CompSci131

Parallel and Distributed Systems

Prof. A. Veidenbaum

Today's topics

- **Communication**
- **RPC**
- **Messaging systems**

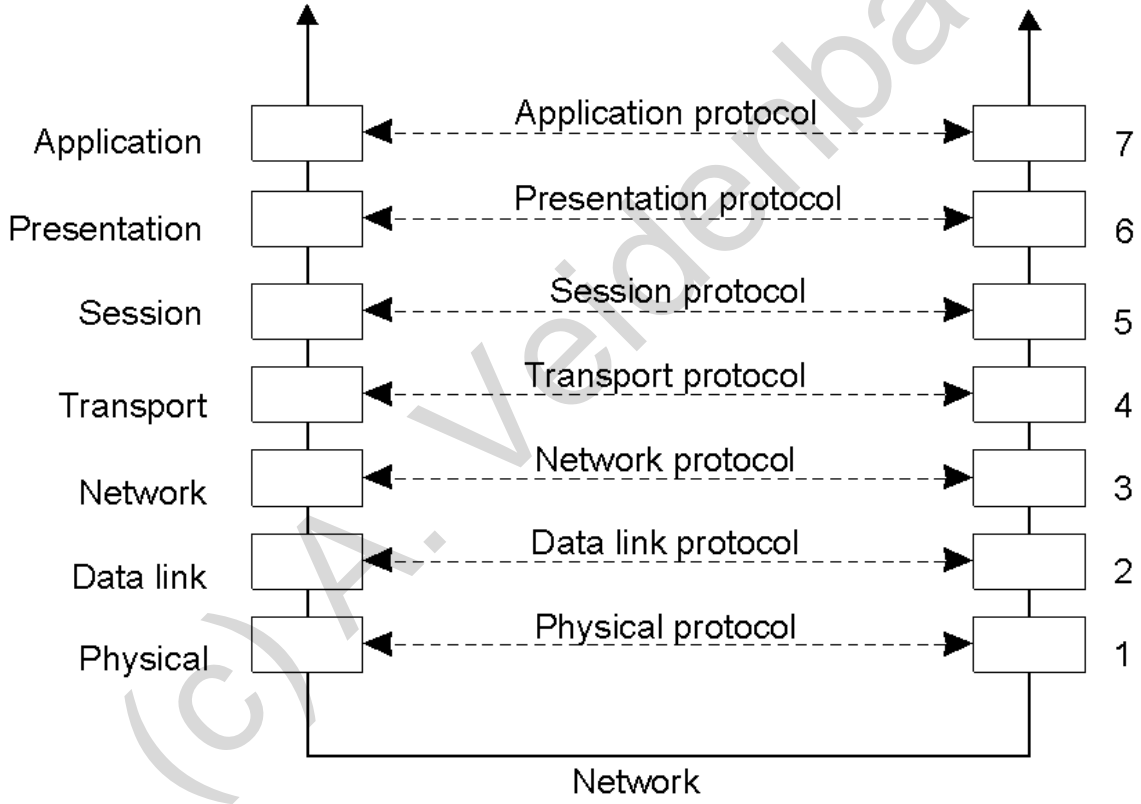
- **Reading assignment:**
  - **Today: 4.2-4.3**
  - **Next 2 lectures: MPI, Sec. 4.3 *and* lecture notes**
    - » **Complete the assignment *before* next class**

