Message Passing Model

Alessio Vecchio
alessio.vecchio@unipi.it
Dip. di Ingegneria dell'Informazione
Università di Pisa

Based on original slides by Silberschatz, Galvin, and Gagne
Operating System Concepts

Inter-Process Communication (IPC) – Message Passing

- Processes communicate with each other without resorting to shared variables
- IPC facility provides two operations:
  - **`send`**(*message*)
  - **`receive`**(*message*)
- The *message* size is either fixed or variable

Message Passing (Cont.)

- If processes *P* and *Q* wish to communicate, they need to:
  - Establish a ***communication link*** between them
  - Exchange messages via send/receive
  - The communication link is provided by the OS

- Implementation issues:
  - How are links established?
  - Can a link be associated with more than two processes?
  - How many links can there be between every pair of communicating processes?
  - What is the capacity of a link?
  - Is the size of a message that the link can accommodate fixed or variable?
  - Is a link unidirectional or bi-directional?

Implementation Issues

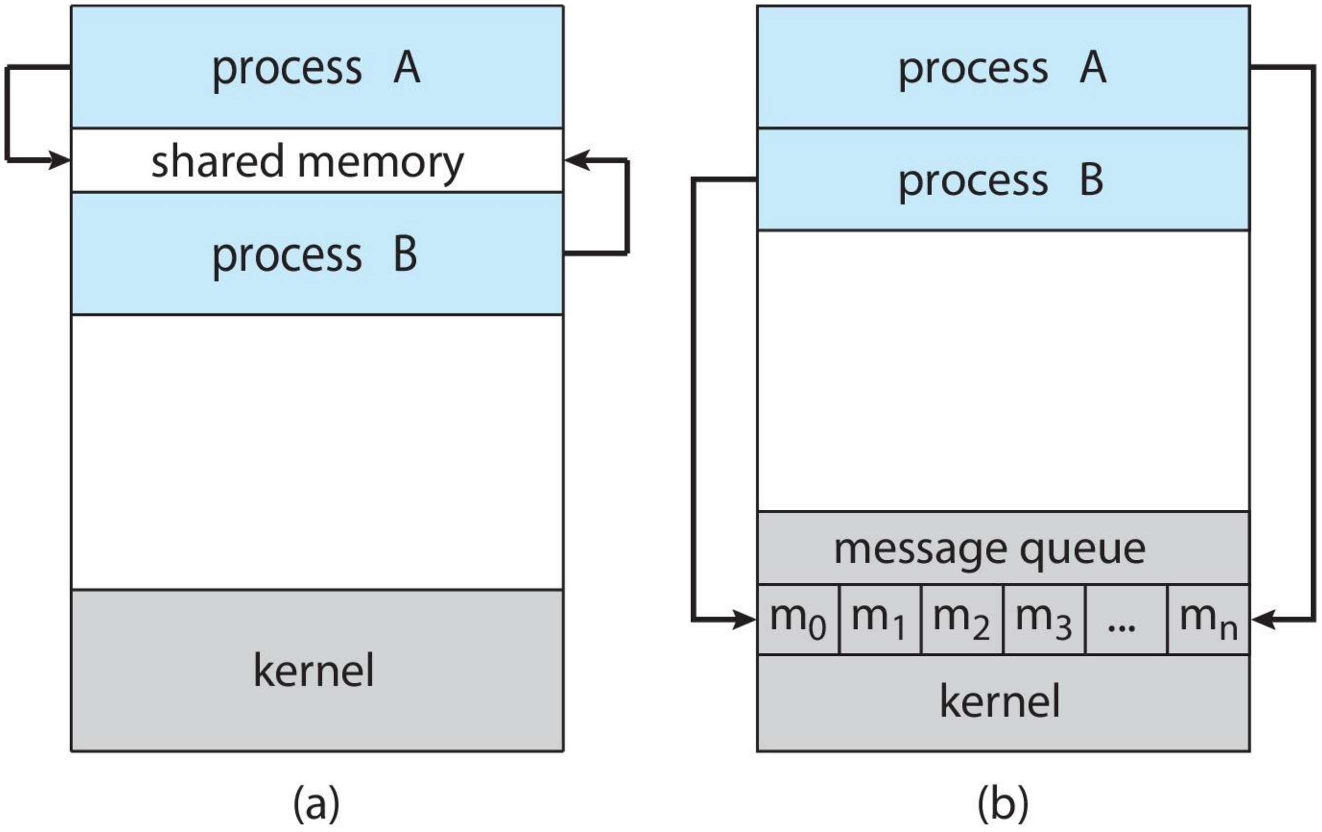
Physical implementation

- Single-processor system
  - Shared memory
- Multi-processor systems
  - Hardware bus
- Distributed systems
  - Networking System + Communication networks

Communications Models

(a) Shared memory.

(b) Message passing.



Implementation Issues

Logical properties

- Direct or indirect
- Synchronous or asynchronous
- Automatic or explicit buffering

Direct Communication

- Processes must name each other explicitly:
  - **`send`** (*P, message*) – send a message to process P
  - **`receive`**(*Q, message*) – receive a message from process Q
- Properties of communication link
  - Links are established automatically
  - A link is associated with exactly one pair of communicating processes
  - Between each pair there exists exactly one link
  - The link may be unidirectional, but is usually bi-directional

Direct Addressing

- Processes must name each other explicitly
- Symmetric scheme
  - send (D, message) – send a message to process D
  - receive(S, message) – receive a message from process S
- Logical properties
  - Links are established automatically
  - A link is associated with exactly one pair of communicating processes
