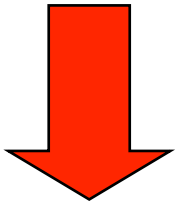
Embedded Networks

Models of Communication

Summer Term 2012

CO-OPERATIVE SYSTEMS

Which model of communication?



What kind of addressing and routing should be supported by the network?

Which abstractions in the programming model?

Abstractions for Communication

- **Message passing**
- **Remote Procedure Call/ Remote Object Invocation**
- **Publish Subscribe**

Abstractions for Communication

Dimensions of Dependencies:

Space Coupling: References must be known

Explicit specification of the destination, i.e. producer must know where to send the message. Message contains an ID specifying an address or name.

Flow coupling: Control transfer with communication

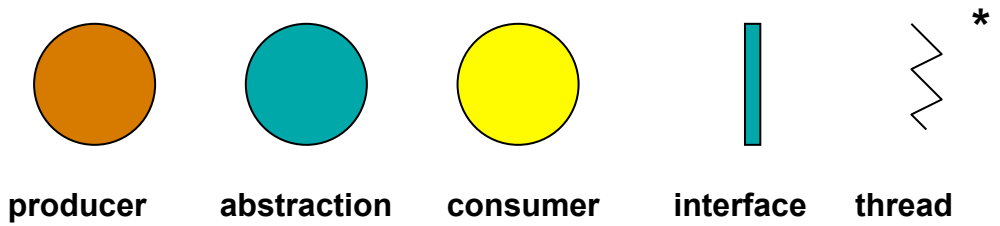
Defines whether there is a control transfer coupled with a message transfer.
E.g. if the sender blocks until a message is correctly received.

Coupling in time: Both sides must be active

Communication can only take place if all partners are up and active.

Message passing

Connected socket, e.g. TCP

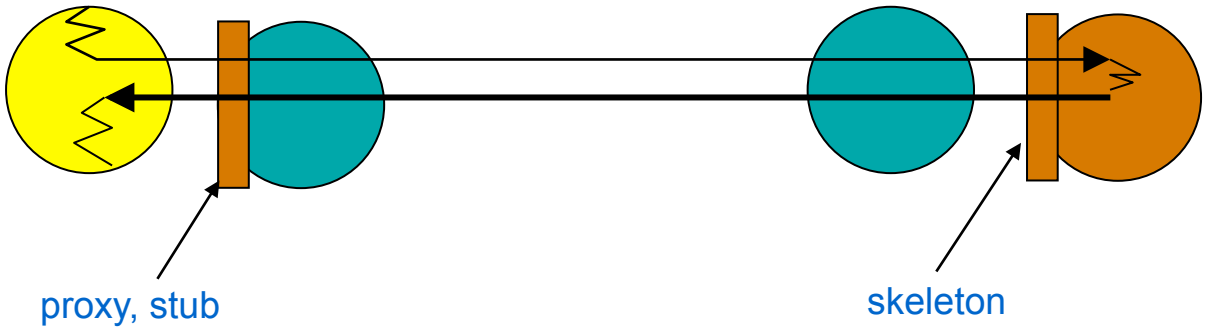


primitives: send (), receive ()

Coupling: space, time

* Notation acc. P. Eugster: Type-Based Publish Subscribe, PhD-thesis, EPFL, Nr. 2503, 2001

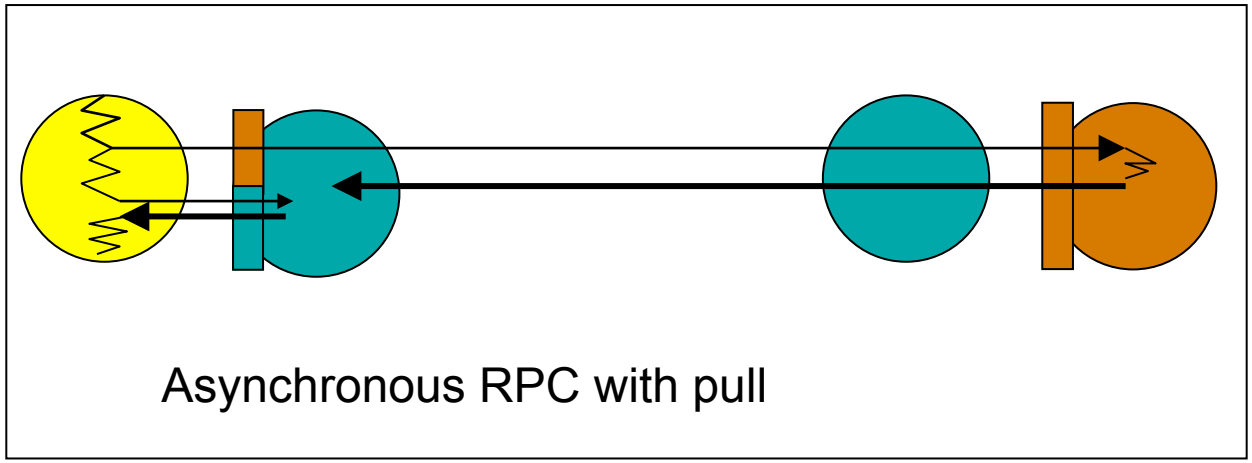
Remote Procedure Call (RPC)