Last Lecture Covered

- **Virtualization**
- **Code migration**

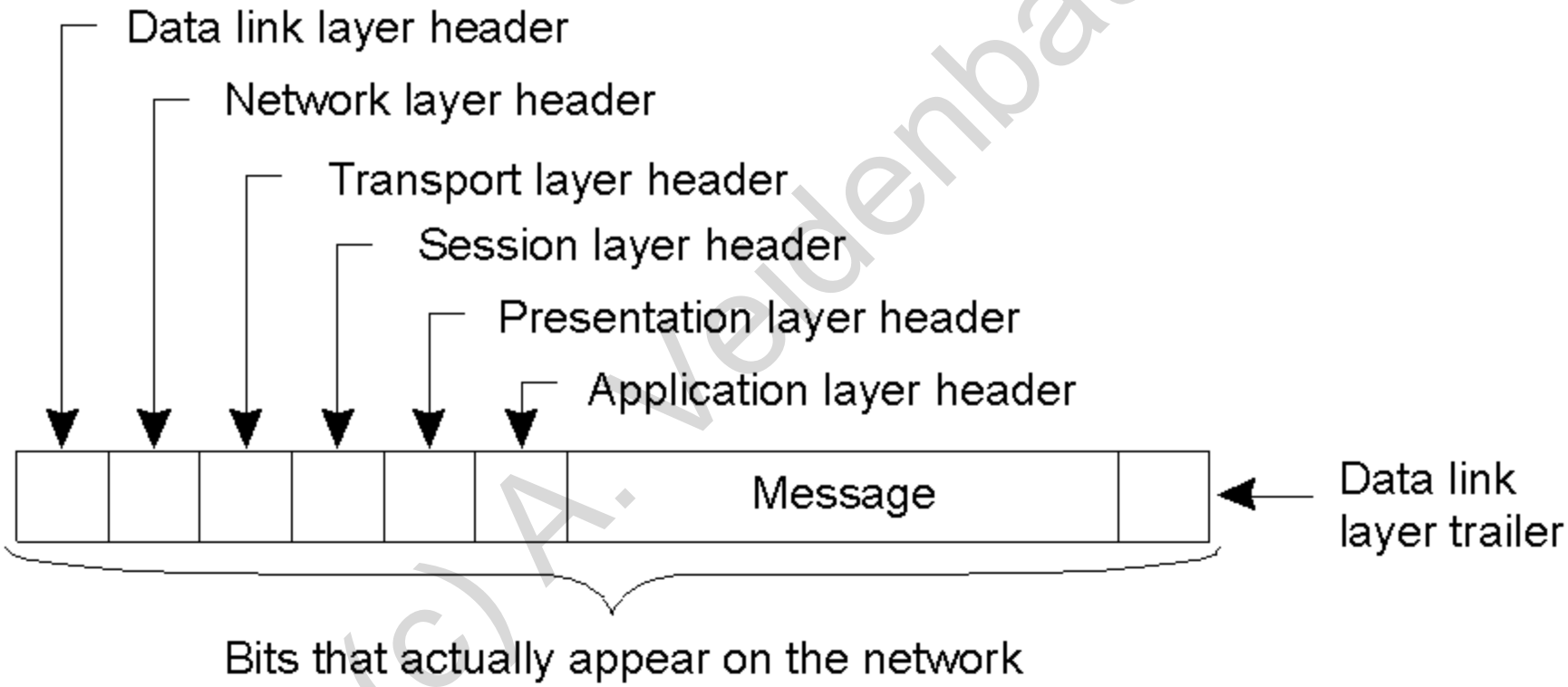
Communication

- **Uses layered protocols**
  - **Here are layers, interfaces, and protocols in the OSI model.**



Layered Protocols (2)

- **A typical message as it appears on the network.**



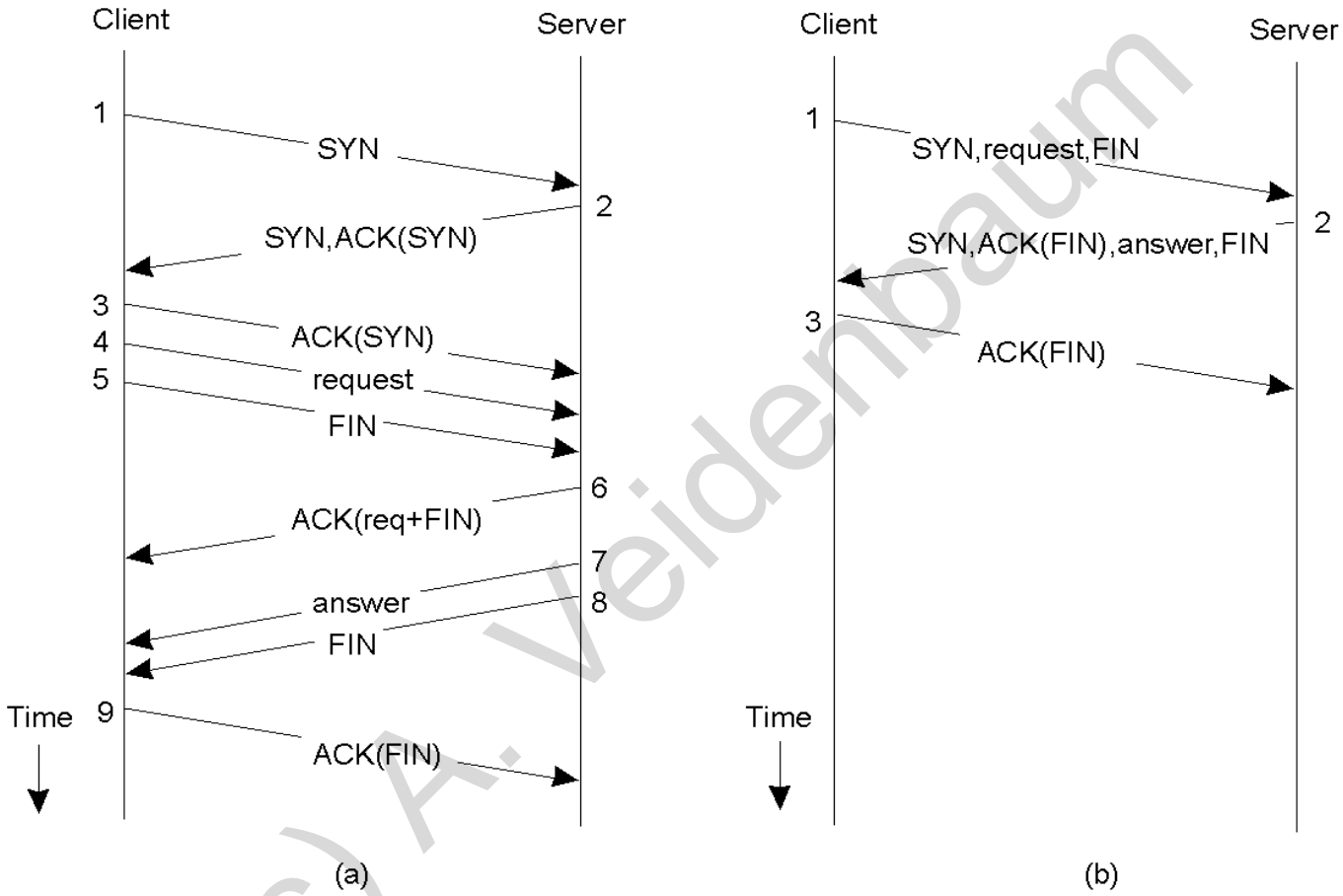
Protocol Layers

1. Data link: correctly deliver frames on a link
2. Network: IP, connectionless protocol, routing
3. Transport: TCP, reliable in-order delivery

