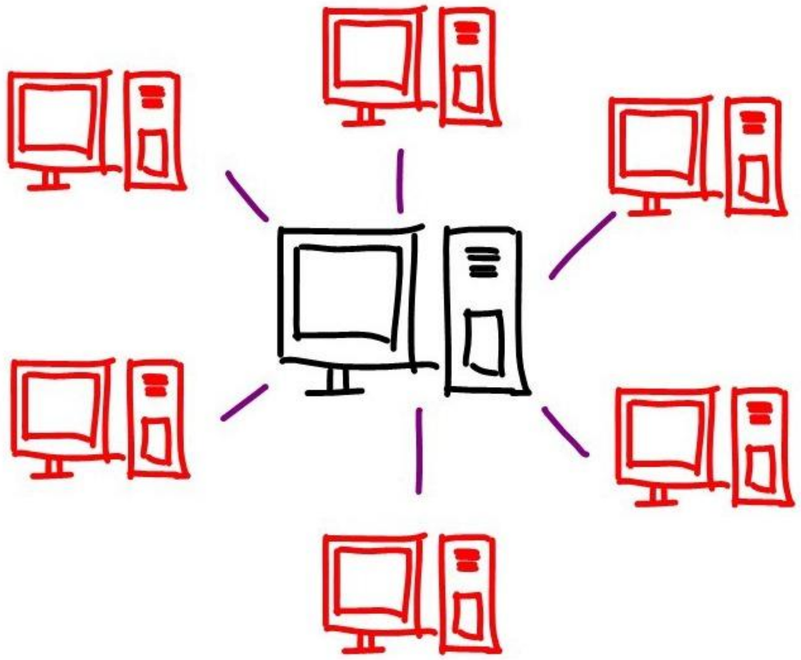


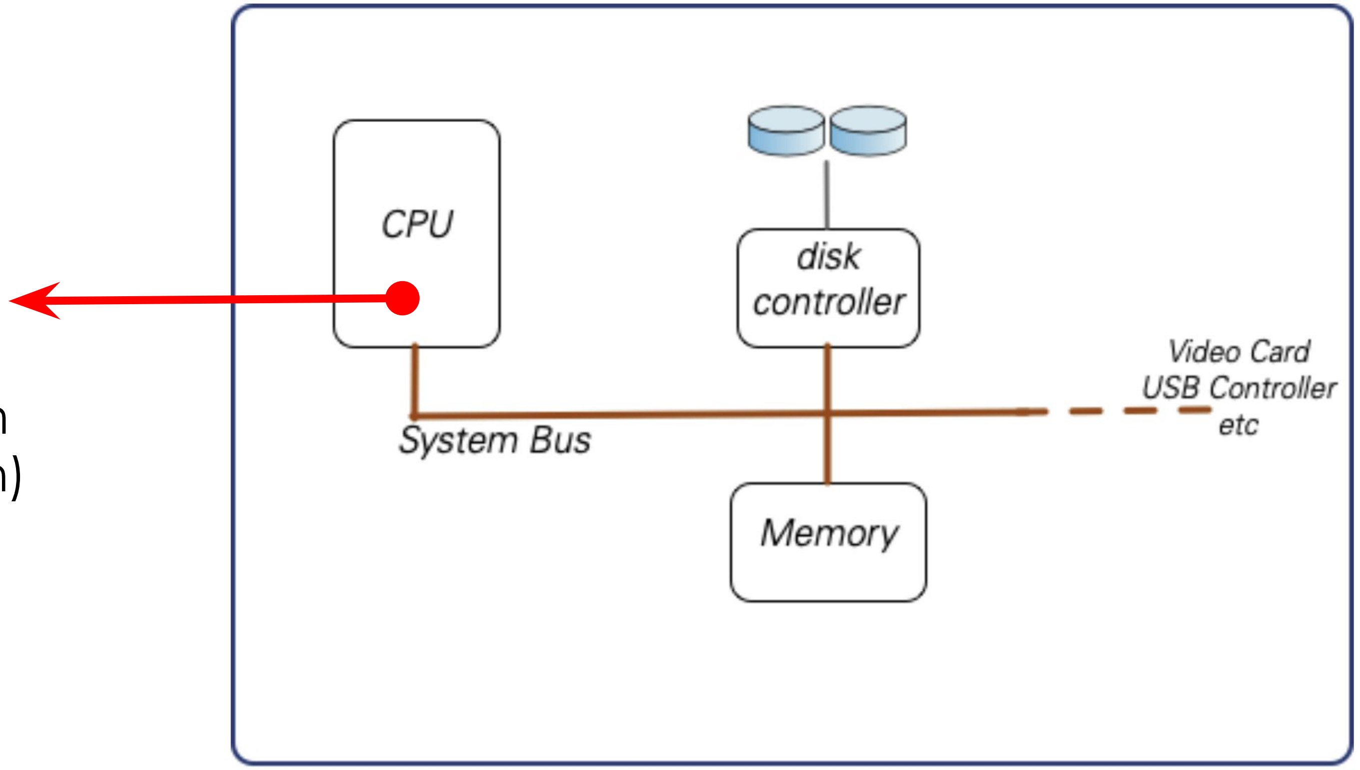
Distributed Systems and CORBA Standard

Rosita Hormann Lobos

Standalone computer

process

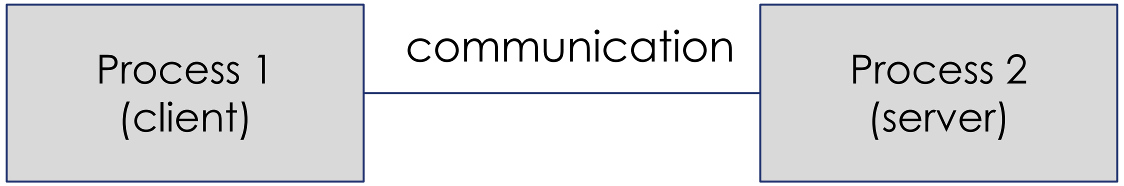
(a program in execution)

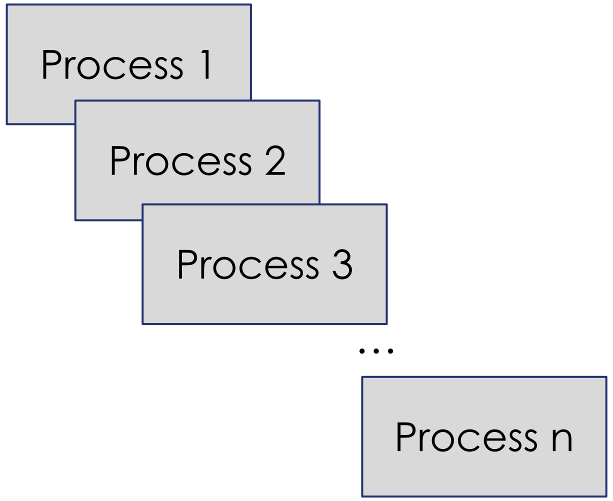


In a fair computing scenario, when multiple processes are executed, they "*take turns*" of computing time of the CPU. This is managed by a kernel-level process called *scheduler* and is what allows multi-processing.

InterProcess Communication (IPC)

IPC are mechanisms to allow the processes to communicate among them



