Remote Procedure Call

30221 - Sistemas Distribuidos

Rafael Tolosana Calasanz

Dpto. Informática e Ing. de Sistemas

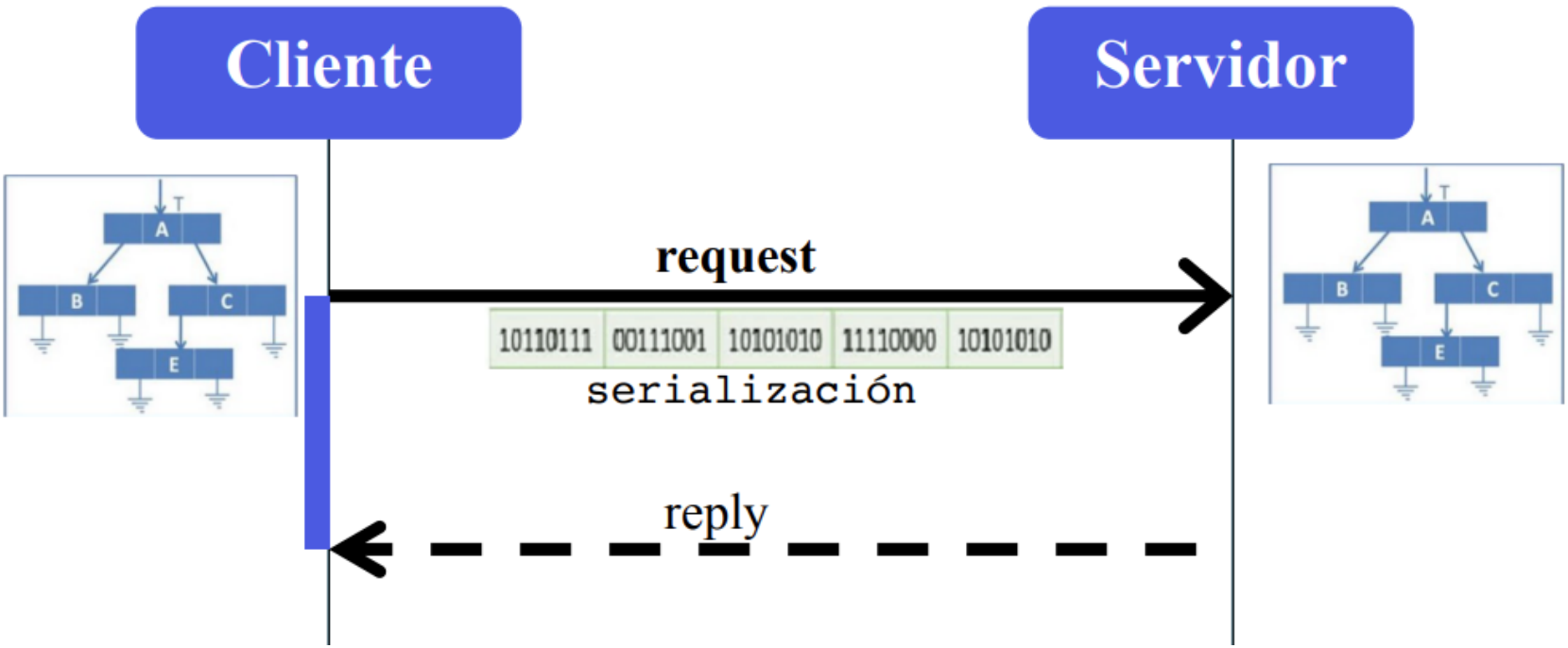
Lectura Recomendada

- Tanenbaum Van Steen, Distributed Systems: Principles and Paradigms, 3e, (c) 2017. **Chapter 4**

Motivación

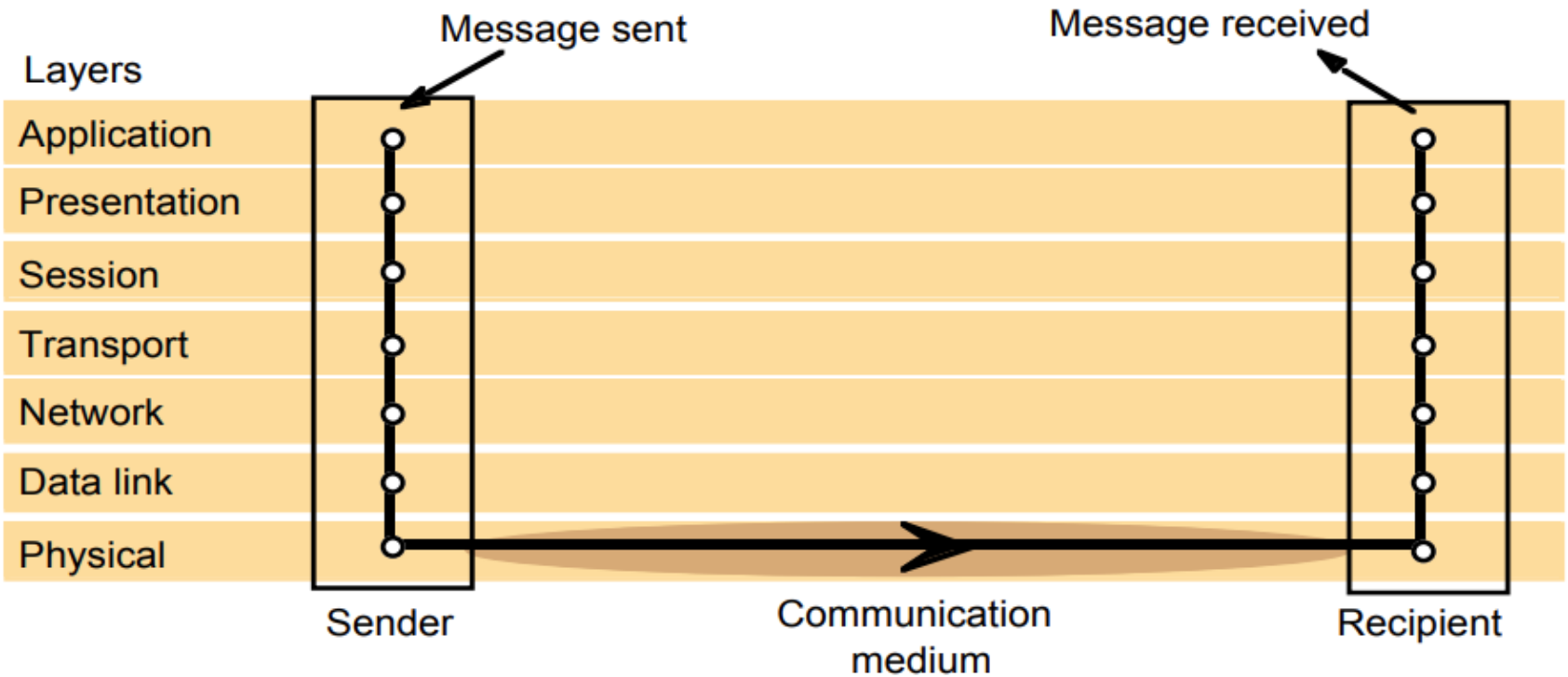
Motivación

El reto de la Serialización



Motivación

La importancia de la comunicación para construir aplicaciones en SSDD

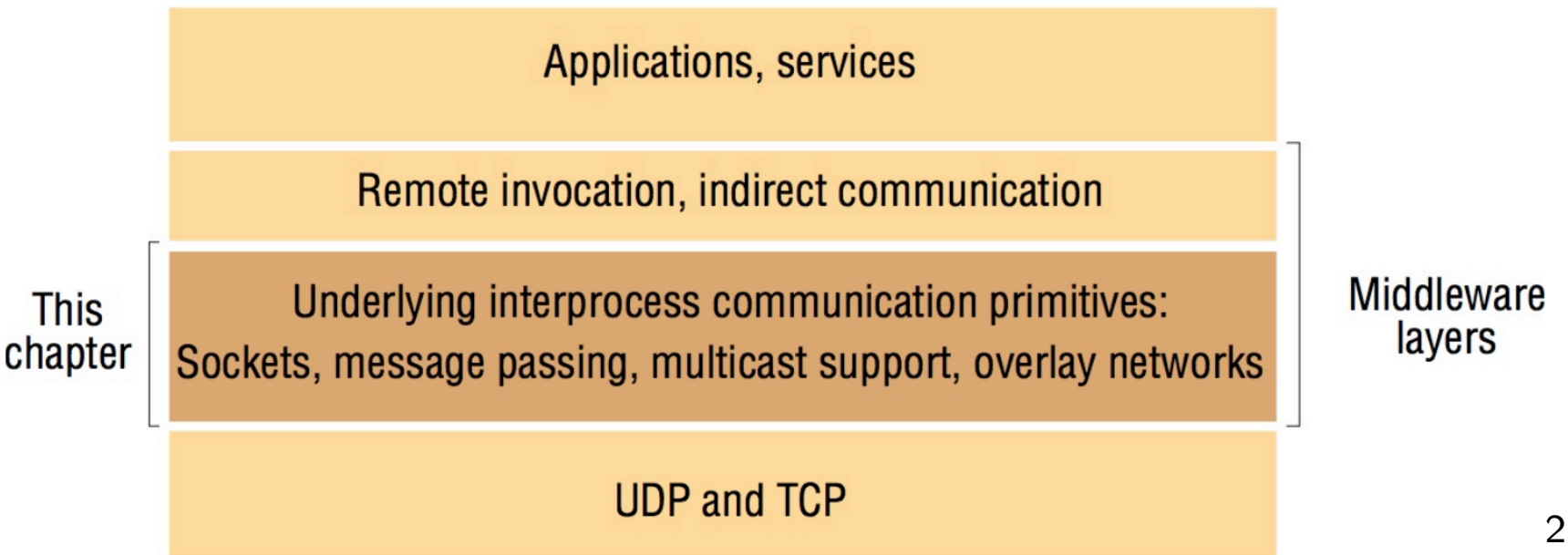


[1]

