

Sistemas Distribuídos

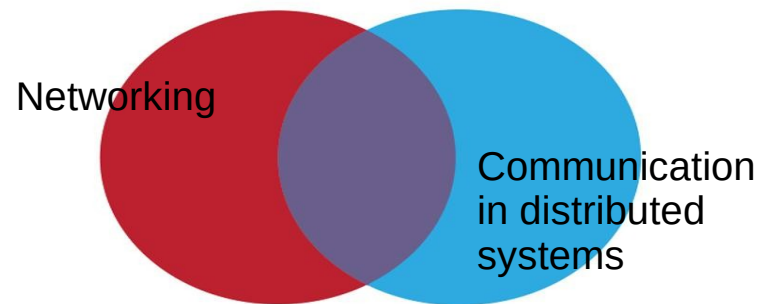
José Orlando Pereira

Departamento de Informática
Universidade do Minho

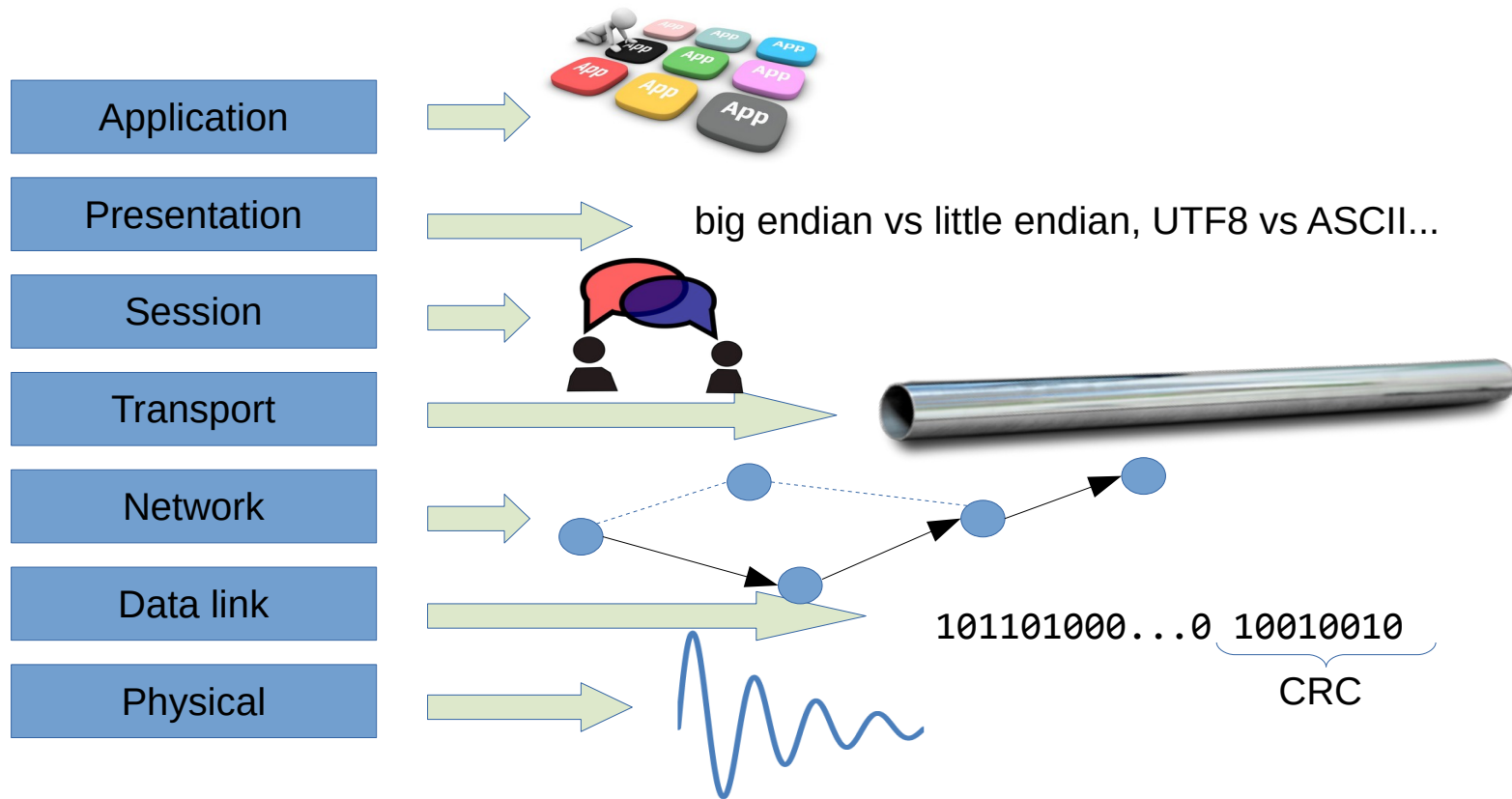


Communication

- Networking is used for communication in distributed systems
- What aspects of networking are relevant to distributed systems?
- How does communication in distributed systems goes beyond networking?