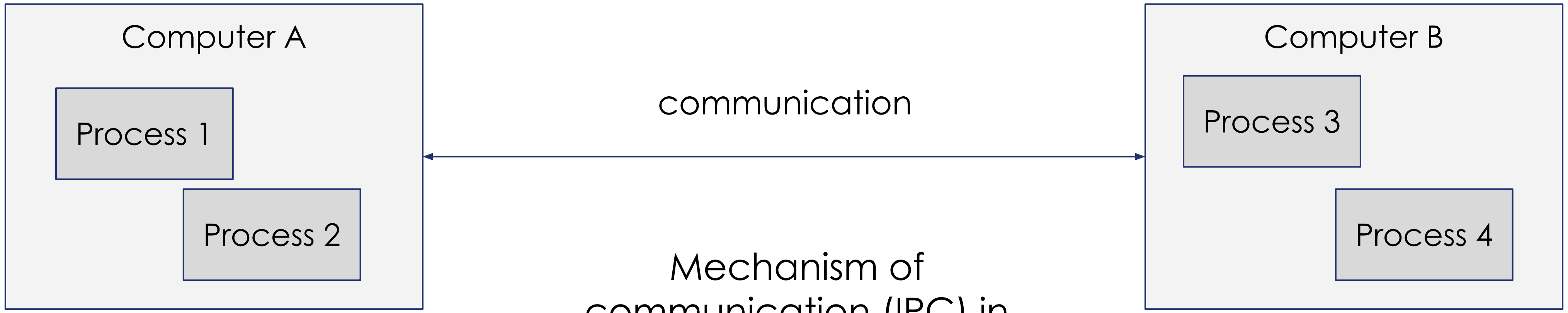


Mechanisms of communication in standalone computer:

- Shared memory
- Mailboxes
- Pipes (ex: `ls -l | grep txt`)

Distributed System

A collection of autonomous computing elements that appears to its users as a single coherent system



Mechanism of communication (IPC) in distributed systems:
through the Network
depends on network IPC protocol (dce/rpc, mcrpc, gIOP (CORBA), grpc)

Distributed systems paradigms

- Message-exchange pattern
- Publish-Subscribe Message Model (ex: RabbitMQ, ActiveMQ, Kafka)
- Request-reply
- Remote Procedure Call (RPC)
  - Distributed Objects

