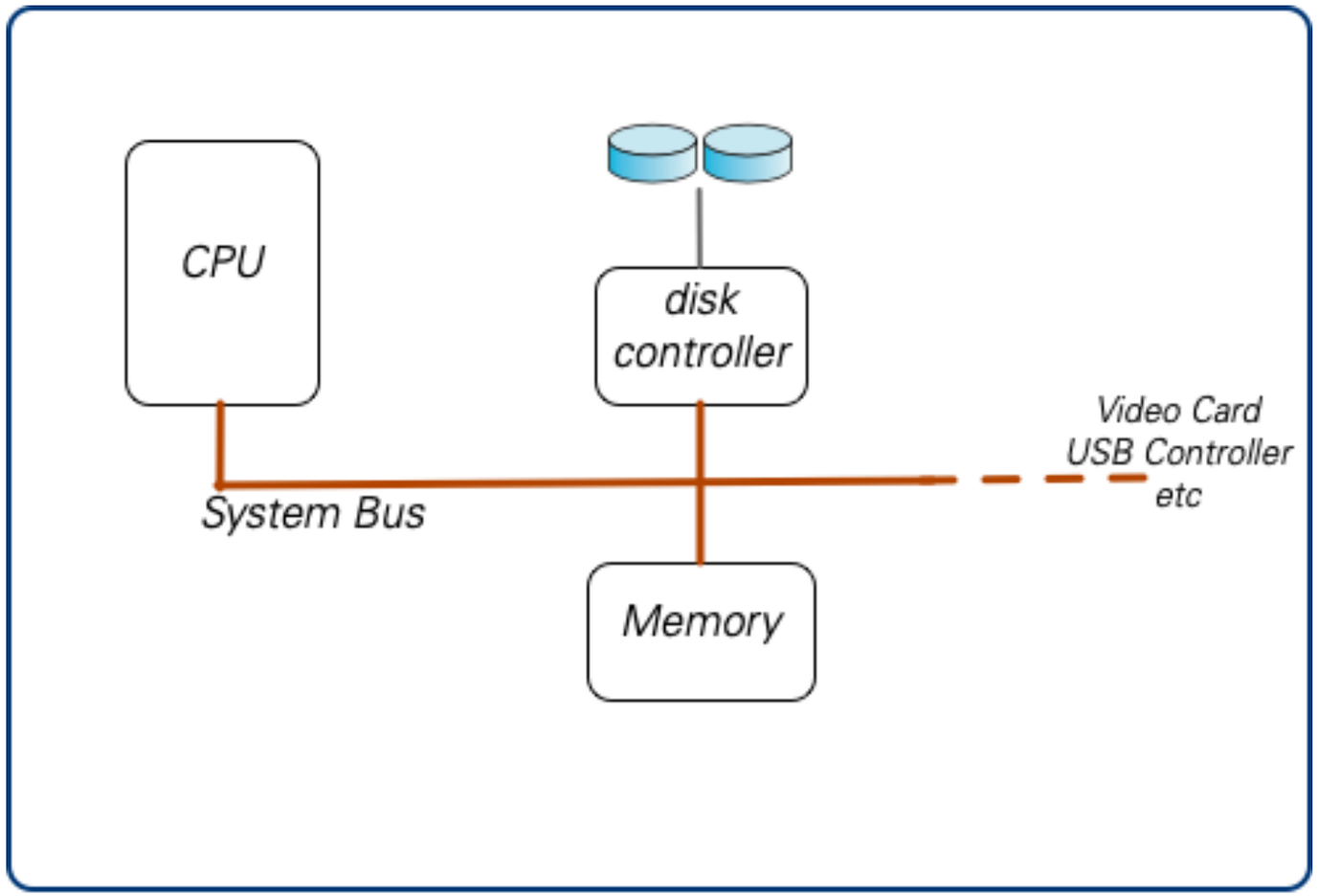
Distributed Systems and CORBA Standard

Pablo Burgos

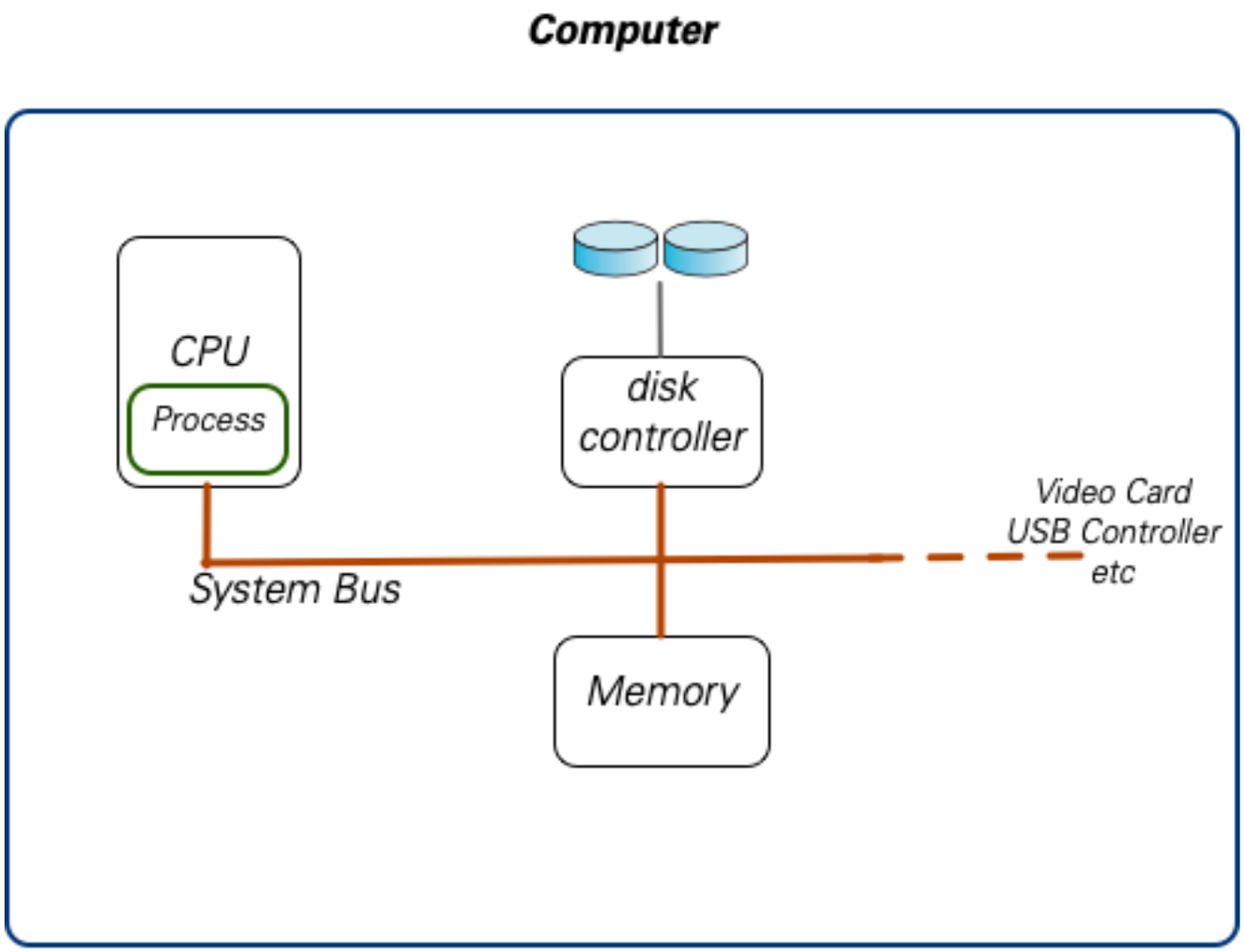
Introduction

Standalone Computer



Computer

Process



Process: A program in execution

```
❯ tail -f /var/log/system.log
```

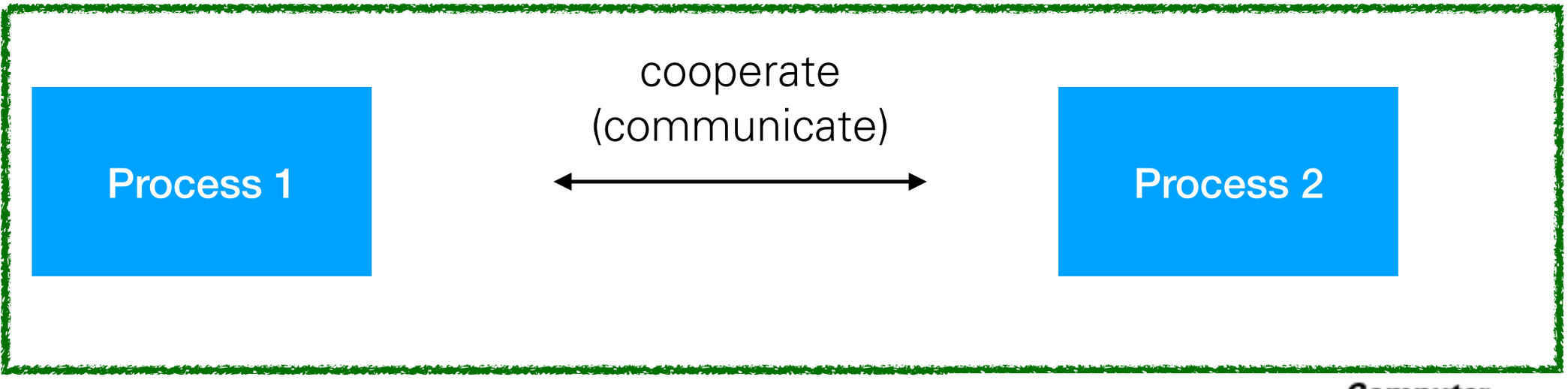
```
❯ ps -ef | grep 'tail '
  501 96417 16091   0 10:47PM ttys006    0:00.04 tail -f /var/log/system.log
```

InterProcess Communication (IPC)

Process is **cooperating** if it can affect or be affected by the other processes in execution in the system

Driver to have cooperation in a system:
- Information Sharing
- Computation Speed up
- Modularity

InterProcess Communication (IPC)