- These are 3 main networking layers/protocols

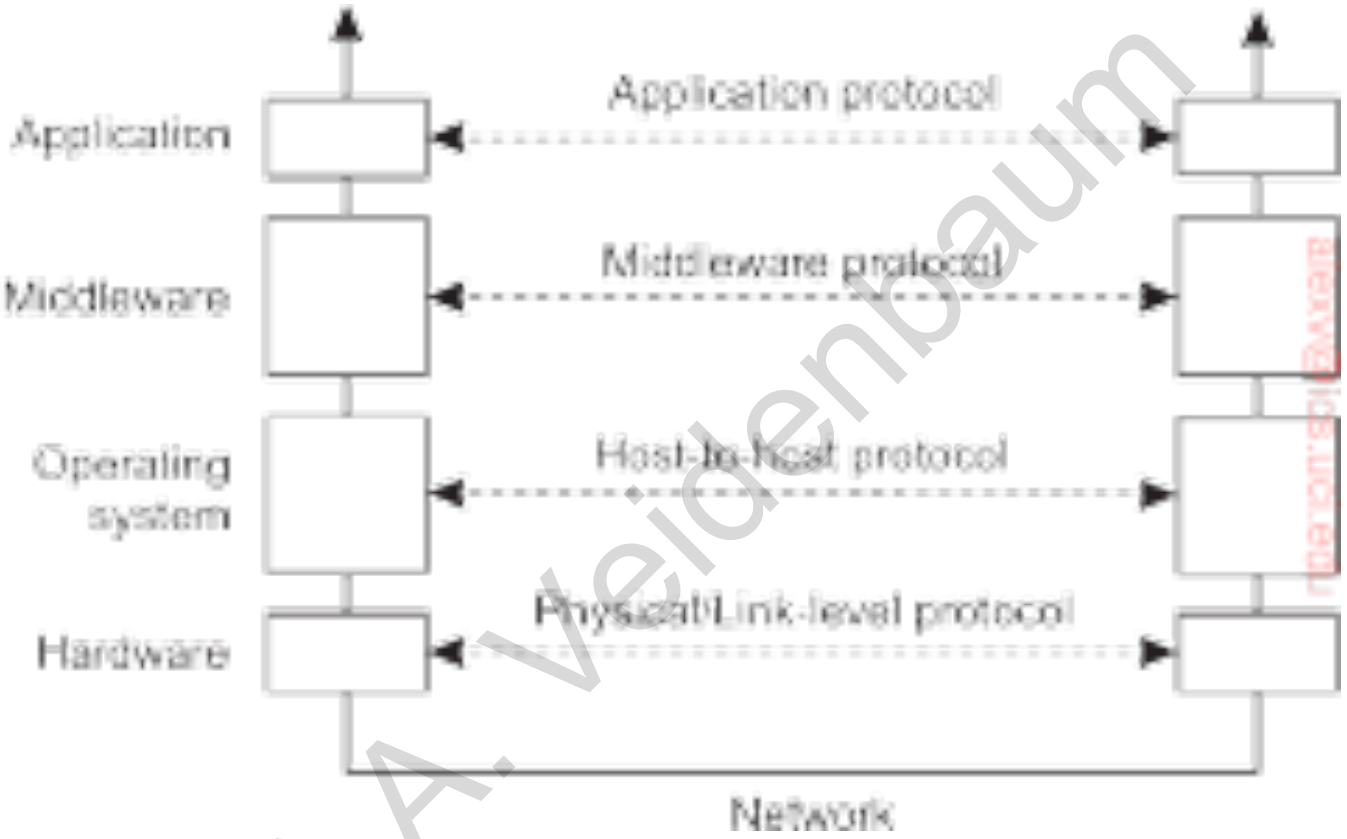
- Sometimes UDP is used - a datagram protocol
  - Connectionless
  - Unreliable – delivery no guaranteed

Client-Server TCP



a) Detailed TCP operation b) Transactional TCP.

Middleware Protocols



- **An adapted reference model for networked communication.**
- **Examples of middleware protocols:**
  - **Authentication, DS locking, commit, RPC, multicast**

Types of communication

- **Persistent**
  - Message stored by middleware until delivery
- **Transient**
  - Message can be dropped if cannot deliver
- **Synchronous**
  - Sender blocks until request accepted
- **Asynchronous**
  - Sender continues after submitting a message

Middleware protocols

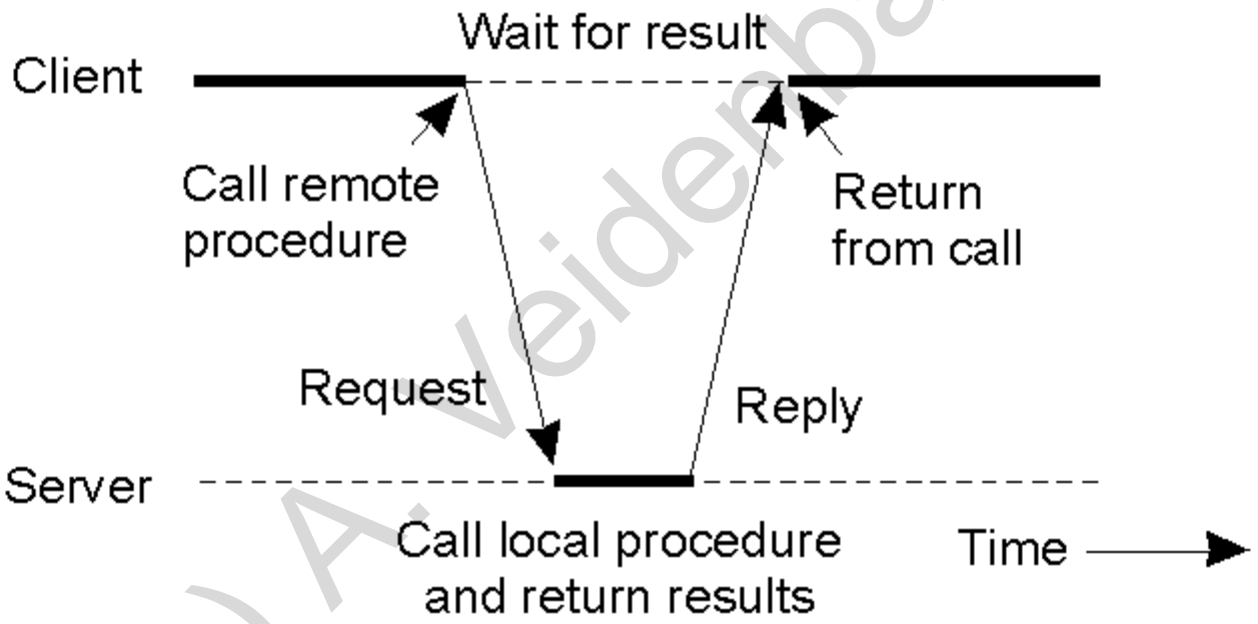
- **RPC**
- **Message-oriented communication**
- **Data Streaming**

