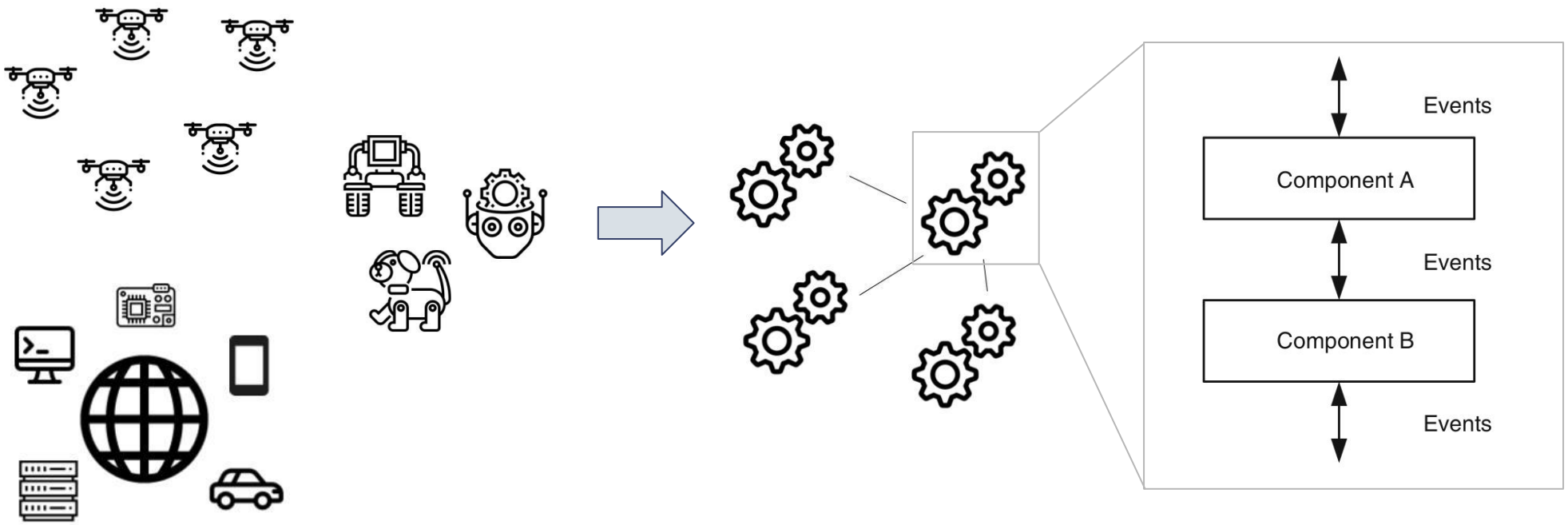
Dependable Distributed Systems Master of Science in Engineering in Computer Science

AA 2023/2024

LECTURE 2C: DESIGNING A DISTRIBUTED PROTOCOL

Recall: From a Physical System to an Abstraction



Recall: Composition Model and its code

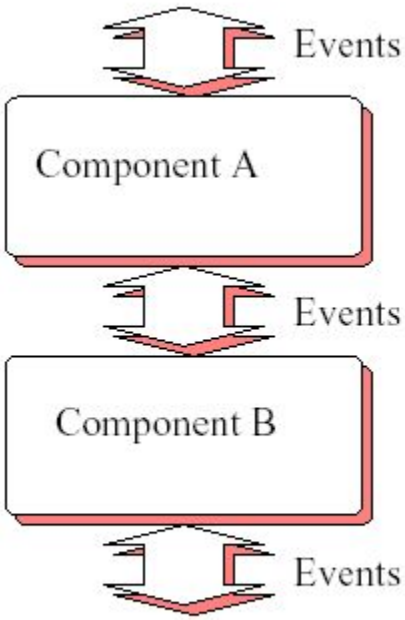


Figure 1.1. Composition model

upon event $\langle\, co_1, Event_1 \mid att_1^1, att_1^2, \ldots \,\rangle$ **do**
    do something;
    trigger $\langle\, co_2, Event_2 \mid att_2^1, att_2^2, \ldots \,\rangle$; *// send some event*

upon event $\langle\, co_1, Event_3 \mid att_3^1, att_3^2, \ldots \,\rangle$ **do**
    do something else;
    trigger $\langle\, co_2, Event_4 \mid att_4^1, att_4^2, \ldots \,\rangle$; *// send some other event*

Recall: Composition Model and its code

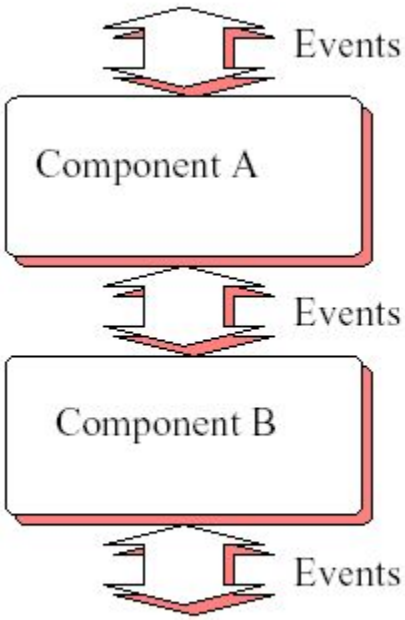
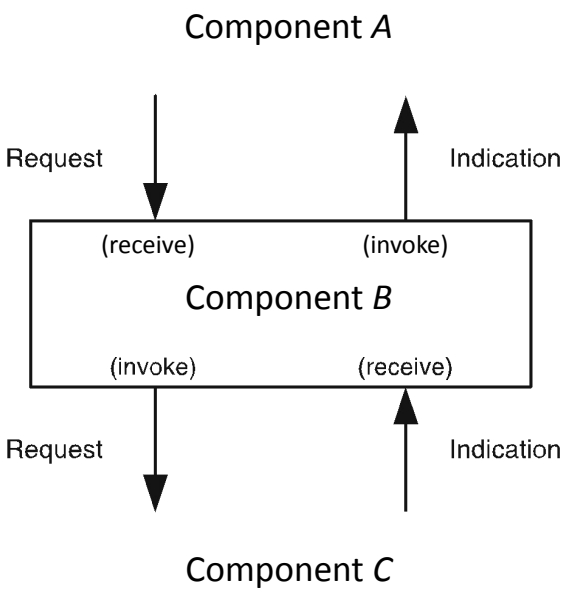