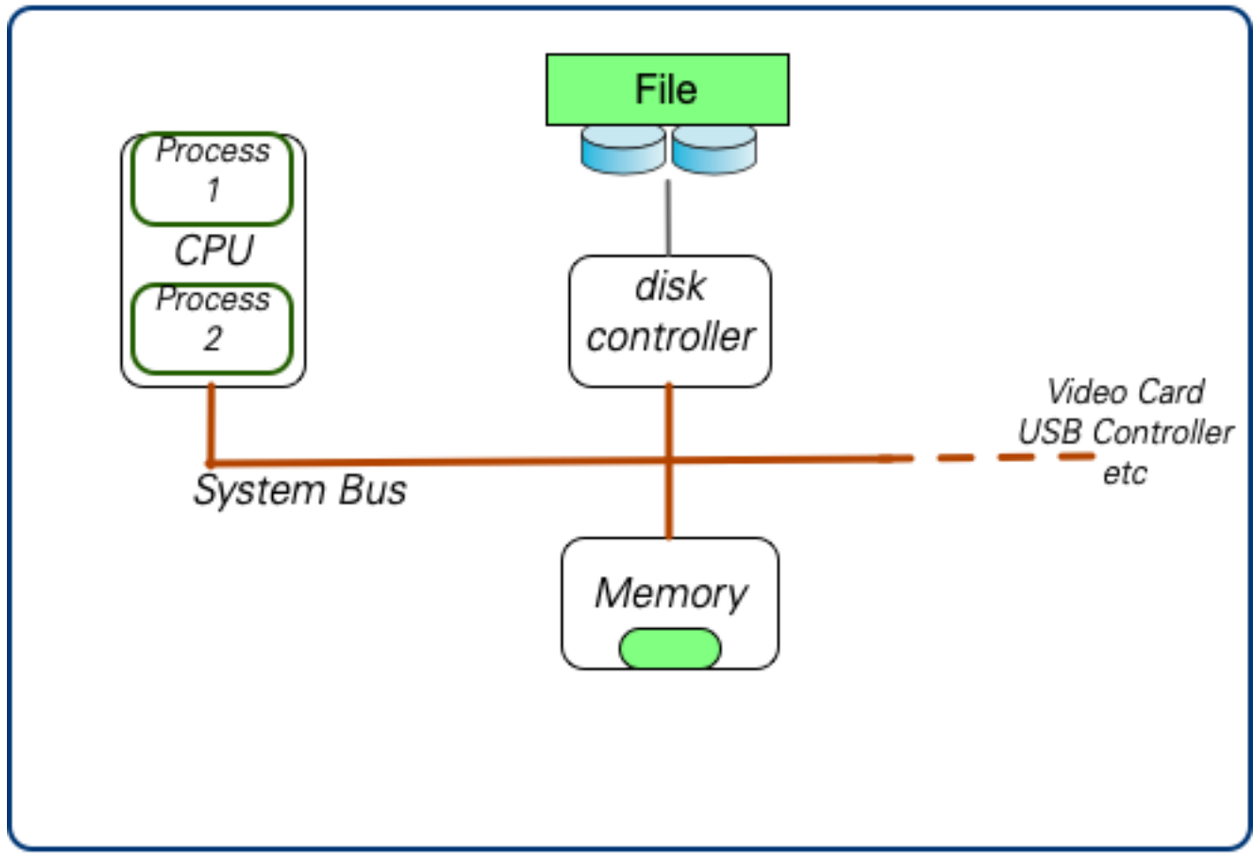


Some Mechanisms

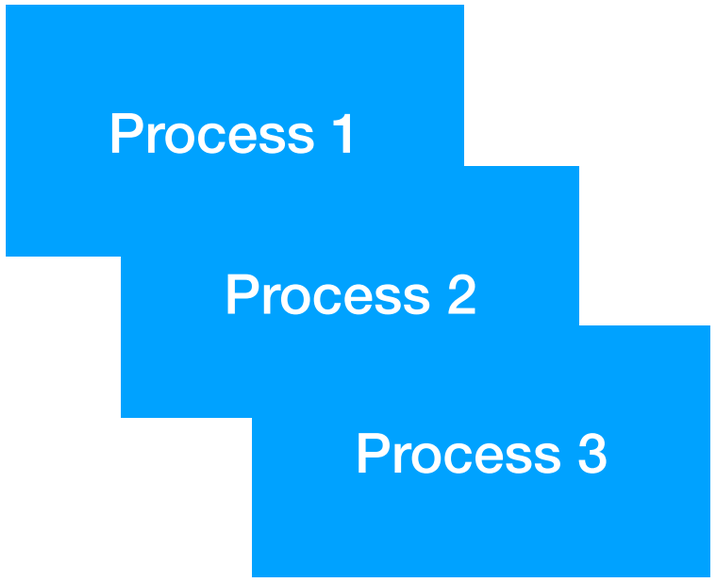
- **Shared Memory**
- **Named Pipes**

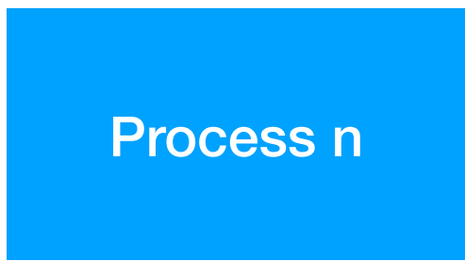
```
❯ mkfifo WSPipe
❯ ls -la WS*
prw-r--r--  1 pburgos  staff  0 Jul 27 06:05 WSPipe
❯ echo "Hello ACS Workshop 2020" > WSPipe
```

```
..nsible_acsen
❯ cat < WSPipe
Hello ACS Workshop 2020
```