Distributed Objects

- Distributed applications using OOP paradigm.
- Application Objects distributed over the network
- Objects provide methods. Through them other Objects in the network access to services → Inter-Process Communication is done via methods calling through the network
- Examples: Enterprise Java Beans, Microsoft DCOM, Java RMI, ZeroC ICE, CORBA

CORBA: A Distributed Object-Based System

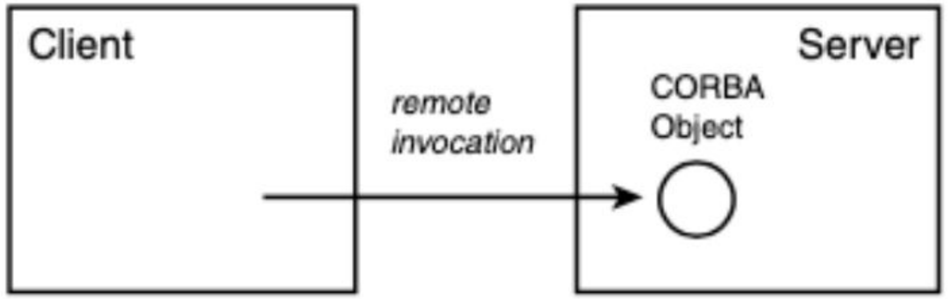
CORBA = Common Object Request Broker Architecture

- Is a standard designed to facilitate the communication of systems that are deployed on diverse platforms.
- Enables collaboration between systems on different operating systems, programming languages, and computing hardware.
- Uses an object-oriented model.

< ACS is built on top of CORBA >

CORBA characteristics

- Object-orientation
  - Remote Operations grouped into interfaces
  - An instance of an Interface is a CORBA object
  - The identity of a Corba object (object reference) is unique. Encapsulates all information about the object like location information



CORBA characteristics

- Location-transparency
  - It does not matter where the CORBA object is located (local or remote). The operations are invoke using the same syntax.

- Programming Language Neutral
  - Designed to work with multiple programming Languages.
  - Interfaces definition is implemented in a common-language (IDL)
  - In ACS we use C++, Java and Python.

IDL and Language Mapping

IDL = Interface Definition Language

IDL is a declarative language, it defines an interface with:

- Methods
- datatypes

It DOES NOT define an implementation, just the interface.

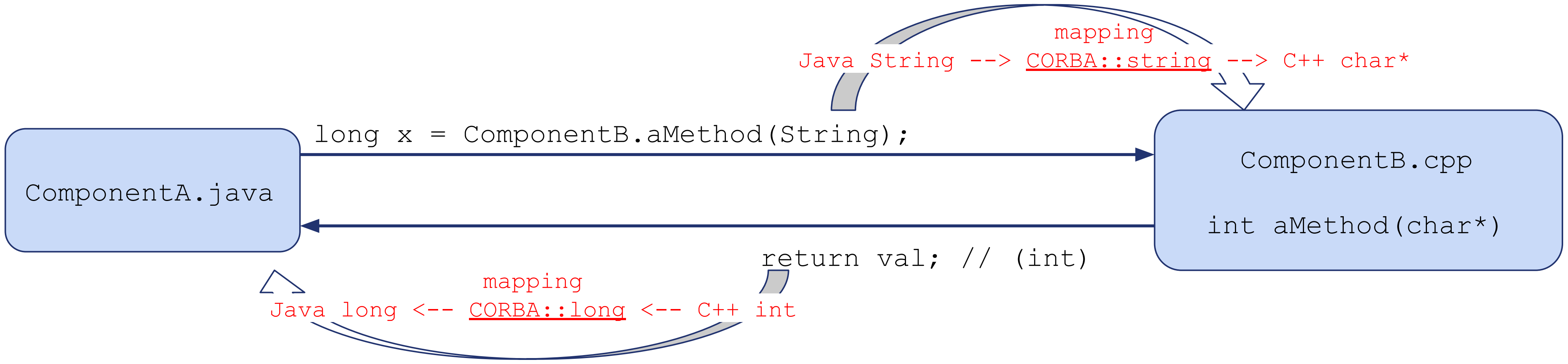
```
module HelloApp {

  interface Hello {
    string sayHello();
    oneway void shutdown();
  }

}
```

IDL and Language Mapping

CORBA uses IDL to specify the interfaces that objects present to the outer world. CORBA then specifies a mapping from IDL to a specific implementation language like C++ or Java.


mapping
Java String --> CORBA::string --> C++ char*
long x = ComponentB.aMethod(String);
ComponentA.java
ComponentB.cpp
int aMethod(char*)
return val; // (int)
mapping
Java long <-- CORBA::long <-- C++ int

When an object calls remote method, CORBA handles the values passed in the call and returns the method return value. The passing values are **serialized** (CDR)and transported in a wire-protocol (GIOP).

ACS services on top of CORBA

- Distributed error
- Distributed logging
- Distributed Events (Notification Channel)
- Threading Support