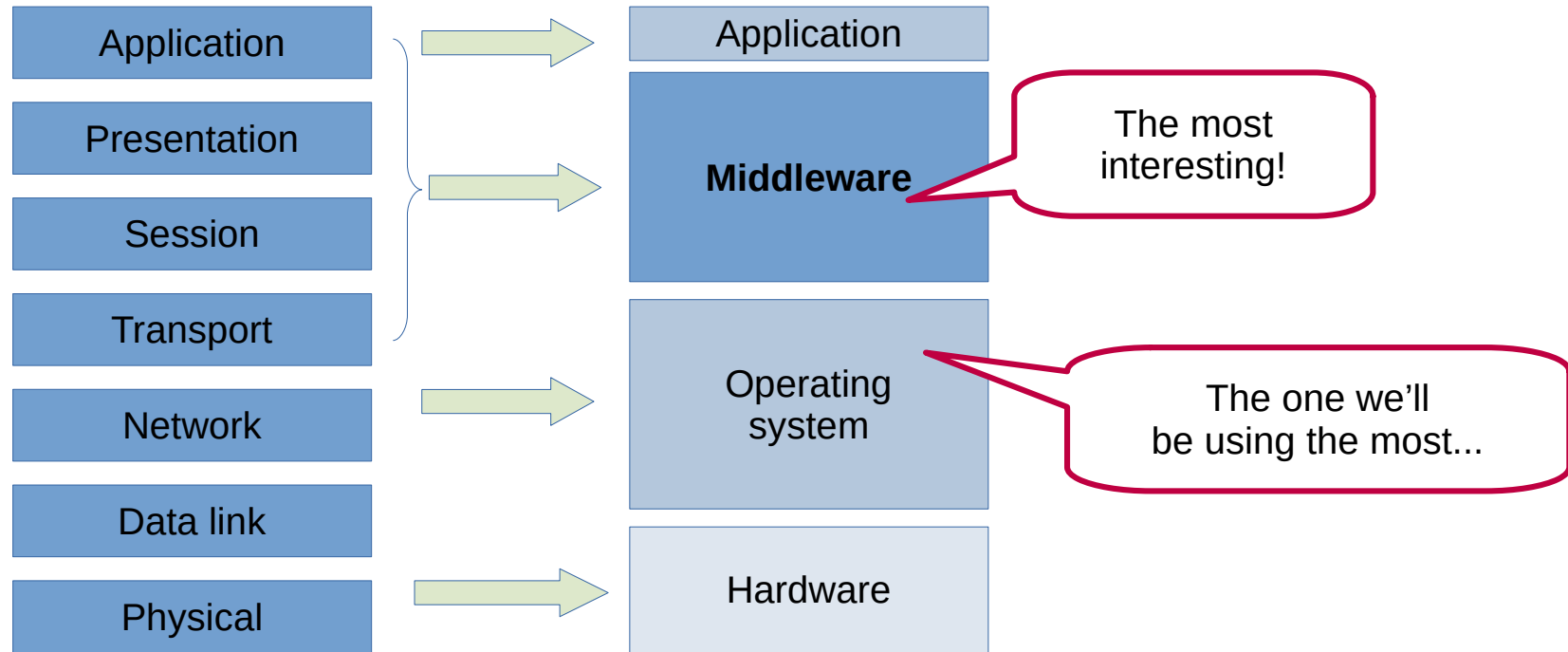
Networking 101: OSI model



Networking 101: OSI model

- Layers as implementation modules
 - Standard interfaces
 - Interchangeable alternatives
 - No longer used
- Layers as abstractions
 - Different ways to interpret the same thing
 - E.g. electric signal vs bits vs channel
 - Still useful