[1] Instructor's Guide for Coulouris, Dollimore, Kindberg and Blair, Distributed Systems: Concepts and Design Edn. 5© Pearson Education 2012

Motivación

La importancia de la comunicación para construir aplicaciones en SSDD



[2]

[2] Instructor's Guide for Coulouris, Dollimore, Kindberg and Blair, Distributed Systems: Concepts and Design Edn. 5© Pearson Education 2012

Remote Procedure Call

Remote Procedure Call

En 1976, James E White

- publicó los detalles de lo que más tarde daría lugar a RPC.
- describió su Procedure Call Model *"as a way of making the networked environment seem completely familiar to application developers, rather than exposing the network directly to them and thus presenting them with a development model so different than they be scared away from writing distributed programs."*

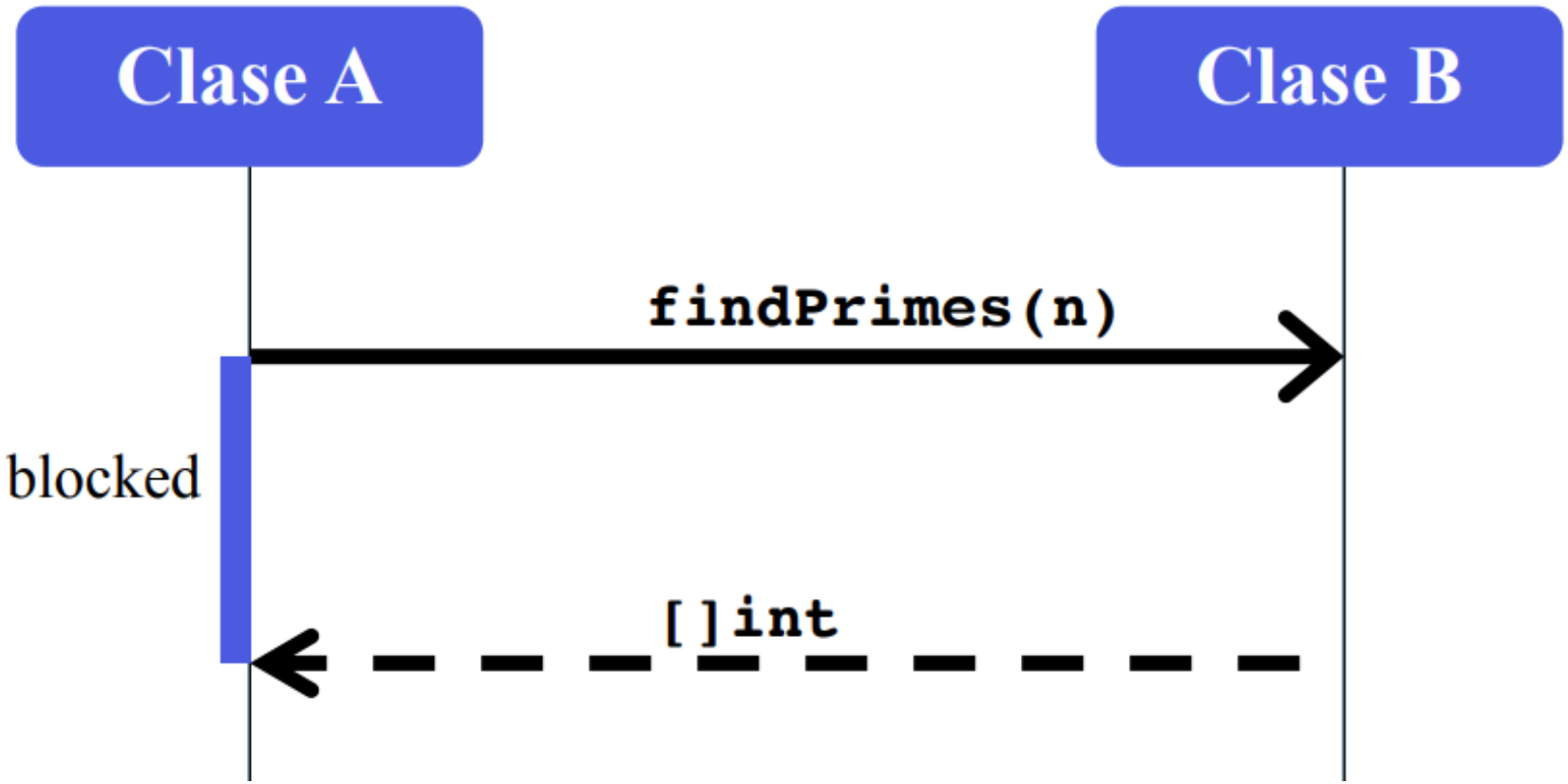
Remote Procedure Call

Consiste en invocar un procedimiento / método remoto como si fuera local

- Cuando un proceso A invoca a un proceso B, A se queda bloqueado
- Los parámetros se envían de A a B
- La ejecución se realiza en B
- El resultado va de B a A
- **Para A ni para B hay noción de paso de mensajes**

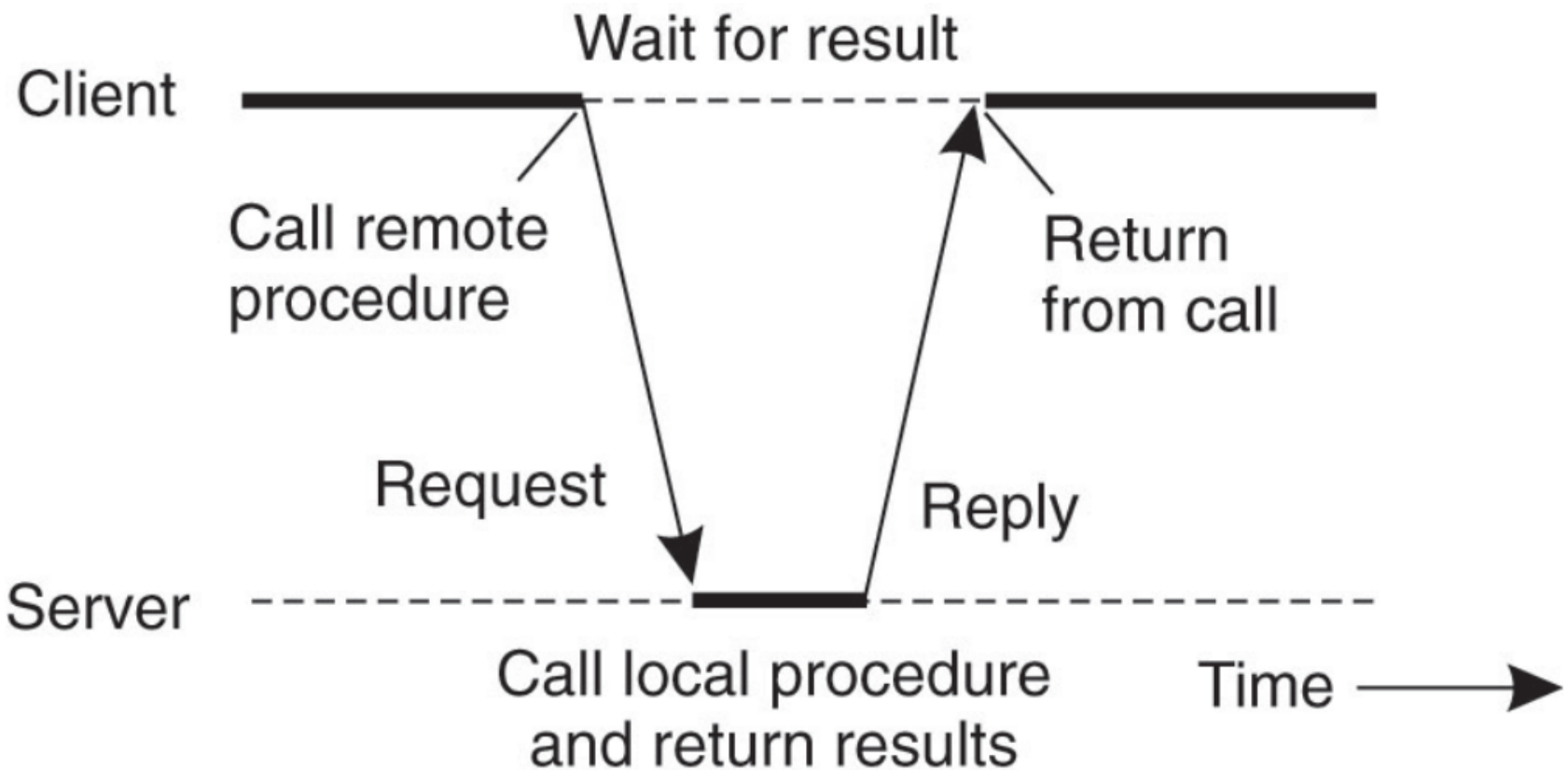
Remote Procedure Call

Interacción / Invocación entre A y B



Remote Procedure Call

Interacción / Invocación



[3]

