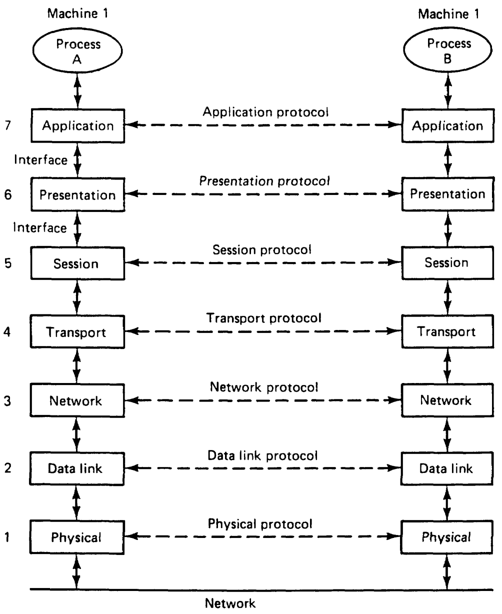
Network Communication

- Systems communicate according to a *protocol* — a set of rules that govern the sequence, format, content, and meaning of messages sent between the systems
- *Connection-oriented* communication
  - Information delivered as a *stream* of bytes, in correct order
  - Connect, exchange data, release
- *Connectionless* communication
  - Information delivered as a set of *packets*
  - Packets may be delivered out of sequence, must be reassembled
  - May be *reliable* — data will reach destination, otherwise sender will be notified of an error
  - May be *unreliable* — data may not reach destination, sender never notified of errors

Protocol Layers

- Network communication is divided up into seven layers
  - Each layer deals with one particular aspect of the communication
  - Each layer uses a set of routines provided by the layer below it
  - Each layer ignores lower-level (and higher-level) details and problems
- Each layer takes a message handed down to it by a higher layer, adds some header information, and passes the message on to a lower layer
  - Each layer has the illusion of peer-to-peer communication
  - Eventually the message reaches the bottom layer, and get physically sent across the network

ISO OSI 7-Layer Protocol



Distributed Operating Systems, Tanenbaum, Prentice Hall, 1995

ISO OSI 7-Layer Protocol Summary

- Application layer — provides network access to application programs
  - Telnet, ftp, email, web browsers
- Presentation layer — provides freedom from machine-dependent representations
- Session layer — provides communication between processes, error recovery
  - Not required in connectionless commun.
  - Example: Remote Procedure Call (RPC)
- Transport layer — reliably transfers messages (broken into *packets*) between hosts, error control for out-of-sequence and missing packets
  - Examples: TCP (connection-oriented), UDP (connectionless)

ISO OSI 7-Layer Protocol Summary (cont.)

- Network layer — provides switching and routing needed to (1) establish, maintain, and terminate switched connections, and (2) transfer data (packets) between end systems
  - Examples: IP (connectionless), X.25 (connection-oriented)
- Data link layer — reliably transfers packets (broken up into *frames*) over a communication link, error / flow control
  - Examples: Ethernet
- Physical layer —converts 1s and 0s into electrical or optical signals, and transmits frames of bits across a wire / cable
  - Examples: RS-232-C (serial communication lines), X.21

TCP / IP Protocol

- Upper layers
  - ftp — file transfer protocol
    - Sends files from one system to another under user command
    - Handles both text and binary files
    - Supports userids and passwords
  - telnet — remote terminal protocol
    - Lets a user at one terminal log onto a remote host
  - smtp — simple mail transfer protocol
    - Transfers mail messages between hosts
    - Handles mailing lists, forwarding, etc.
    - Does not specify how mail messages are created
  - nsp — name server protocol
    - Maps names into IP addresses
    - A domain may be split into subdomains
    - Name severs are usually replicated to improve reliability

TCP / IP Protocol (cont.)

- Transport layer (messages & packets)
  - TCP — Transmission Control Protocol
    - Connection-oriented (3-way handshake)
    - On transmit side, breaks message into *packets*, assigns sequence numbers, and and sends each packet in turn
      - Sends to a particular IP address and port
      - Flow control — doesn’t send more packets than receiver is prepared to receive
    - On receive side, receives packets, reassembles them into messages
      - Computes a checksum for each packet and compares it to checksum sent, discards packet if checksums don’t agree
      - Reorders out-of-order packets
    - Reliable
      - Packets must be acknowledged
      - If sender doesn’t receive an acknowledgment after a short period, it retransmits that packet
    - Congestion control — don’t overwhelm the network

TCP / IP Protocol (cont.)

- Network layer (routing packets)
  - IP — Internet Protocol
    - Connectionless
    - Unreliable
      - Packets may be lost, duplicated, or delivered out of order
    - Forward packet from sender through some number of gateways until it reaches the final destination
      - A *gateway* accepts a packet from one network and forwards it to a host or gateway on another network
    - Destination has specific Internet address, which is composed of two parts:
      - network part — network the host is on
      - address part — specific host on network
    - Routing is dynamic — each gateway chooses the next gateway to send the packet to
      - Gateways send each other information about network congestion and gateways which are down

TCP / IP Protocol (cont.)