Problems with a Monolithic Computer





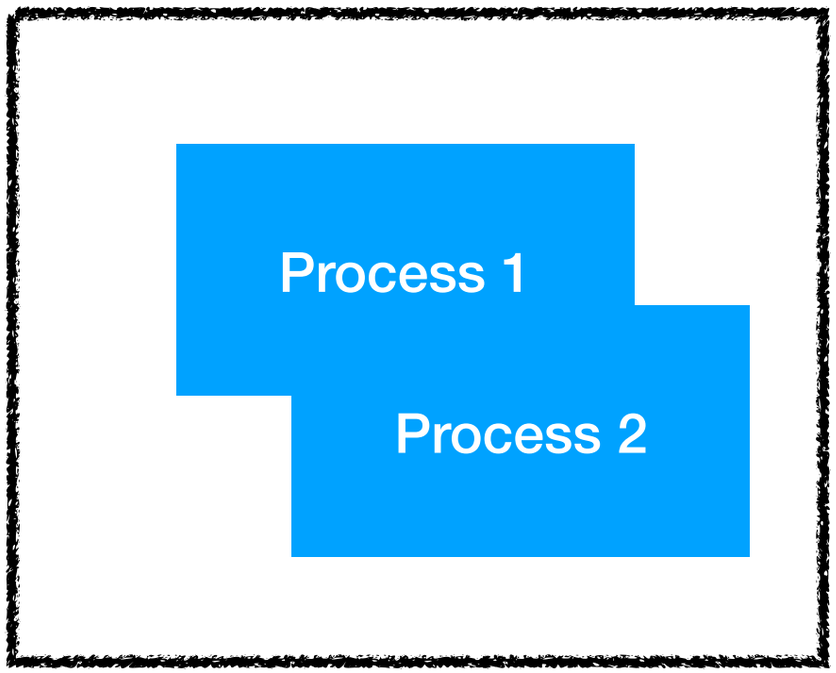
Service Capacity

When utilization is low response to service ratio close to 1

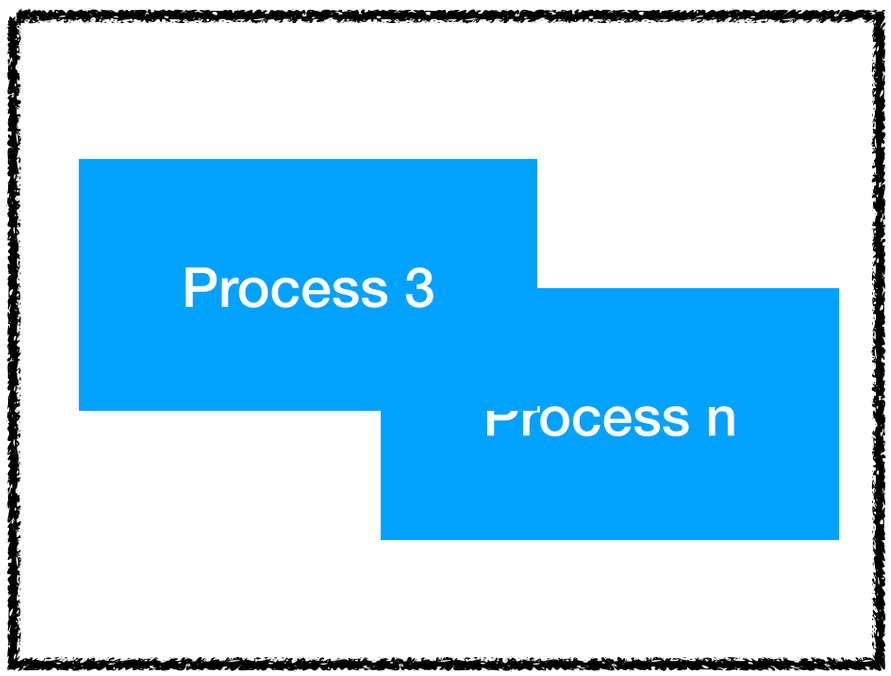
Requests instantly processed

When utilization is close to 1 response to service ratio increases quickly (Not lineal)

Divide and Conquer

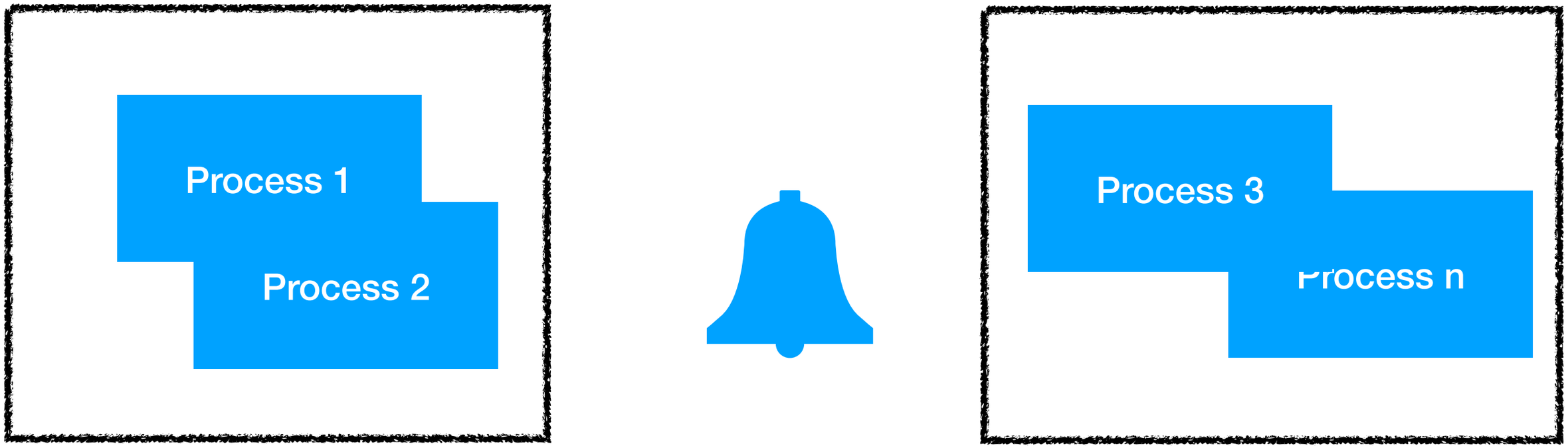


Computer 1

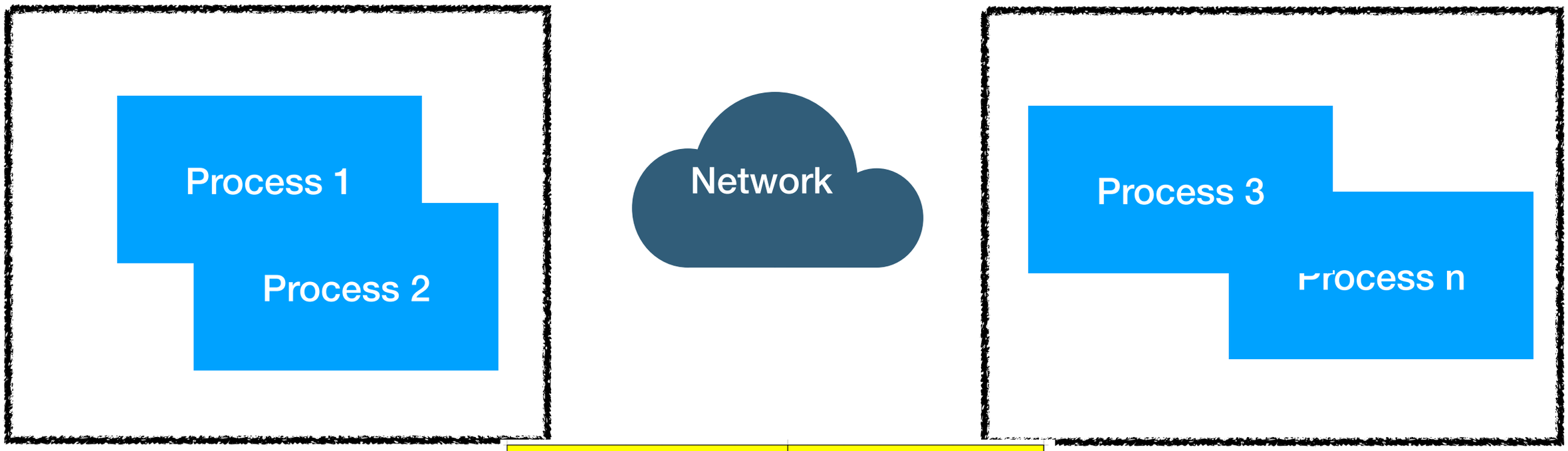


Computer 2

Are we missing something?