[3] Instructor's Guide for Steen, Tanenbaum, Distributed Systems 2nd Edition

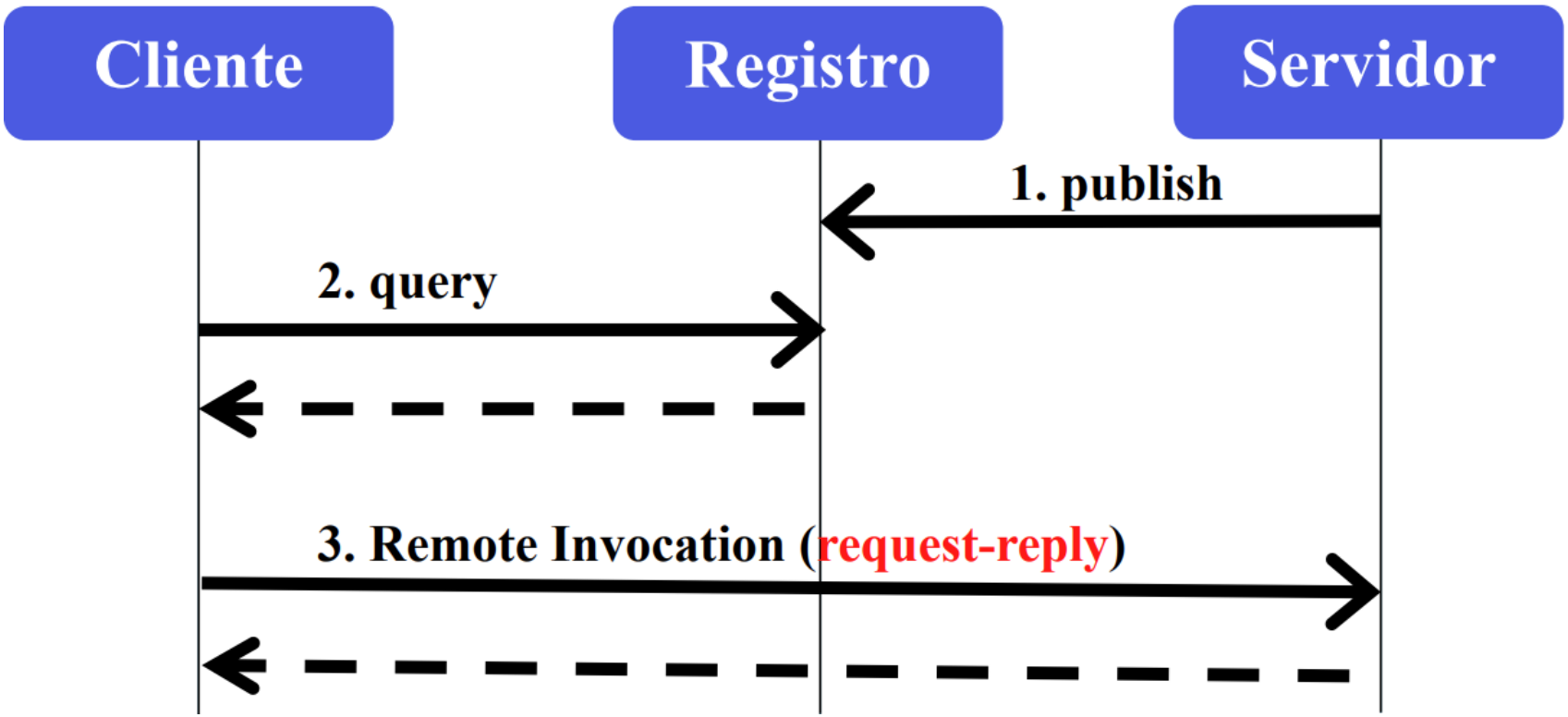
Remote Procedure Call

Evolución de RPC

- En 1984, Bruce J. Nelson y Andre D. Birrell (Xerox) presentaron RPC para implementar SSDD
- De RPC se derivarían en las décadas siguientes, diferentes framework:
  - Common Object Request Broker Arquitecture, Java Remote Method Invocation, Sun RPC, RPC-based Web Services, gRPC etc.

