DISTRIBUTED COMPUTING, CLUSTER AND GRID COMPUTING

DR. SHAKTI MISHRA

ROAD MAP: OVERVIEW

- Why are distributed systems interesting?
- Why are they hard?

GOALS OF DISTRIBUTED SYSTEMS

Take advantage of cost/performance difference between microprocessors and shared memory multiprocessors

Build systems:

1. with a single system image
2. with higher performance
3. with higher reliability
4. for less money than uniprocessor systems

In wide-area distributed systems, information and work are physically distributed, implying that computing needs should be distributed. Besides improving response time, this contributes to political goals such as local control over data.

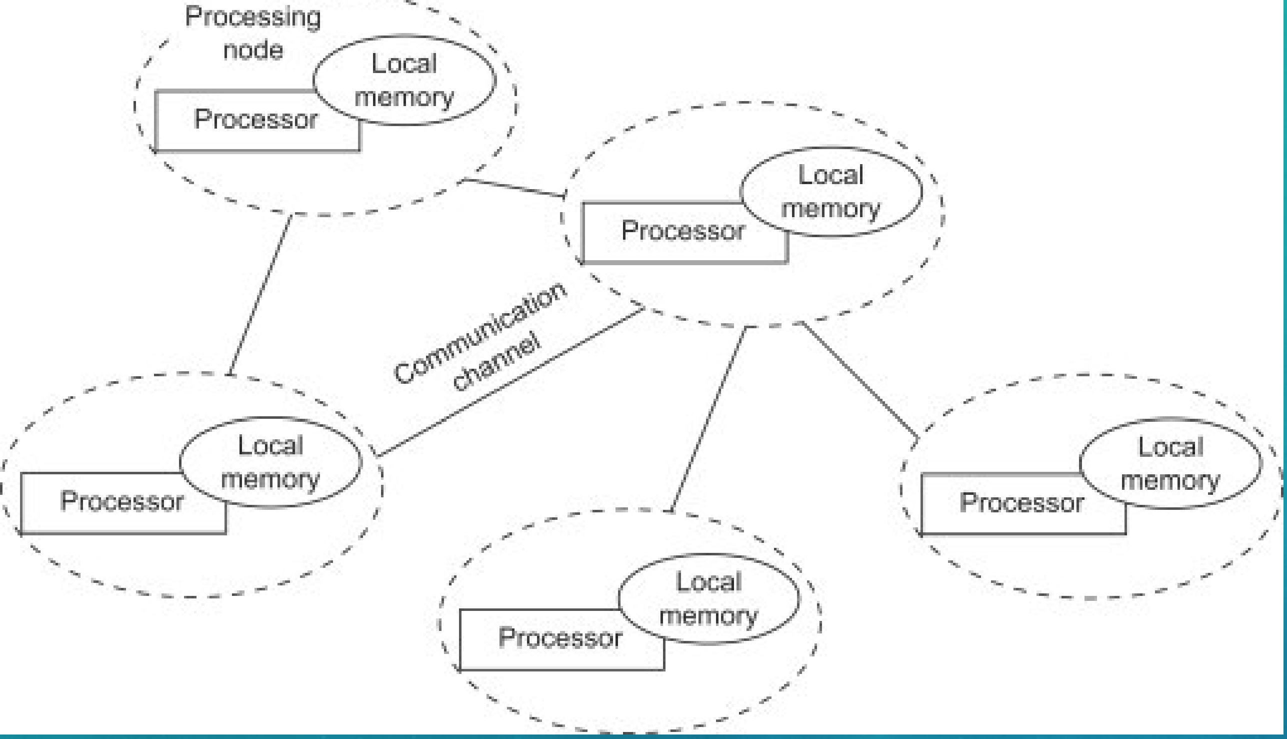
WHY SO HARD?