Simplified model



Simplified model

- The hardware is not very interesting...
- The operating system is standard:
 - TCP/IP (mainly...)
 - Sockets API
- The middleware layer encapsulates solutions to hard problems:
 - Much more than session/presentation layers
 - More than just communication (e.g. persistence) *→ possibilidade de guardar os dados em disco*
 - Determines how applications are developed

Operating system

- A simplified understanding of TCP/IP
 - What do we need to know for distributed systems?
- The Sockets application programming interface

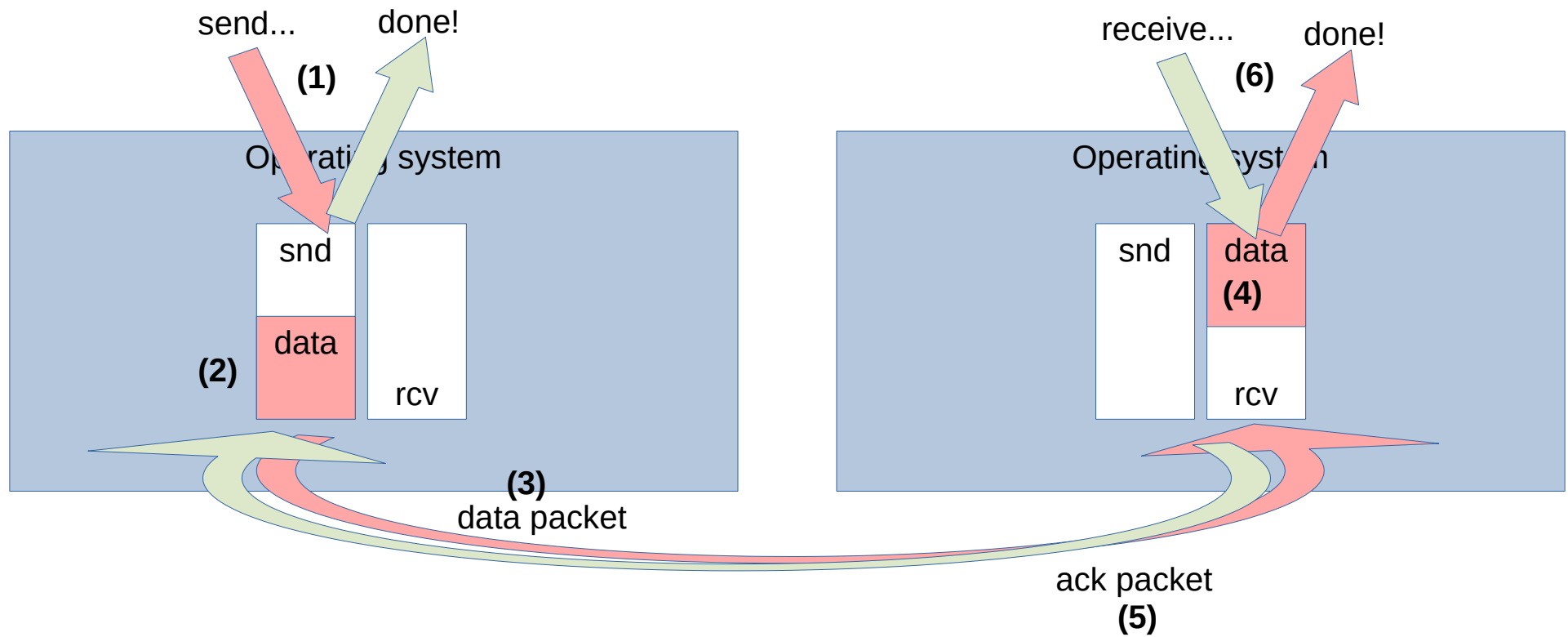
TCP/IP

- A bi-directional reliable FIFO connection:
 - Bi-directional: both participants send and receive
 - Reliable: no data is lost or corrupted
 - FIFO: data is received in the same order as sent
- When the connection is broken:
 - A prefix of data sent has been received
(“Prefix” includes possibly all or none)

TCP/IP

- A connection is identified at each participant by:
 - A local IP address
 - A local port
 - A remote IP address
 - A remote port
- It is possible to have more than one connection on the same port
- It is possible that addresses and ports are different in participants due to Network Address Translation (NAT)

