Arquitecturas de Sistemas Distribuidos

30221 - Sistemas Distribuidos

Rafael Tolosana Calasanz

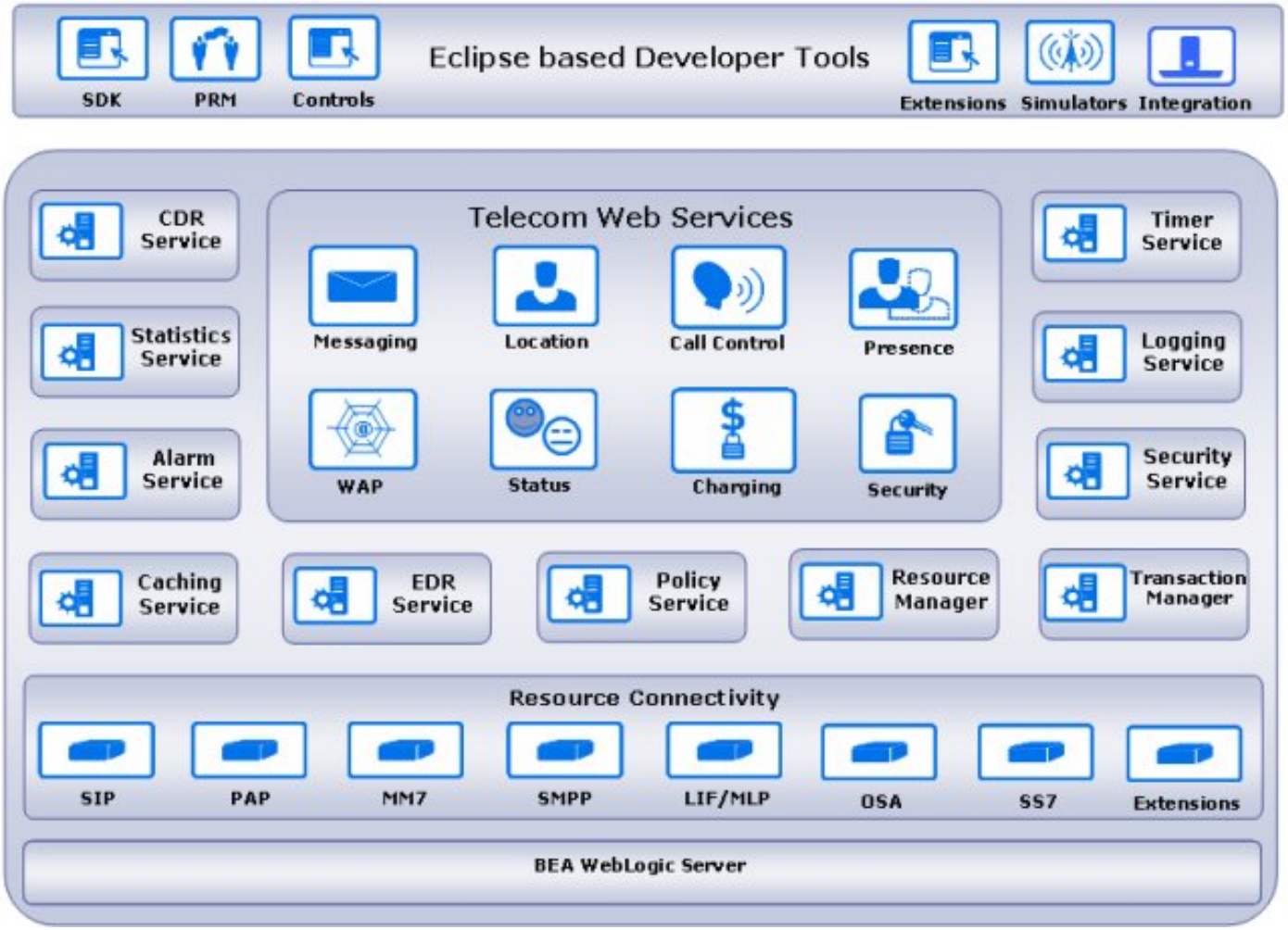
Dpto. Informática e Ing. de Sistemas

Lectura Recomendada: capítulo introductorio:

- G. Colouris, J. Dollimore, T. Kindberg and G. Blair. Distributed systems: Concepts and Design. 5th Edition. Addison-Wesley. May, 2011. ISBN: 978-0132143011.

Arquitectura Software

Arquitectura Software



[1]

[1] Oracle's WebLogic Network Gatekeeper's software architecture

Arquitectura Software

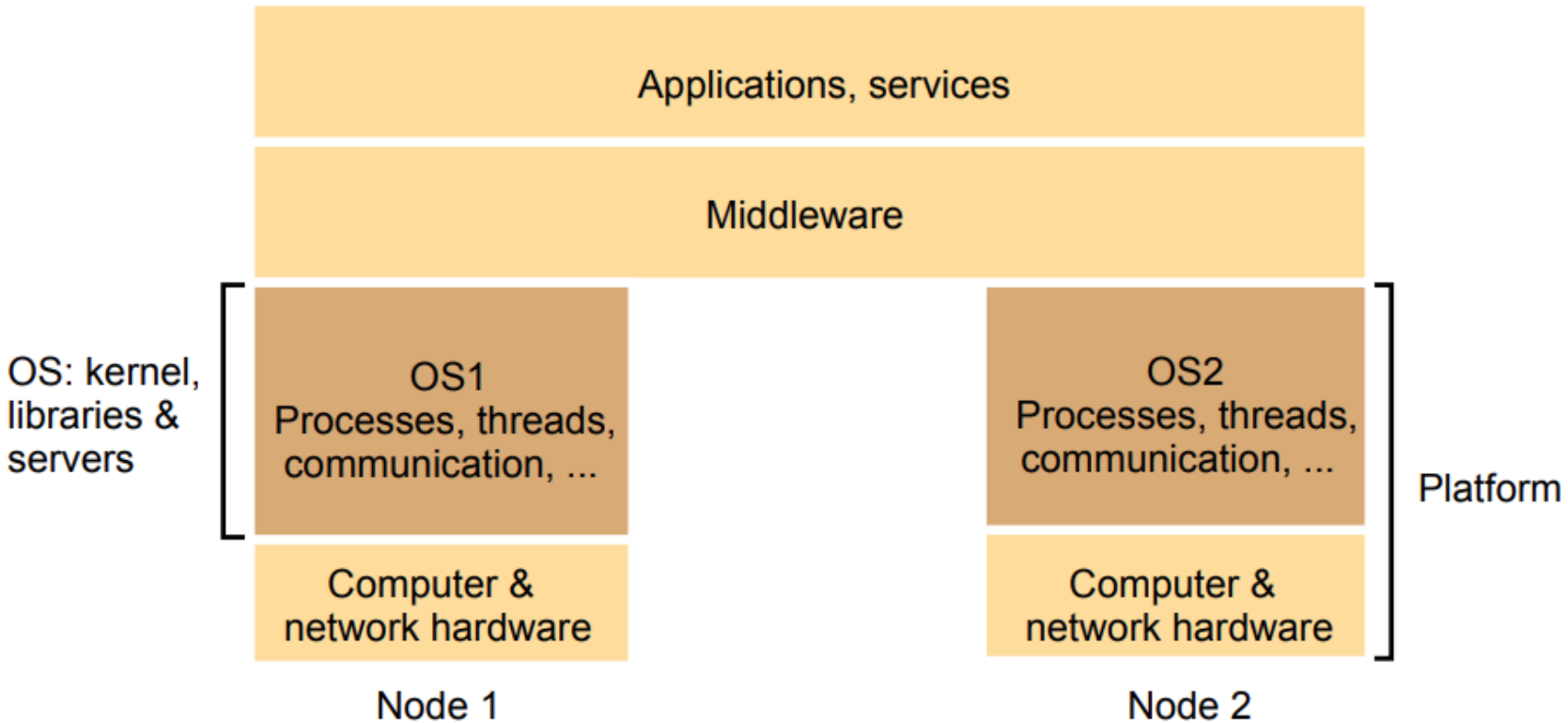
Definición: [Colouris]