- Data link / (packets & frames)
  physical layers (1s and 0s)
  - Ethernet
    - Connectionless
    - Unreliable
    - Network is a bus
      - Broadcast to anyone who cares to listen
    - Transmission
      - Carrier sense: listen before broadcasting, defer until channel is clear, then broadcast
      - Collision detection: listen while broadcasting
        - If two hosts transmit at same time —*collision* — the data gets garbled
        - Each jams network, then waits a random (but increasing) amount of time, and tries again
      - This is called CSMA/CD (carrier sense multiple access, with collision detection)
      - Frames contain checksum
    - Every Ethernet device (everywhere in the world!) has a unique address

Contention

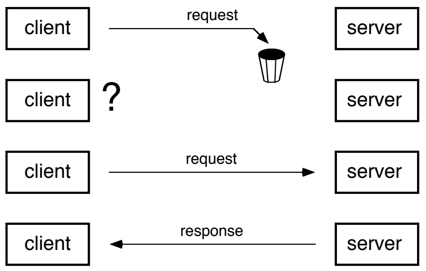
- Collision detection
  - Before sending a message, listen to see if another process is sending
    - If one is, wait a random time and try again
  - While transmitting, watch for collisions
- Token passing
  - A unique message (a token) continuously circulates through the network
    - To transmit, a host waits for a free token, attaches its message to it, sent the token status to busy, and sends it on
    - Destination removes the message, sets the token status to free, and sends it on
- Message slots
  - A number of fixed-length message slots circulate through the network
    - Wait for an empty slot and fill it

Failure Handling in Client / Sever Communication

- Potential failures:
  - Loss of request
    - Server never performs request
  - Loss of response message
    - Client doesn't know server performed request
  - Server may die or become unreachable
    - Did server perform request or not?
- 3-message reliable protocol:
  - Client sends request; blocks
  - Server sends reply; blocks
  - Client unblocks, sends acknowledgment; server unblocks
- 2-message protocol:
  - Client sends request; blocks
  - Server sends reply; client unblocks

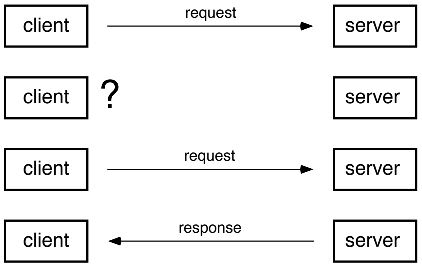
Semantics in Presence of Failure (Client Can't Locate Server, Lost Request)

- Client can't locate server
  - Reasons: server down, new version of server code
  - Can't just return error code always
  - Raise an exception (if supported)
- Lost request
  - Start timer after issuing request
    - If time expires, send request again
  - No problem if request was really lost

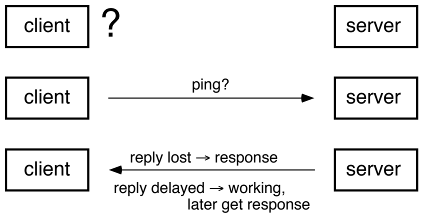


Semantics in Presence of Failure (Lost Request (cont.))

- Lost / delayed reply
  - OK to retransmit request **only** if remote procedure is *idempotent* (calling it multiple times is same as calling it once)

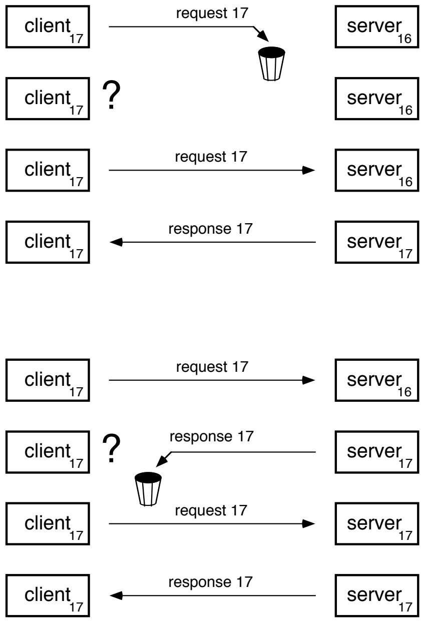


  - If not idempotent, be more conservative:



Semantics in Presence of Failure (Error Recovery — Sequence Numbers)

- More general solution: attach *sequence number* to every request and reply



Semantics in Presence of Failure (Server Crash)

- Possible scenarios
  - Request arrives, server crashes
  - Request arrives, request processed, server crashes
  - Request arrives, request processed, reply sent, server crashes
  - Desired response is different for each, but neither client nor server knows what it is
- Three (unattractive) alternatives:
  - Client keeps trying until it gets a response
    - Action carried out *at least once*
  - Client gives up and reports failure
    - Action carried out *at most once* (but maybe not at all)
  - Whatever…
    - No guarantees at all… easy to implement!
  - Ideal (unachievable)
    - Action carried out *exactly once*