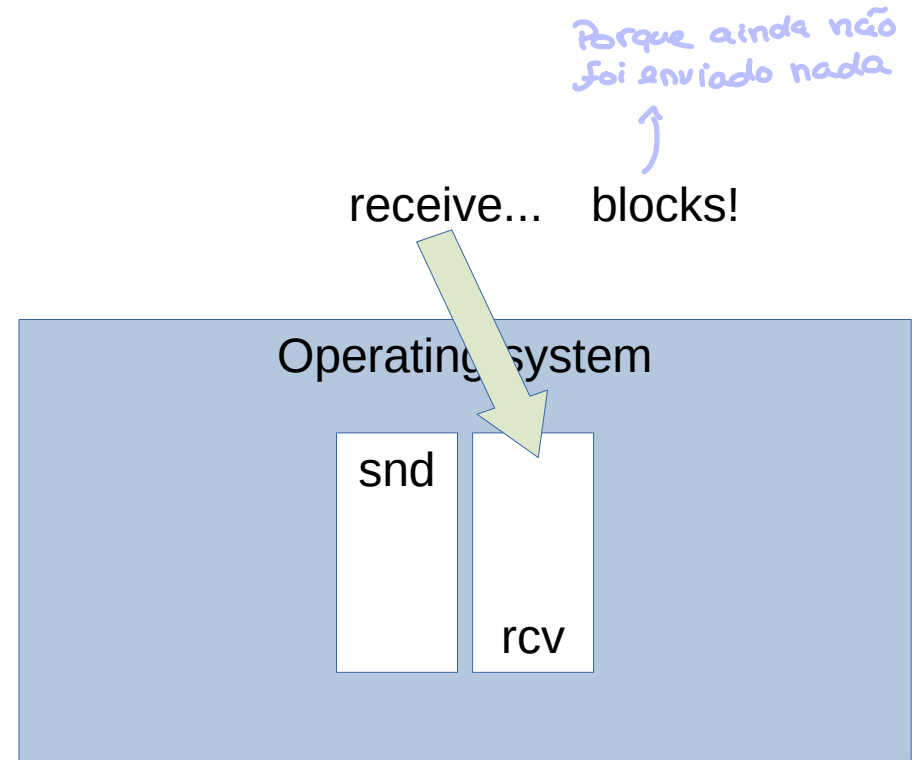
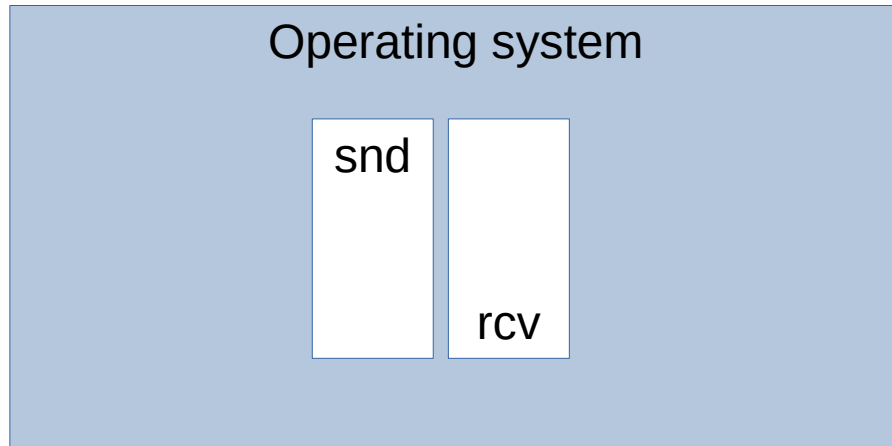
TCP/IP



TCP/IP

- The protocol controls when to send packets
 - (2)→(3) and (4)→(5)
 - Important for efficient/safe use of network
 - Limited by network capacity
- Consuming data (6) makes receiver buffer space available for more packets
- Acknowledgment (5) makes sender buffer space available for more data