- La arquitectura software de un sistema es su estructura en términos de los componentes que la *conforman* y sus *relaciones* entre sí.
- Una arquitectura software es una *abstracción* de un sistema computacional, una abstracción sobre el *código fuente*.
  - Nos muestra aspectos del sistema que nos ayudan a comprenderlo
  - Las arquitecturas nos sirven para trabajar en equipo, comunicación y para mejorar el desarrollo del sistema

Arquitectura Software

- **Vista estática**
  - Describe qué componentes tiene la arquitectura
- **Vista funcional**
  - Describe funcionalmente qué hace cada componente
- **Vista dinámica**
  - Describe cuál es el comportamiento del sistema en ejecución
- **Vista de despliegue**
  - Describe dónde y cómo se realiza la instalación del sistema
  - Es el resultado de la fusión entre un modelo (visión) estático y un modelo físico (hardware + red)

Arquitectura Software



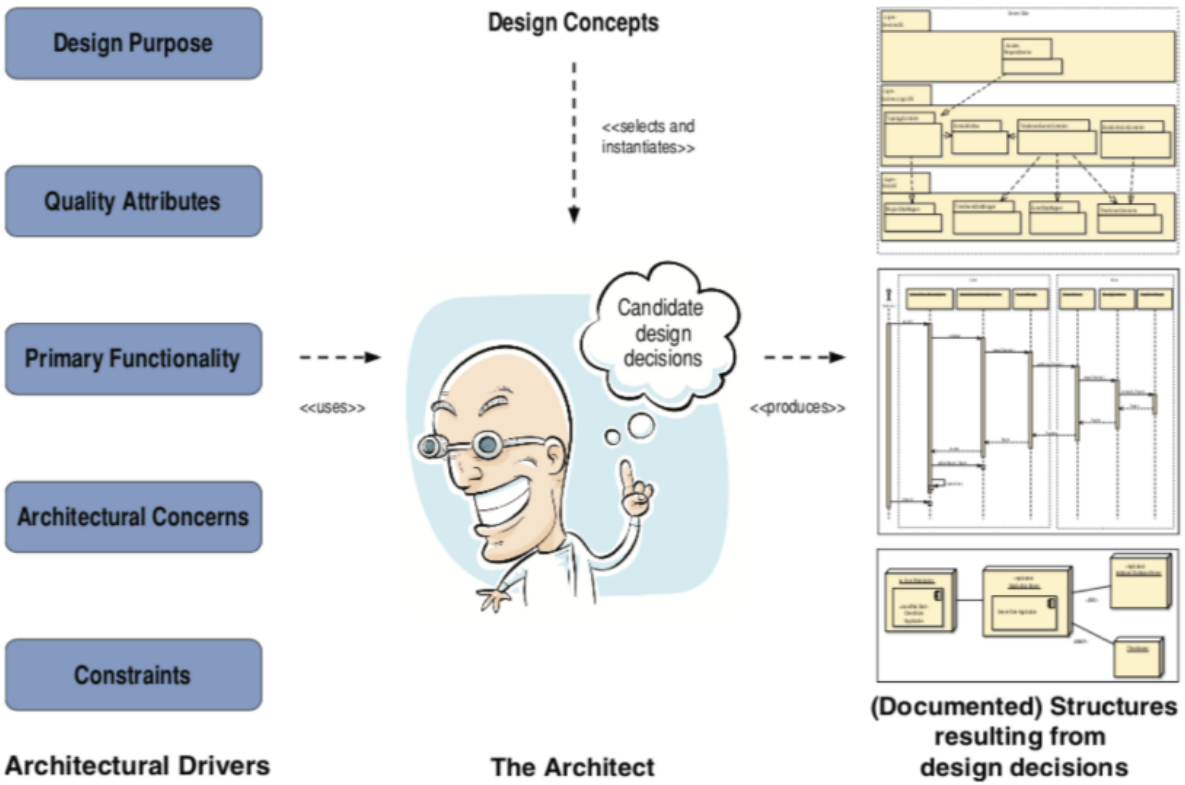
2

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

Arquitectura Software

Diseño Arquitectural

- Conjunto de decisiones en el software con objeto de satisfacer requisitos y restricciones



[3]

[3] Overview of the architecture design activity (Architect Image © Brett Lamb Dreamstime.com

Arquitectura Software

Diseño Arquitectural

- Conjunto de decisiones en el software con objeto de satisfacer requisitos y restricciones
- Las decisiones afectan al:
  - Propósito, funcionalidad, QoS y restricciones del sistema
- Es importante porque las decisiones de diseño son:
  - se toman al principio
  - difíciles de cambiar
  - tienen impacto a largo plazo

Arquitectura Software

- Se guía por principios arquitecturales:
  - Modularidad, grado de acoplamiento y cohesión
- En los SSDD habrá que tener en cuenta:
  - Heterogeneidad, Escalabilidad, Seguridad, QoS, etc.
- NO se debe diseñar desde cero!
  - Se debe partir de patrones arquitecturales, arquitecturas de referencia, etc.
  - Constituyen bloques de construcción arquitectural

Arquitectura Software

Patrones Arquitecturales

- Distintas combinaciones de componentes software y sus relaciones que son habituales y por tanto conocidas
- Elementos de un Patrón:
  - Descripción del patrón y su utilidad
  - Modelo arquitectural
  - Código fuente

Modelos Físicos

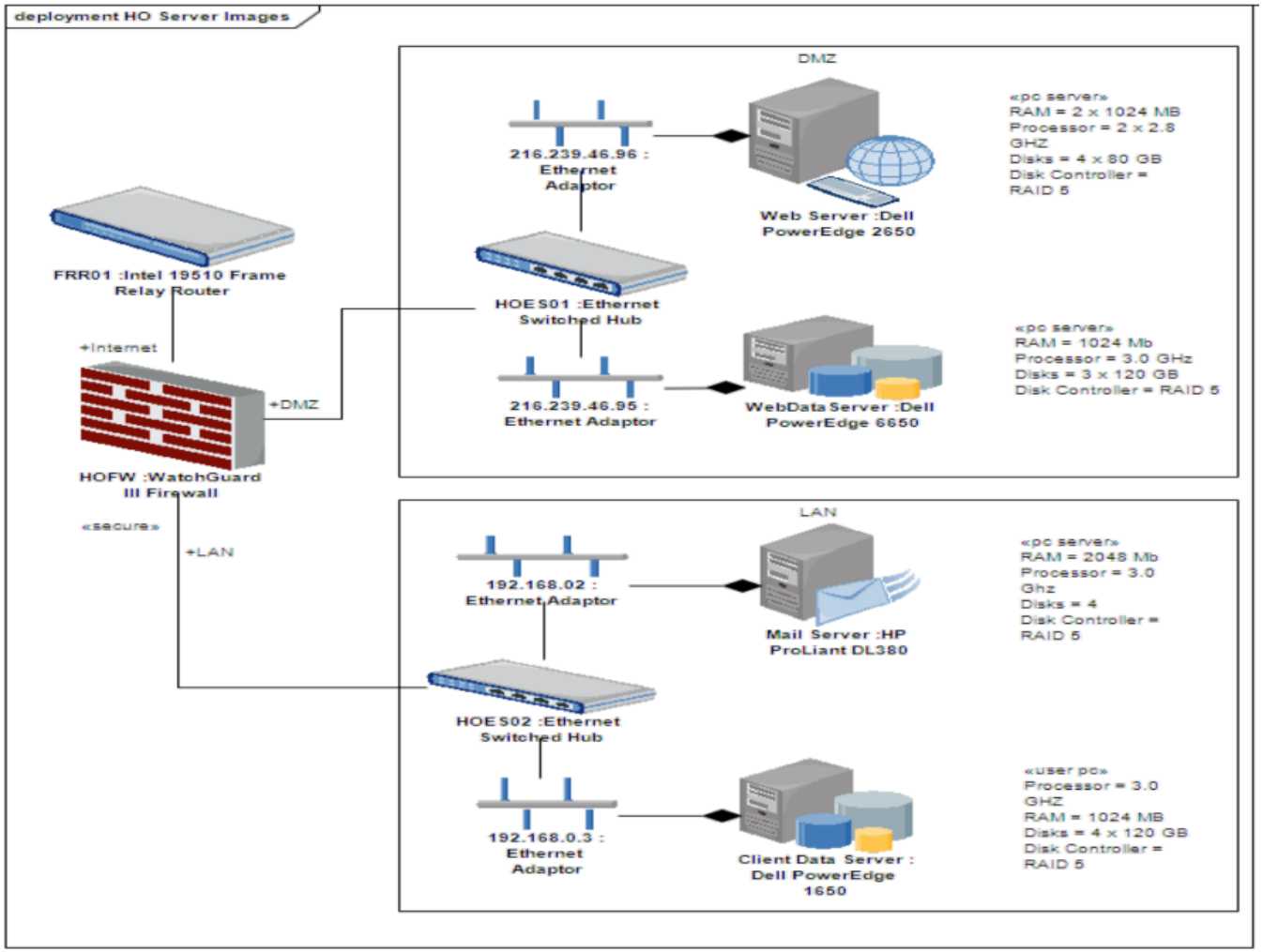
Modelos Físicos

Un modelo físico es:

- Una *representación* de los elementos hardware subyacentes de un SD
- Una *abstracción* de los detalles específicos de los *computadores* y de las *redes* de comunicación

Modelos Físicos

Extraído de https://www.sparxsystems.com.au



Modelos Físicos

Evolución Histórica

- Años 70-80
  - Auge de Ethernet y las redes de área local
- Años 90
  - Auge de Internet, SSDD interconectados mediante una red de redes
  - Mayor cantidad de nodos que en el pasado
  - PCs, infraestructura sin muchos cambios
- Actualidad
  - Auge de la computación móvil
  - PCs, portátiles, sensores, smart-phones, etc.

Modelos Físicos

Distributed systems:	*Early*	*Internet-scale*	*Contemporary*
Scale	Small	Large	Ultra-large
Heterogeneity	Limited (typically relatively homogenous configurations)	Significant in terms of platforms, languages and middleware	Added dimensions introduced including radically different styles of architecture
Openness	Not a priority	Significant priority with range of standards introduced	Major research challenge with existing standards not yet able to embrace complex systems
Quality of service	In its infancy	Significant priority with range of services introduced	Major research challenge with existing services not yet able to embrace complex systems

4

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

Tipos de Componentes

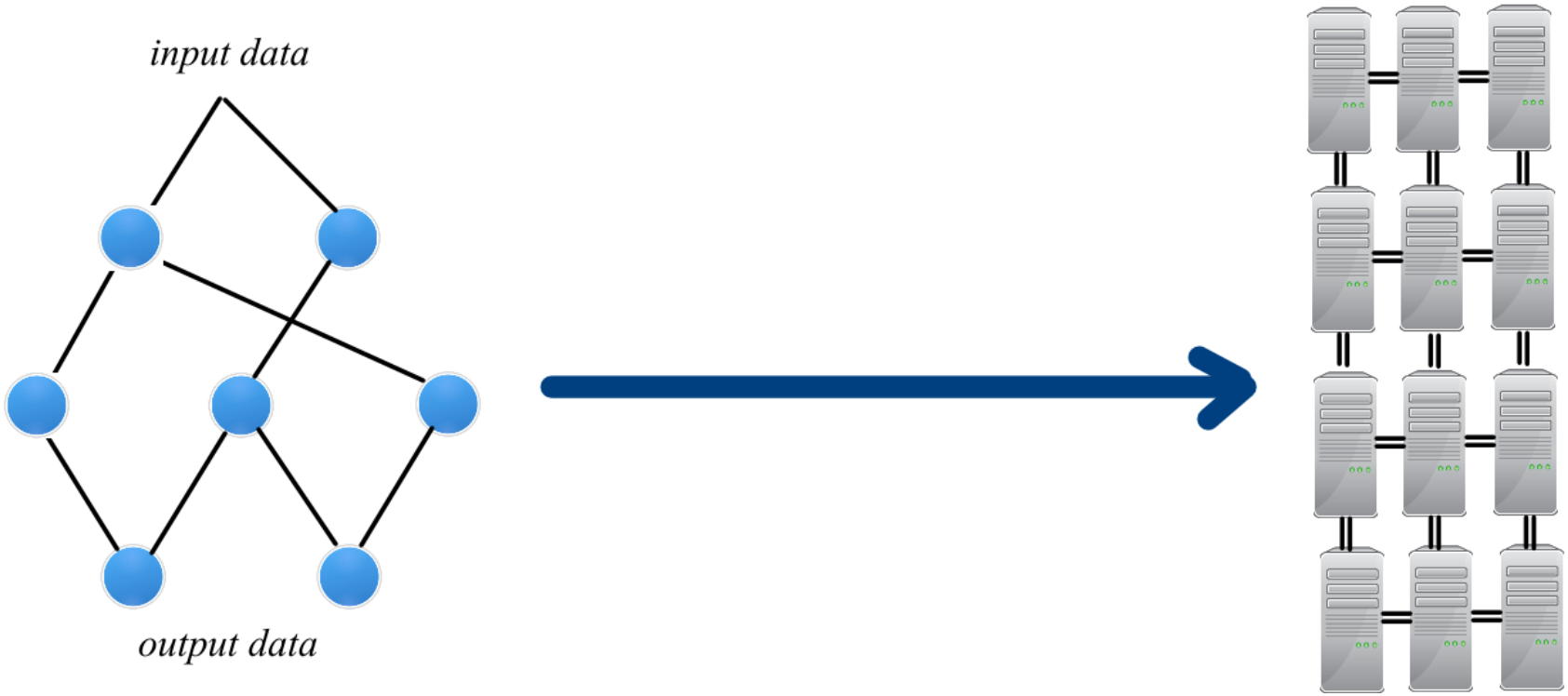
Tipos de Componentes

Desde un punto de vista de la programación:

- Programas, Módulos, Objetos
  - Componentes (CON gestión de recursos)
- Servicios
  - SOA, serverless (SIN gestión de recursos)

Tipos de Componentes

Programación Distribuida



Comunicación

Comunicación

Formas de Comunicación entre Componentes

- Comunicación Directa
- Comunicación Indirecta (mediada)
  - Linda, publish-subscribe, colas de mensajes, etc.

Comunicación

Formas de Comunicación entre Componentes

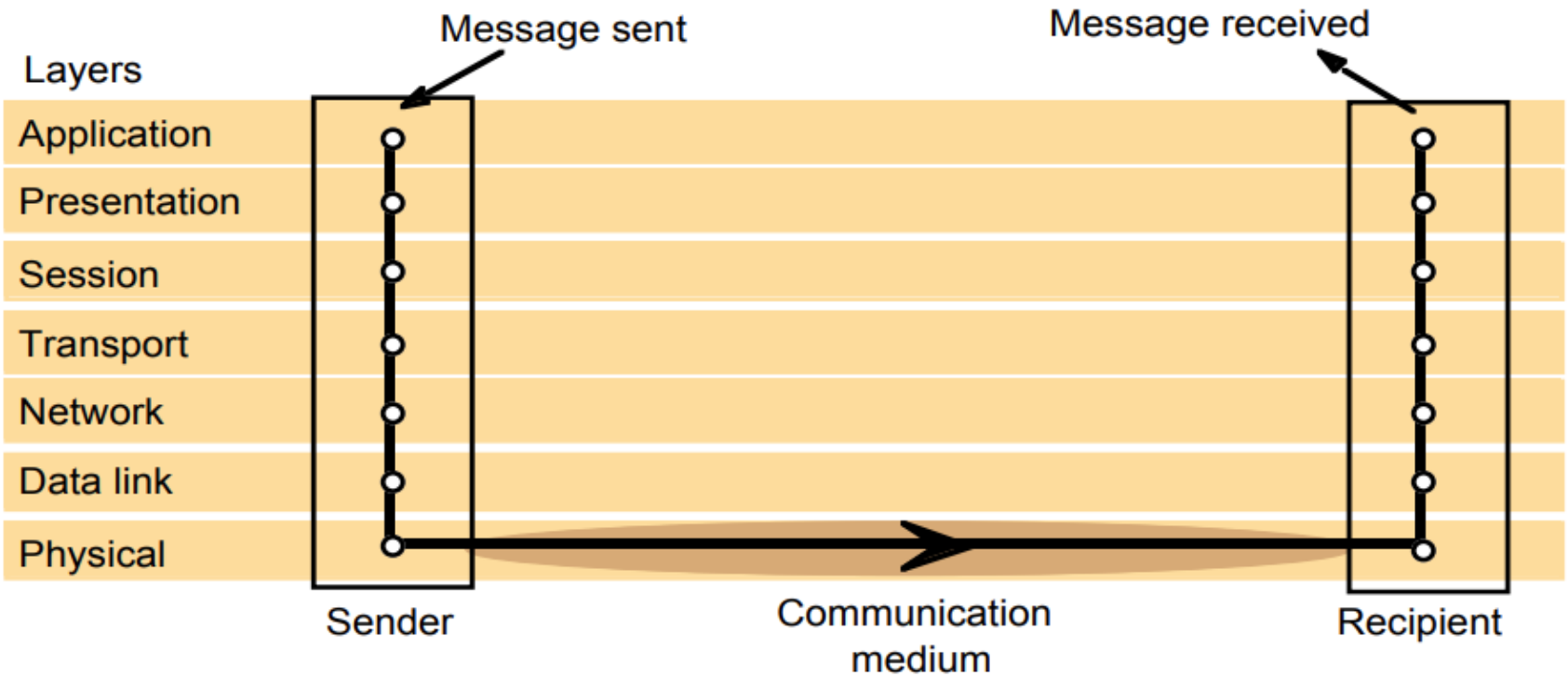
- Comunicación Directa
- Comunicación Indirecta (mediada)
  - Linda, publish-subscribe, colas de mensajes, etc.