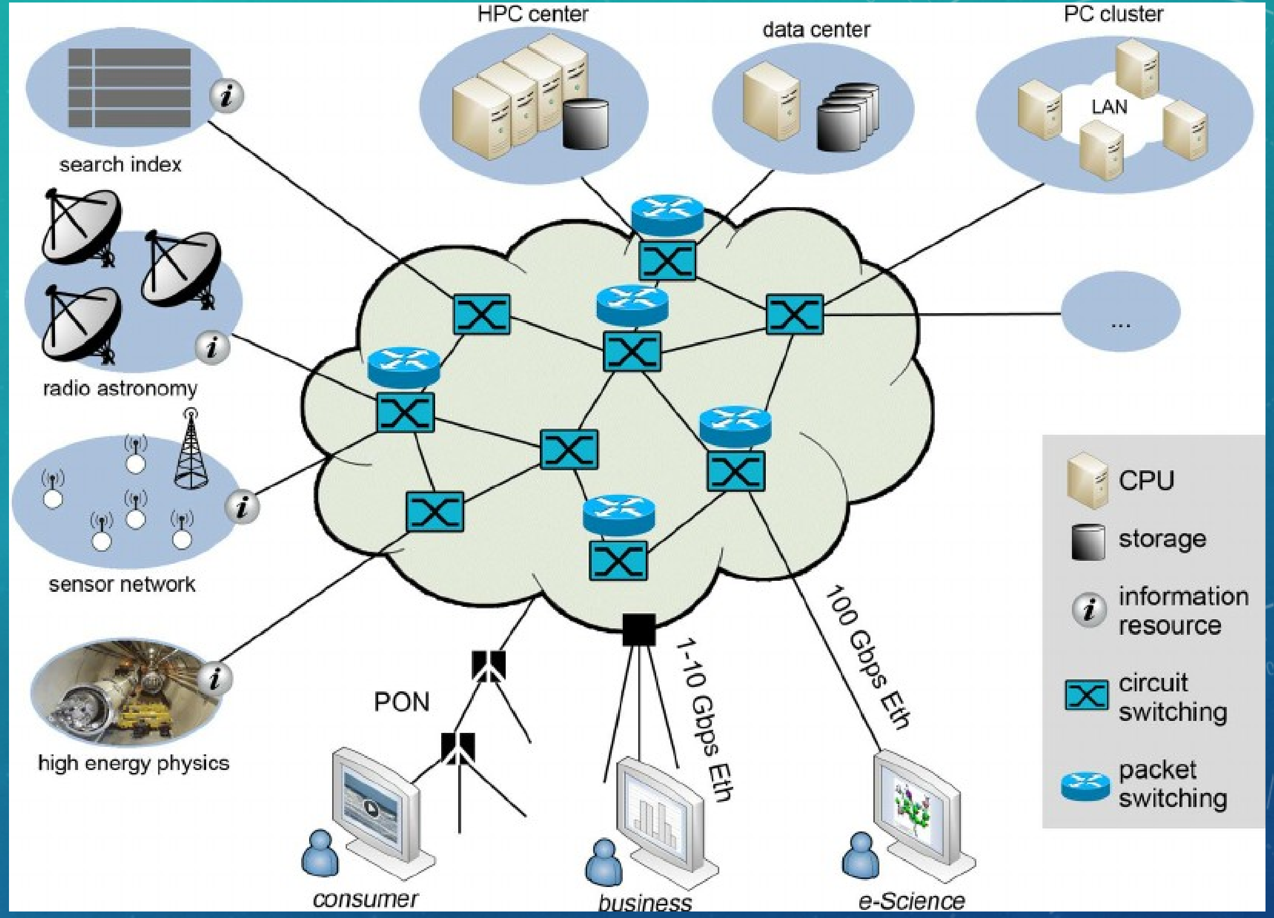
A distributed system is one in which each process has imperfect knowledge of the global state.