Relation: one-to-one

Coupling:
Space: destination is explicitely specified
Flow: blocks until message is delivered
Time: both sides must be active

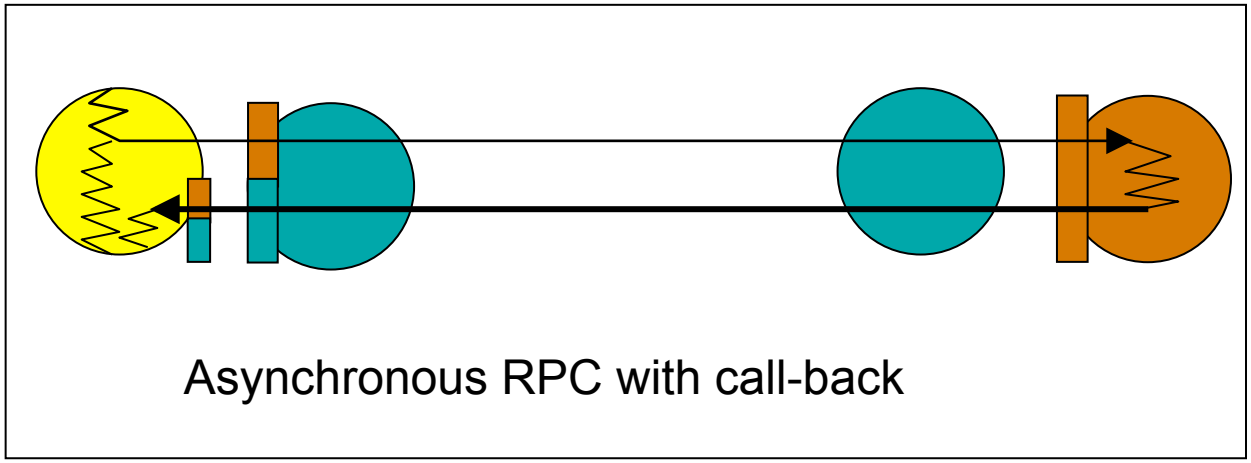
Variations of RPC



Example: Concurrent Smalltalk

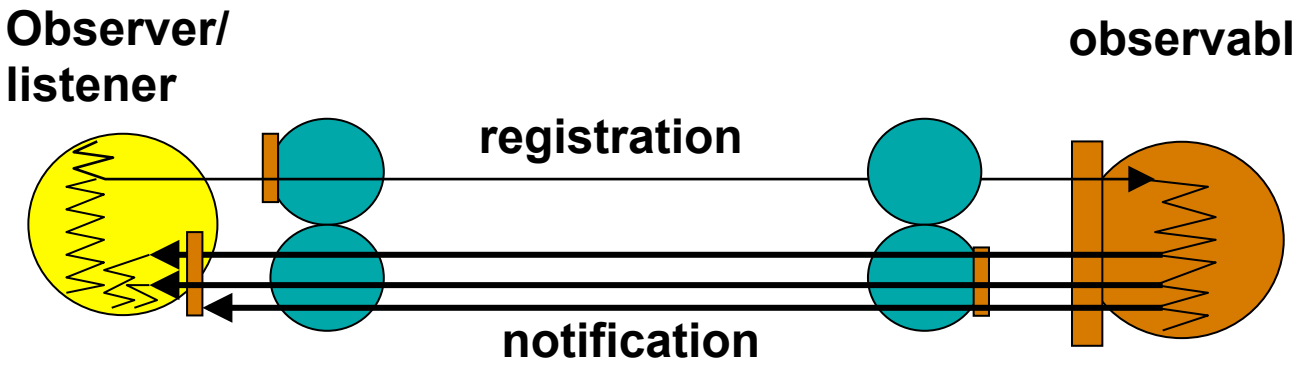
Relation: one-to-one

Coupling:
Space: destination is explicitely specified
Flow: no flow coupling
Time: both sides must be active