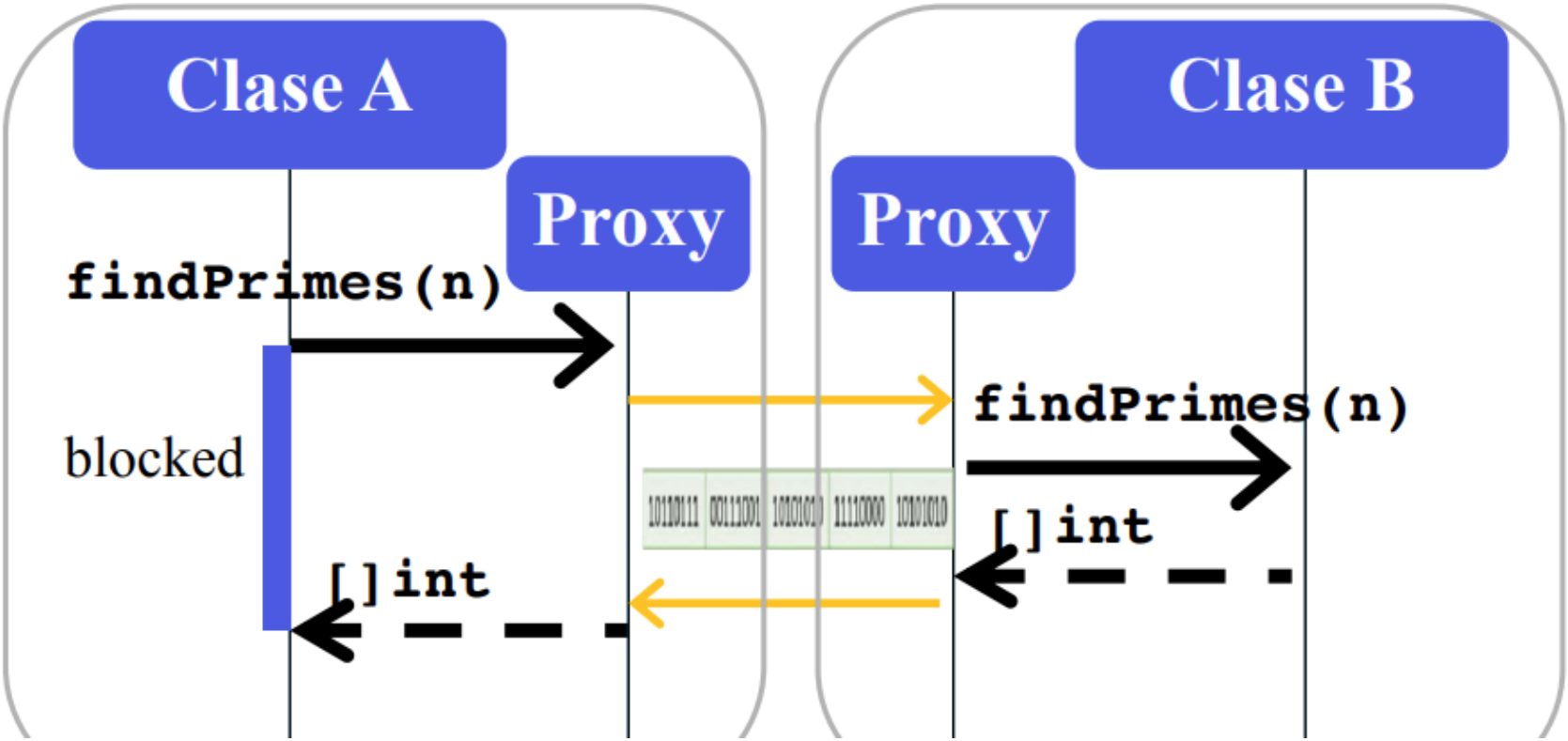
Remote Procedure Call

Interacción en RPC



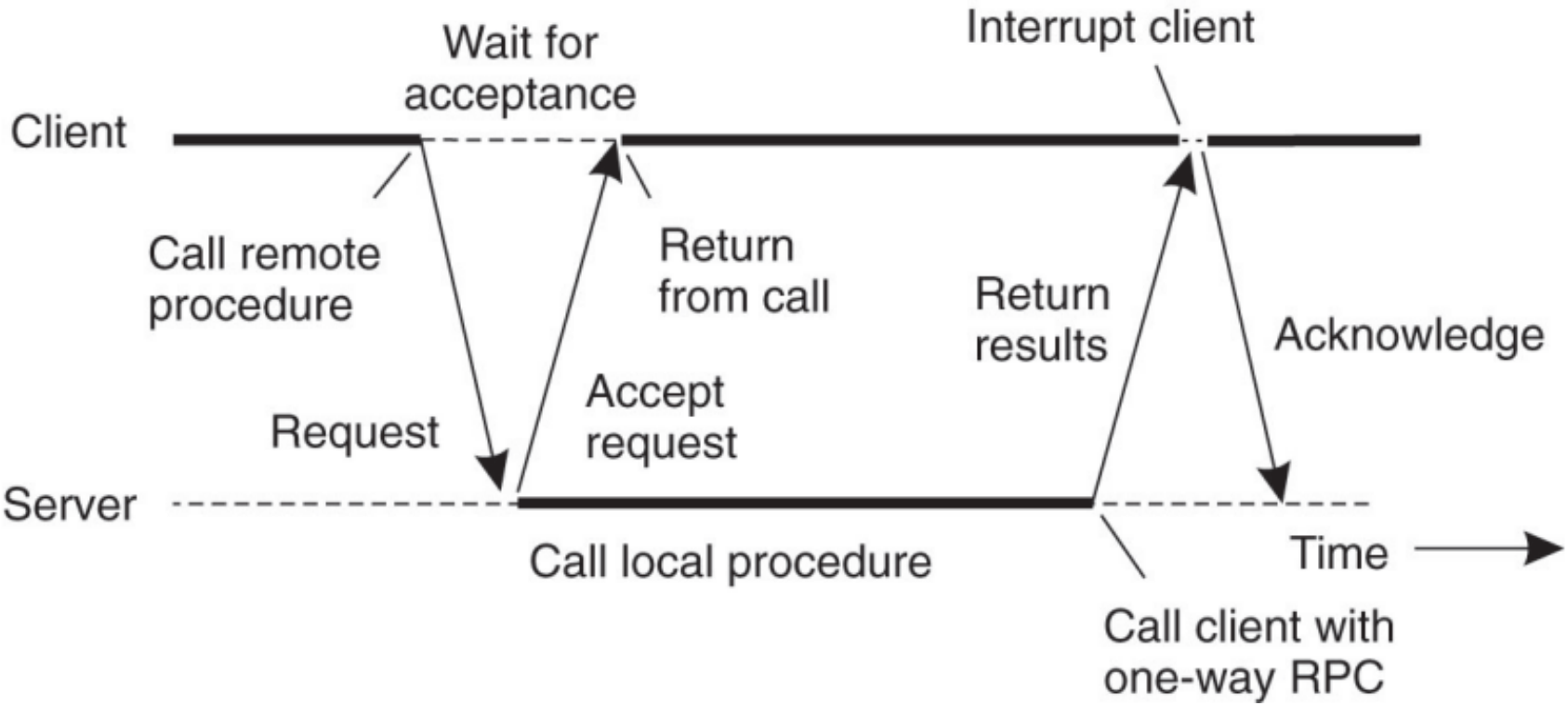
Remote Procedure Call

Fundamentos Arquitecturales de RPC



Remote Procedure Call

Modelo RPC Asíncrono



Retos RPC

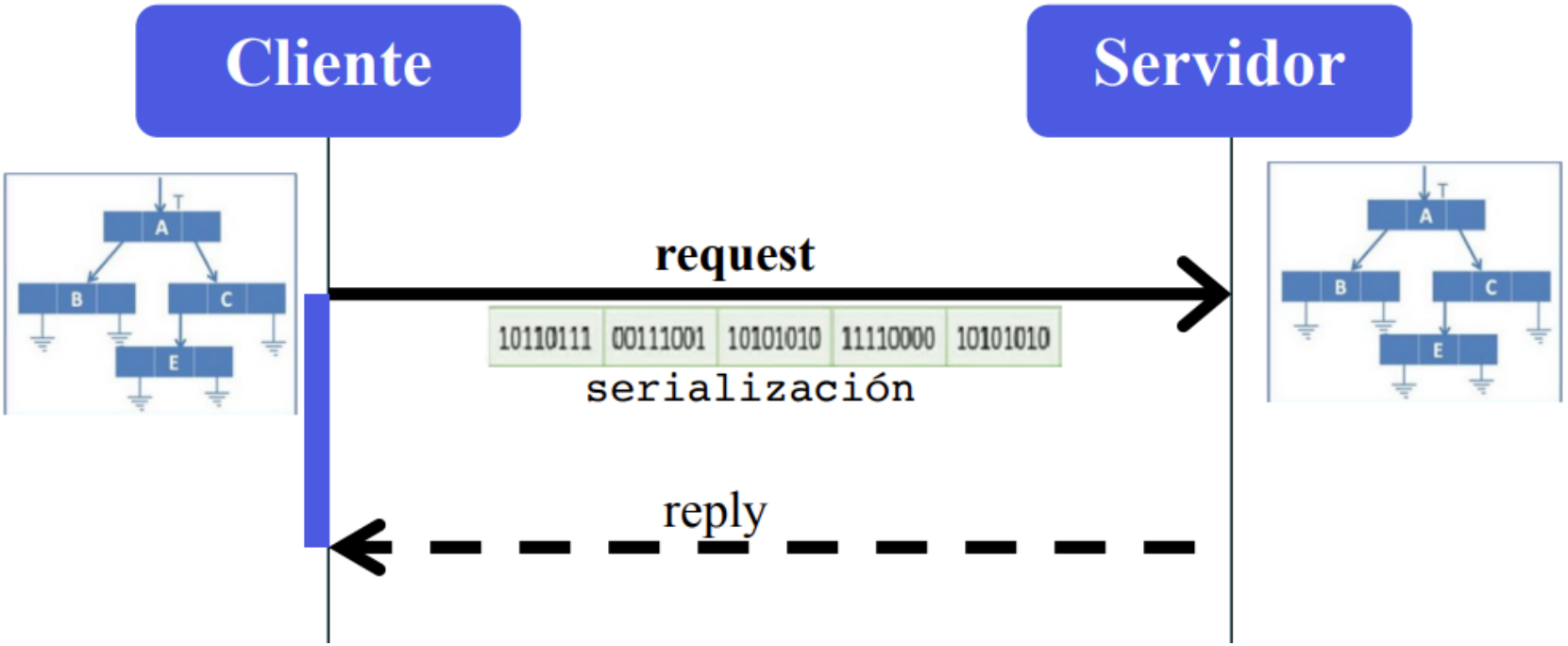
Datos intercambiados emisor / receptor

- Acuerdo emisor / receptor
- Serialización de los datos

Interacción emisor / receptor

- Intentar que una interacción sea parecida a una invocación
- Interacción Síncrona vs Interacción Asíncrona

Retos RPC


Cliente
Servidor
A
B
C
E
request
10110111
00111001
10101010
11110000
10101010
serialización
reply

Retos RPC

Datos intercambiados emisor / receptor

- Acuerdo mediante una interfaz común
  - **Interface Definition Language (IDL)**
  - Solución para la heterogeneidad

Reto 1: Interface Definition Language

Interface Definition Language (IDL)

- Un IDL es un lenguaje de especificación de interfaces
- Se utilizan para describir la API de un componente software de manera independiente de cualquier lenguaje de programación
- Se utilizan habitualmente para tender puentes entre lenguajes de programación diferentes en la programación RPC
- Ejemplos:
  - Web Service Definition Language (WSDL)
  - eXternal Data Representation (XDR) [RPC]

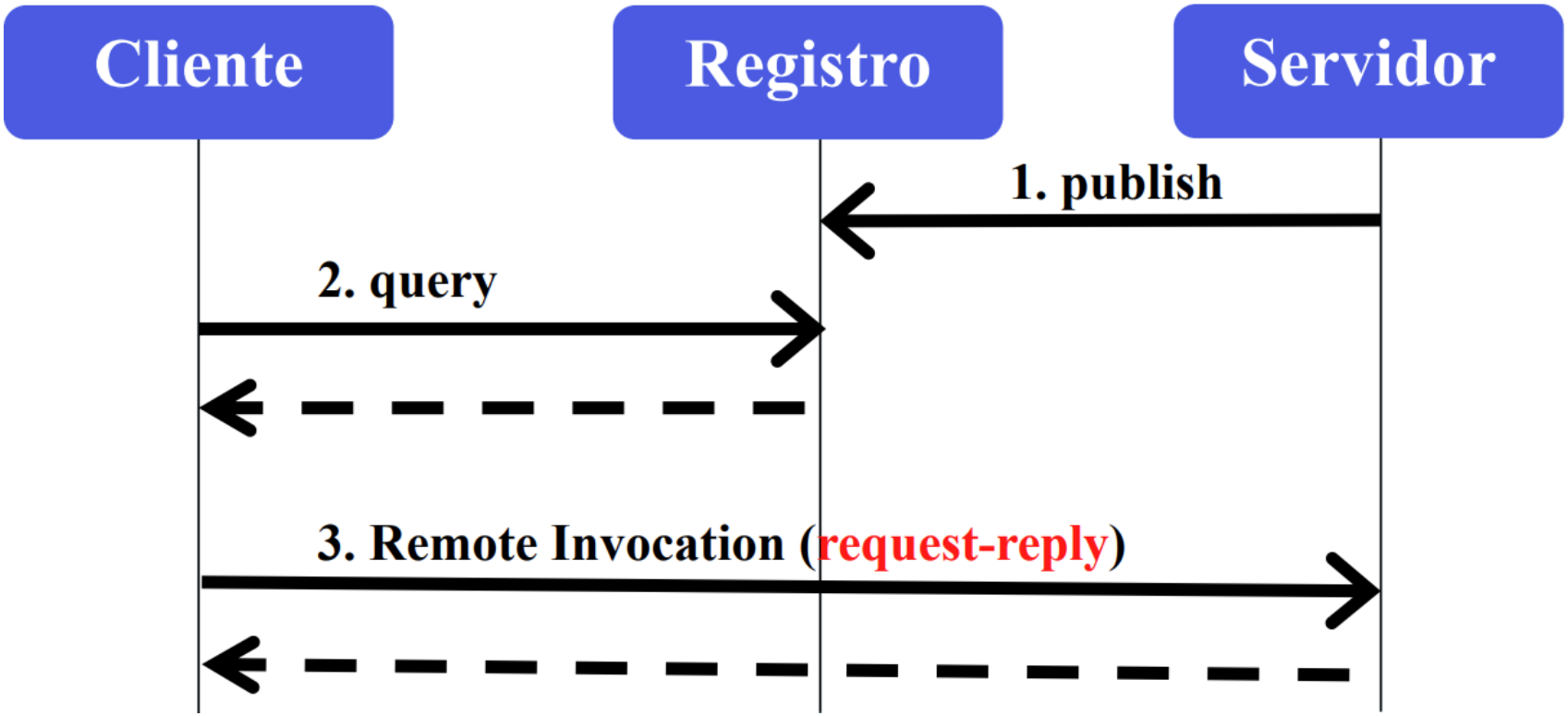
Reto 1: Interface Definition Language

Ejemplo de Interfaz en C

```
1  /*
2   * date.x  Specification of the remote date and time server
3   */
4
5  /*
6   * Define two procedures
7   *       bin_date_1() returns the binary date and time (no arguments)
8   *       str_date_1() takes a binary time and returns a string
9   *
10  */
11
12 program DATE_PROG {
13     version DATE_VERS {
14         long BIN_DATE(void) = 1;     /* procedure number = 1 */
15         string STR_DATE(long) = 2;   /* procedure number = 2 */
16     } = 1;                           /* version number = 1 */
17 } = 0x31234567;                      /* program number = 0x31234567 */
```

Reto 1: Interface Definition Language

La Interfaz permite aclarar aspectos morfo-sintácticos



¿Y la semántica?