Communication Network



OSI Model	TCP/IP Model
Application Layer	Application layer
Presentation Layer	
Session Layer	
Transport Layer	Transport Layer
Network Layer	Internet Layer
Data link layer	Link Layer
Physical layer	

Distributed Systems

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

Goals and Pitfalls

Goals:

- Support resource Sharing
- Distribution Transparency
  - Location
  - Access
  - Concurrency
  - Failure
- Scalable
- Increase computing power
- Economy

Pitfalls/ False assumptions:

- Network is reliable
- Network is secure
- Network is homogeneous
- Topology don't change
- Latency is Zero
- Bandwidth is infinite
- Transport cost is zero
- There is one administrator

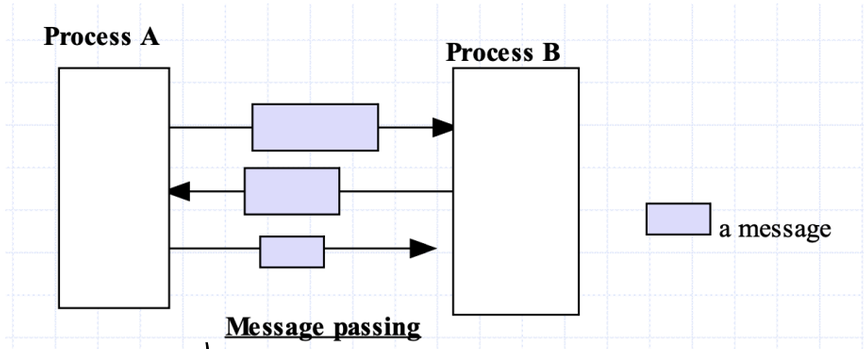
Components disperse across the network

Paradigms

Level of Abstraction

\- +

Message-Passing



send(receiver, &message)

receive(sender, &message)

Message-System

Publish-Subscribe Msg Model

Examples: Sockets, RabbitMQ, Apache Kafka, ActiveMQ

Remote Procedure Call

Distributed Objects

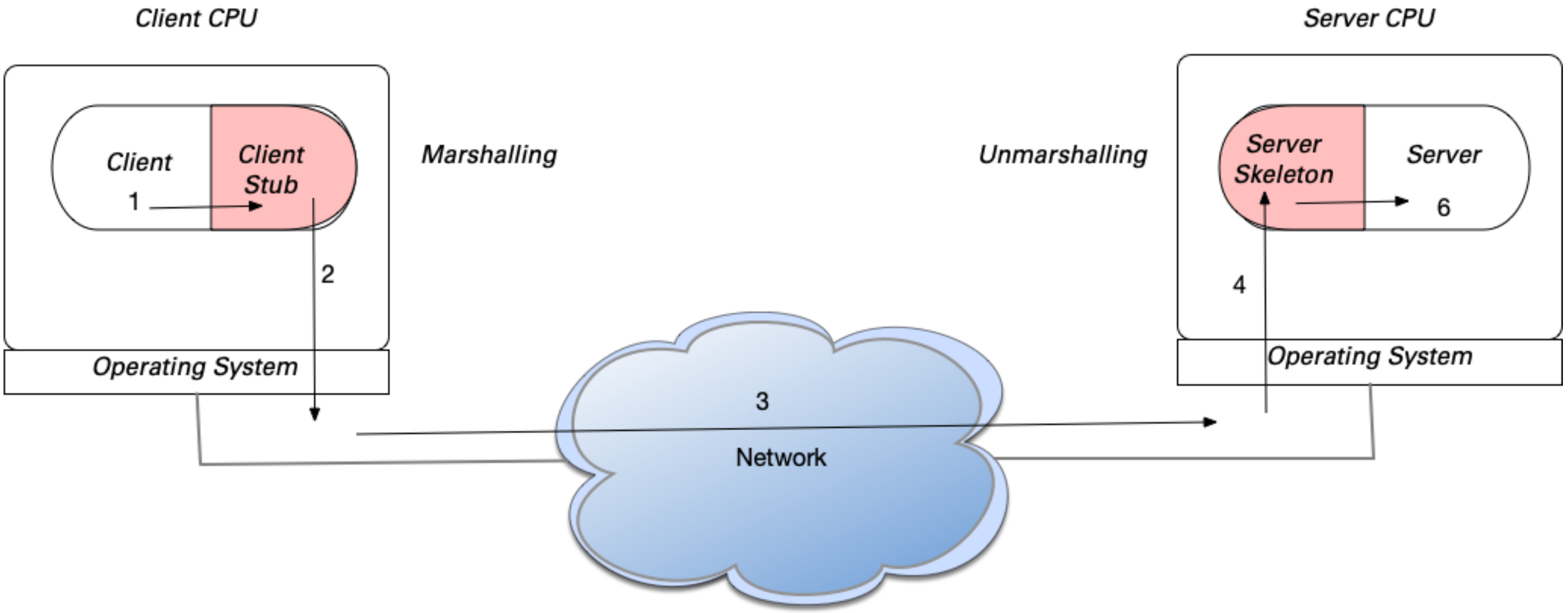
RPC Remote Procedure Call

As application get more complex, it is desired to have a paradigm to allow for distributed software to be programmed in a similar way than conventional applications that run on a single processor

Birrel&Nelson (1984) suggested to allow programs running in a CPU to call procedures located in other CPU

RPC attempts to make a remote procedure call look as much as possible as a local one