Figure 1.1. Composition model

upon *condition* **do** // an internal event
do something;

upon event $\langle$ *co*, *Event* | $att_1^1, att_1^2, \ldots \rangle$ **such that** *condition* **do**
do something;

Recall: Programming Interface



Request events

- used by a component c1 to request a service to component c2
- e.g., c1 may ask to c2 to disseminate a message in a group

Confirmation events

- used by a component to confirm the completion of a request

Indication events

- used by a component c1 to deliver information to a component c2
- e.g., the notification of a message delivery

On Distributed Protocol/Components

Perfect Point-to-Point Link: Implementation

Algorithm 2.2: Eliminate Duplicates

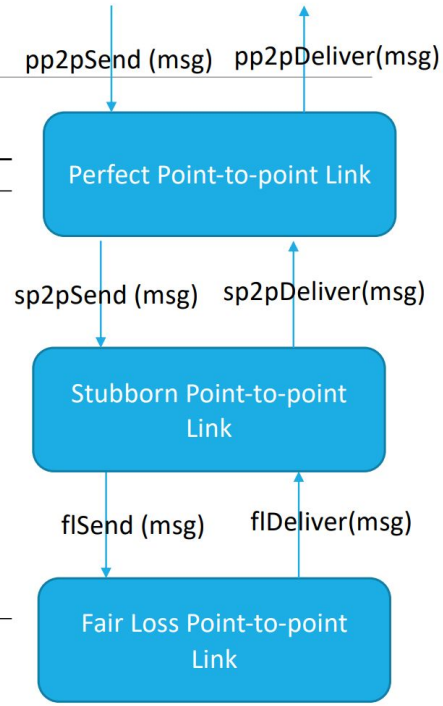
Implements:
PerfectPointToPointLinks, **instance** *pl*.

Uses:
StubbornPointToPointLinks, **instance** *sl*.

upon event $\langle$ *pl*, *Init* $\rangle$ **do**
delivered := $\emptyset$;

upon event $\langle$ *pl*, *Send* | q, m $\rangle$ **do**
trigger $\langle$ *sl*, *Send* | q, m $\rangle$;

upon event $\langle$ *sl*, *Deliver* | p, m $\rangle$ **do**
if $m \notin$ *delivered* **then**
delivered := *delivered* $\cup \{m\}$;
trigger $\langle$ *pl*, *Deliver* | p, m $\rangle$;



- The code of a distributed protocol is INDEPENDENTLY EXECUTED by each process
- A process can ACCESS ONLY ITS LOCAL VARIABLE (i.e., its local instance of a variable)
- The same local variable can have different values in different processes
- Inter-process and Inter-component communication : EVENTS

p1.send(p2,m) _> p2.deliver(p1,m)

Example: *SystemMax* Component

Consider a distributed system composed of **n** processes $\Pi = \{p_1, p_2, \ldots, p_n\}$, each one identified by a unique integer.

Processes communicate by exchanging messages over **perfect point-to-point links**.

Every process knows Π and can communicate with every other processes in the system.

Each process stores an integer positive number in a local variable called *l_max*.

Design a distributed protocol implementing a software component called SystemMax, which aim is to retrieve the maximum value l_max stored among all processes.

More in detail, SystemMax handles two events: **getMax()** and **returnMax(max_value)**. The former triggers the retrieval of the information, whereas returnMax(max_value) returns the maximum retrieved value.

Example: *SystemMax* Component

Module: SystemMax (SM)

Events:

- Request⟨**SM, getMax** ⟩: retrieve the maximum value of l_max.
- Indication ⟨**SM, returnMax | max_value** ⟩: return the maximum value of l_max.

component, event | parameters

Properties (implicit):

- **Liveness** : if a process triggers ⟨SM, getMax ⟩, then it eventually returns ⟨SM, returnMax | max_value ⟩
- **Exact response (Safety)** : ⟨SM, returnMax | max_value ⟩returns the maximum value max_value among all l_max

Recall - Perfect point-to-point link specification

Module 2.3: Interface and properties of perfect point-to-point links

Module:

Name: PerfectPointToPointLinks, **instance** *pl*.

Events:

Request: $\langle$ *pl*, *Send* | q, m $\rangle$: Requests to send message m to process q.

Indication: $\langle$ *pl*, *Deliver* | p, m $\rangle$: Delivers message m sent by process p.

Properties:

PL1: *Reliable delivery:* If a correct process p sends a message m to a correct process q, then q eventually delivers m.

PL2: *No duplication:* No message is delivered by a process more than once.

PL3: *No creation:* If some process q delivers a message m with sender p, then m was previously sent to q by process p.

Back from Theory to Practice

- *“Every process knows Π”*
  - Bootstrap
  - Discovery

- *“Communicate by exchanging messages over perfect point-to-point links”*
  - Direct implementation https://github.com/giovannifarina/DDS_primitives_and_protocols
  - Networking Library https://libp2p.io/
  - Message Middleware https://zeromq.org/ https://www.rabbitmq.com/

- *“Synchronous distributed system”*