  - Between each pair there exists exactly one link

Direct Addressing

- Asymmetric scheme
  - send (D, message) – send a message to process D
  - receive(proc, message) - receive a message from any process proc

Indirect Addressing

- Messages are sent/received through mailboxes
  - shared data structures where messages are queued temporarily. Sometimes referred to as ports
- Processes can communicate only if they share a mailbox
  - Each mailbox has a unique id
- Primitives are defined as:
  - send(mb, message) – send a message to mailbox mb
  - receive(mb, message) – receive a message from mailbox mb

Indirect Communication

- Operations
  - create a new mailbox
  - send and receive messages through mailbox
  - destroy a mailbox
- Properties of communication link
  - Link established only if processes share a common mailbox
  - A link may be associated with many processes
  - Each pair of processes may share several communication links
  - Link may be unidirectional or bi-directional
- Relationships

One-to-one	(private communication)
Many–to-one	(client-server communication)
Many-to-many	(multicast communication)

Synchronization

Message passing may be either blocking or non-blocking

- **Blocking** is considered **synchronous**
  - **Blocking send** -- the sender is blocked until the message is received
  - **Blocking receive** -- the receiver is blocked until a message is available
- **Non-blocking** is considered **asynchronous**
  - **Non-blocking send** -- the sender sends the message and continue
  - **Non-blocking receive** -- the receiver receives:
    - A valid message, or
    - Null message

Synchronization

- Blocking send, blocking receive
  - Rendez-vous between sender and receiver
- Non-blocking send, blocking receive
  - Most useful combination (used by servers)
  - Variations: receive with timeout, select, proactive test
- Non-blocking send, Non-blocking receive
  - Neither party is required to wait

Buffering

- Queue of messages attached to the link
- Implemented in one of three ways.
  - Zero capacity – 0 messages
    Sender must wait for receiver (in fact, this introduces a rendezvous).
  - Bounded capacity – finite length of n messages
    Sender must wait if the link full.
  - Unbounded capacity – infinite length
    Sender never waits.