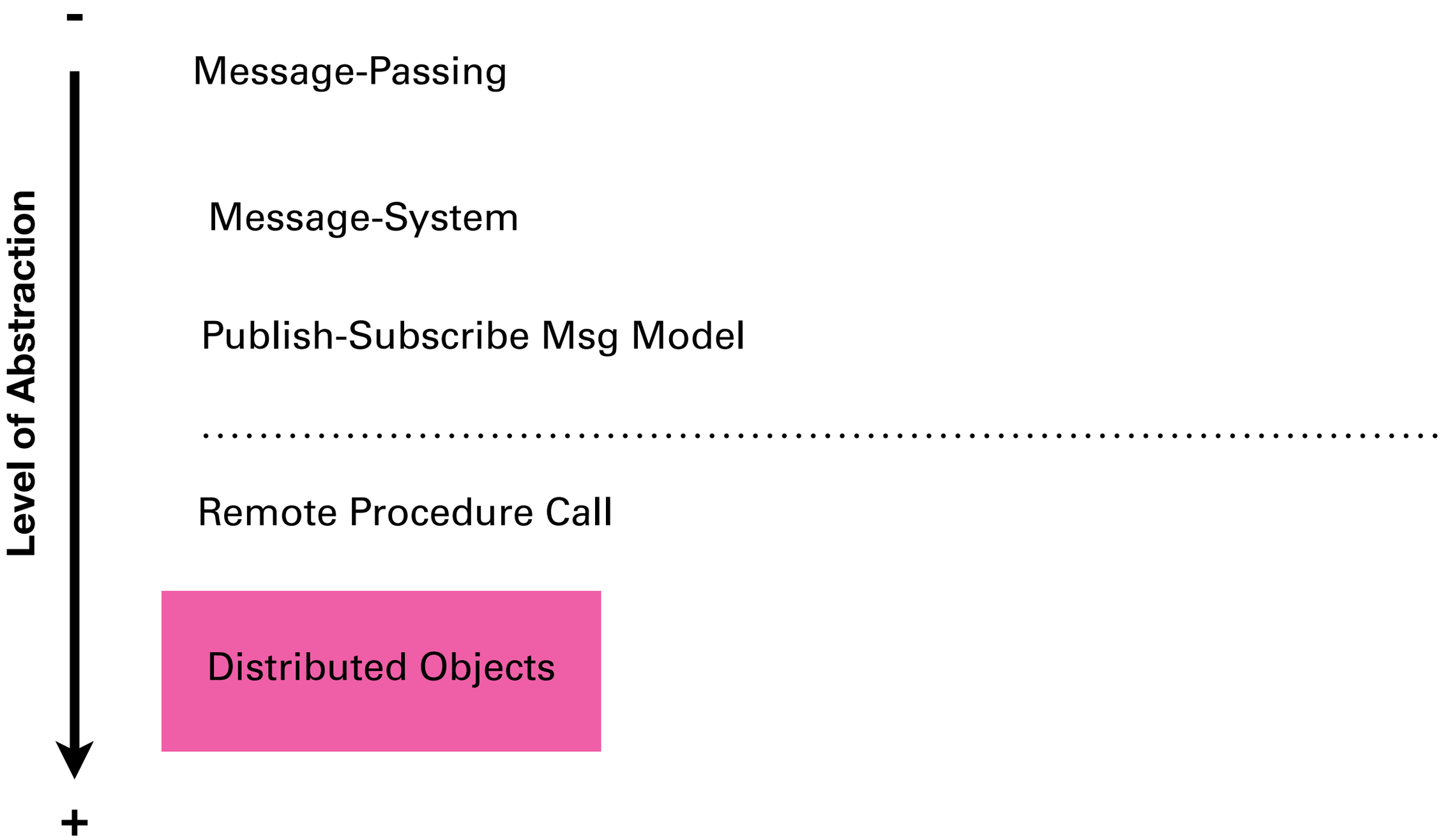
RPC: Remote Procedure Call



Stub: **converts** parameters passed between client and server during a RPC.

Client Stub: represents server procedure in the client address space.
Server Stub or Skeleton: likewise on the server side

Distributed Objects Paradigm



Distributed Objects Paradigm

Apply Object Orientation to distributed Applications natural extension of OOP

Application Objects distributed over the network

Objects provide methods. Through them other Objects in the network access to services.

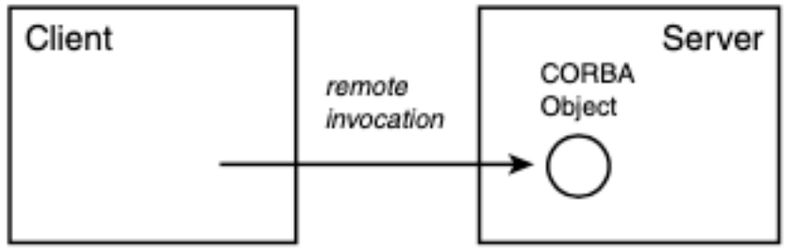
Examples: Enterprise Java Beans, Microsoft DCOM, Java RMI, ZeroC ICE, CORBA

CORBA: A Distributed Object-Based System

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

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.

C++, Python, Java, Ruby

Basic CORBA Concepts

Interface Definition Language (IDL)

time.idl

```
struct TimeOfDay {
    short   hour;       // 0 - 23
    short   minute;     // 0 - 59
    short   second;     // 0 - 59
};

interface Time {
    TimeOfDay   get_gmt();
};
```

IDL is a declarative language. Can not be used to describe algorithms or evaluate expressions.

IDL was optimized to fit different programming languages.

Language Mappings

How to translate IDL definitions to the particular constructs of the target language

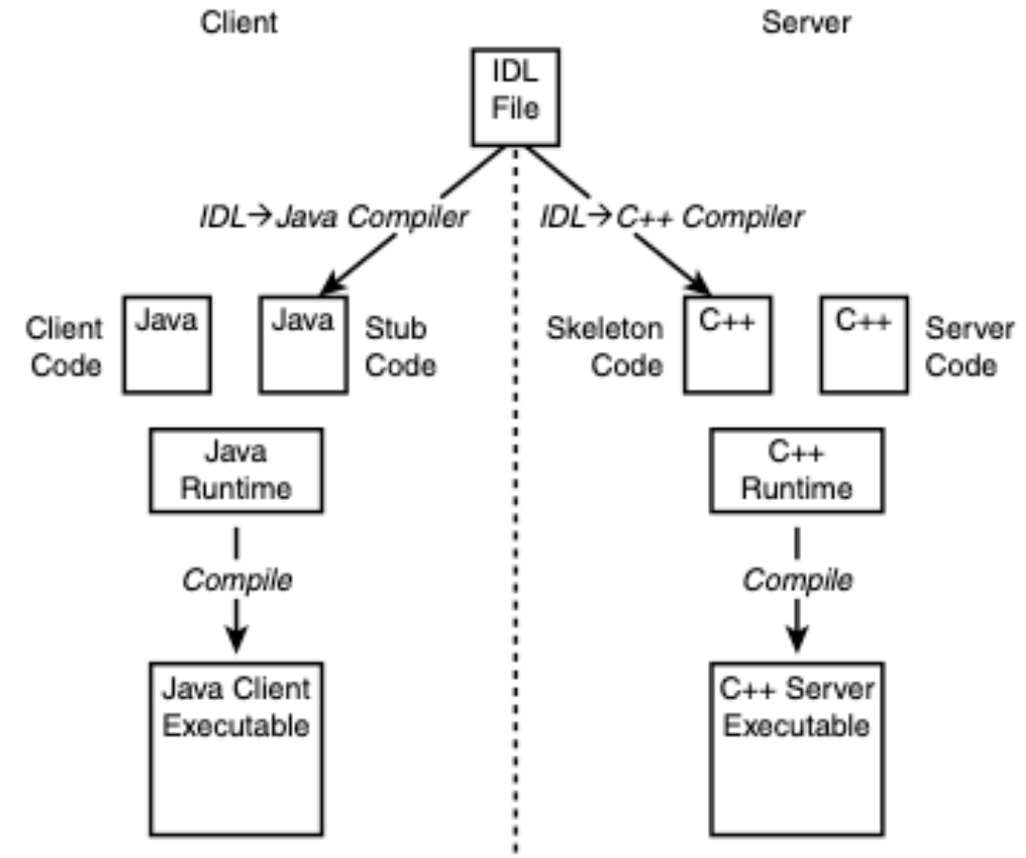
How are data types translated?