RPC

- **RPC implements a Remote Procedure Call**
  - **Works very much like a regular C call**
- **Synchronous, transient communication**
- **Implementation issues**
  - **How to make RPC *transparent***
  - **Remote function call**
  - **Parameter passing**
  - **Returning results**
  - **Dealing with heterogeneity**

Client / Server RPC

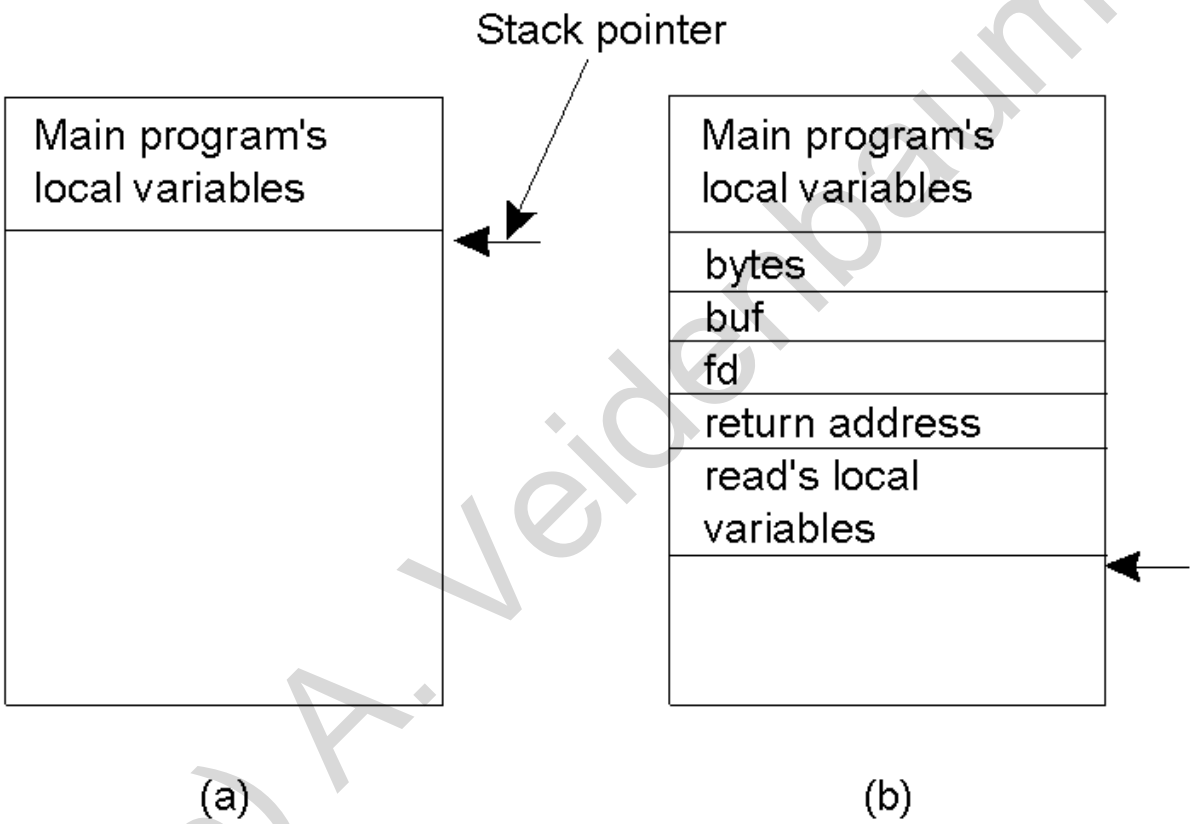
- **RPC uses client and server stubs.**



RPC

- **Transparency is implemented using *stubs***
  - » **A client stub and a server stub**
- **Client function calls client stub**
  - **Stub builds a message and calls local OS**
  - **Local OS sends the message to the remote OS**
- **Server stub**
  - **Remote OS passes the message to server stub**
  - **Stub unpacks the message and calls server**
- **Server does all the work**
- **Result communication is done in reverse**