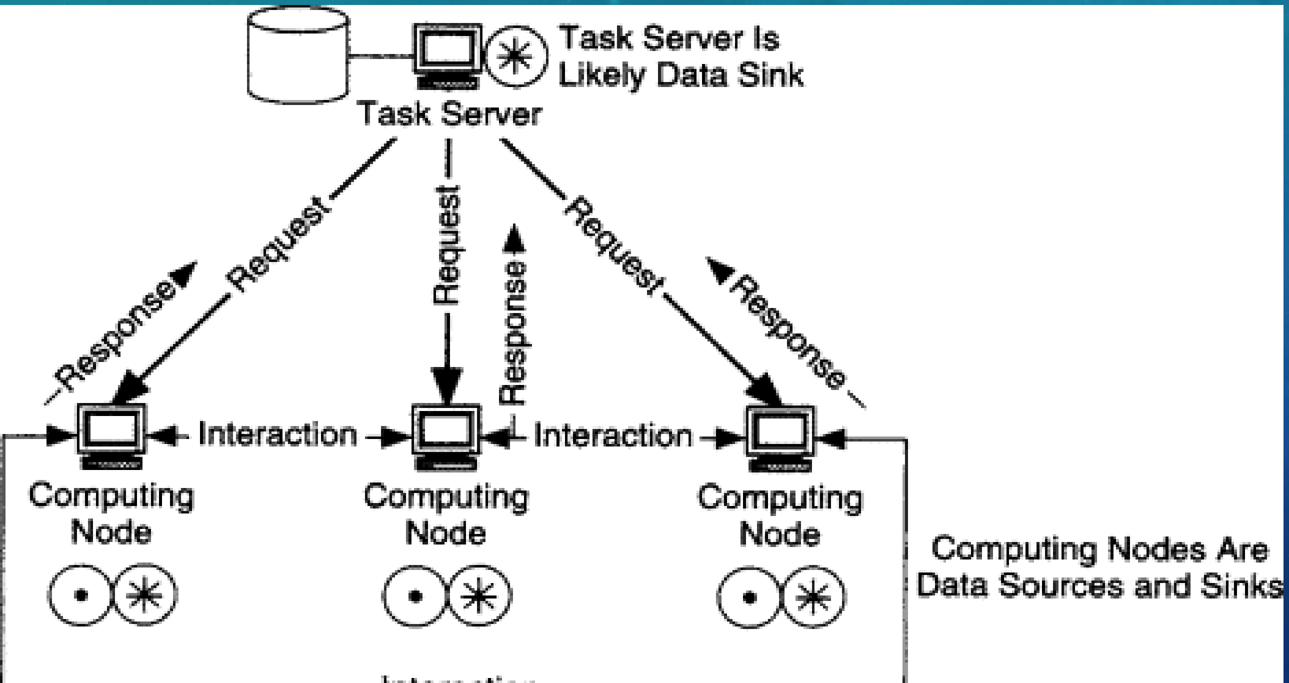
Reasons: Asynchrony and failures

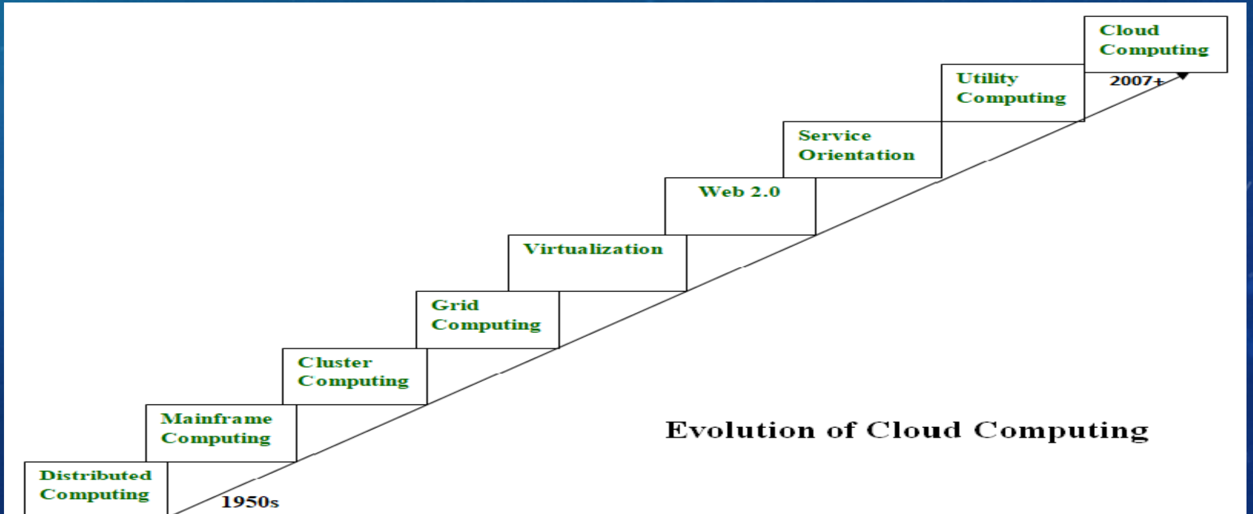
We discuss problems that these two features raise and algorithms to address these problems.

Then we discuss implementation issues for real distributed systems.









Evolution of Cloud Computing

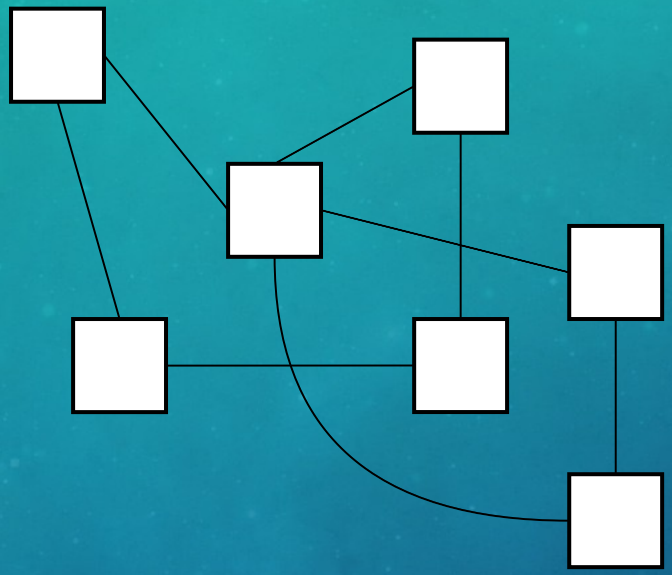
ANATOMY OF A DISTRIBUTED SYSTEM

A set of asynchronous computing devices connected by a network. Normally, no global clock.