Protocolos de Comunicación

- Formato de los Mensajes
- Inter-Process Communication (IPC)
  - Sockets IPC: TCP / UDP
  - RPC
  - Canales síncronos / asíncronos

Protocolos de Comunicación

Protocolos de Comunicación



Protocolos de Comunicación

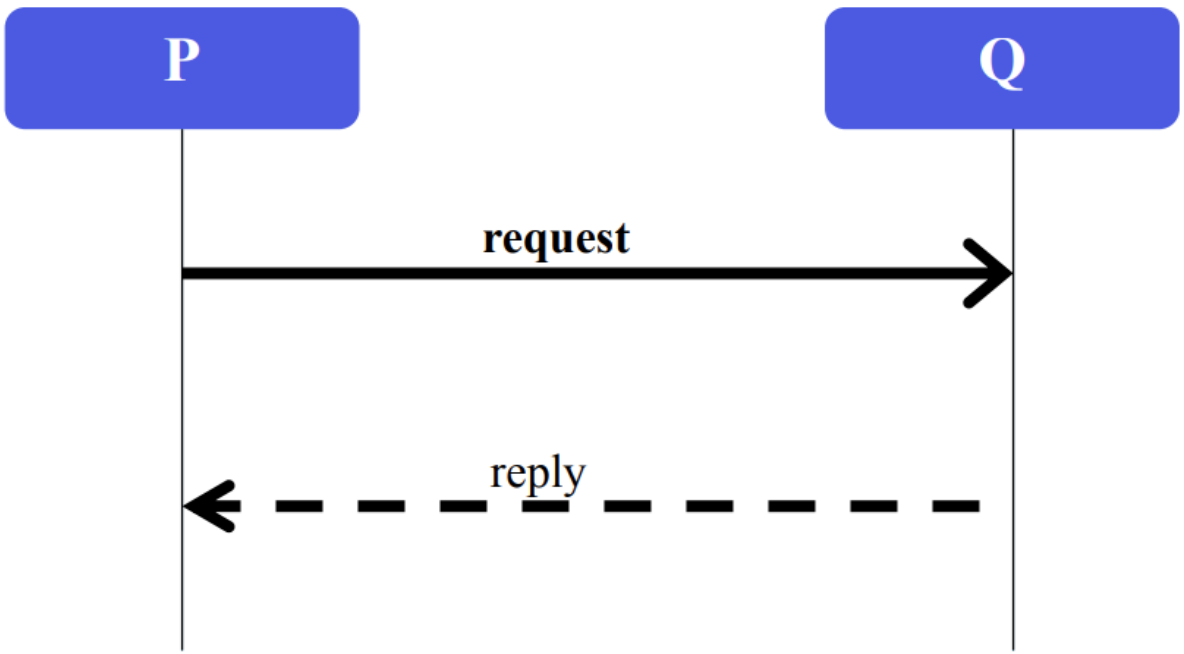
Protocolos de Comunicación

- Formato de los Mensajes
- Inter-Process Communication (IPC)
  - Sockets IPC: TCP / UDP
  - RPC
  - Canales síncronos / asíncronos

Protocolos de Interacción

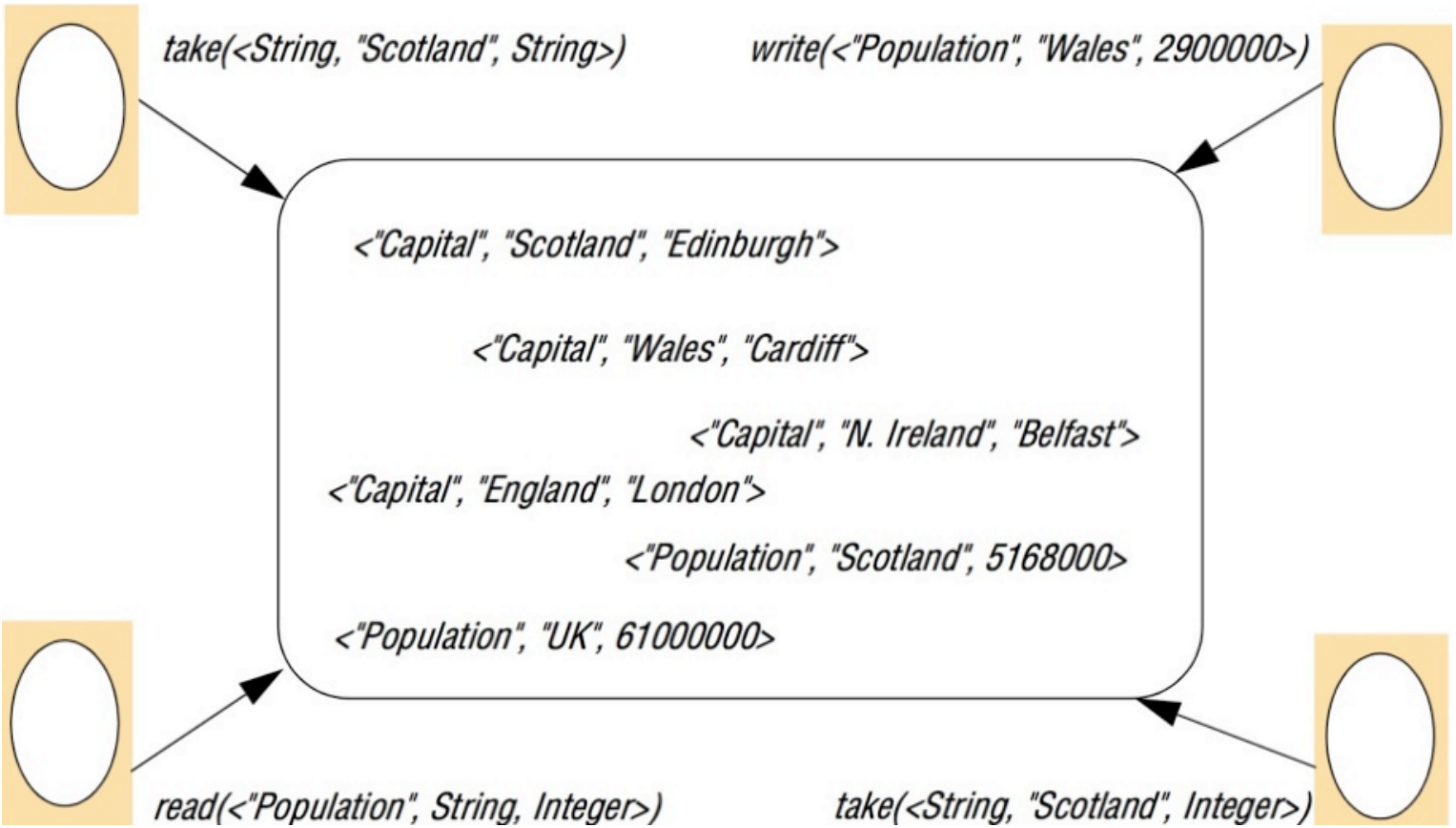
Protocolos de Interacción

- Conjunto de mensajes intercambiados entre procesos para conseguir una sincronización / coordinación



