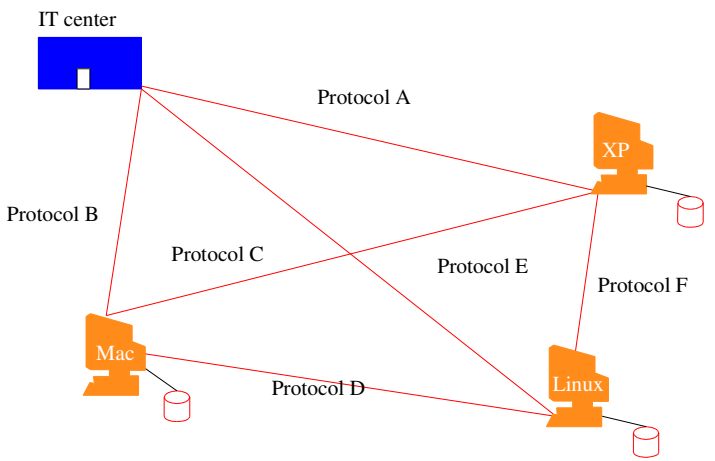
Computer Communication

It's all about standardization:

- Standardized protocols
- Standardized terminology for Computer Communication
  - ISOs reference model
    the OSI model (Open System Interconnect)
- Standardized Computer Networks
  - IEEE
- Standardized Middleware
  - CORBA
  - RMI
  - DCOM

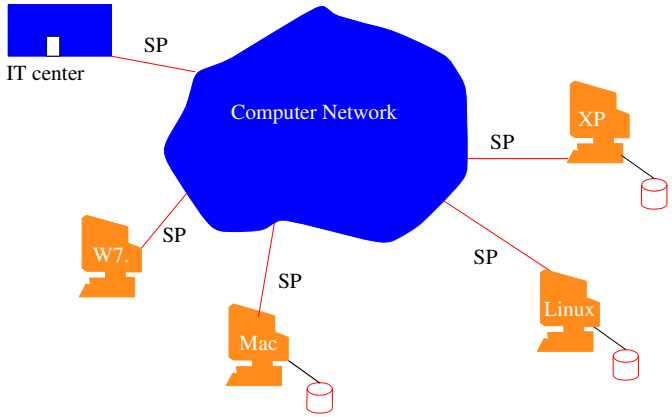
Open Systems

Not an open system:



Open System

Open System:



SP = Standardized Protocol

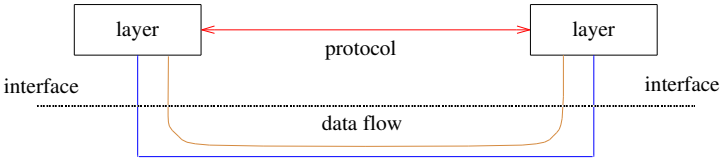
Open System

- System:
  - computers
  - programs
  - peripheral units
  - terminals
  - data communication links
  - operators
- Open system:
  - the system units can cooperate with other systems units according to official standards.

This does not mean that it has to cooperate,
It only means that it should be able to do so when it agrees to do so with another unit.

Layer Model

Communication rules



Protocol: A set of rules that are agreed for the communication between units.

Interface: Rules for how a unit can use another unit on the layer below.

ISOs Reference Model

OSI model, Open Systems Interconnect

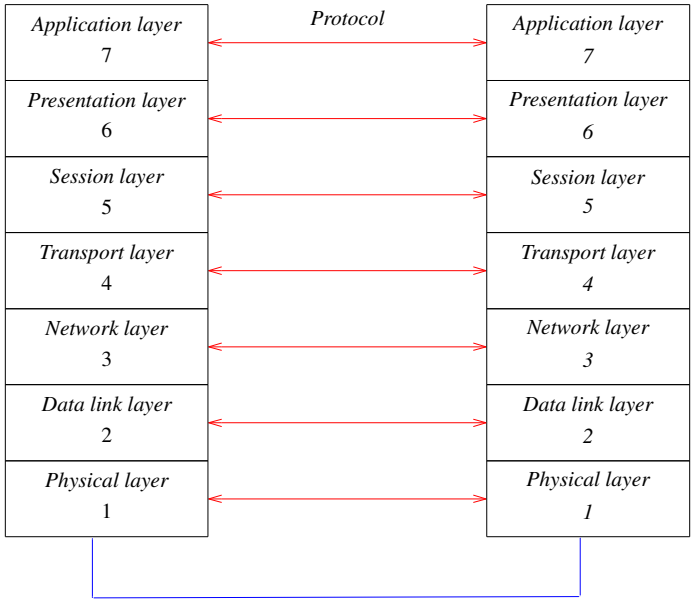
- Application layer:
  - Protocols for specific applications:
    - HTTP
    - FTP
    - SMTP
    - CORBA IIOP
- Presentation layer:
  - Standardized data transportation representation:
    - CORBA DATA
  - Encryption

