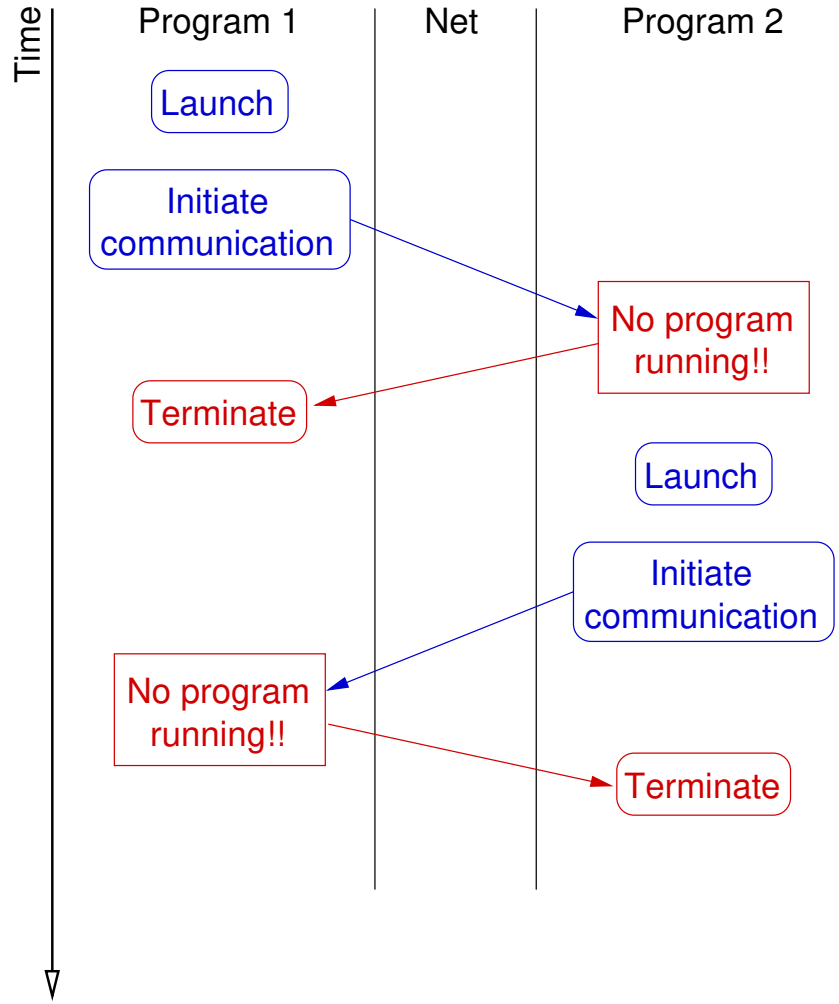
The client-server model

Stefan D. Bruda

CS 464/564, Fall 2023

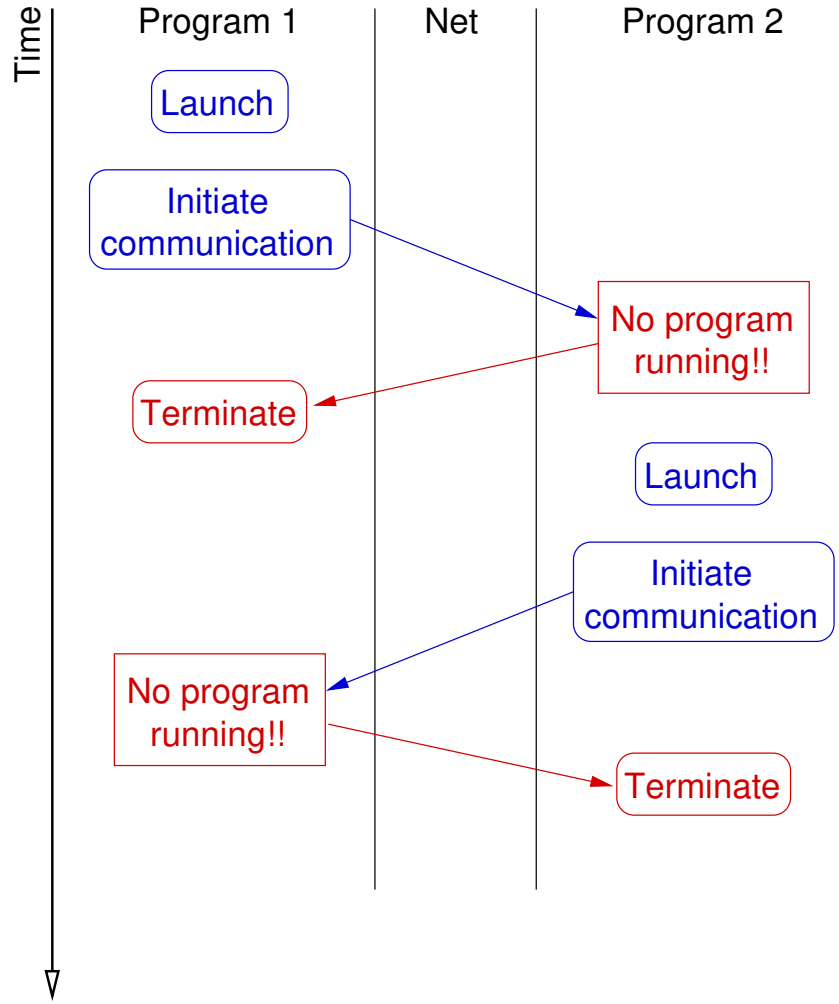
Why client-server applications

- 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



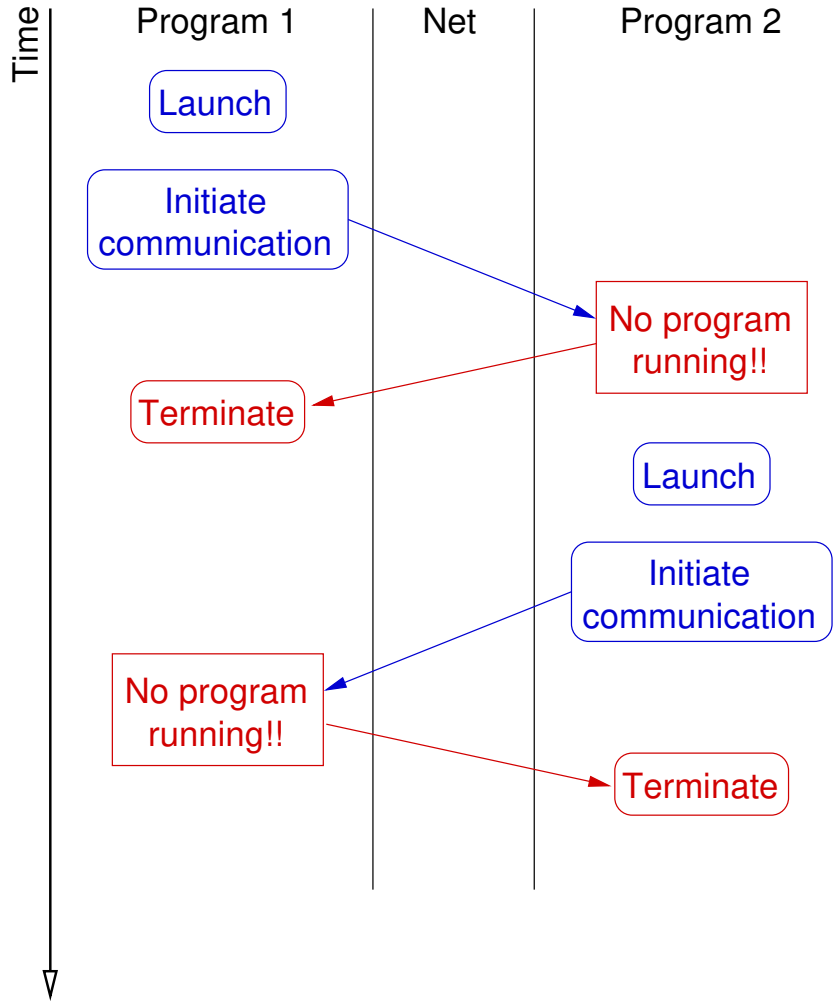
Why client-server applications

- 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

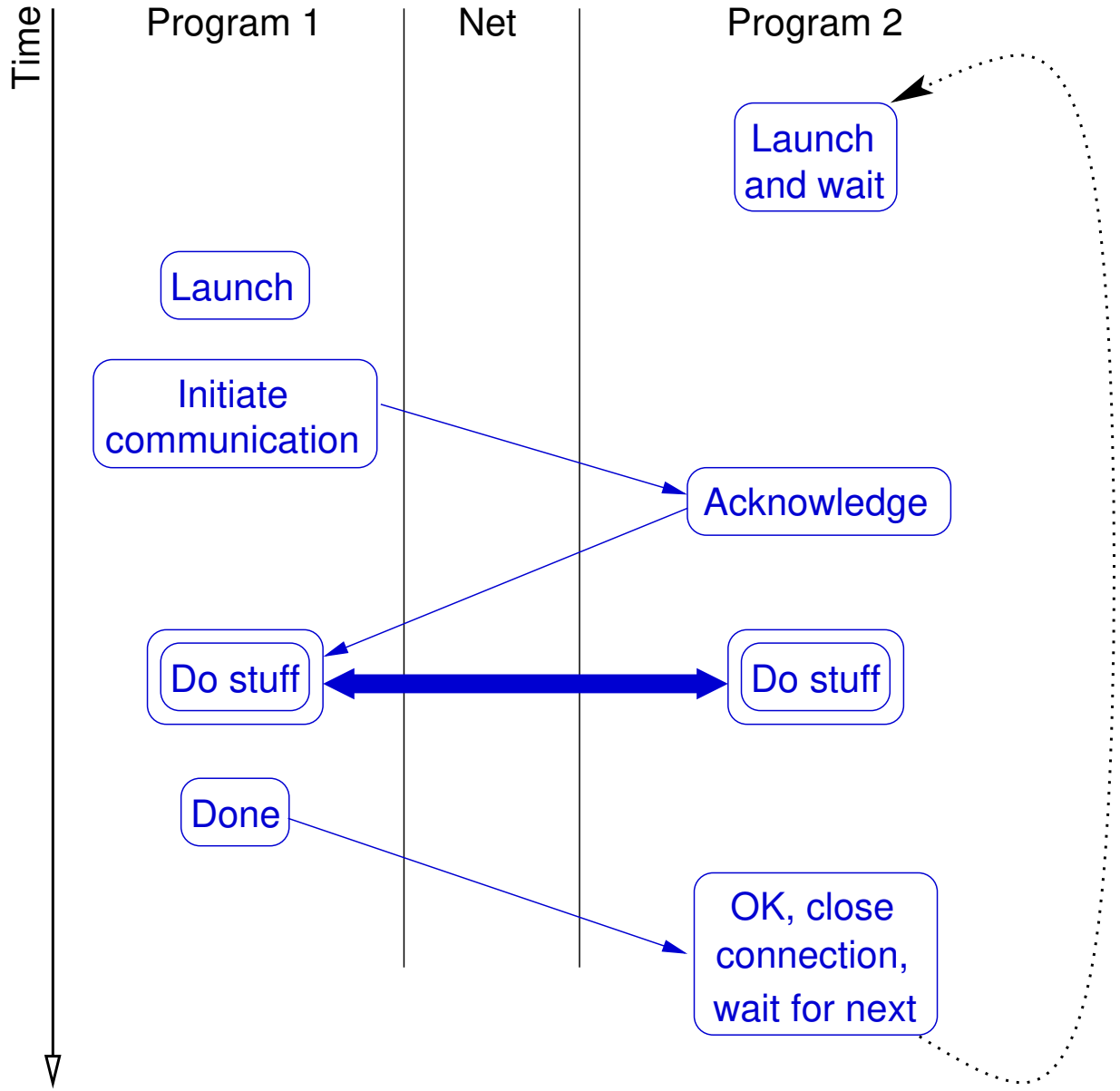


Why client-server applications

- 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 also simplifies the TCP/IP mechanisms



WHY CLIENT-SERVER APPLICATIONS (CONT'D)



Client issues

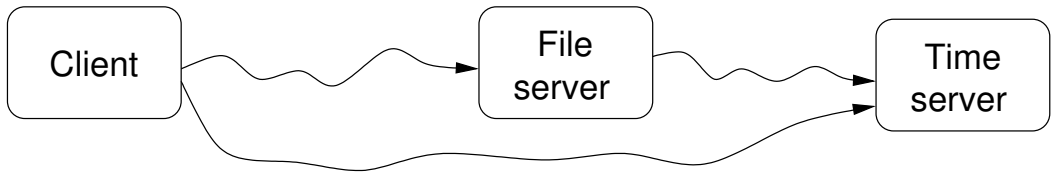
- When connecting to a server, a client has to know the address of the machine and a port number
  - Port numbers identify the actual server to connect to
  - Note incidentally the concept of server (program) vs server (machine)
- Standard versus nonstandard
  - No matter what, the client must speak the server's "language" (protocol)
- Parameterization
  - Some clients do one thing only e.g., manage file transfers
  - Some other (parameterized) clients can access many services
    - `telnet` is a fully parameterized client (though not a very smart one at it)

Server issues

- Connection or connectionless
  - Connection-oriented servers assume that all the data packets arrive correctly and in order (TCP)
  - A connectionless server does no 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

Server issues

- Connection or connectionless
  - Connection-oriented servers assume that all the data packets arrive correctly and in order (TCP)
  - A connectionless server does no 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
- Simultaneous servers and clients (for other servers) e.g.,



State information

- 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
  - IMAP server, that allows clients to retrieve their email messages which have not been previously received
  - HTTP server for an e-commerce site

State information

- 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
  - IMAP 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 or computer crash
  - Problems with identifying clients
- A stateless server
  - Operations must be idempotent
  - Copes well with loss of communication/computer crash