Comunicación Indirecta

Linda

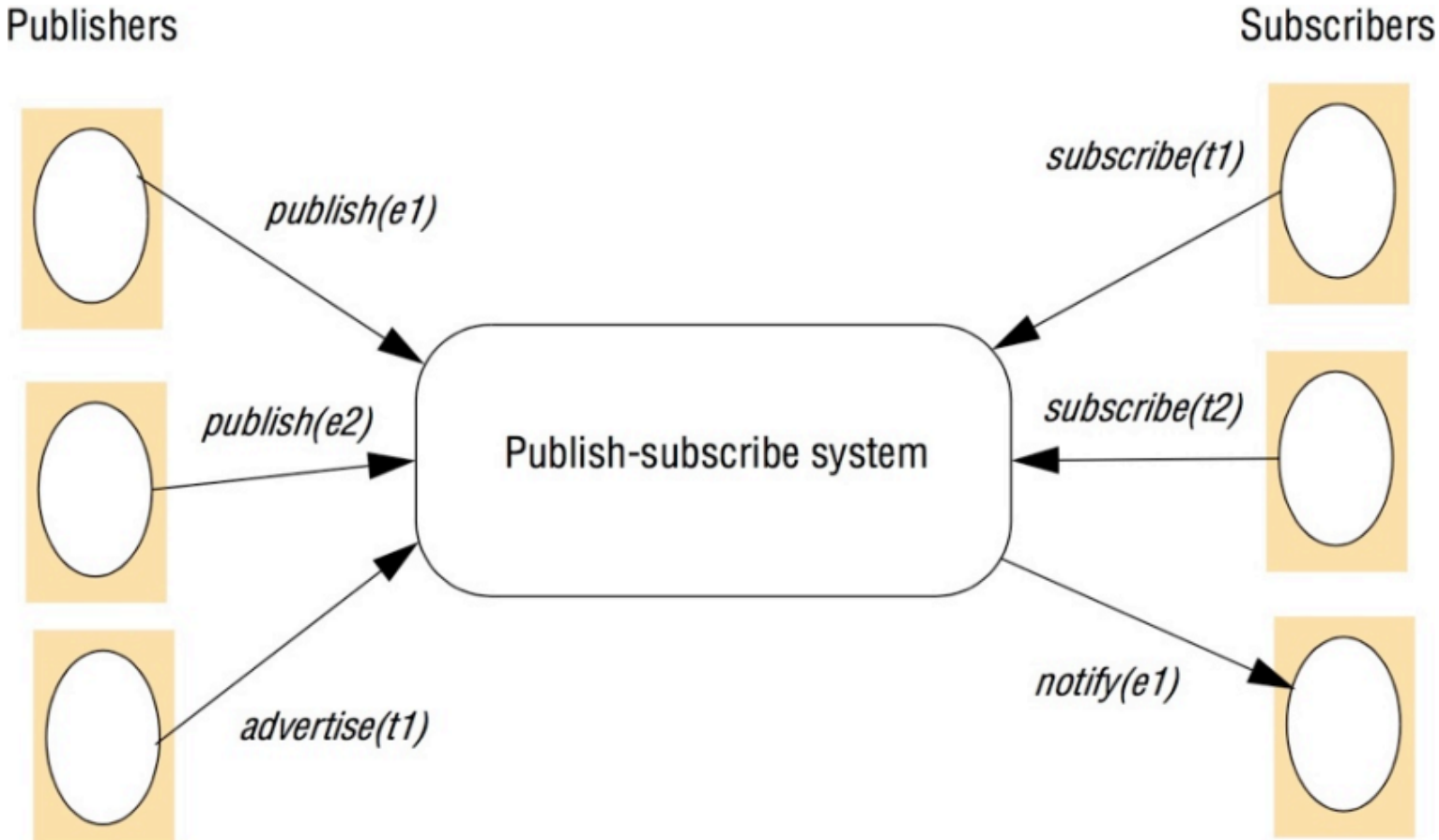


[5]

[5] Instructor's Guide for Coulouris, Dollimore and Kindberg Distributed Systems: Concepts and Design Edn. 4©Pearson Education 2005

Comunicación Indirecta

Publish-Subscribe



6

[6] Instructor's Guide for Coulouris, Dollimore and Kindberg Distributed Systems: Concepts and Design Edn. 4©Pearson Education 2005

Tipo y Grado de Acoplamiento

Acoplamiento Espacial

- Los procesos tienen que conocerse unos a otros
  - IP + puerto
  - Formato del mensaje (morfo-sintaxis)
  - Semántica del mensaje
  - Mensajes intercambiados