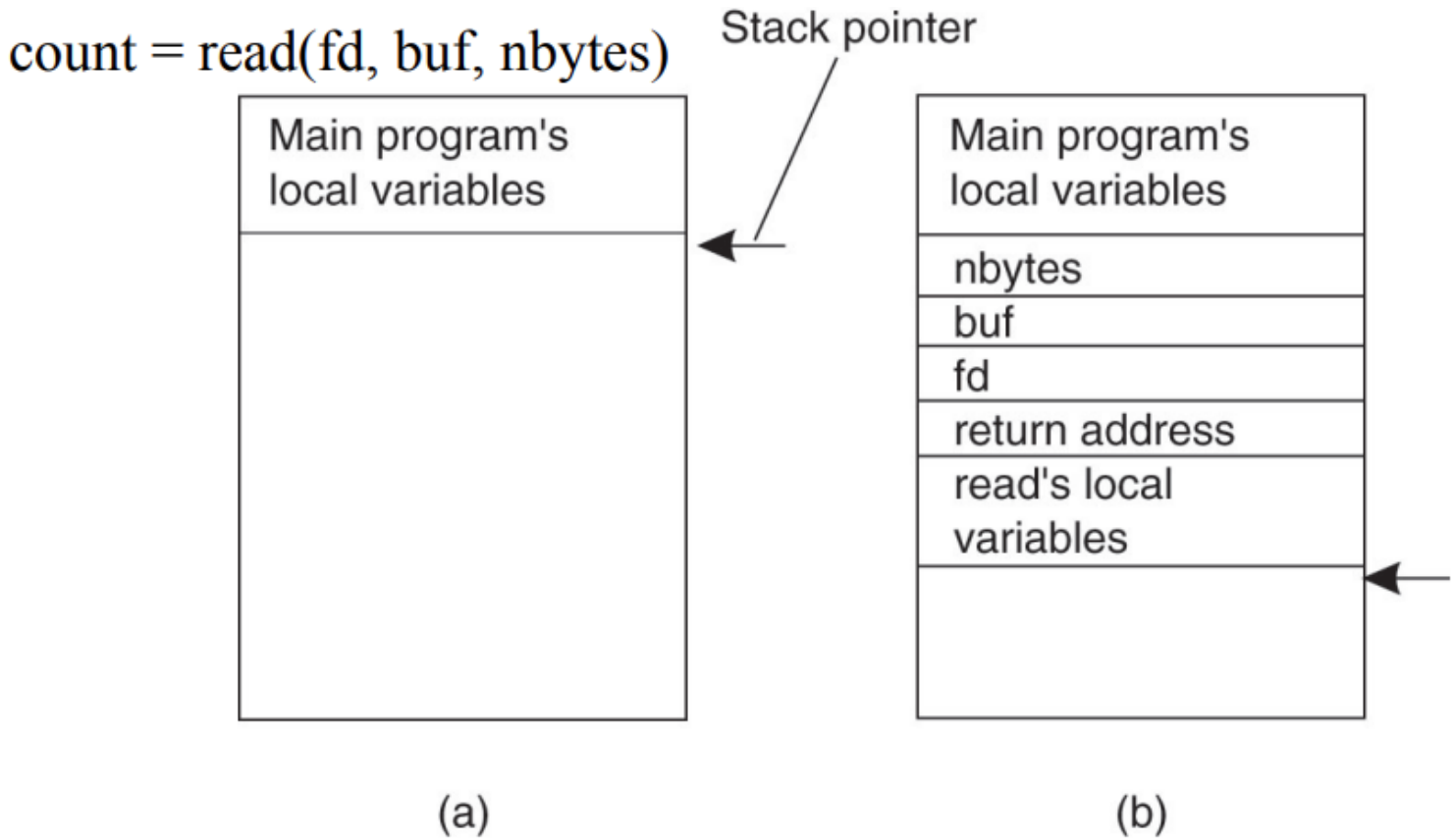
Reto 1: Serialización de Datos

Estrategias para Serializar Datos

- Acuerdo explícito entre emisor y receptor
  - Por ejemplo, de forma manual
- Las estructuras de datos incorporan metadatos sobre cómo se han codificado
  - paquete gob de Go
- Datos en formato texto
  - XML, JSON

Reto 2: Interacción vs invocación

Paso de Parámetros Convencional



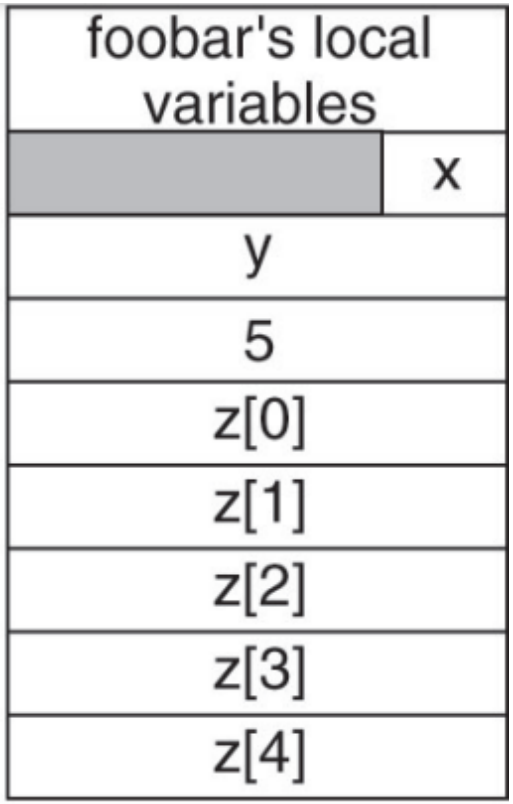
(a) Parameter passing in a local procedure call: the stack before the call to read. (b) The stack while the called procedure is active.

Reto 2: Interacción vs invocación

Paso de Parámetros Convencional

```
foobar( char x; float y; int z[5] )
{
   ....
}
```

(a)



(b)

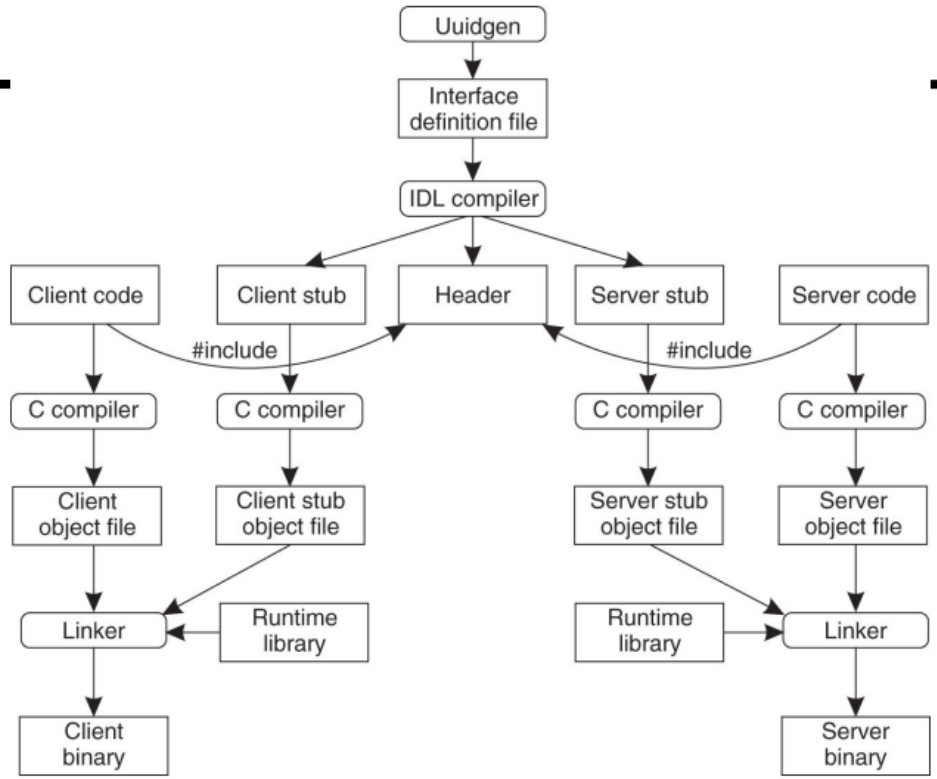
Reto 2: Interacción vs invocación

Paso de Parámetros

- Por valor
  - Envío una copia
- Por referencia
  - ¿Qué hago con una referencia?