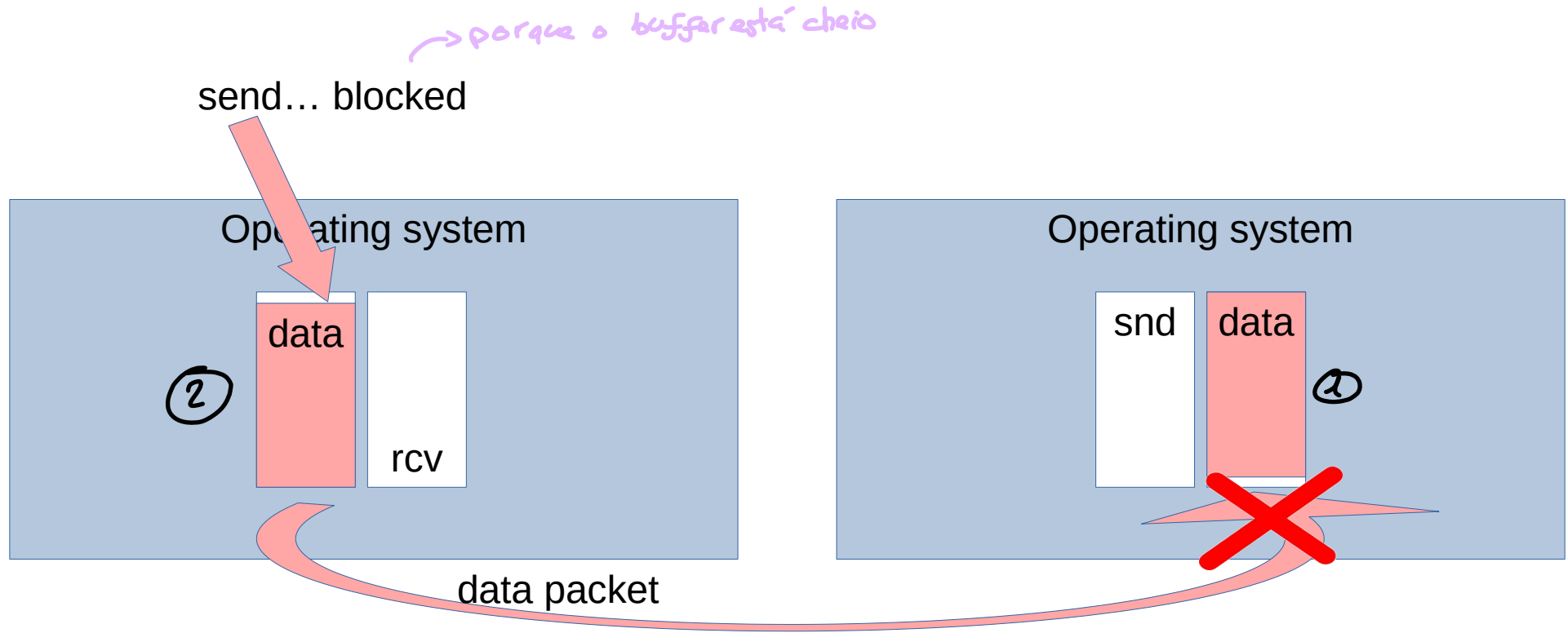
TCP/IP



TCP/IP

- When the sender does not write:
 - Sender buffer becomes empty
 - Stops sending packets
 - Receiver buffer becomes empty
- The receiver is also blocked

TCP/IP



TCP/IP

- When the receiver does not read:
 - Receiver buffer fills up
 - Stops accepting new data
 - Stops sending acknowledgments
 - Sender buffer fills up
- The sender is also blocked