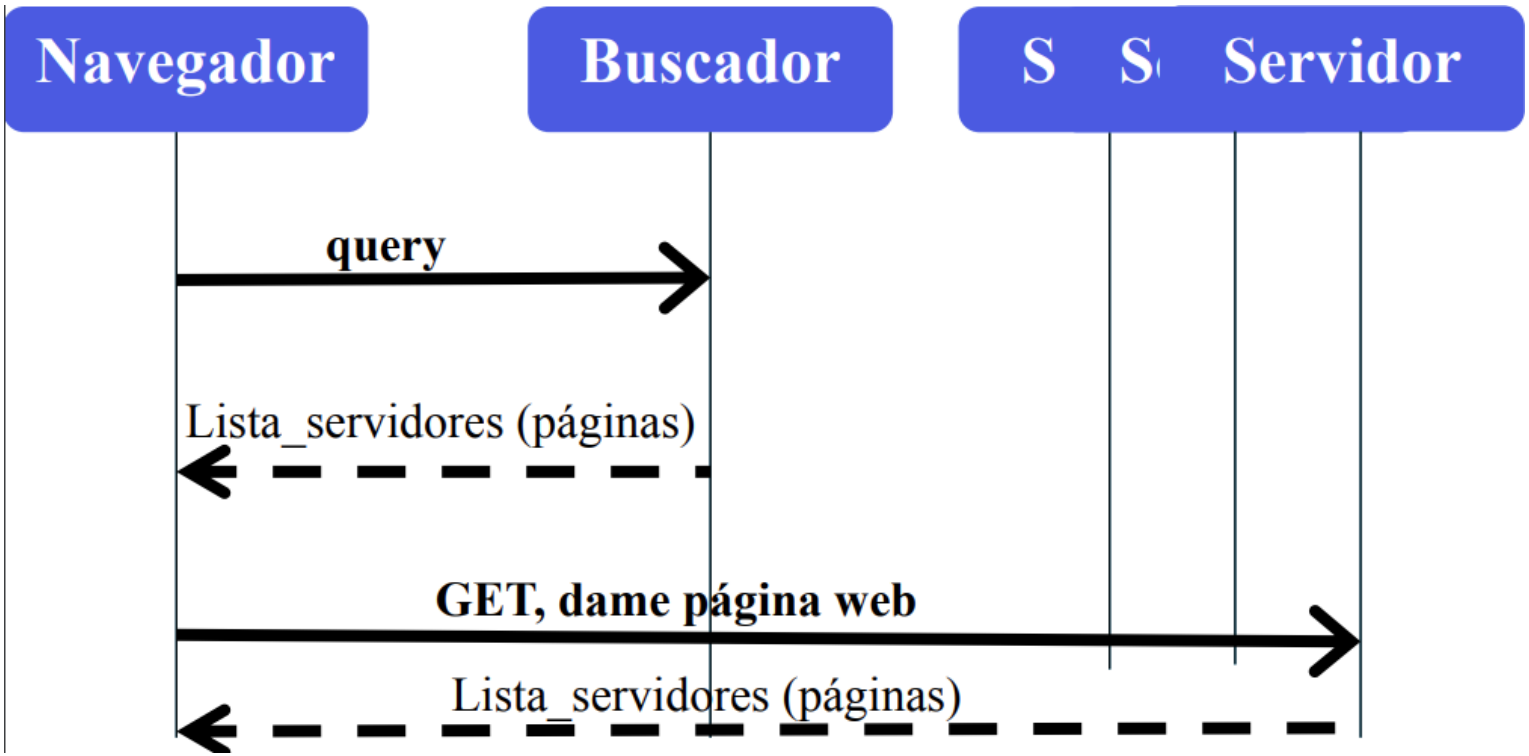
Acoplamiento Temporal

- Los procesos tienen que coincidir en el tiempo

Lo ideal es minimizar el grado de acoplamiento

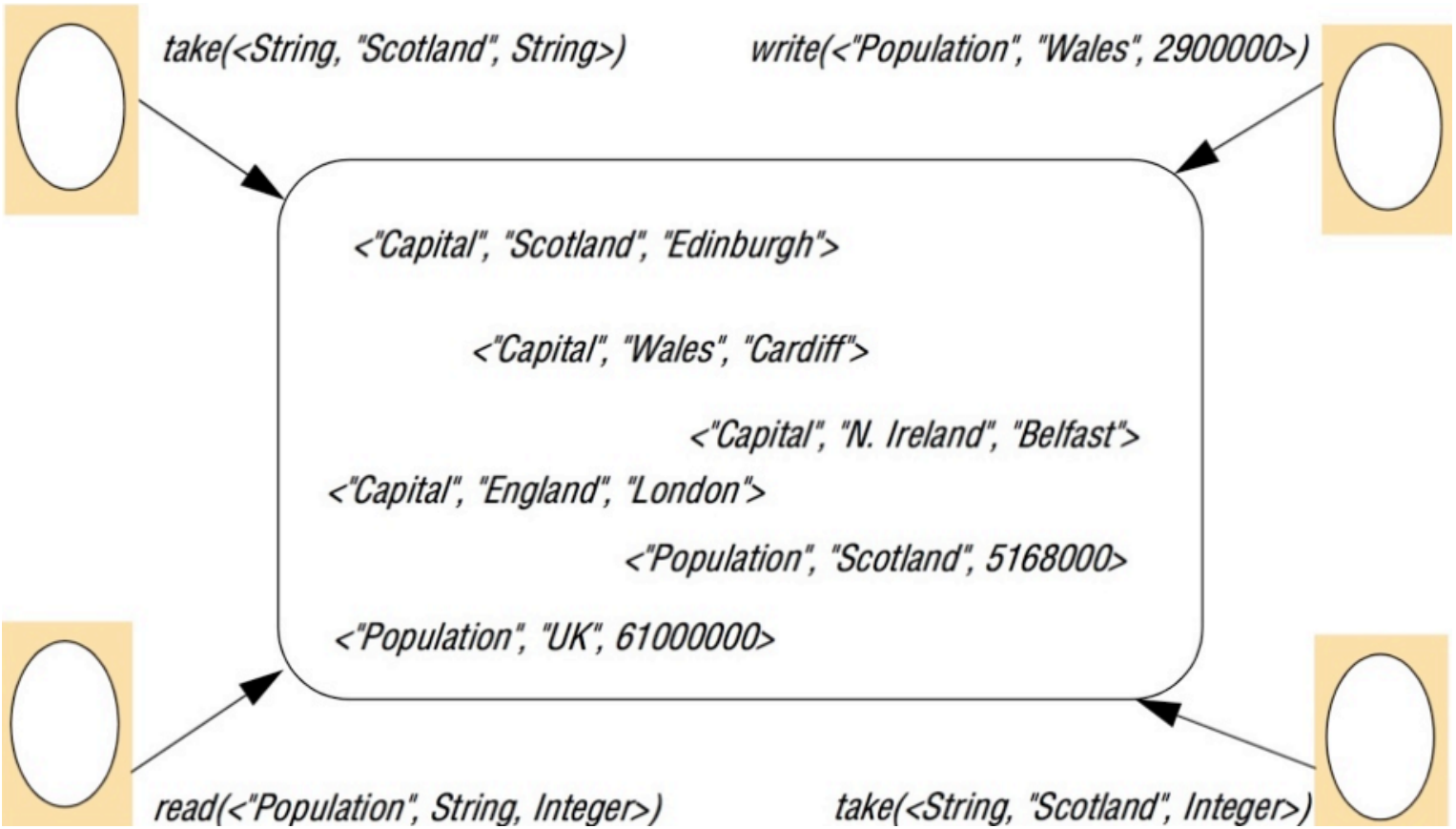
Tipo y Grado de Acoplamiento

Ejemplo: Búsqueda de Servidor Web



Tipo y Grado de Acoplamiento

Ejemplo: Análisis Grado de Acoplamiento en Linda



[7]

[7] Instructor's Guide for Coulouris, Dollimore and Kindberg Distributed Systems: Concepts and Design Edn. 4 © Pearson Education 2005

Tipo y Grado de Acoplamiento

Ejemplo: Análisis Grado de Acoplamiento en Linda
Los procesos no se conocen entre sí, solo conocen a Linda

- Los procesos ponen tuplas en el tuplespace, op write
- Se minimiza el acomplamiento espacial
- No hay acomplamiento temporal

Tipo y Grado de Acoplamiento

Ejemplo: Análisis Grado de Acoplamiento en Linda
Los procesos no se conocen entre sí, solo conocen a Linda

- Los procesos ponen tuplas en el tuplespace, op write
- Se minimiza el acomplamiento espacial
- No hay acomplamiento temporal

Linda inspiró el patrón broker de mensajes (message broker)

- Un message broker tiene además un conjunto de componentes wrappers que saben invocar distintos servicios

Patrones Arquitecturales

Patrones Arquitecturales

Patrones Arquitecturales en SSDD

- Cuando aparecen problemas con cierta frecuencia es una buena idea aplicar siempre la misma solución
- Los patrones arquitecturales son soluciones reutilizables para problemas que ocurren frecuentemente
- Muchas veces los patrones no son soluciones cerradas, sino que pueden adaptarse al contexto

Cliente-Servidor

Descripción

- Esta es la arquitectura más utilizada y la más importante históricamente
- Consiste en N procesos, N - 1 clientes y 1 servidor
- El servidor ofrece una funcionalidad o acceso a un recurso(s)
- El cliente interactúa con el servidor para poder acceder a la funcionalidad o al recurso