Click to add text

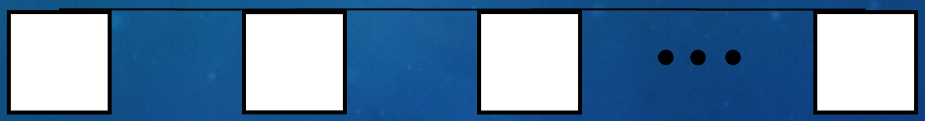
Communication is either through messages or shared memory. Shared memory is usually harder to implement.

ANATOMY OF A DISTRIBUTED SYSTEM (CONT.)



EACH PROCESSOR HAS ITS OWN CLOCK

+ ARBITRARY NETWORK



BROADCAST MEDIUM

Special protocols will be possible for the broadcast medium.

BASIC COMMUNICATION PRIMITIVE: MESSAGE PASSING

Paradigm:

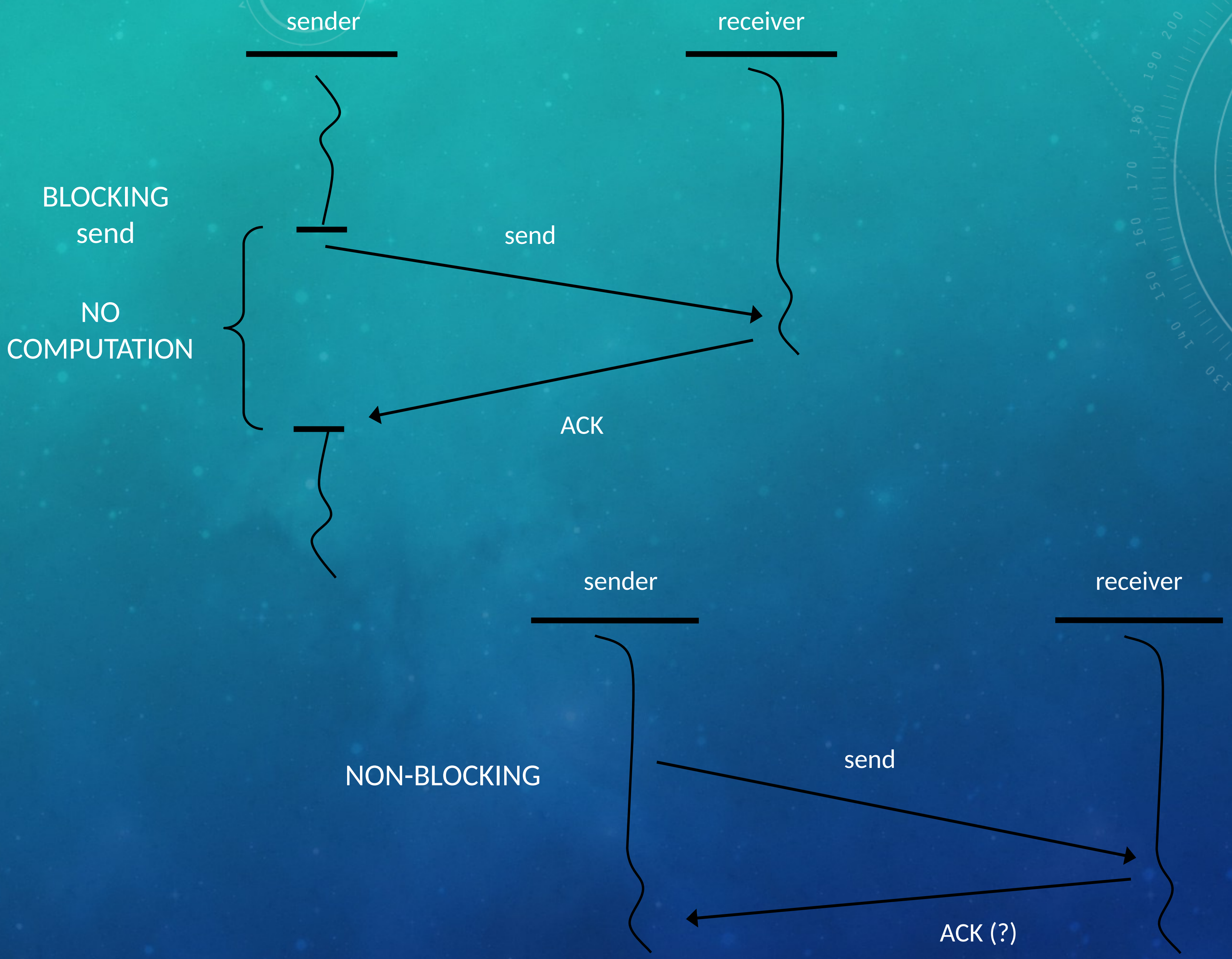
- Send message to destination
- Receive message from origin