Conventional Procedure Call

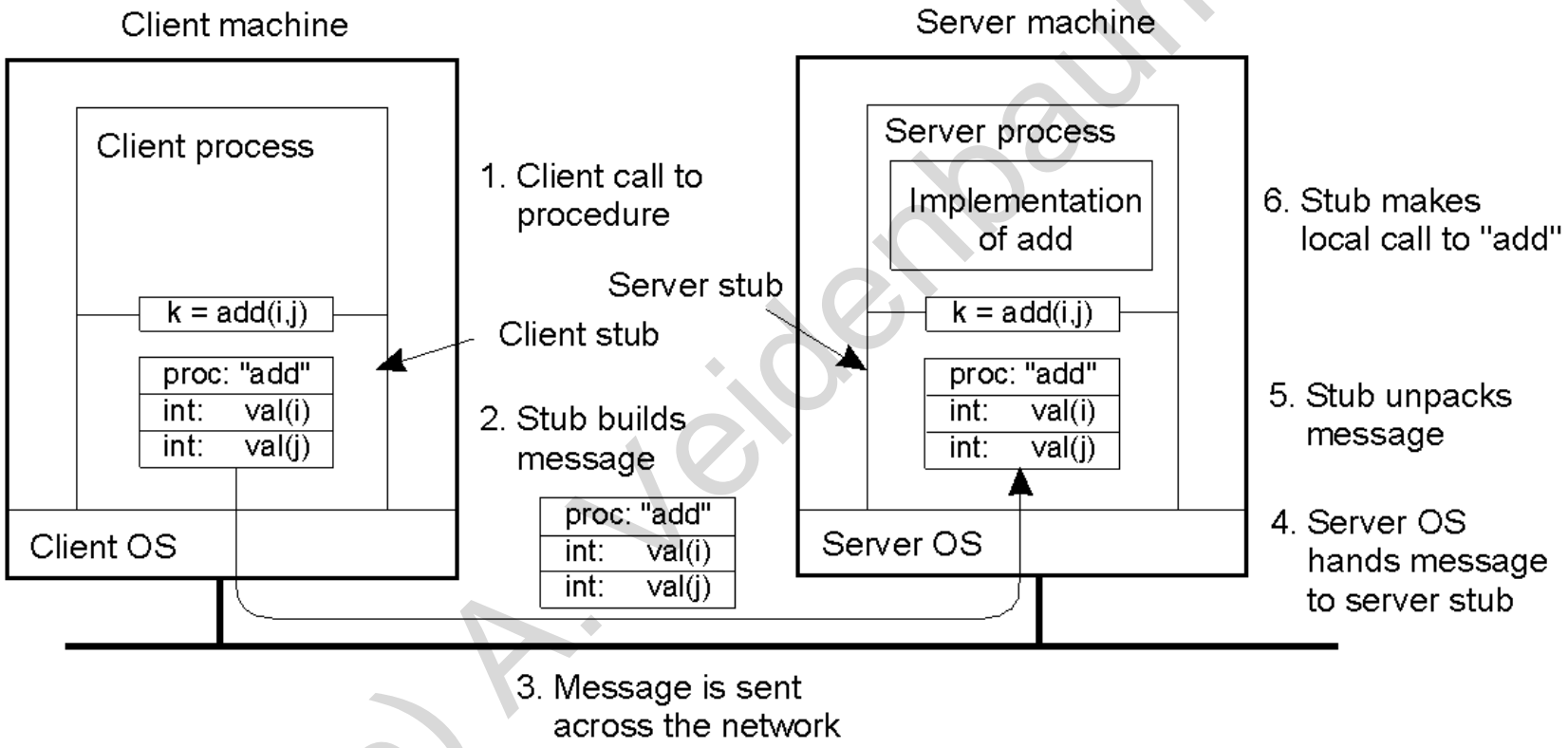


a) Parameter passing in a procedure call: the stack before the call

b) The stack while the called procedure is active

RPC Steps

- **Steps involved in doing remote computation**



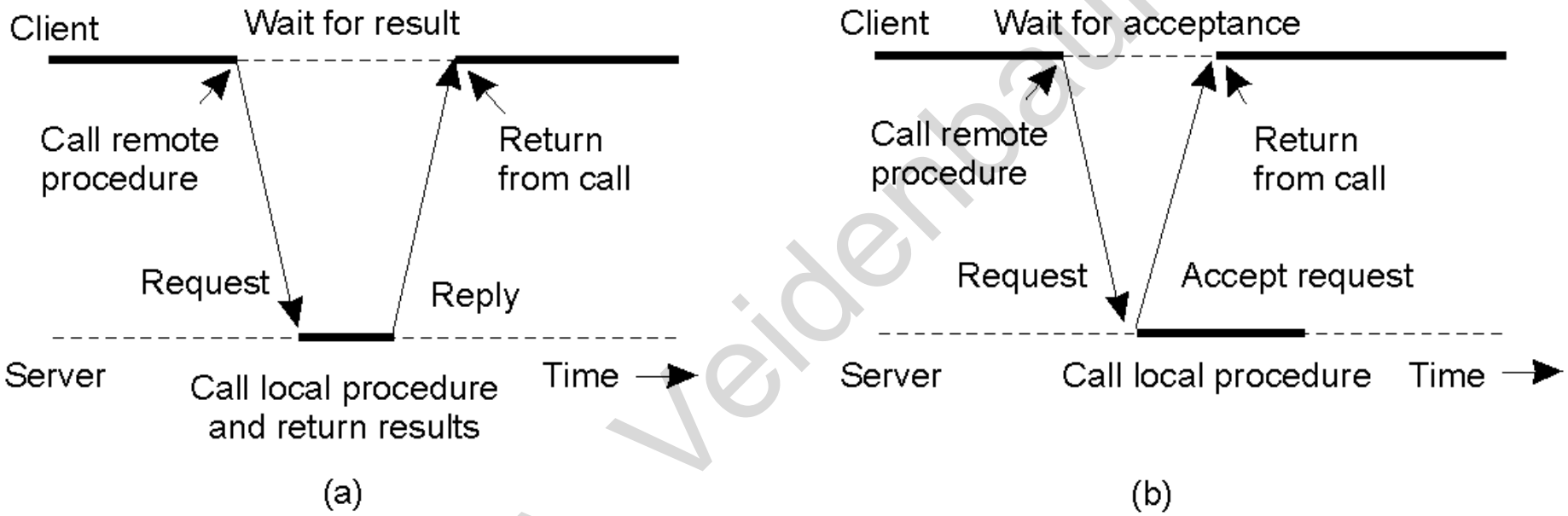
Steps in the RPC

1. Client procedure calls a client stub
   - A regular function call
2. Client stub builds message, calls local OS
3. Client's OS sends message to remote OS
4. Remote OS gives message to server stub
5. Server stub unpacks parameters, calls server
6. Server does work, returns result to the stub

Steps of an RPC return

1. Server stub packs it in message, calls local OS
2. Server's OS sends message to client's OS
3. Client's OS gives message to client stub
4. Stub unpacks result, returns to client

Asynchronous RPC (1)