- Session layer:
  - Reliability issues
    - detection of failures
    - automatic recovery
- Transport layer:
  - Transfer messages between communication ports associated with processes.
    Uses packet service from the layer below.
    Connection-oriented or connectionless transfer.
    - TCP
    - UDP
- Network layer:
  - Transfer packets between computers in a specific network.
    Routing
    - IP
    - X.25

- Data link layer:
  - Transmits packets between nodes that are directly connected with a physical link.
    - Ethernet MAC
    - X.25
- Physical layer:

  The electrical circuit and hardware connecting the nodes.
    - Ethernet baseband signaling
    - X.21

ISOs Reference Model

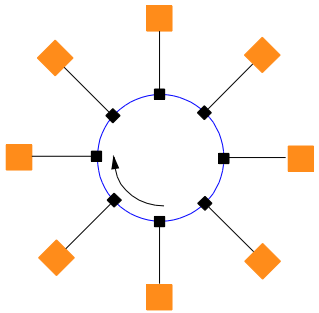


Local Networks

- When using networks in a local area they can by designed for higher performance.
  - Loop networks
  - Bus networks

Loop Networks

- Three main principles:
  - *control token loop*, "*Check-and-Forward*"
  - *register insertion loop*
  - *slotted loop*, "*Check-and-Forward*"



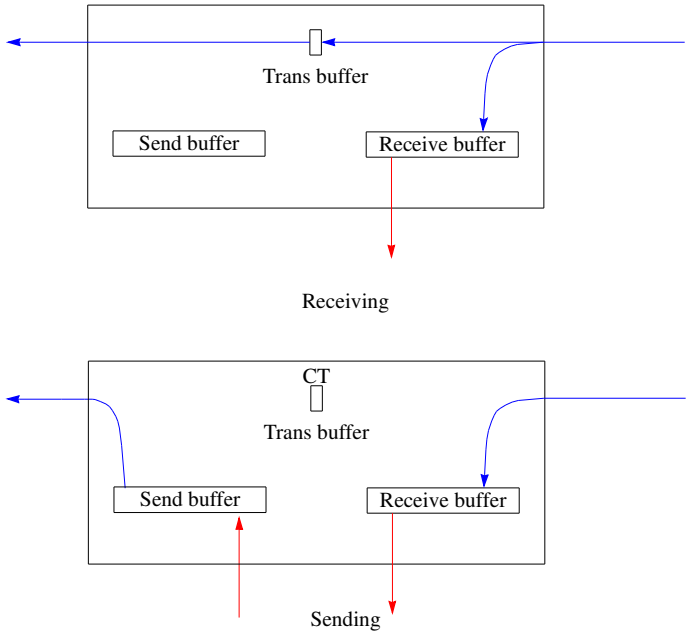
Control Token Loop

- only one bit in the node buffers, "*Check-and-Forward*"
- all nodes receives all packages
- only those with the node's host's address is sent further to the host
- each sender erases its packets from the loop
- the node holding the CT (Control Token) is allowed to send

- coaxial cable
- optical fibre

SOM	Destination Address	Source Address	Data	Ack.	CT

Example of packet format.



Bus Networks

- common medium
  - coaxial cable
  - twinned pairs
  - radio frequency - radio channel
- ALOHA-network
- ETHERNET - CSMA/CD-network

ETHERNET

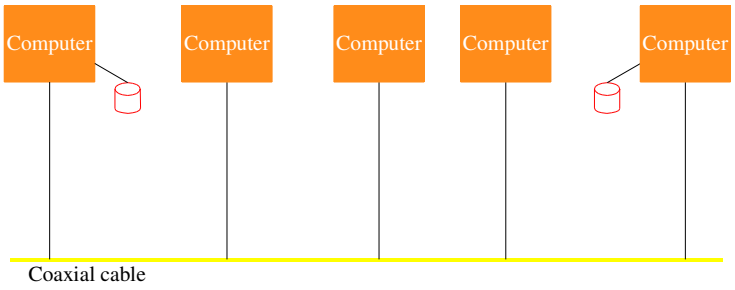
CSMA/CD

- *Carrier detection*

  Carrier Sense Multiple Access, CSMA
- *Interference detection*

  Collision Detect, CD
- *Packet error detection*
- *Truncated packet filtering* i hardware
- *Collision consensus enforcement*

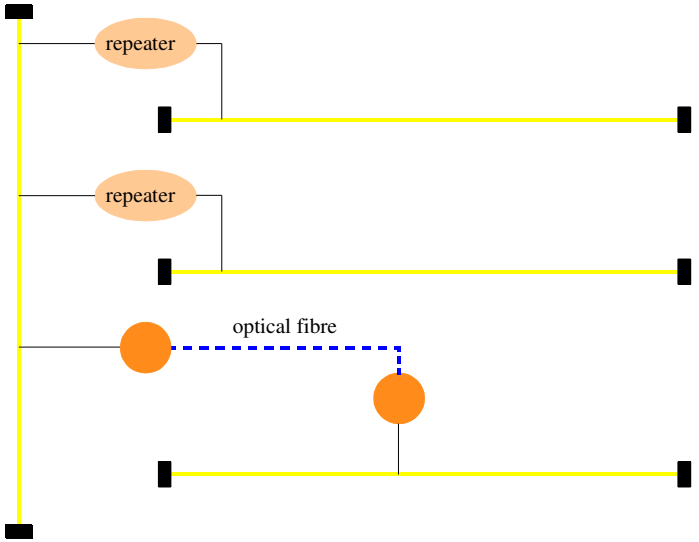
ETHERNET



- passive medium, a coaxial cable
- active component, *repeater*
- Minimum message size to fill up whole cable (*important*!)