TCP/IP Summary

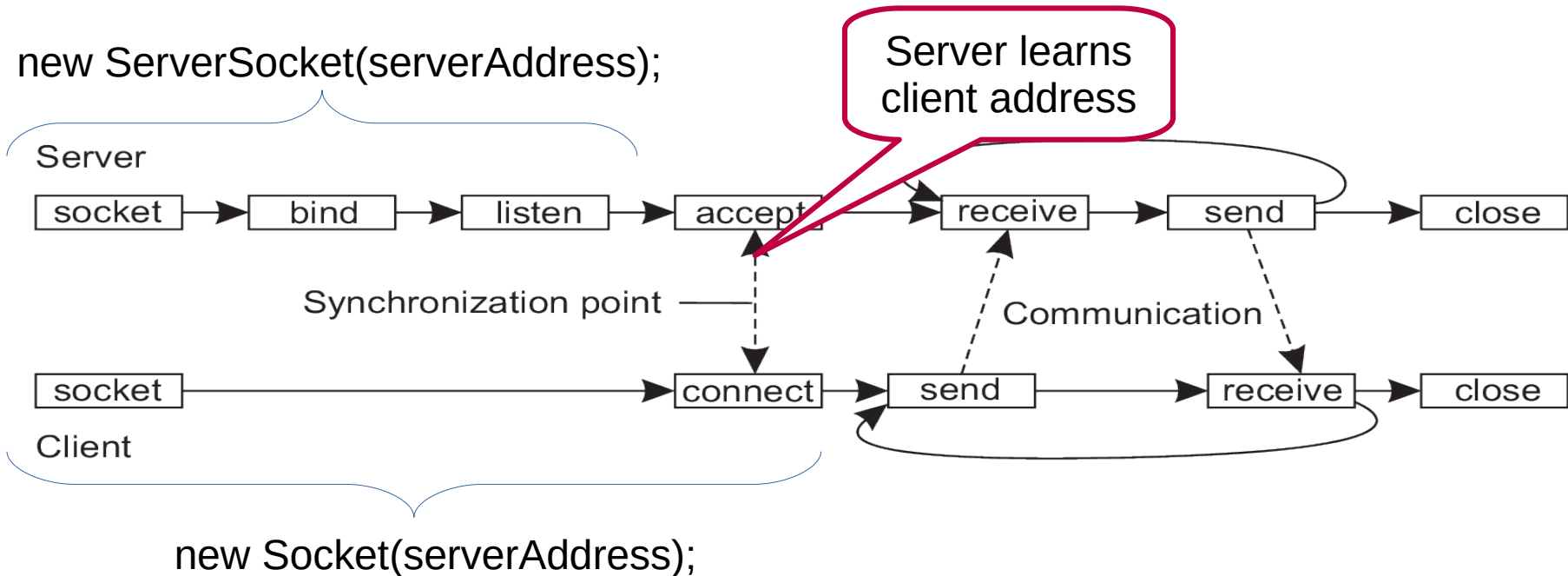
- Connection behaves like:
 - a pair of operating system pipes
 - a pair of bounded buffer concurrency constructs
- It is both a communication and a distributed synchronization primitive
 - Allows waiting for other processes/threads in different hosts
 - (Think await()/signal()...)

Sockets

- Interface for setting up a new connection
- Interface for sending and receiving data
- Interface for closing an existing connection

Sockets: Connecting

- How to discover the peer address?
- Asymmetric interface:
 - “Server” is well known and is contacted by “Client”



Sockets: Sending and receiving

- In the Unix tradition:
 - An open connection is a file descriptor
 - Sending and receiving with `read()` and `write()`
- In Java, it is wrapped in I/O streams:
 - `Socket sock = ...`
 - `sock.getOutputStream()` → for writing
 - `sock.getInputStream()` → for reading
- Closed connection equivalent to EOF (read doesn't block and returns 0 bytes)

Sockets: Sending and receiving

- Sending and receiving lines of text in Java:
 - `BufferedWriter pw = new BufferedWriter(
 new OutputStreamWriter(sock.getOutputStream())
);`
 - `rw.write(...);`
 - `BufferedReader br = new BufferedReader(
 new InputStreamReader(sock.getInputStream())
);`
 - `br.readLine();`

Sockets: Disconnecting

- In the Unix tradition, by closing the file descriptor. In Java, wrapped as:
 - `sock.close();`
 - `sock.getInputStream().close();`
 - `sock.getOutputStream().close();`
- Can be partially closed:
 - `sock.shutdownOutput();`
 - `sock.shutdownInput();`