How are IDL interface definitions map to closes and how IDL operations map to methods

Stubs and Skeletons

Stubs and Skeletons are code generated by the idl compiler

Comes with the language translations and boiler plate code needed.



IDL Compiler maps the Interface Definition to a specific Language

Object References

*Standard format **IOR** (Interoperable Object Reference)*

Contains:

- *The IP hostname of the host where server runs*
- *The port number that the server is listening to*
- *A unique object identity*

Object References contains all information needed to find and use CORBA Objects.

Object References originates in Server---> Catch 22 situation

Some solutions.

- *Write the stringify reference to file*
- *Use a naming Service*

2020-07-23T11:01:22.417 INFO [Manager] Manager activated with
IOR:000000000000001C49444C3A696A732E73692F6D6163692F4D616E616765723A312E3000000000010000000000000B00001020000000000A31
302E302E322E3135000BB8000000164F52422F4D616E61676572504F412F4D616E6167657200000000000400000003000000160000000000000000C
31302E31302E31302E3130000BB8000000000003000000180000000000000000E3139322E3136382E312E313535000BB8000000000000000800000000
04A4143000000000100000024000000000001000100000002000100F0501000100010109000000020501000100010100

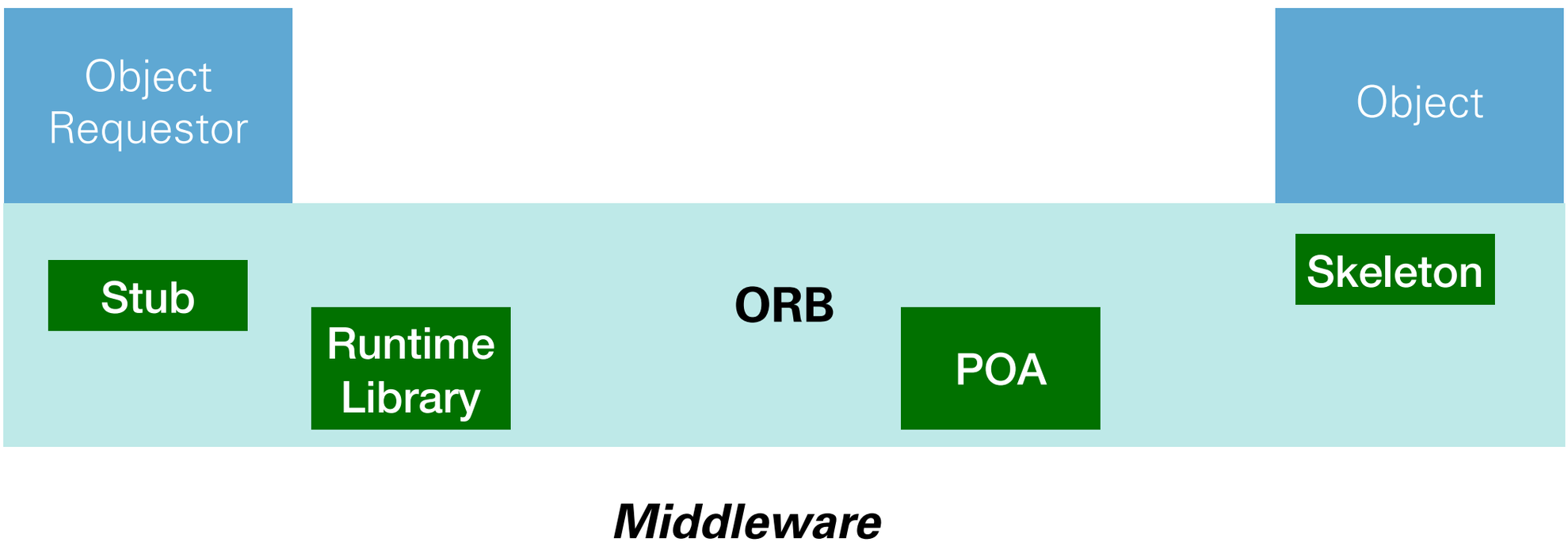
Portable Object Adapter (POA)

- *Provides mechanism to associate a concrete class (lets say a C++ or python class) with the particular IDL interface*

- *Manage lifecycle of CORBA objects, ie how to activate CORBA objects to make them accessible to clients, and how to deactivate them.*

Object Request Broker(ORB)

- *ORB concept is an Abstraction*
- *it is everything needed, the total sum of infrastructure that allow us to make remote invocations*



ORBs and Examples

ACE TAO - C++ ORB

IDL	C++
short	CORBA::Short
long	CORBA::Long
long long	CORBA::LongLong
unsigned short	CORBA::UShort
unsigned long	CORBA::ULong
unsigned long long	CORBA::ULongLong
float	CORBA::Float
double	CORBA::Double
long double	CORBA::LongDouble
char	CORBA::Char
wchar	CORBA::WChar
string	char *
wstring	CORBA::WChar *
boolean	CORBA::Boolean
octet	CORBA::Octet
any	CORBA::Any

```
struct Details {
    double          weight;
    unsigned long   count;
};
```

```
class Details_var;

struct Details {
    CORBA::Double   weight;
    CORBA::ULong    count;
    typedef Details_var _var_type;
    // Member functions here...
};
```