Compilación y Ejecución de RPC

Proceso de Compilación en DCE RPC



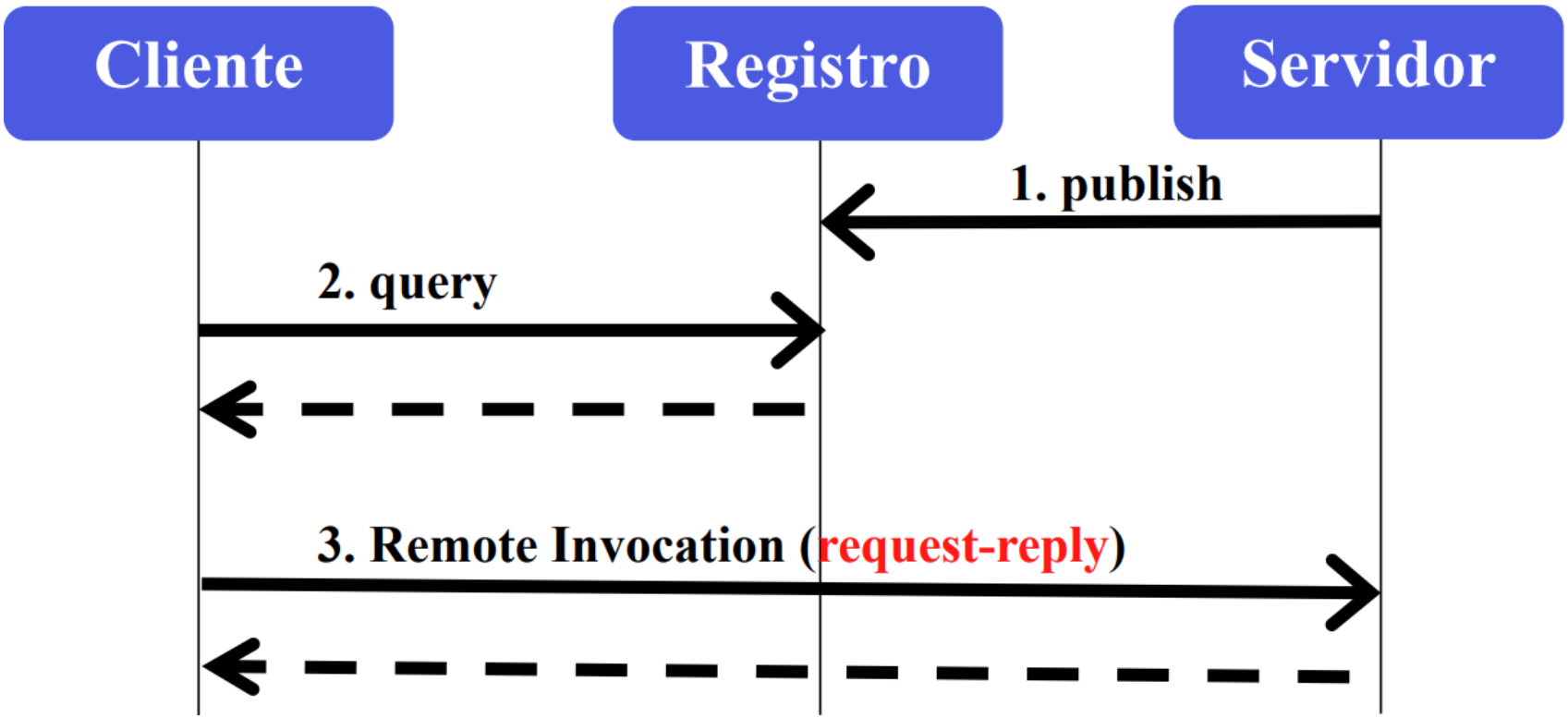
The steps in writing a client and a server in DCE RPC.

[4]

[4] Instructor's Guide for Coulouris, Dollimore, Kindberg and Blair, Distributed Systems: Concepts and Design Edn. 5© Pearson Education 2012