Middleware: Synchrony and persistence

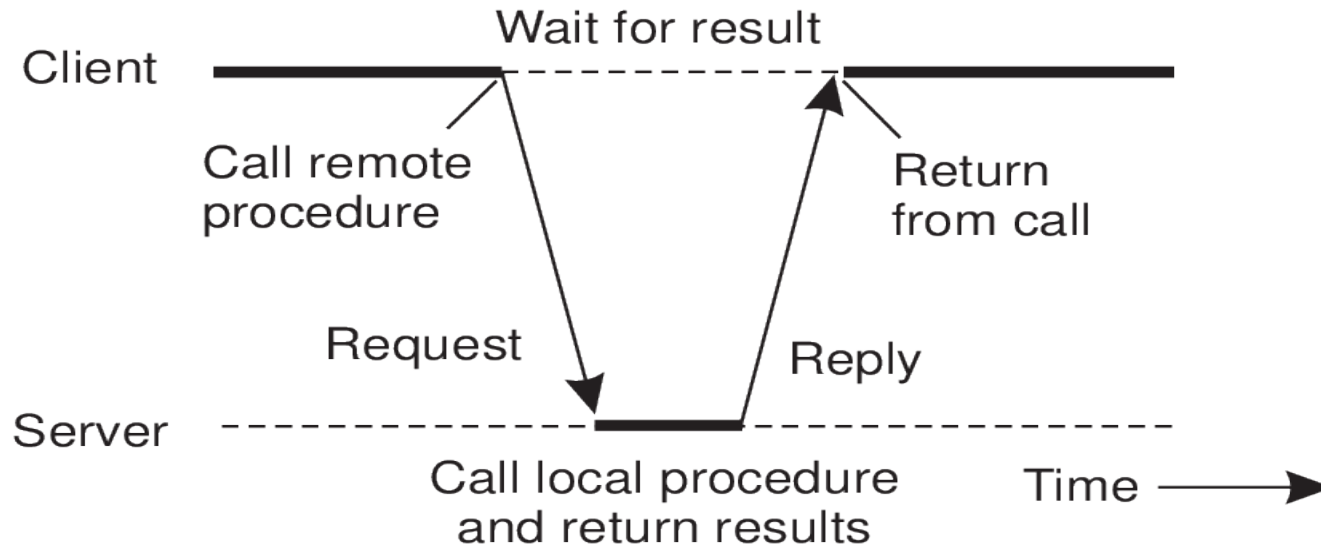
- Asynchronous vs synchronous:
 - Thread not blocked while interacting
 - Thread blocked only until...
 - To thread blocked until reply received
- Transient vs persistent:
 - Persistent messages stored on disk at various stages of processing
 - Used to restart interaction

Middleware: Destination and addressing

- Point-to-point vs multipoint:
 - Interaction between two vs multiple participants
- Explicit vs implicit:
 - Explicit addressing: Join group, send to group
 - Implicit addressing: Subscribe to interest, publish content

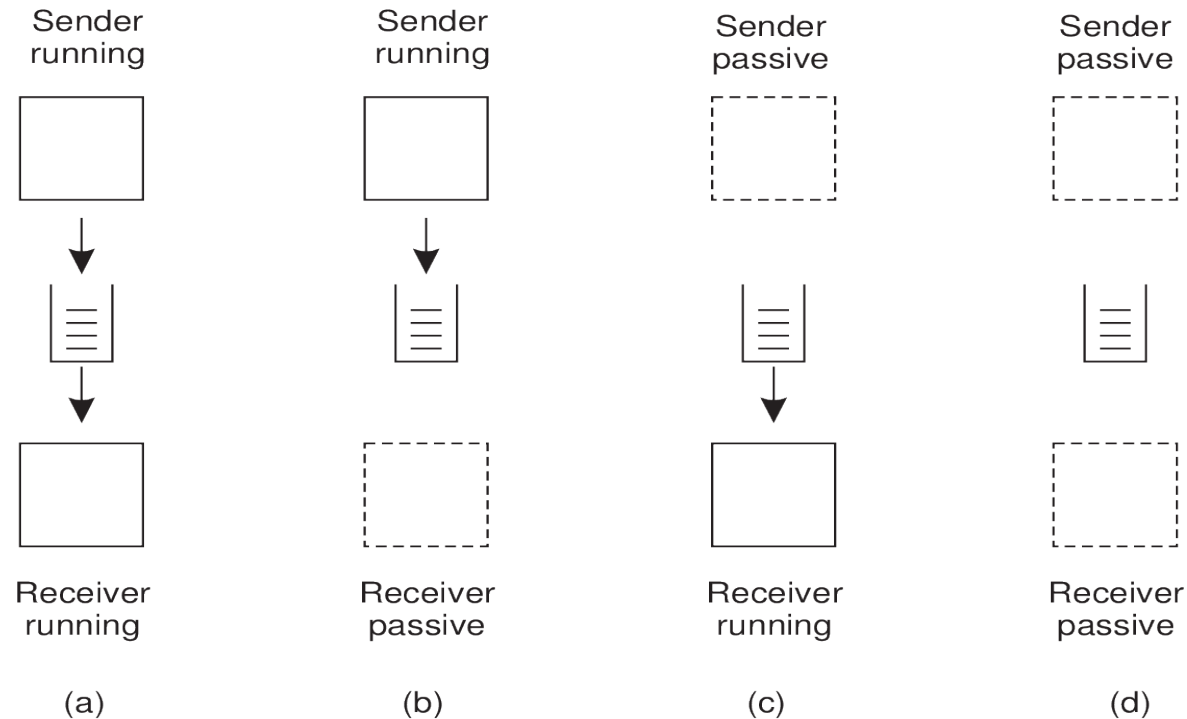
Example: Client/server RPC

- Synchronous / transient / point-to-point / explicit
- Interaction looks like a local method/procedure invocation