ETHERNET

ETHERNET's segment, *repeaters*



Ethernet Messages

- What is important for us is that the standard proposes a minimum packet size which guarantees that every message will completely fill up the whole network.
  - If two messages starts at the same time from different ends of the network they will be perceived as colliding by all receiving nodes in the network.
  - There can only be one message successful message transfer at any time.
  - All nodes will receive all successful messages in the same order.
  - The network can be used as a synchronization tool
    Likewise the bus is used in a traditional computer.
- To assure this there was a new much higher minimum size when the standard allowed for higher speed and longer networks.

Broadcast Messages

- *broadcast message*
  is a message that will be received by all nodes
- *multicast message*
  is a message that will be received by a group of nodes

  Normally only *multicast messages* are used but by tradition they are often called *broadcast messages*

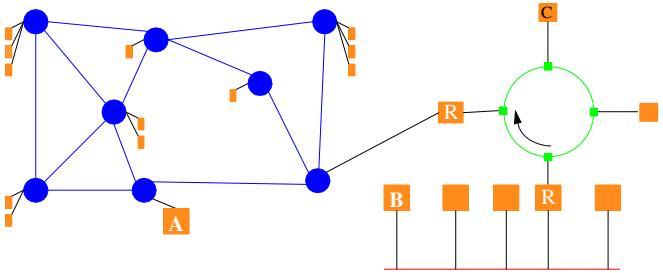
- In loop networks and bus networks broadcast/multicast is **not expensive**
- In general (mesh) networks broadcast/multicast is **expensive**

  Thus we might want to choose different distributed algorithms for bus networks than what is used in general mesh networks.

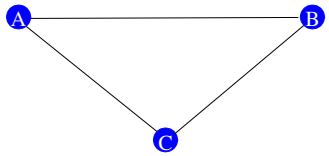
Connecting Computer Networks

Logical views

- If you have a network like this:



and let A, B and C cooperate using TCP/IP you will logically have a network like:



The protocol hides the underlying topology

Types of Communication

- Reliability
  - Persistent communication
    - guaranteed delivery
  - Transient communication
    - no guarantees
- Thread Control
  - Synchronous communication
    - wait for reply
  - Asynchronous communication
    - don't wait for reply

Programming Constructs for Communication

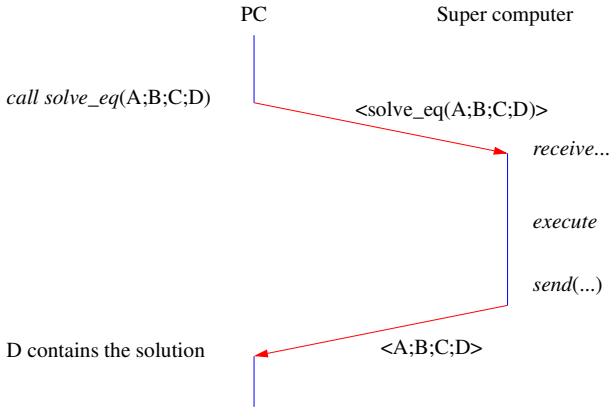
- Message Oriented:
  - using Ports from which you can send or receive

    send(*port_address*, *message*)

    or

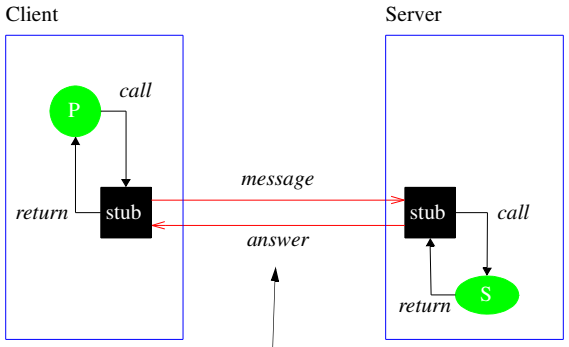
    receive(*port_address*, *message*) port addressing
  - Internet uses Ports called Sockets
- *Remote Procedure Call* RPC
- Object Oriented
  - Middleware CORBA DCOM
- Distributed Transactions

Remote Procedure Call



- Network Transparency
  - the PC program is the same as for a local call.
- Drawbacks:
  - no parallel work, the calling process is waiting without doing anything.

Remote Procedure Call



The data formats and special events (e.g. exceptions) must be standardized here

Asynchronous RPC

- The Client does not wait for the return message after sending an RPC message.
  - Promotes concurrency
  - Much more complicated to take care of errors.