Producer-Consumer: Solution (1)

```
Mailbox mb;
```

```
Process Producer {
  while (true) {
    // message in nextProduced
    send(mb, nextProduced);
  }
}
```

```
Process Consumer {
     while (true) {
        receive(mb, msg);
        // consume message
      }
}
```

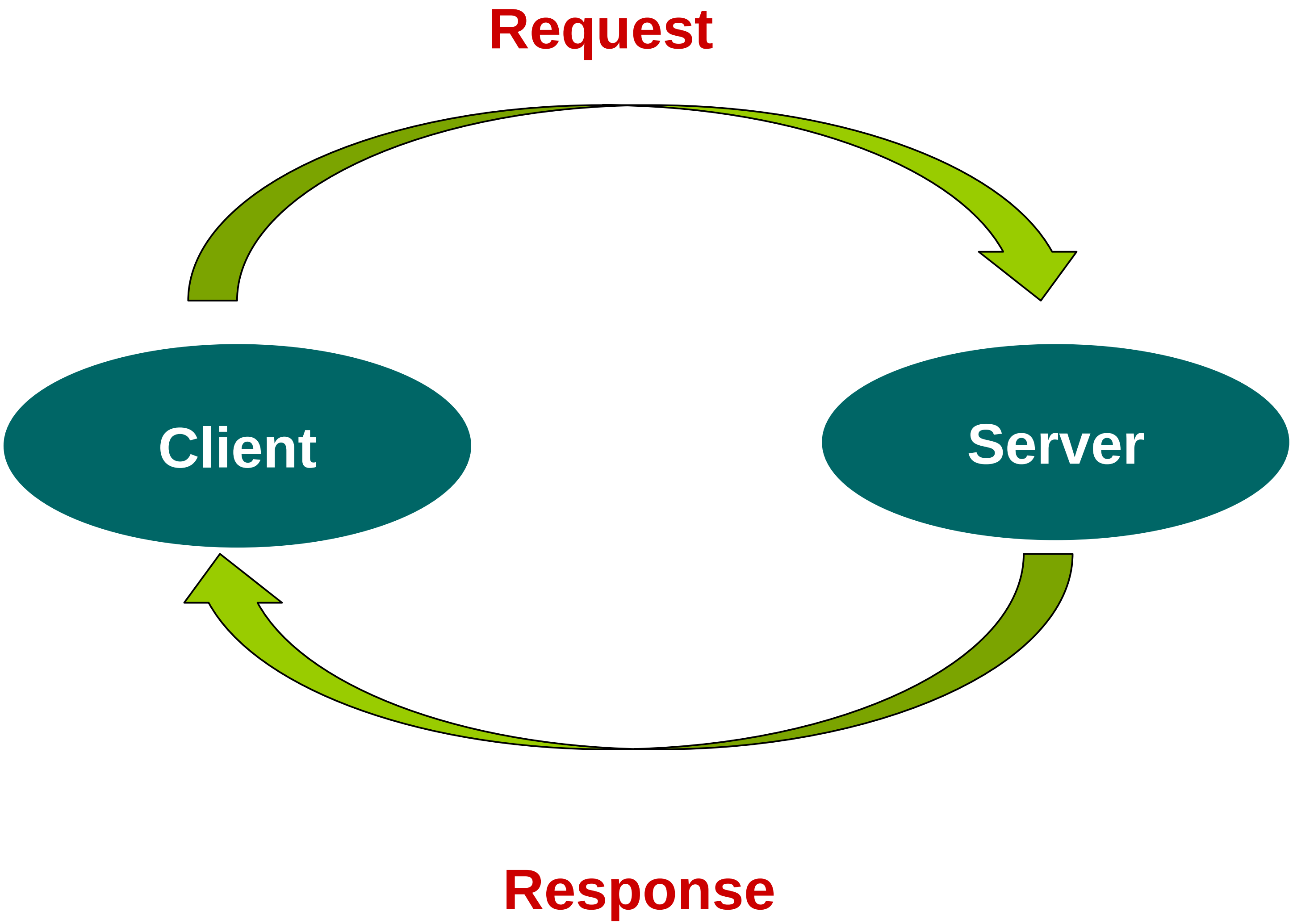
Producer-Consumer: Solution (2)

```
Mailbox mb1, mb2;
```

```
Process Producer {
  while (true) {
    // message in nextProduced
    receive(mb2, ack);
    send(mb1, nextProduced);
  }
}
```

```
Process Consumer {
    while (true) {
      send(mb2, READY);
      receive(mb1, msg);
      // consume message
    }
}
```

Client-Server Communication


Request

Client

Server

Response