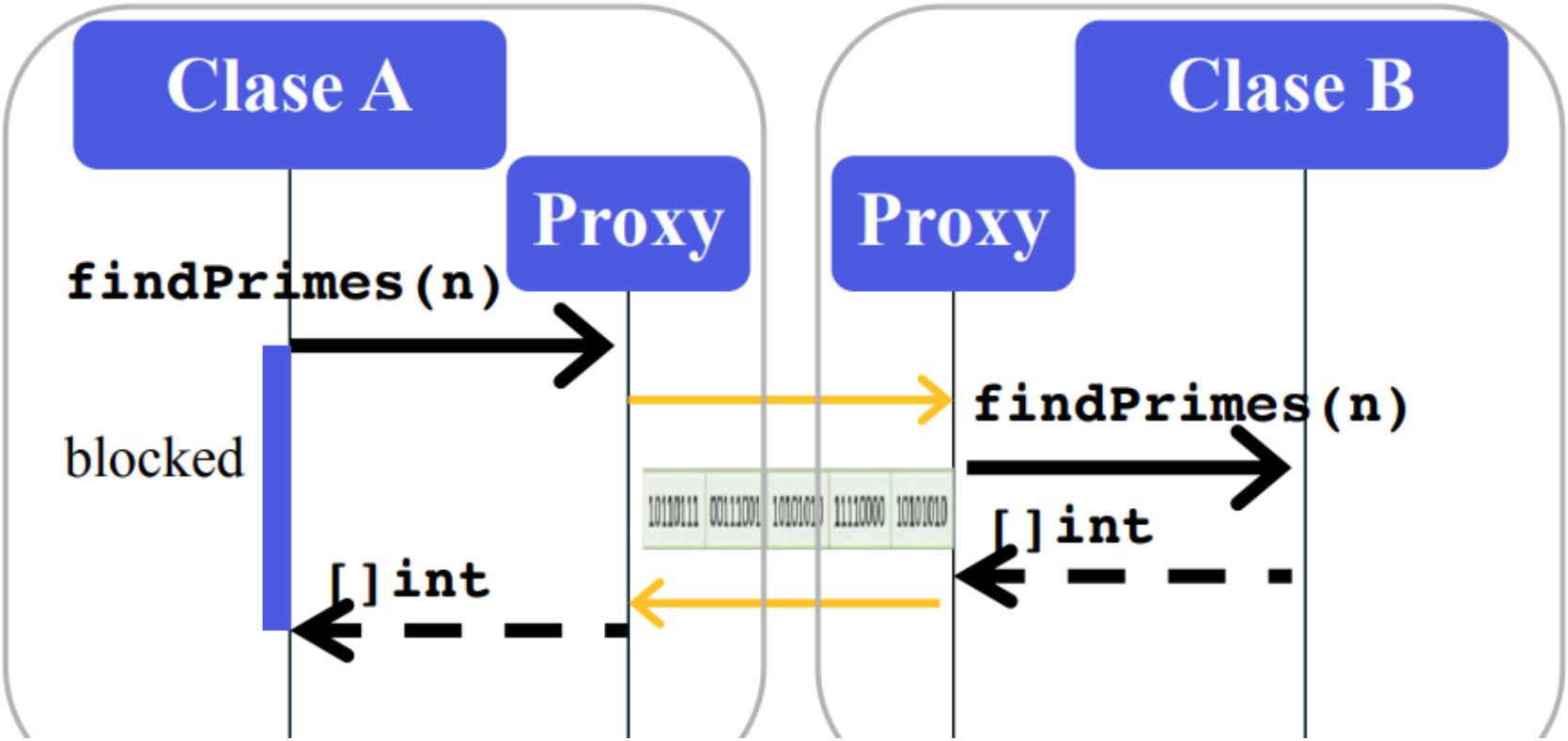
Compilación y Ejecución de RPC

Proceso de Ejecución en RPC



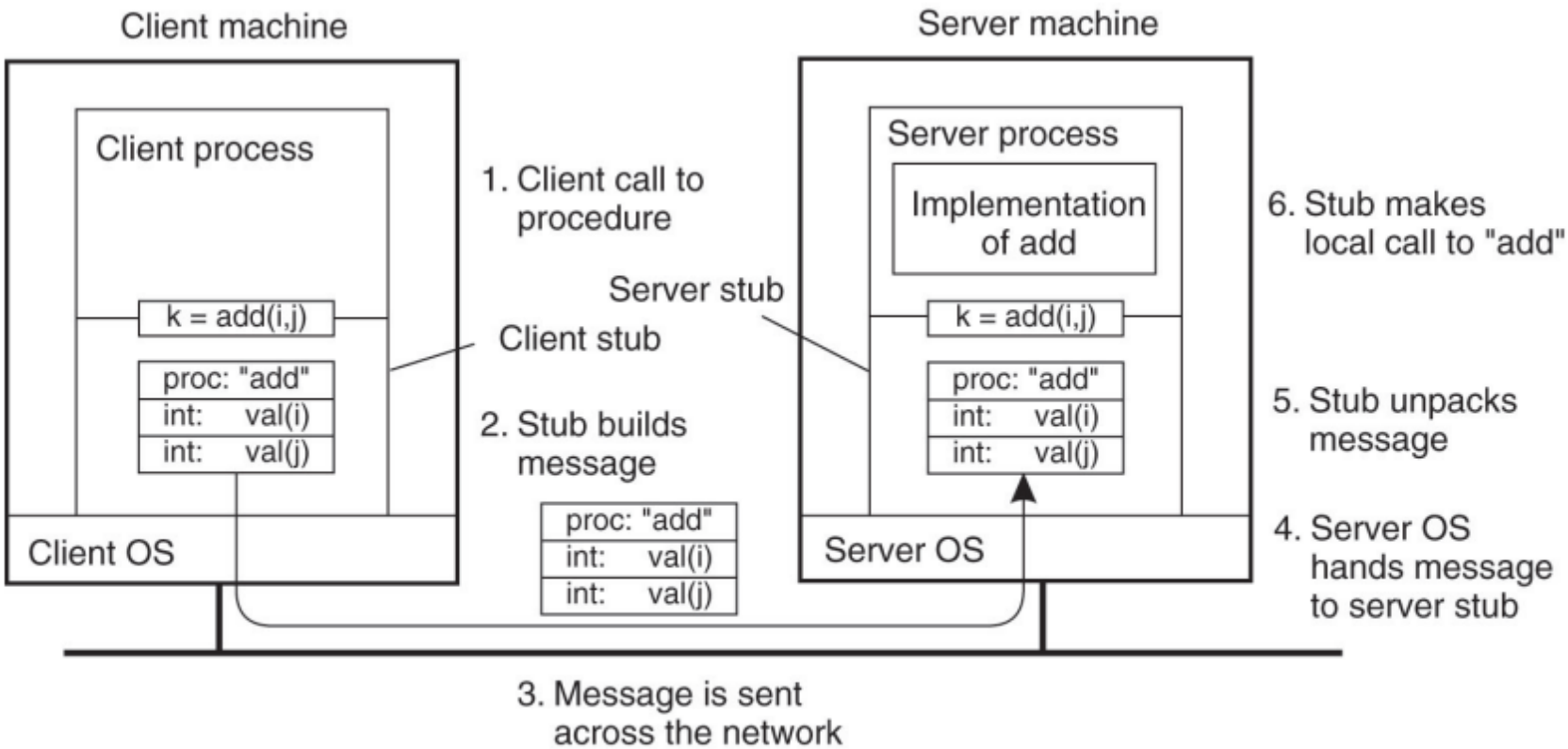
Compilación y Ejecución de RPC

Proceso de Ejecución en RPC: Fase 3 invocación



Compilación y Ejecución de RPC

Proceso de Ejecución en RPC: Fase 3 invocación



The steps involved in a doing a remote computation through RPC.

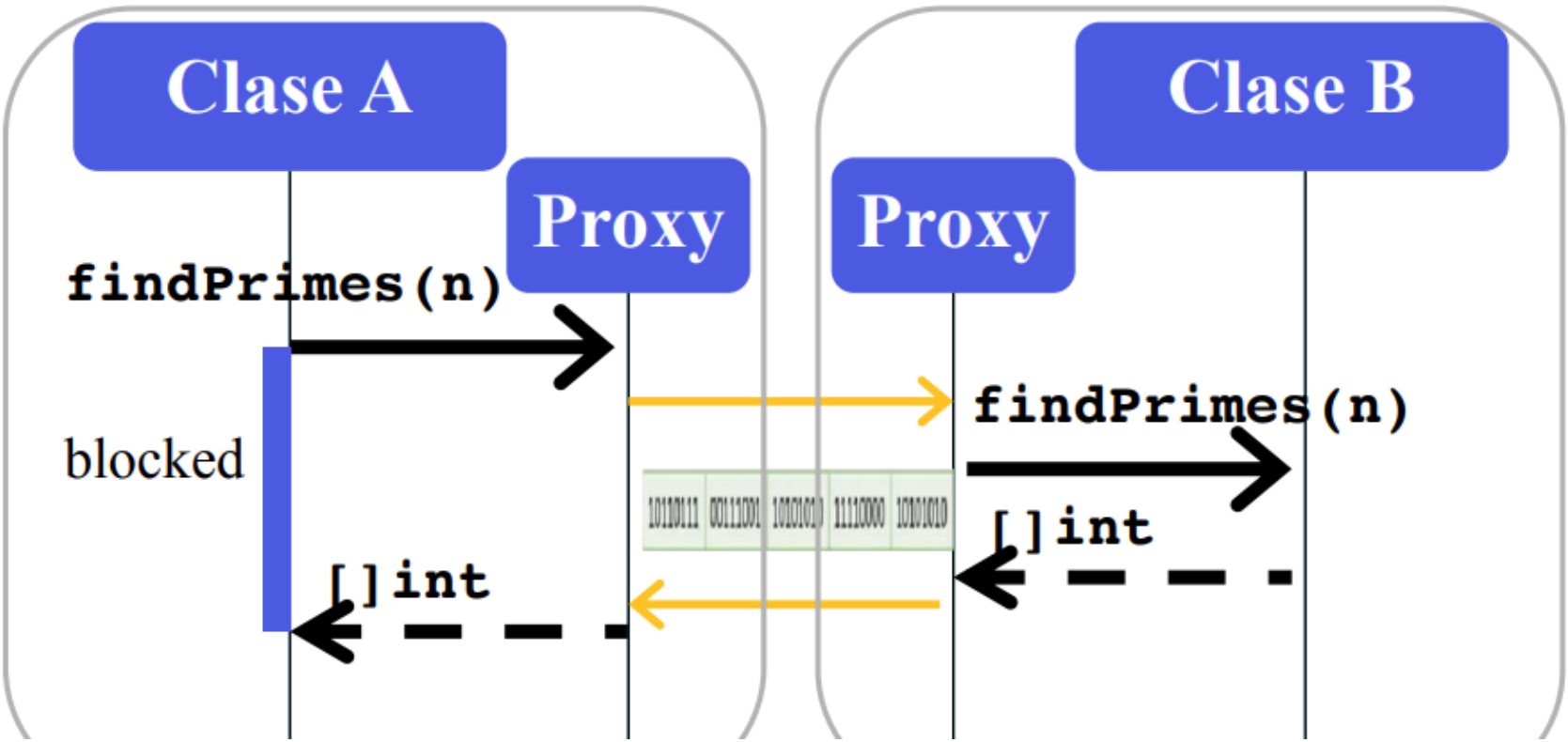
[5]

[5] Instructor's Guide for Coulouris, Dollimore, Kindberg and Blair, Distributed Systems: Concepts and Design Edn. 5© Pearson Education 2012

Ejemplos de RPC

Ejemplos de RPC

RPC en C



RPC en Java: RMI

Remote Method Invocation

- Equivalencia **OO** de RPC
- Está diseñado (casi) exclusivamente para procesos Java
- El protocolo de interacción ya no es un Request-Reply
- Dynamic Class Loading
  - Client and server sides
  - **Security**
- Java RMI tutorial [6]

[6] https://docs.oracle.com/javase/tutorial/rmi/index.html

RPC en Java: RMI

```
package client;

import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
import java.math.BigDecimal;
import compute.Compute;

public class ComputePi {
    public static void main(String args[]) {
        if (System.getSecurityManager() == null) {
            System.setSecurityManager(new SecurityManager());
        }
        try {
            String name = "Compute";
            Registry registry = LocateRegistry.getRegistry(args[0]);
            Compute comp = (Compute) registry.lookup(name);
            Pi task = new Pi(Integer.parseInt(args[1]));
            BigDecimal pi = comp.executeTask(task);
            System.out.println(pi);
        } catch (Exception e) {
            System.err.println("ComputePi exception:");
            e.printStackTrace();
        }
    }
}
```

RPC en Java: RMI

```java
package engine;

import java.rmi.RemoteException;
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
import java.rmi.server.UnicastRemoteObject;
import compute.Compute;
import compute.Task;

public class ComputeEngine implements Compute {

    public ComputeEngine() {
        super();
    }

    public <T> T executeTask(Task<T> t) {
        return t.execute();
    }

    public static void main(String[] args) {
        if (System.getSecurityManager() == null) {
            System.setSecurityManager(new SecurityManager());
        }
        try {
            String name = "Compute";
            Compute engine = new ComputeEngine();
            Compute stub =
                (Compute) UnicastRemoteObject.exportObject(engine, 0);
            Registry registry = LocateRegistry.getRegistry();
            registry.rebind(name, stub);
            System.out.println("ComputeEngine bound");
        } catch (Exception e) {
            System.err.println("ComputeEngine exception:");
            e.printStackTrace();
        }
    }
}
```

RPC en Go

RPC en Go

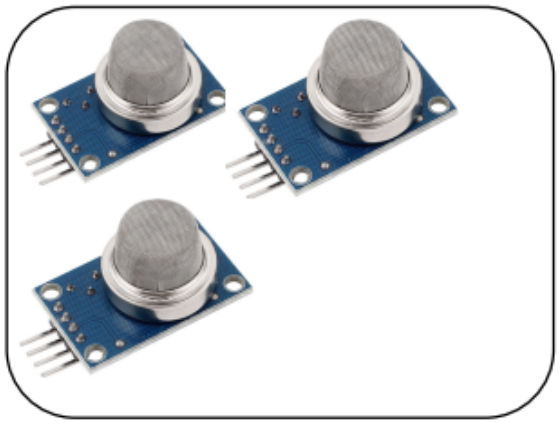
- net / RPC [7]
  - gob + HTTP
- net / RPC JSON
  - JSON + HTTP
- gRPC
  - Protocol buffers de Google

[7] https://pkg.go.dev/net/rpc

Limitaciones de RPC

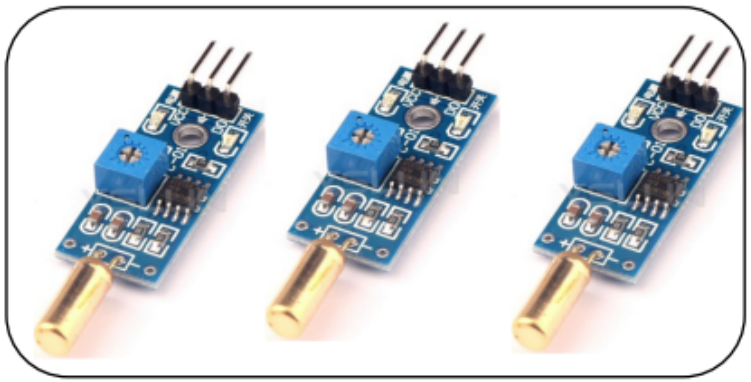
Limitaciones de RPC

No todas las interacciones son request-reply



Sensores operando bajo CoAP (request-reply*)