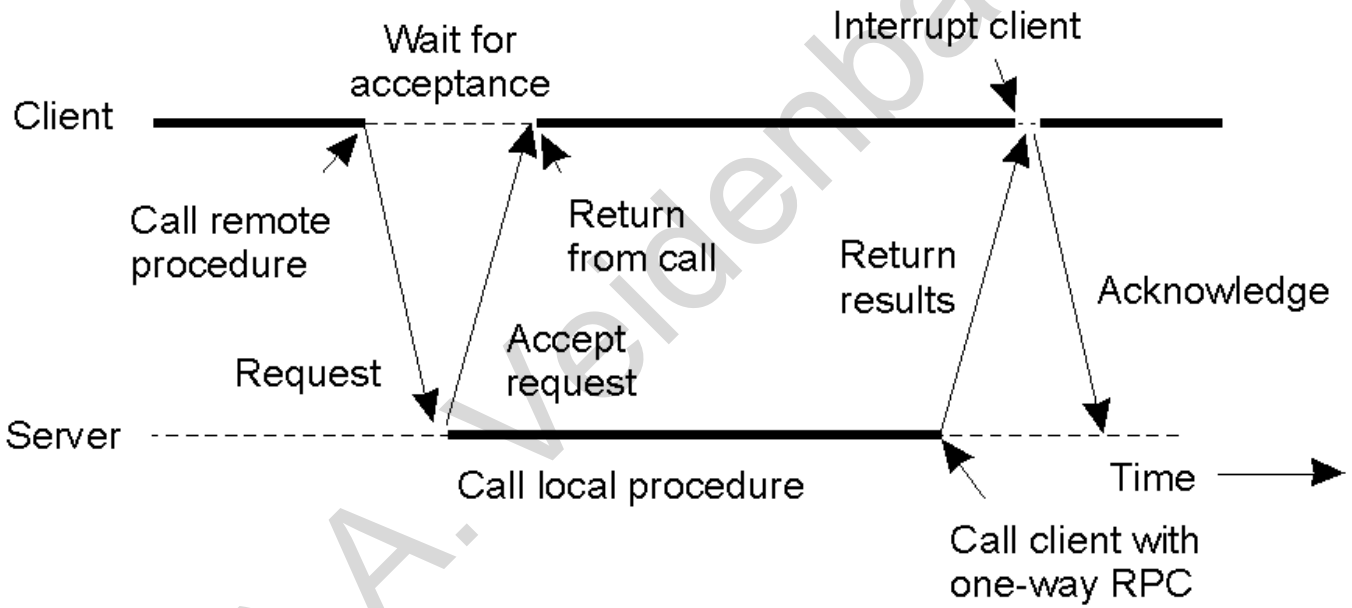


a) **The interconnection between client and server in a traditional RPC**

b) **The interaction using asynchronous RPC**

Asynchronous RPC (2)

- **A client and server interacting through two asynchronous RPCs**

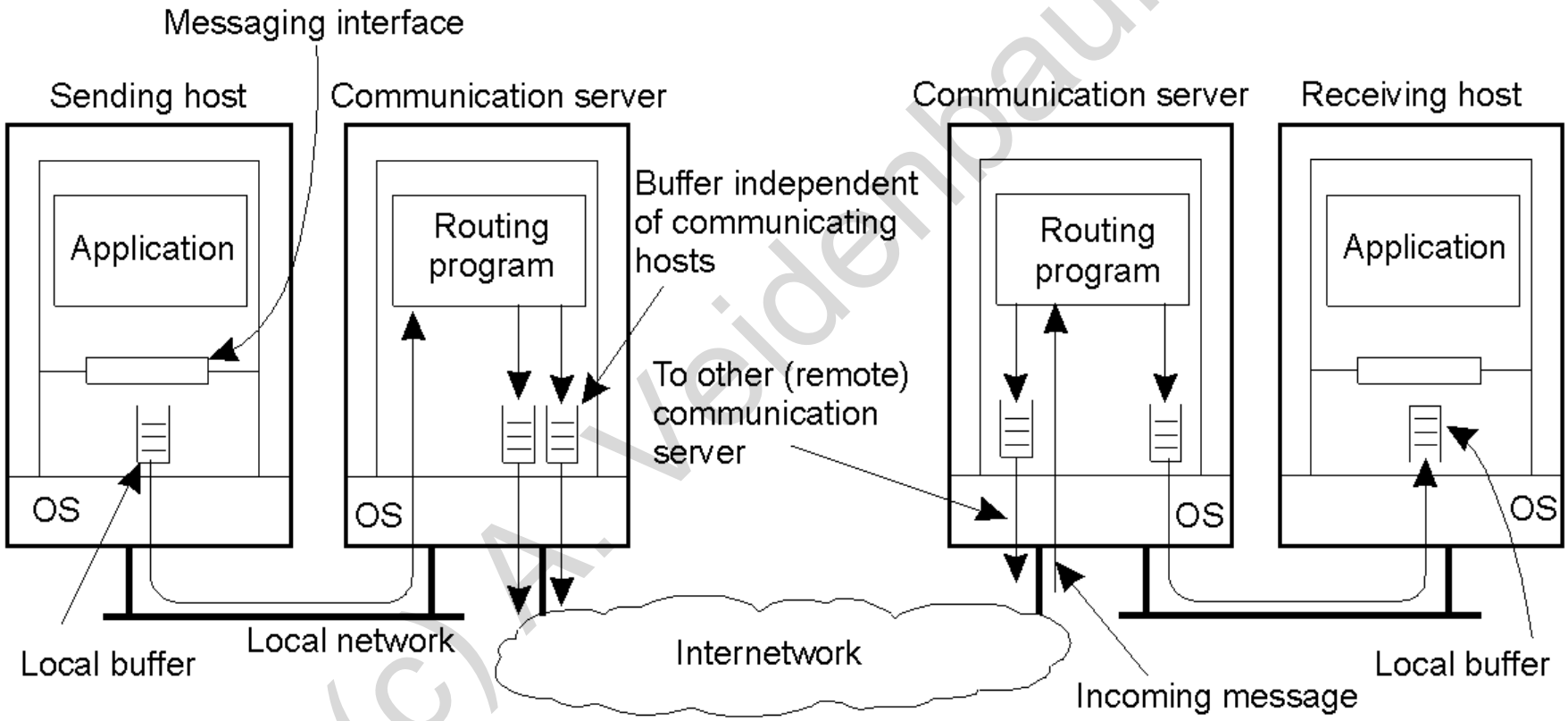


Message Oriented Communication

- RPC/RMI is not always be desirable
  - Two processes may not run at the same time!
  - RPC's blocking sender is not always a good idea
    - » Although one can implement the async. RPC
- One solution is to use messaging
  - include all info needed to do the equivalent of RPC
- Need protocols for message exchange
- Let's first look at the general architecture