Example: MoM

- Asynchronous / persistent / * / *
- Loosely coupled communication:

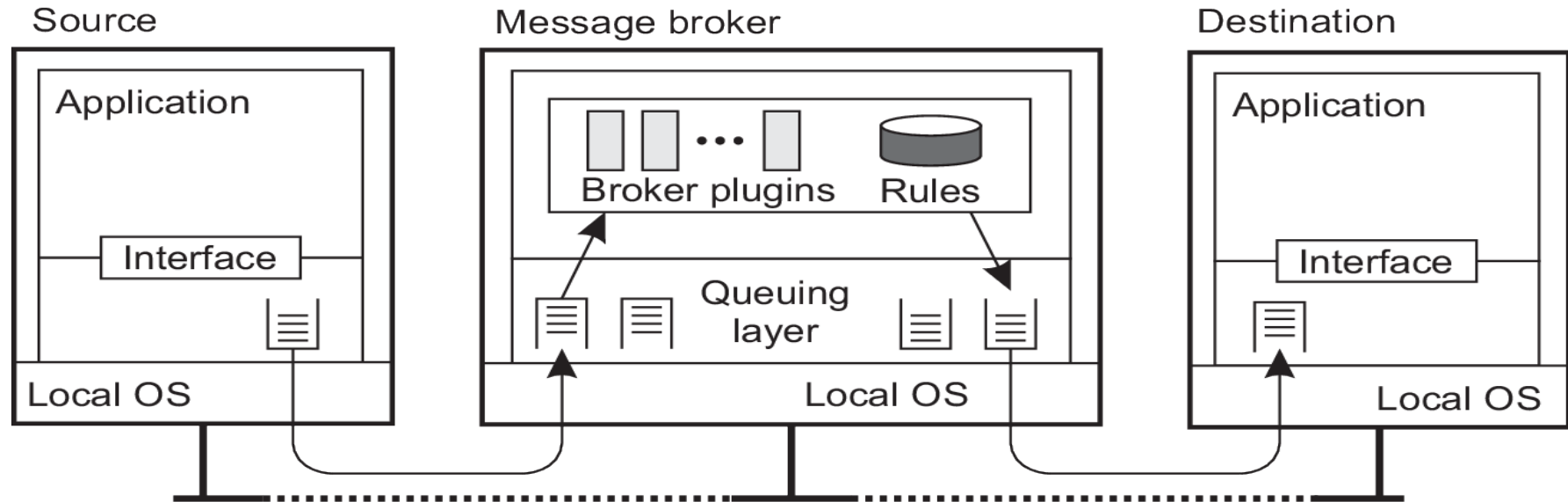


Example: MoM

- Point-to-point:
 - Send to named queue
 - Receive from named queue
- Topic based multipoint:
 - Subscribe to topic
 - Publish to topic
- Content based multipoint:
 - Subscribe to expression: “stock='IBM' and price>10”
 - Publish data: “(stock='IBM', ..., price=11.5, ...)”

Example: MoM

- Use with a message broker / application bus component:



Example: Group communication

- Asynchronous / transient / point-to-multipoint / explicit
- Expects a variable group of processes
- Targets fault-tolerant systems
- Provides notifications of current group composition and predictable delivery context

Example: MPI

- * / transient / multipoint / explicit
- Expects a fixed group of processes
- Targets parallel processing
- Provides a spectrum of asynchronous/ synchronous send and receive operations:
 - Asynchronous needed for parallelism
 - Synchronous needed to wait for completion of interim stages

Summary

- Operating system provides the basic communication primitive:
 - TCP/IP + Sockets
- Various middleware layers provide communication primitives tailored for different applications
- Our next goal: Build useful middleware on TCP/IP + sockets