Jacorb - a java ORB

https://www.jacorb.org/releases/3.3/ProgrammingGuide.pdf

Hello.idl

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

HelloServer.java

```
// Copyright and License
import HelloApp.*;
import org.omg.CosNaming.*;
import org.omg.CosNaming.NamingContextPackage.*;
import org.omg.CORBA.*;
import org.omg.PortableServer.*;
import org.omg.PortableServer.POA;

import java.util.Properties;

class HelloImpl extends HelloPOA{
  private ORB orb;

  public void setORB(ORB orb_val){
    orb = orb_val;
  }

  public String sayHello(){
    return "\nHello world !!\n";
  }

  public void shutdown(){
    orb.shutdown(false);
  }
}
```

omniORBpy- a Python ORB

https://www.omniorb-support.com/omnipy3/omniORBpy.pdf

```
module Fortune {
  interface CookieServer {
    string get_cookie(  );
  };
};
```

```
$ omniidl -bpython fortune.idl
```

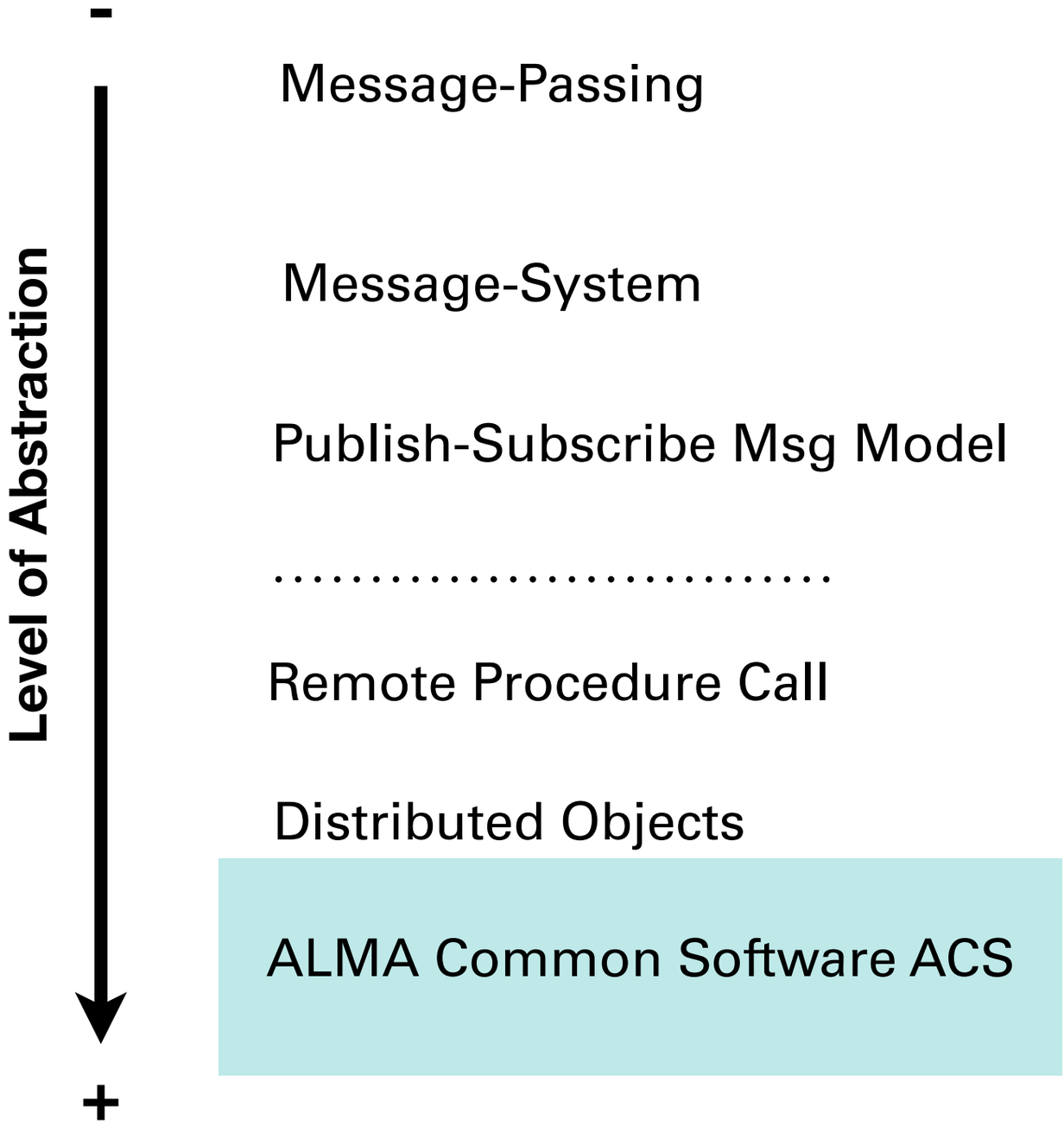
Generates 2 python modules

```
import sys, os
import CORBA, Fortune, Fortune_ _POA

FORTUNE_PATH = "/usr/games/fortune"

class CookieServer_i(Fortune_ _POA.CookieServer):
    def get_cookie(self):
        pipe   = os.popen(FORTUNE_PATH)
        cookie = pipe.read(  )
        if pipe.close(  ):
            # An error occurred with the pipe
            cookie = "Oh dear, couldn't get a fortune\n"
        return cookie

orb = CORBA.ORB_init(sys.argv)
poa = orb.resolve_initial_references("RootPOA")

servant = CookieServer_i(  )
poa.activate_object(servant)

print orb.object_to_string(servant._this(  ))

poa._get_the_POAManager().activate(  )
orb.run(  )
```

```
>>> import CORBA, Fortune
>>> orb = CORBA.ORB_init(  )
>>> o = orb.string_to_object(
...   "corbaloc::host.example.com/fortune")
>>> o = o._narrow(Fortune.CookieServer)
>>> print o.get_cookie(  )
```

OK, I understood CORBA. Why do we need ACS?

Why ACS?



Services on Top of CORBA:

- Distributed error
- Distributed logging
- Distributed Events (Notification Channel)
- Transparent Remote Object Invocation
- Threading Support

Component/Container Paradigm

Thank You!

References

Message Passing, Remote Procedure Calls and Distributed Shared Memory as Communication Paradigms for Distributed Systems, Silcock, Goscinski

Modern Operating Systems, Andrew Tanenbaum, 2014

Pure CORBA, Fintan Bolton, 2001

Lectures from Concurrent and Distributed Systems, Cambridge University. https://www.cl.cam.ac.uk/teaching/1617/ConcDisSys/

An overview of the ALMA Common Software (ACS). P Di Marcantonio et all, 2007

Distributed Systems and CORBA Standard

Pablo Burgos

Credits

- Photo by Shwetha Shankar on Unsplash