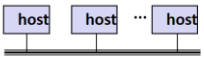
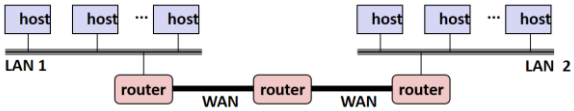
- A *network* is a hierarchical system of boxes and wires organized by geographical proximity
  - SAN (System Area Network) spans cluster or machine room
  - LAN (Local Area Network) spans a building or campus
  - WAN (Wide Area Network) spans country or world
- An *internetwork* (*internet*) is an interconnected set of networks
  - The Global IP Internet (uppercase "I") is the most famous example of an internet (lowercase "i")

- Simplified view of a LAN
  - A collection of hosts attached to a single wire



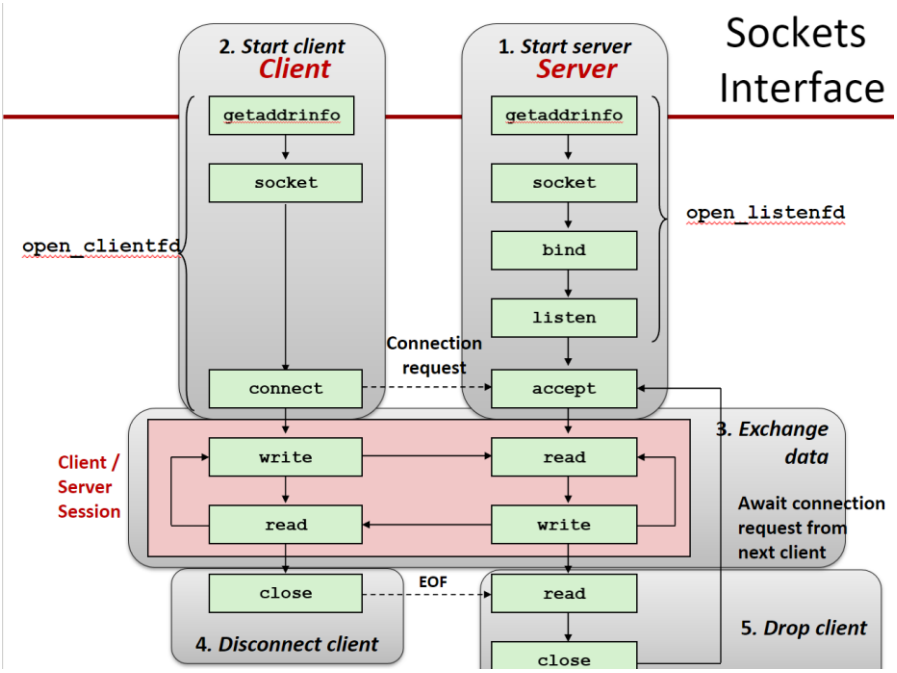
- An *internet* (lower case): multiple LANs connected by *routers*



LAN 1 and LAN 2 might be different and incompatible

- *Solution*: an internet *protocol* running on hosts and routers
  - Protocol is a set of rules that governs how hosts and routers should cooperate when they transfer data from network to network
- It provides a *naming scheme*
  - It defines a uniform format for ***host addresses***
  - Each host (and router) is assigned at least one of these internet addresses that uniquely identifies it
- It provides a *delivery mechanism*
  - It defines a standard transfer unit (***packet***)
  - Packet consists of ***header*** and ***payload***
    - **Header**: contains info such as packet size, source and destination addresses
    - **Payload**: contains data bits sent from source host

- Clients and servers communicate by sending streams of bytes over *connections*. Each connection is:
  - *Point-to-point*: connects a pair of processes
  - *Full-duplex*: data can flow in both directions at the same time
  - *Reliable*: stream of bytes sent by the source is eventually received by the destination in the same order it was sent
- *A socket* is an endpoint of a connection
  - *Socket address* is an `IPaddress:port` pair
- A *port* is a 16-bit integer that identifies a process:
  - ***Ephemeral port***: Assigned automatically by client kernel when client makes a connection request.
  - ***Well-known port:*** Associated with some *service* provided by a server
    - echo server: 7/echo
    - email server: 25/smtp
    - ssh servers: 22/ssh
    - Web servers: 80/http



Sockets Interface

Function	Parameters	Usage
socket	`int domain, int type, int protocol`	Clients & servers use the `socket` function to create a ***socket descriptor***
bind	`int sockfd, SA *addr, socklen_t addrlen`	A server uses `bind` to associate the server's socket address with a socket descriptor
listen	`int sockfd, int backlog`	A server calls the `listen` function so that a descriptor will be used by a server (for listening) rather than a client
accept	`int listenfd, SA *addr, int *addrlen`	A server waits for connection requests from clients by calling `accept`
connect	`int clientfd, SA *addr, socklen_t addrlen`	A client establishes a connection with a server by calling `connect`