Example: Eiffel

Notification



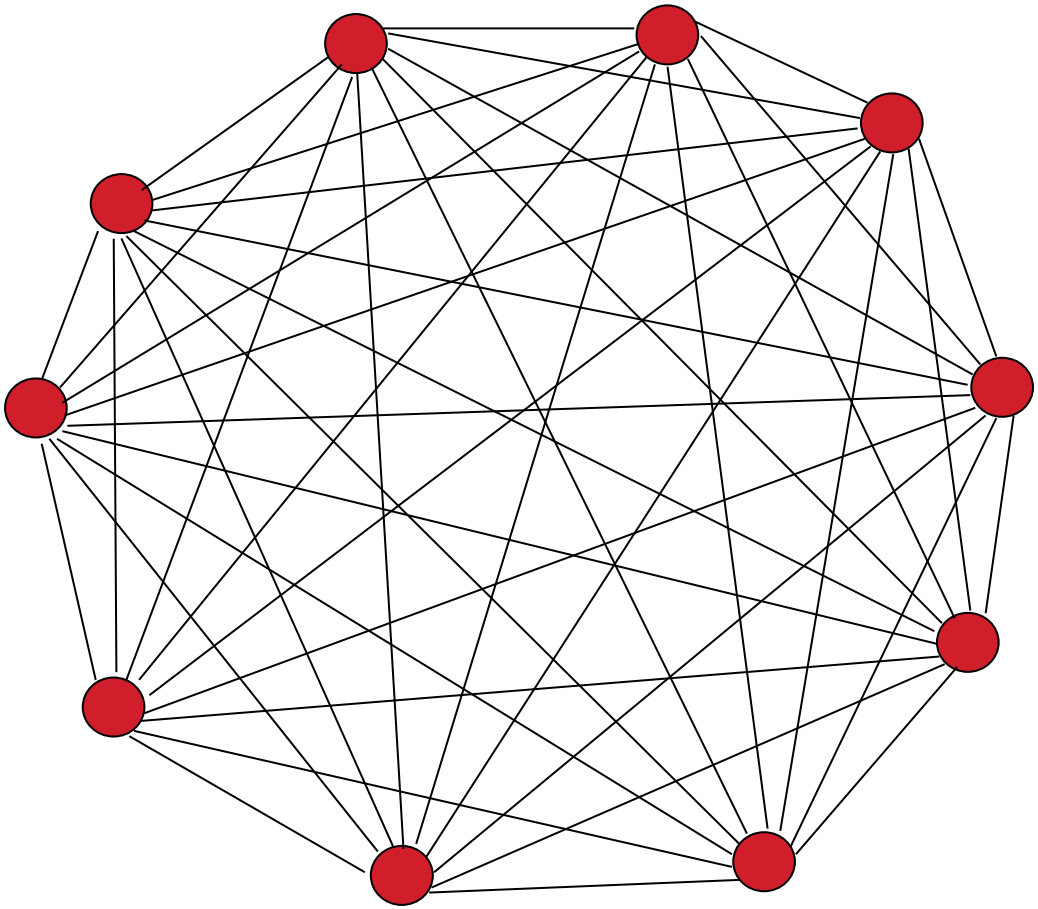
Examles:
Java

Relation: one-to-many

Coupling:
Space: Yes (Observable/Observer pattern (delegation))
Flow: none
Time: both sides must be active (notification performed by RMI)

