A Communication in a Distributed System
   - Shared memory vs message passing
   - Communication modes

B Communication Abstractions

Why Communication?
Cooperating processes need to communicate.
   - For synchronisation and control
   - To share data

In a Non-Distributed System:
Two approaches to communication:
   - Shared memory

Shared Memory:

![Shared Memory Diagram](image)
In a Non-Distributed System:
Two approaches to communication:
- Shared memory
  - Direct memory access (Threads)
  - Mapped memory (Processes)
- Message passing

Message Passing:

**Communication in a Distributed System**

Previous slides assumed a uniprocessor or a multiprocessor. In a distributed system (multicomputer) things change:

Shared Memory:
- There is no way to physically share memory

Message Passing:
- Over the network
- Introduces latencies
- Introduces higher chances of failure
- Heterogeneity introduces possible incompatibilities
MESSAGE PASSING

Basics:
- send()
- receive()

Variations:
- Connection oriented vs Connectionless
- Point-to-point vs Group
- Synchronous vs Asynchronous
- Buffered vs Unbuffered
- Reliable vs Unreliable
- Message ordering guarantees

Data Representation:
- Marshalling
- Endianness

COUPLING

Dependency between sender and receiver

Temporal: do sender and receiver have to be active at the same time?

Spatial: do sender and receiver have to know about each other? explicitly address each other?

Semantic: do sender and receiver have to share knowledge of content syntax and semantics?

Platform: do sender and receiver have to use the same platform?

Tight vs Loose coupling: yes vs no

COMMUNICATION MODES

Data-Oriented vs Control-Oriented Communication:

Data-oriented communication
- Facilitates data exchange between threads
- Shared address space, shared memory & message passing

Control-oriented communication
- Associates a transfer of control with communication
- Active messages, remote procedure call (RPC) & remote method invocation (RMI)

Synchronous vs Asynchronous Communication:

Synchronous
- Sender blocks until message received
  - Often sender blocked until message is processed and a reply received
- Sender and receiver must be active at the same time
- Receiver waits for requests, processes them (ASAP), and returns reply
- Client-Server generally uses synchronous communication

Asynchronous
- Sender continues execution after sending message (does not block waiting for reply)
- Message may be queued if receiver not active
- Message may be processed later at receiver’s convenience

When is Synchronous suitable? Asynchronous?
Slide 13

Transient vs Persistent Communication:

Transient
→ Message discarded if cannot be delivered to receiver immediately
→ Example: HTTP request

Persistent
→ Message stored (somewhere) until receiver can accept it
→ Example: email

Coupling?

Slide 14

Provider-Initiated vs Consumer-Initiated Communication:

Provider-Initiated
→ Message sent when data is available
→ Example: notifications

Consumer-Initiated
→ Request sent for data
→ Example: HTTP request

Slide 15

Direct-Addressing vs Indirect-Addressing Communication:

Direct-Addressing
→ Message sent directly to receiver
→ Example: HTTP request

Indirect-Addressing
→ Message not sent to a particular receiver
→ Example: broadcast, publish/subscribe

Coupling?

Slide 16

Combinations:

Examples?
Distributed Systems (COMP9243)

Lecture 4 (B): Communication

A. Communication in a Distributed System
B. Communication Abstractions
   - RPC and RMI
   - Message-oriented communication (MPI, MQ, event-based)
   - Group communication (multicast, gossip)