SDJ 3 - Distributed System

Friday, 24 August 2018 08.22

A system of sub-systems communicating over networks by sending messages.

Architectural Overview

Topics:

- Sub-systems
- Communication between Sub-systems
- Sending messages (how, which and etc.)

Problems with distributed systems

- Heterogeneity (Different platforms and languages).
- Security (If the system is secure).
- Scalability (How is performance impacted when number of users grow?).
- Failure Handling (When something fails, find a different way of accomplishing your goal).
- Concurrency
- Transparency (The user doesn't care about the distribution, and the developer does not care).
- Quality of Service
  - Reliability
  - Performance
  - Maintainability
  - Consistency - Is the System consistence?
  - Availability - When the System is down, but needs to be available

- A system is heterogeneous when different platforms and languages are used in different parts of the system
- Architectural impact
  - A heterogeneous system is best avoid whenever possible because it can be difficult to maintain, the same logic is repeated many places in the system, and it can be difficult for different parts of the team to communicate.
- Technological impact
  - The technologies that can handle heterogeneous systems are more flexible, but sometimes also more complex.
  - Beware of making too much complexity to cater to heterogeneity that you don't need.

Architectures

Thursday, 30 August 2018 13.08

- **Client / Server**
  - **Always online**
  - **Passive mode (waits for clients to send message)**
  - **Clients are active**

- **Communication paradigms** such RMI, Message queues

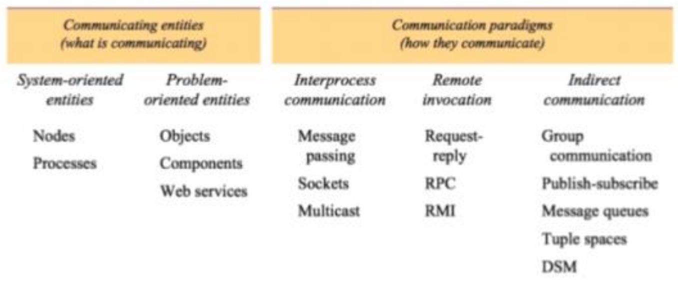
- **Peer-to-peer**

- **Tiers**

- **Communicating entities (objects, components, or services)**

- **Communication paradigms(request/reply, RPC, RMI, publish/subscribe, message queues, tuple spaces, distributed storage management)**
  - **Indirect**
    - **Decoupling**
      - **Time - Sender + Receiver not available at same time**
      - **Space - Sender doesn't know the location of the receiver**
    - **Remote Invocation**
      - **Request / Reply**
      - **Remote Method Invocation**
      - **Remote Procedure Call**

Figure 2.2
Communicating entities and communication paradigms



Communicating entities (what is communicating)		*Communication paradigms (how they communicate)*		
System-oriented entities	*Problem-oriented entities*	*Interprocess communication*	*Remote invocation*	*Indirect communication*
Nodes	Objects	Message passing	Request-reply	Group communication
Processes	Components	Sockets	RPC	Publish-subscribe
	Web services	Multicast	RMI	Message queues
				Tuple spaces
				DSM

In computer science, **marshalling or marshaling** is the process of transforming the memory representation of an object to a data format suitable for storage or transmission, and it is typically used when data must be moved between different parts of a computer program or from one program to another.