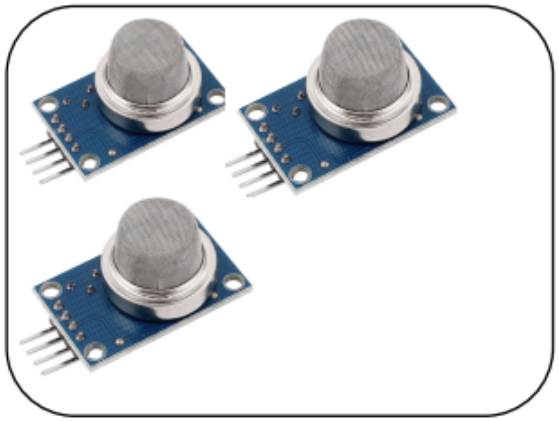
* otros protocolos pueden usarse



Sensores operando bajo MQTT (publish-subscribe)

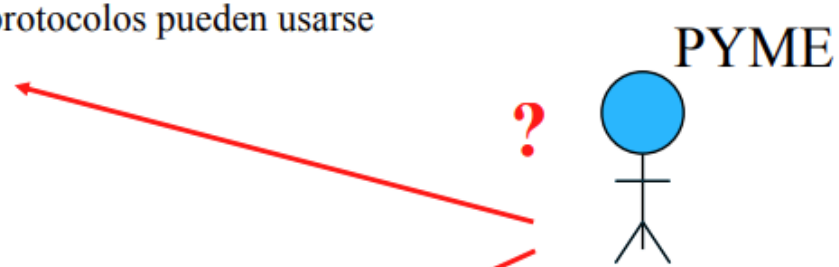
59 Sistemas Distribuidos

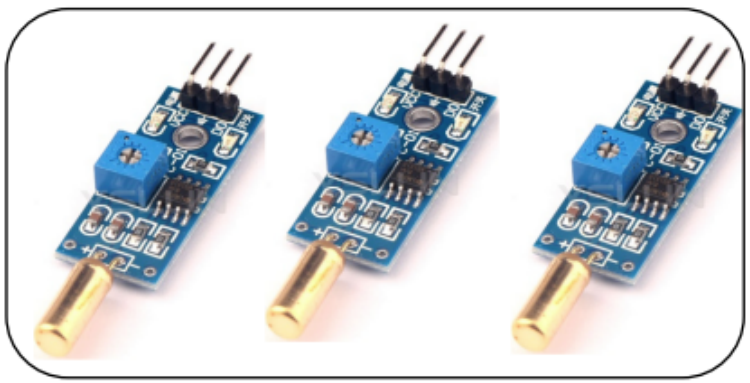
Limitaciones de RPC



Sensores operando bajo CoAP
(request-reply*)

* otros protocolos pueden usarse

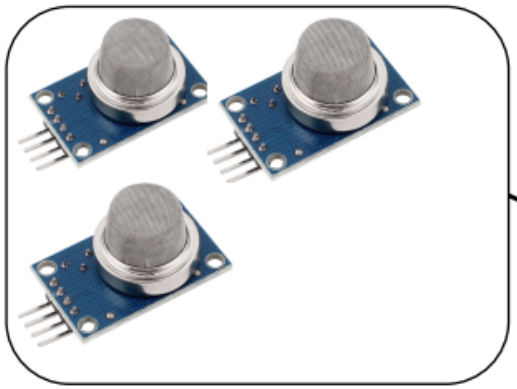




Sensores operando bajo MQTT
(publish-subscribe)

Limitaciones de RPC

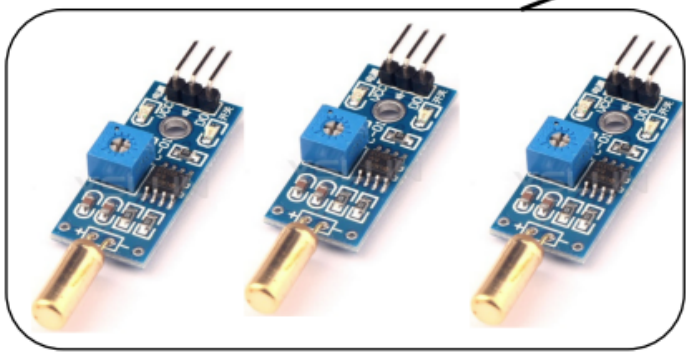
Message Oriented Middleware (MOM)



Transformación de los mensajes
(y el protocolo de interacción)

Message Broker

PYME

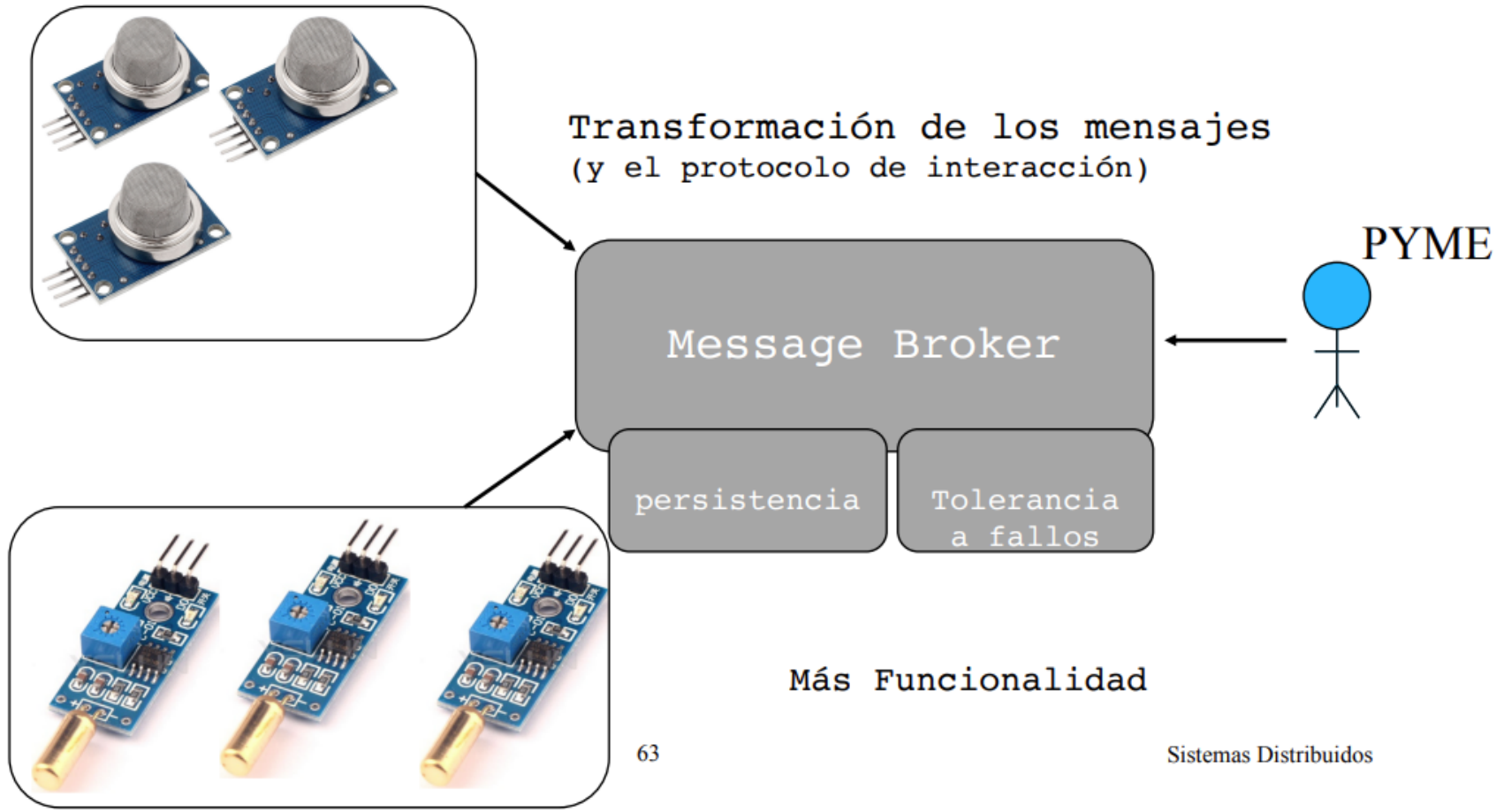


-Desacoplamiento espacial
(mejora interoperabilidad)
-Desacoplamiento temporal
(asincronía)

61 Sistemas Distribuidos

Limitaciones de RPC

Apache Kafka [8]



[8] https://kafka.apache.org/

Resumen

Resumen

Remote Procedure Call

- Comenzó siendo una tecnología
- Es una de las métodos de comunicación más habituales en SSDD
- Trata de simular una comunicación remota como si fuera una invocación a procedimiento local
- La interfaz es la forma en que emisor y receptor acuerdan y especifican la interacción
- Si bien suele ser compleja de utilizar, el RPC de Go es sencillo y práctico

Remote Procedure Call

30221 - Sistemas Distribuidos

Rafael Tolosana Calasanz

Dpto. Informática e Ing. de Sistemas