Use of Communication Servers

- **Applications communicate messages via comm. servers**

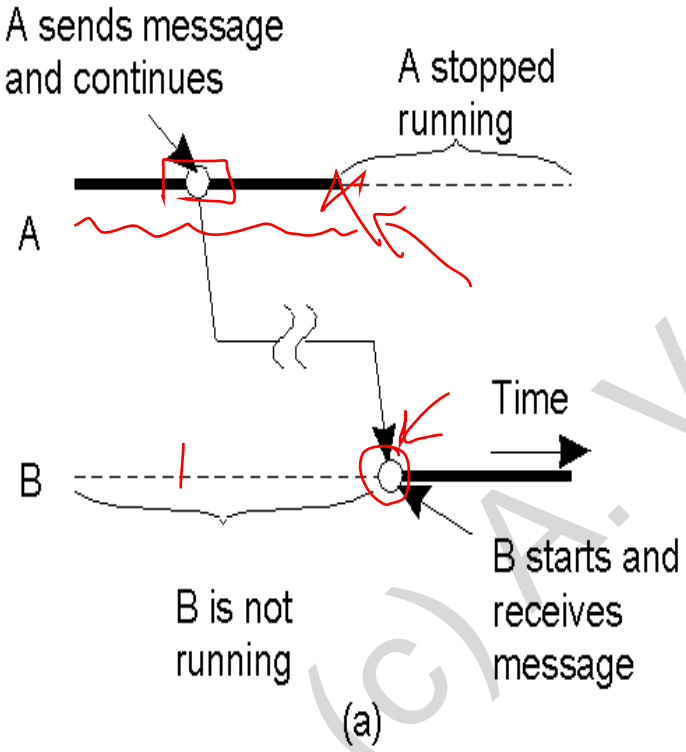


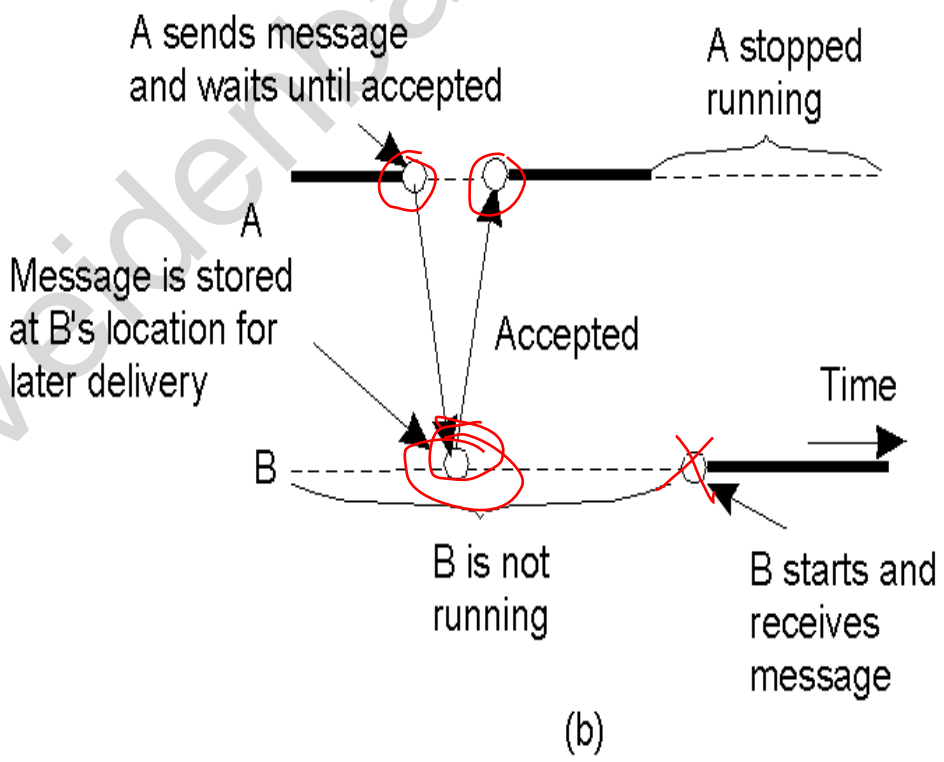
Message Communication

- **Persistent Communication:**
  - Message stored until it can be delivered
- **Transient Communication:**
  - A message is discarded if it cannot be delivered
- **Synchronous:**
  - Sender blocked till message is stored in receiver's buffer
    - » Receiving host's local buffer
      - Or even actually delivered to receiver
- **Asynchronous:**
  - Sender continues after submitting message
    - » Message stored in sending host's local buffer