Cliente-Servidor

Descripción

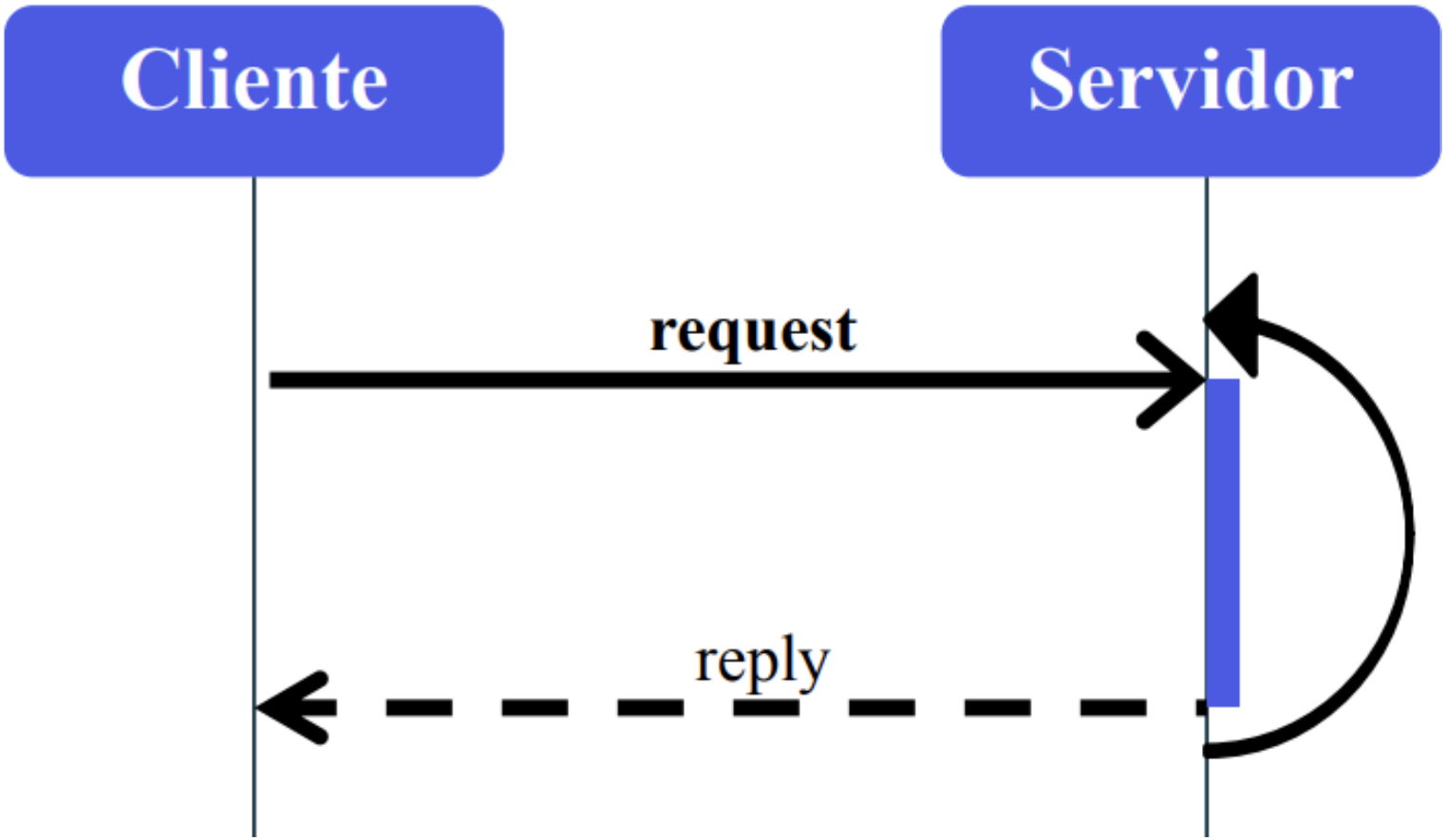
- Esta es la arquitectura más utilizada y la más importante históricamente
- Consiste en N procesos, N - 1 clientes y 1 servidor
- El servidor ofrece una funcionalidad o acceso a un recurso(s)
- El cliente interactúa con el servidor para poder acceder a la funcionalidad o al recurso

Variantes

- Servidor Secuencial (atiende peticiones una tras otra)
- Servidor Concurrente (atiende peticiones simultáneamente)

Cliente-Servidor

Diagrama Arquitectural: servidor secuencial



Cliente-Servidor

Código Fuente Servidor Secuencial

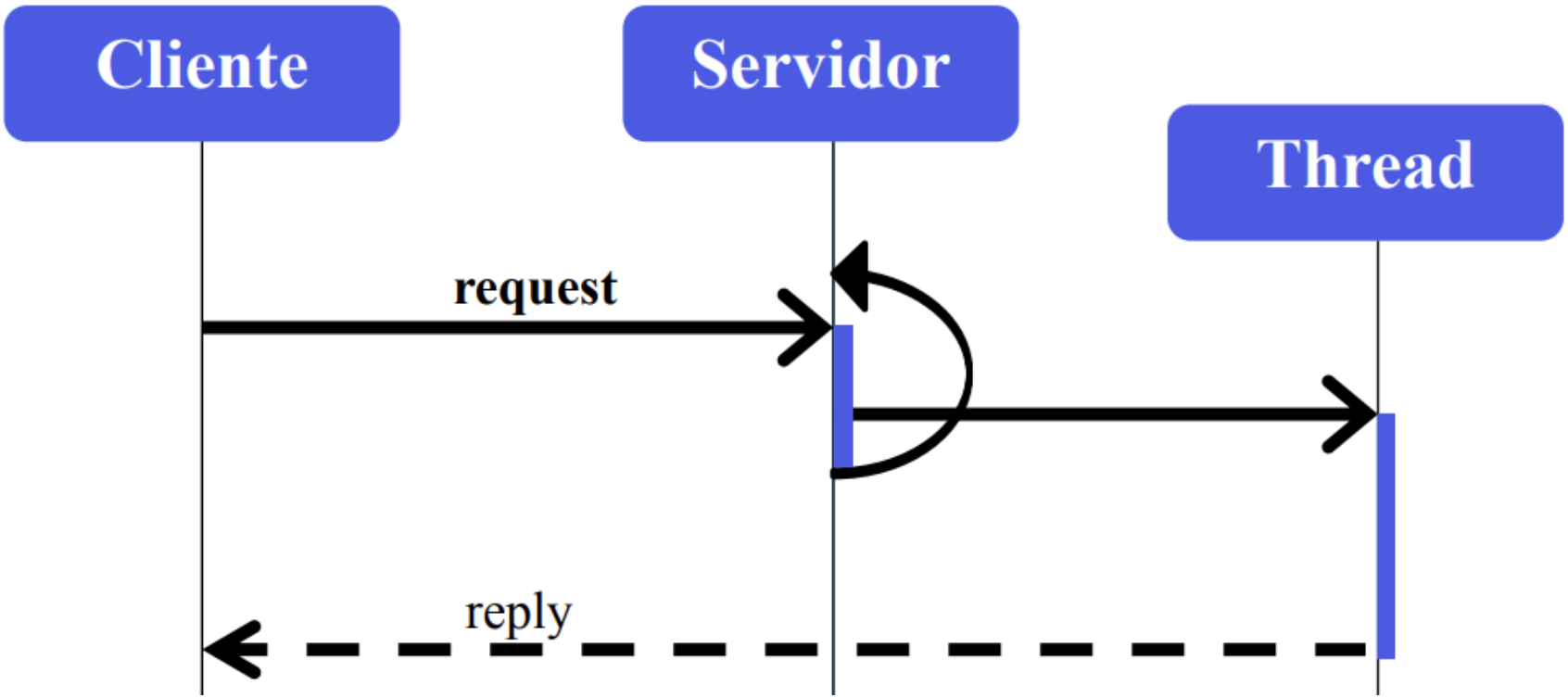
Cliente

```
send(s_pid, msg)
receive do
  {:res, result} -> result
end
```

Servidor

```
def loop do
  receive do
    {:req, msg} ->
          send(c_pid, op(msg))
  end
  loop()
end
```

Diagrama Servidor Concurrente



Código Servidor Concurrente

Cliente

```
send(s_pid, {:req, msg})
receive do
  {:res, result} -> result
end
```

Servidor

```
def loop do
  receive do
    {:req, msg} ->
      go (func {send(c_pid, op(msg))}
  end
end
```

