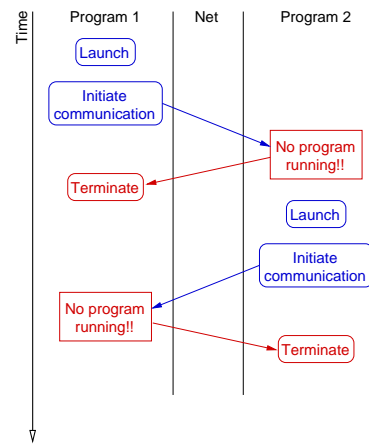
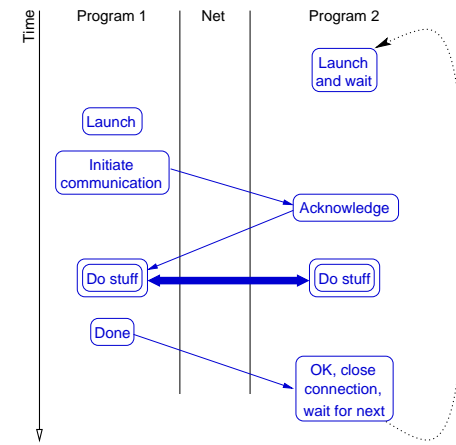


WHY CLIENT-SERVER APPLICATIONS

- TCP/IP provides **peer-to-peer** communication.
- We launch two programs and want them to communicate with each other.
 - Chances are, we will not be able to convince them to meet.
- So we split responsibilities:
 - One party (the **server**) must start execution and wait indefinitely for incoming requests.
 - So the other party (the **client**) will simply connect, knowing that somebody at the other end will listen.
- This way, we also simplify the TCP/IP mechanisms, which do not need to create programs or something equally hairy.



WHY CLIENT-SERVER APPLICATIONS (CONT'D)

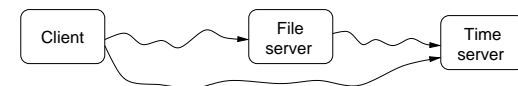


CLIENT ISSUES

- When connecting to a server, a client has to know the **address** of the machine and a **port number**.
 - Port numbers identify the actual server to connect to.
- **Standard** versus **nonstandard**.
 - In any case, the client must speak the server's language.
- Parameterization.
 - Some clients do one thing only, e.g., manage file transfers.
 - Some (**parameterized**) clients can access many services.
 - * telnet is a **fully parameterized client**.

SERVER ISSUES

- Connection or connectionless.
 - Connection-oriented servers assume that all the data packets arrive correctly and in order (TCP).
 - A connectionless server does not assume any delivery guarantee (there might be lost packets, duplicates, and out of order packets).
 - The application (both client and server) should contain code that deals with losses, duplication, etc.
 - Major design issue. TCP introduces some overhead, but is in general preferred because it simplifies design.
- Servers and clients (for other servers), e.g.,



- To keep or not to keep state information, that is the question.
- A **stateless** server does not remember what the client did, a **stateful** one does.
 - Stateless or stateful?
 - * File server, that allows clients to access a given piece of data from a given file.
 - * POP server, that allows clients to retrieve their email messages which have not been previously received.
 - * HTTP server for an e-commerce site.

- Statelessness is a protocol issue.
- A stateful server
 - may be more efficient
 - is difficult to maintain in case of loss of communication and/or computer crash
 - problems with identifying clients
- A stateless server
 - operations must be idempotent
 - copes well with loss of communication/computer crash