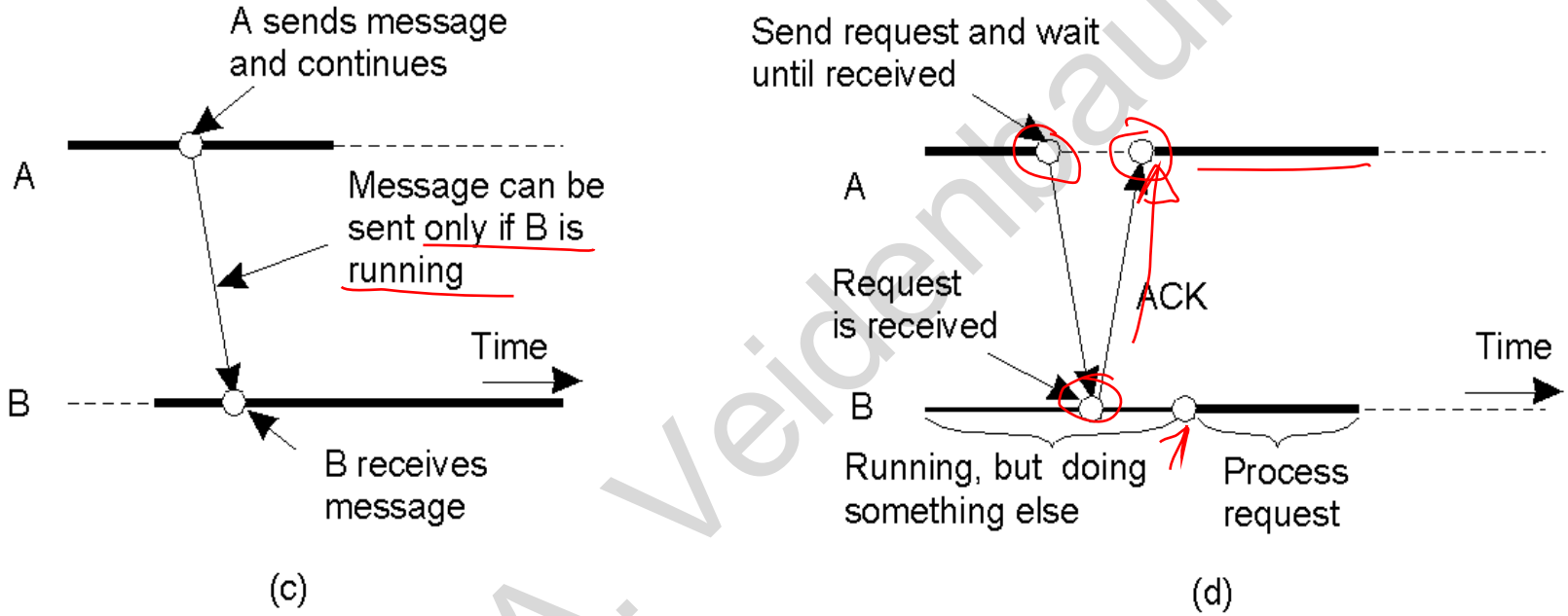
Persistent Communication

a) **Asynchronous**
b) **Synchronous**





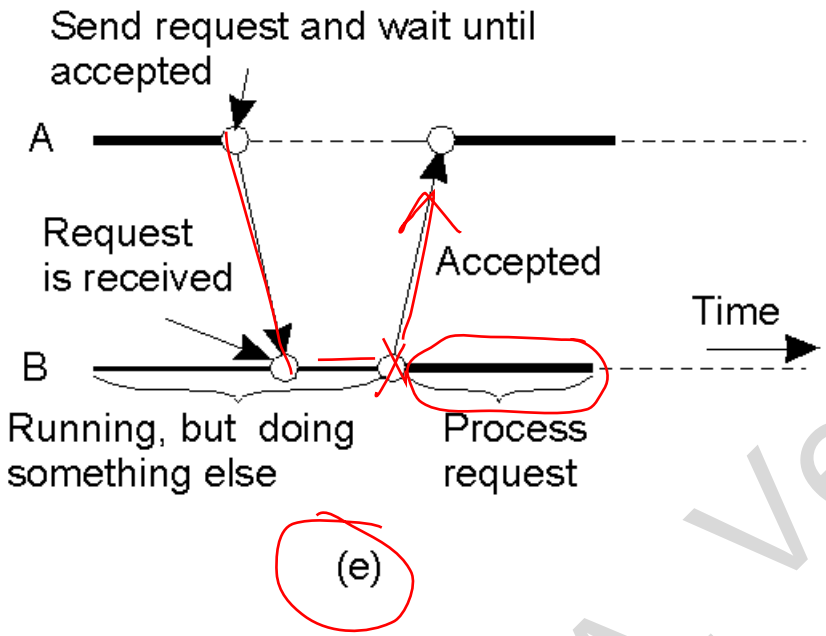
Transient Communication

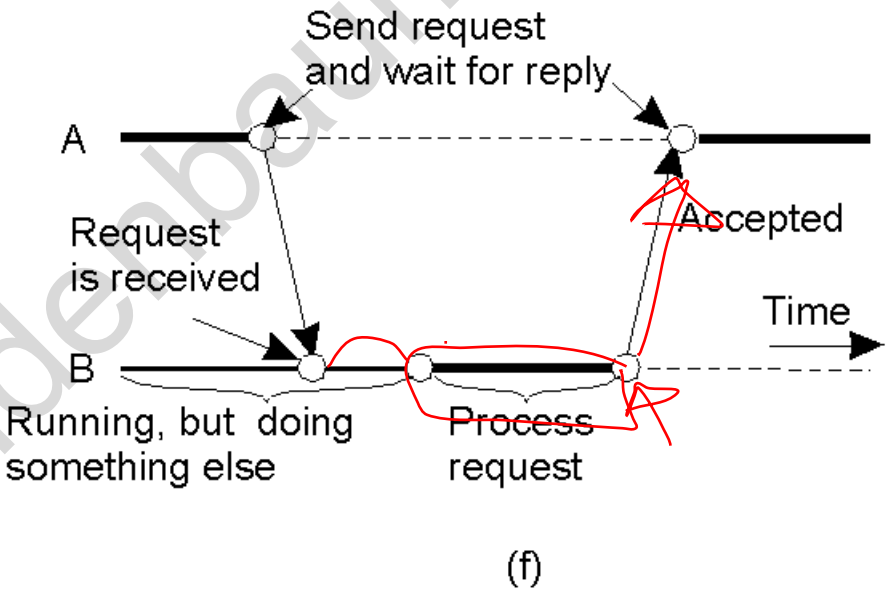


c) Asynchronous communication (e.g. UDP)

d) Receipt-based (ACK) synchronous

Persistence and Synchronicity in Communication (5)





e) Delivery-based transient synchronous (similar to Async. RPC)

f) Response-based transient synchronous communication