QoS Cliente Servidor

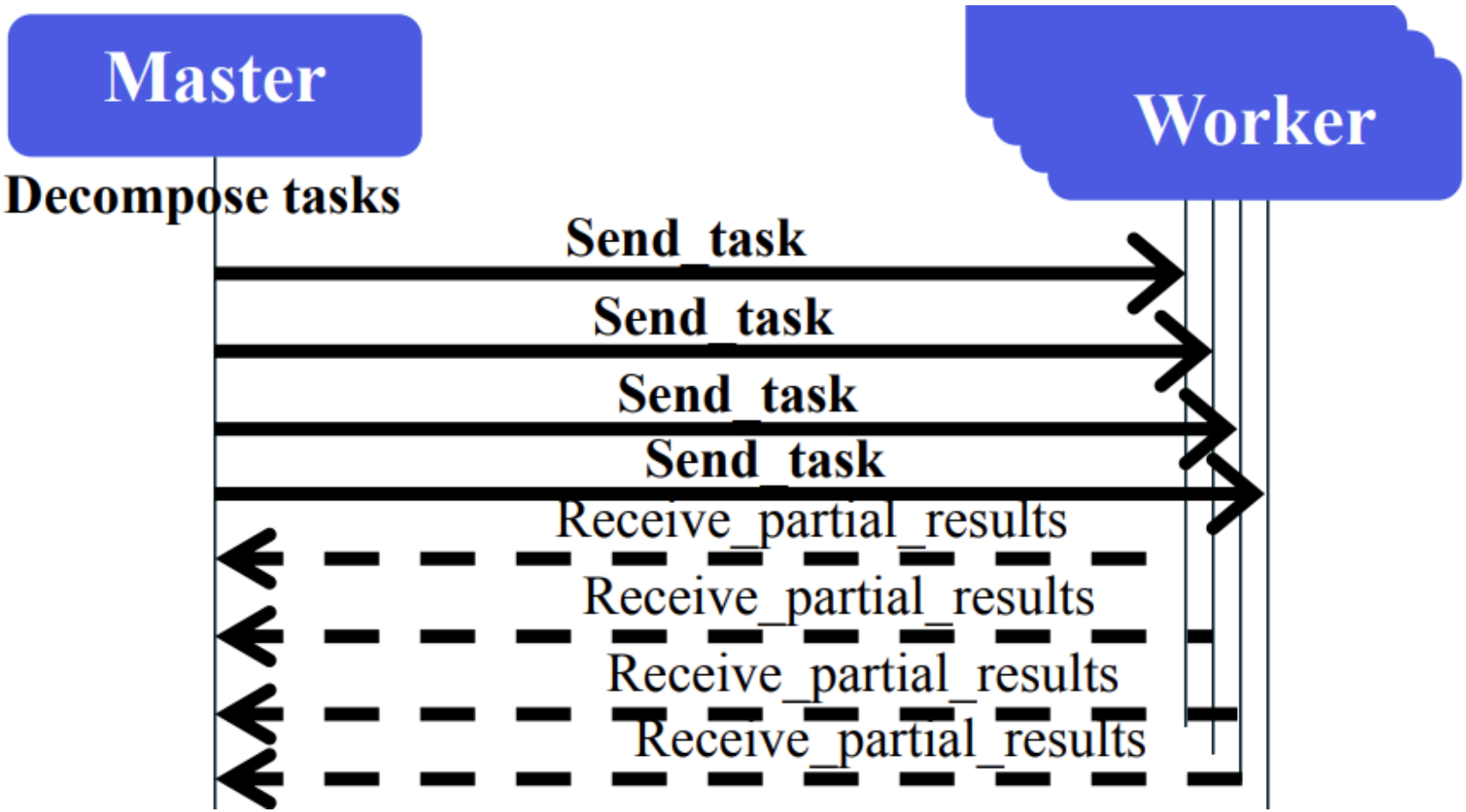
Máster-Worker

Descripción

- Esta arquitectura puede verse como una generalización de la arquitectura cliente servidor
- Consiste en un conjunto de clientes, un proceso máster y un conjunto de workers.
- El proceso máster recibe una carga de trabajo de algún cliente, opcionalmente la puede descomponer para repartirla entre los workers.
- Un worker recibe una tarea del máster y la realiza
- El worker típicamente le devuelve el resultado al máster
- El máster si descompuso las tareas las va a tener que agregar, cuando reciba los resultados de los workers
- Finalmente, el máster le devuelve el resultado al cliente
- Alternativamente, podría darse el caso en que el worker devolviera directamente los resultados al cliente.

Máster-Worker

Diagrama Máster Worker



Máster-Worker

Código Fuente Máster Worker

Código Elixir, Com. Directa, req-reply

master

```
func assign(tasks){
   for t := range tasks {
        send(worker, {:req, ,t})
   }
}
func collect(results) {
 for {
   receive do
  {:res, result} ->
          send(client, result)
 end
}
```

21/9/20 62

Worker

```
def worker do
  for{
  receive do
     {:req, {m_pid,m}} ->
              send(master, op(msg))
  end
  }
end
```

Sistemas Distribuidos

QoS Máster Worker

Peer-to-Peer (P2P)

En este tipo de arquitectura todos los procesos involucrados en una tarea juegan un rol similar.

- La interacción entre ellos es cooperativa
- A diferencia de la arquitectura cliente-servidor, que escala muy mal, esta escala bien
- Generalizando, todos los procesos ejecutan el mismo código y proporcionan la misma interfaz de interacción
- Entre este tipo de arquitecturas están las redes para compartir ficheros
  - Napster, Gnutella, eDonkey, Torrent

Asignación Tareas

Asignación Tareas

¿Cómo se distribuyen las tareas en los recursos?

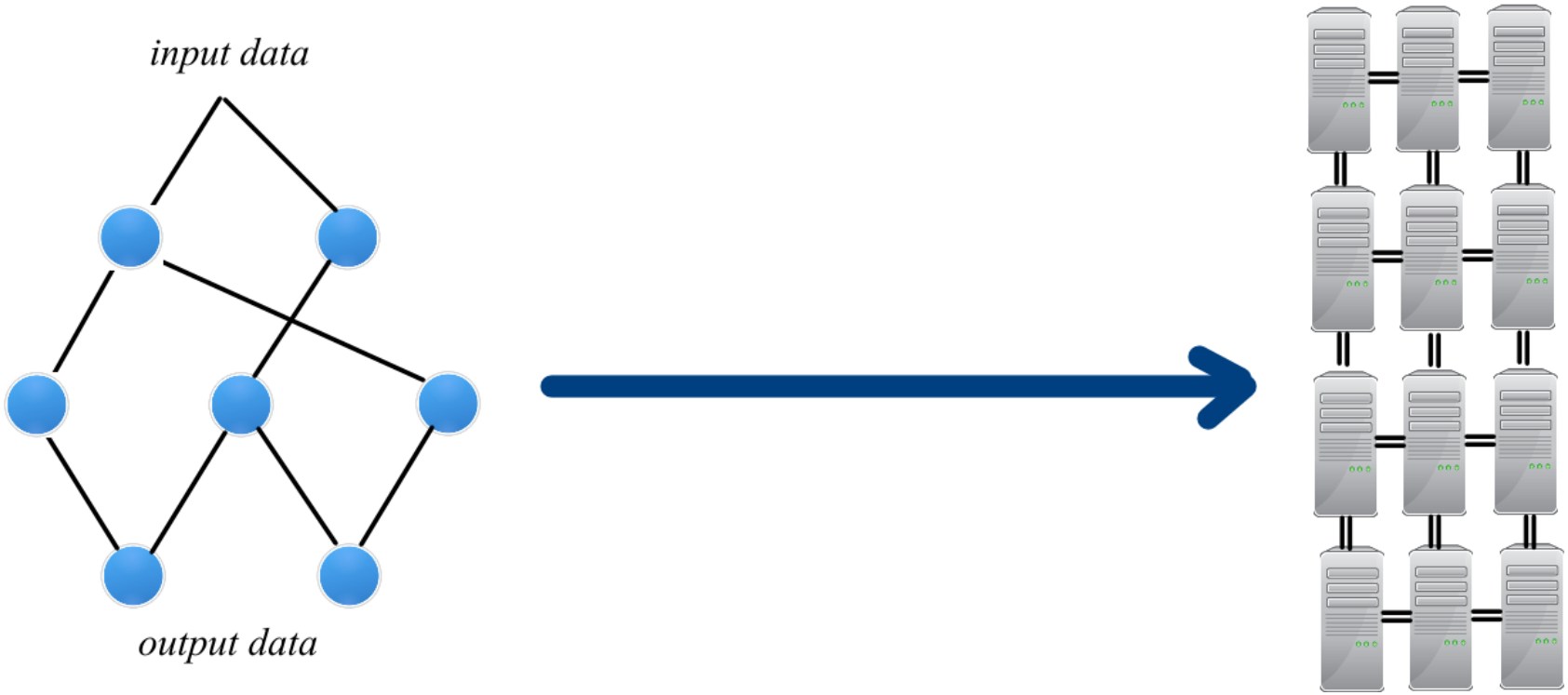
Asignación Tareas

¿Cómo se distribuyen las tareas en los recursos?

Objetivos:

- ¿Qué carga de trabajo tengo? ¿Qué recursos tengo?
- Garantizar el QoS
- Maximizar el uso de recursos compartidos
- Minimizar el coste económico, energético, etc.

Asignación Tareas



Resumen

Resumen

- Concepto de Arquitectura Software
- Modelos Físicos
- Tipos de Componentes
- Comunicación entre Componentes
  - Directa vs Indirecta
  - Protocolos de comunicación / interacción
  - Acoplamiento espacial / temporal
- Patrones Arquitecturales de Sistemas Distribuidos
  - Cliente-Servidor
  - Máster-Worker
  - P2P

Arquitecturas de Sistemas Distribuidos

30221 - Sistemas Distribuidos

Rafael Tolosana Calasanz

Dpto. Informática e Ing. de Sistemas