Interaction Structure in Co-operative Systems

many-to-many

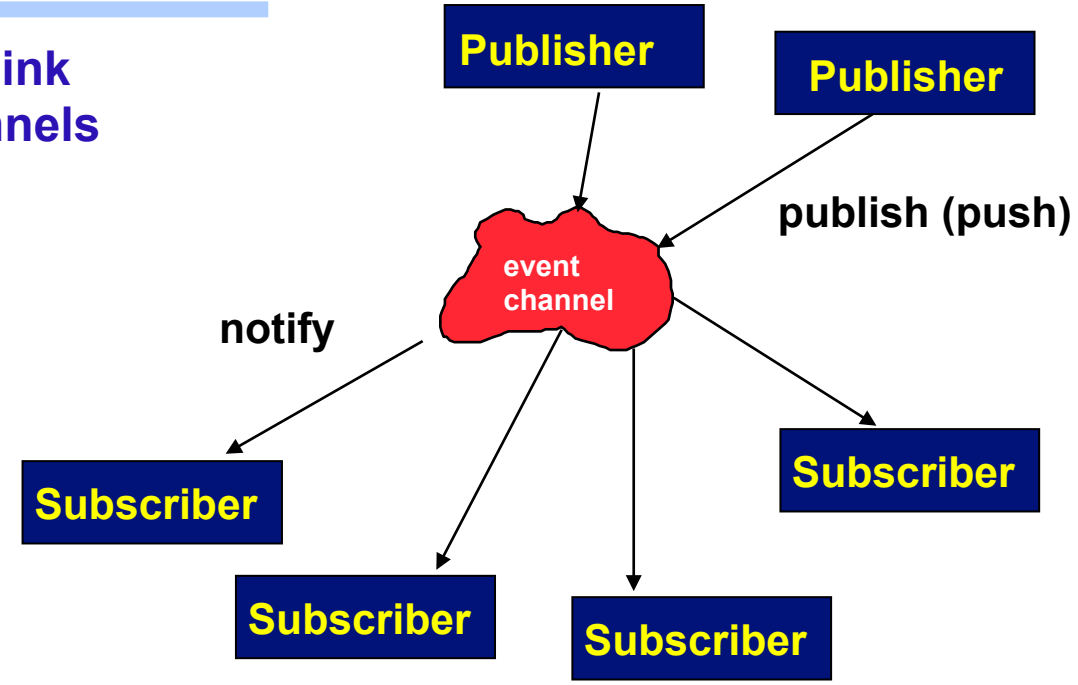


managebility

goal ➡ **sharing information and co-ordinating acitvities**

The Publisher/Subscriber Model

Principle: Keep control local and link systems via event channels



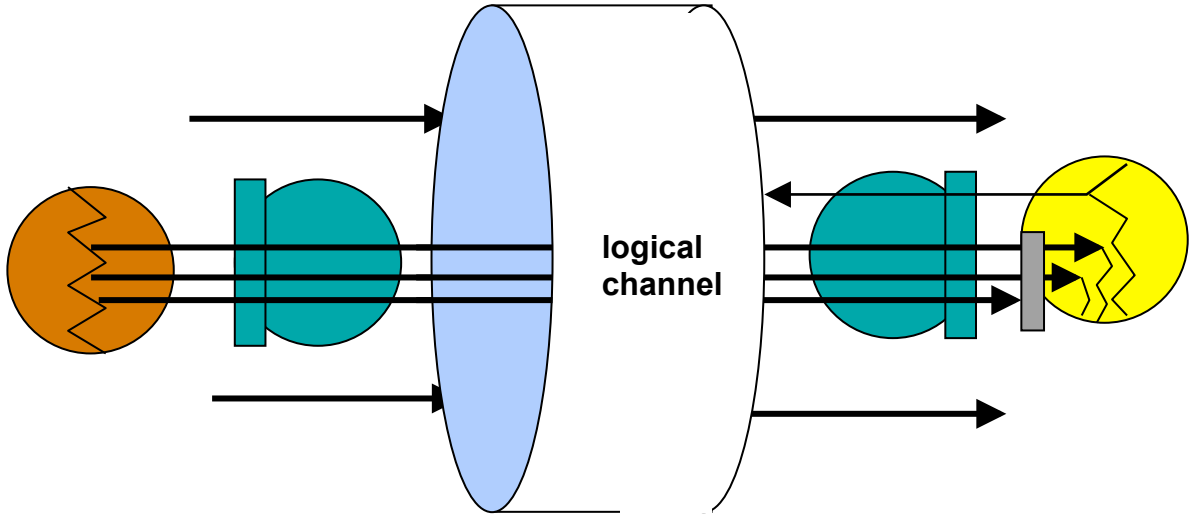
Information Bus *(Oki, Pfluegl, Siegel, Skeen)*
iBus *(Maffeis)*
Real-Time P/S *(Rajkumar, Gagliardi, Sha)*
NDDS *(Real-Time Innovations, Inc.)*
SIENA *(Carzanoga, Rosenblum, Wolf)*
Directed Diff. *(Intanagonwiwat, Govindan, Estrin)*

Many-to-many communication

Support for event-based spontaneous (generative) communication

Anonymous communication

Publish/Subscribe



Relation: many-to-many

Coupling:
Space: none
Flow: none
Time: none

Examples:
Information Bus
NDDS
Real-Time P/S
COSMIC
....
....

P/S in a smart sensor application

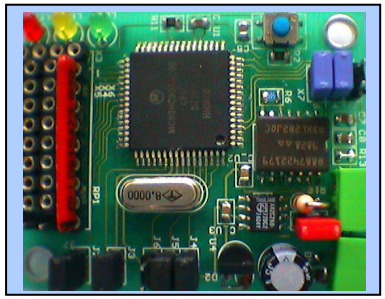
event: acceleration

publish (subject,attr., [acceleration]);

event: alarm

publish (subject, attr., [alarm]);

acceleration sensor



acceleration measurement & crash detection

configuration parameters
subscribe (subject-uid, ...)

Event Handler

network interface

<network address [acceleration]>

<network address [crash alarm]>

<network address, [config.params]>

Comparison

Communication model	Communication abstraction	Communication relation	Routing mechanism	Binding Time
message based	message	asymmetric	address	design time
Remote procedure Call	invocation	client-server	address	design time
Publish-Subscribe	event	Producer-consumer	contents/ subject	run time

Distributed system architecture