Nice property: can make distribution transparent, since it does not matter whether destination is at a local computer or at a remote one (except for failures).

BLOCKING (SYNCHRONOUS) VS. NON-BLOCKING (ASYNCHRONOUS) COMMUNICATION

For sender: Should the sender wait for the receiver to receive a message or not?

For receiver: When arriving at a reception point and there is no message waiting, should the receiver wait or proceed? Blocking receive is normal (i.e., receiver waits).



sender

receiver

BLOCKING
send

NO
COMPUTATION

send

ACK

sender

receiver

NON-BLOCKING

send

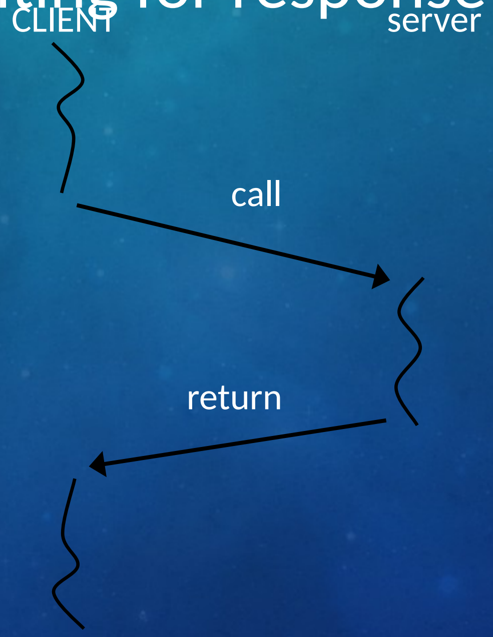
ACK (?)

REMOTE PROCEDURE CALL

Client calls the server using a call server (in parameters; out parameters). The call can appear anywhere that a normal procedure call can.

Server returns the result to the client.

Client blocks while waiting for response from server.



BEYOND SEND-RECEIVE: CONVERSATIONS

Needed when a continuous connection is more efficient and/or only some data at a time.

Bob and Alice: Bob initiates, Alice responds, then Bob, then Alice, …

But what if Bob wants Alice to send messages as they arrive without Bob’s doing more than an ack?

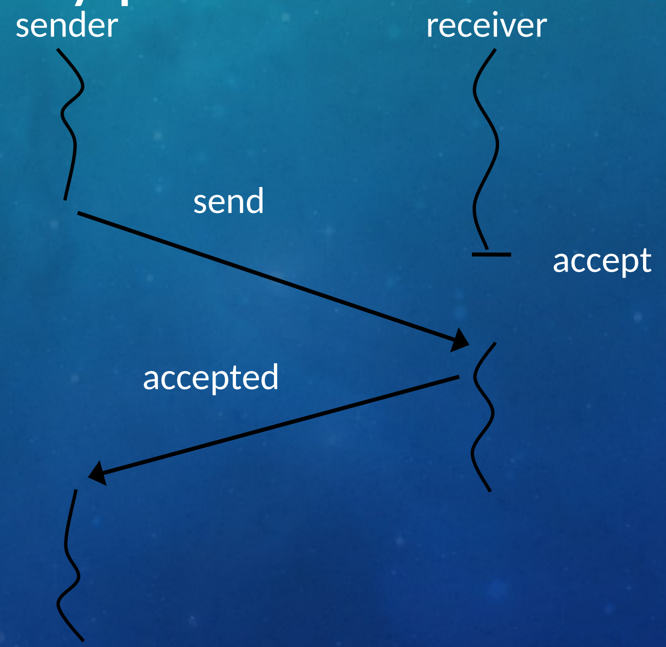
Send only or receive only mode.

Others?

RENDEZVOUS FACILITY

- One process sends a message to another process and blocks at least until that process accepts the message.
- The receiving process blocks when it is waiting to accept a request.

Thus, the name: Only when both processes are ready for the data transfer, do they proceed.


sender
receiver
send
accept
accepted