University of Freiburg, Germany
Department of Computer Science

Distributed Systems

Chapter 2 System Models

Christian Schindelhauer

07. May 2014

2.1: Introduction

Difficulties and threats to distributed systems

- Widely varying modes of use
  millions of accesses to a web-page
  multimedia access versus e-mail
- Wide range of system environments
  heterogeneous hardware, operating systems and networks
- Internal problems
  non-synchronized clocks
  conflicting data updates
  software/hardware failures
- External threats
  attacks on data integrity and security
  denial of service

2.2: Architectural Models

Description of the general structure of a DS

- Placement of the components
- interrelationship between components

Processes may be classified as

- server processes
- client processes
- peer processes

Usually, variations of these classifications are used

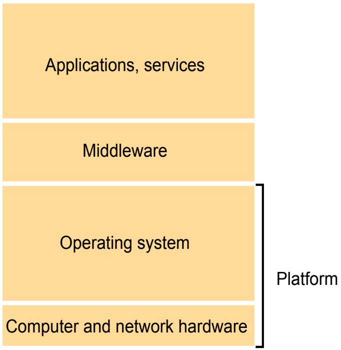
2.2.1: Software Layers

Platform

- Lowest-level hardware and software layers
- Provide services to the layer above
- E.g. Intel x86/Windows, Intel x86/Solaris, Intel x86/Linux

Middleware

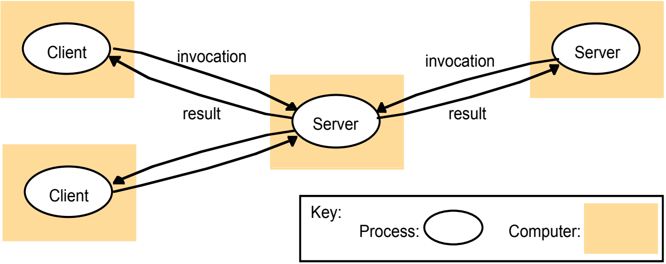
- Layer of software which masks the heterogeneity
- Useful building blocks for the construction of software componetns
- E.g. CORBA, Java RMI, web services, Microsoft DCOM, ISO/ITU-T RM-ODP



2.2.2: System Architectures

Client-Server

- Prevalent architecture
- Server process and client processes
- E.g. Web servers with database, search engines using web crawlers



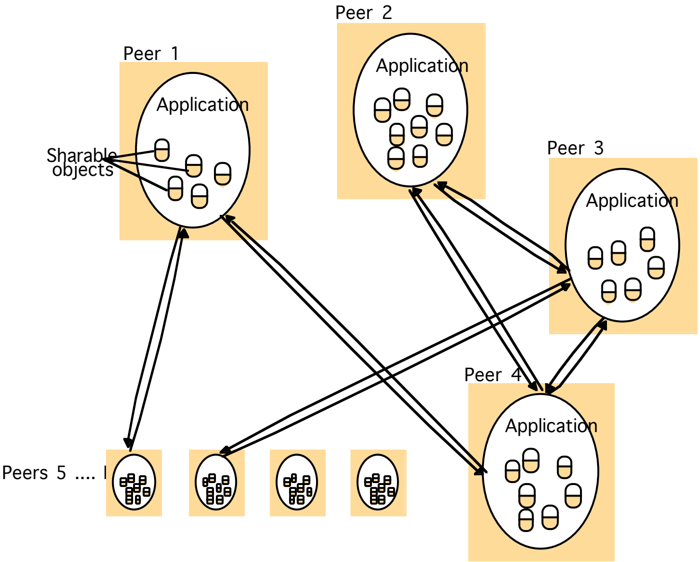
from *Distributed Systems – Concepts and Design*, Coulouris, Dollimore, Kindberg

2.2.2: System Architectures

Peer-to-Peer

- All processes play similar roles
- Interacting as peers (equals)
- Large number of peer processes on separate computers
- Individual servers hold only a small quantity
- E.g. File-sharing, Skype

Peer-to-Peer Architecture



from *Distributed Systems – Concepts and Design*, Coulouris, Dollimore, Kindberg

2.2.2: System Architectures: Variations

- Services provided by multiple servers (based on replicas)
  e.g. Sun NIS (Network Information Service)
- Proxy server and caches
- Mobile code
  e.g. applets
- Mobile agent
  a running program (code and data) that travels from computer to another one
- Network computers
  downloads OS from remote file server; also files are managed there
- Thin clients
  an graphical interface to a remote computer system,
  e.g. terminal to mainframe computer
- Mobile devices and spontaneous interoperation
  e.g. smart phones interacting using GSM, UMTS, Bluetooth

2.2.2: System Architectures: Design Requirements

- Performance issues
  - Responsiveness
  - Throughput
  - Balancing computational loads
- Quality of service
  - Reliability
  - Security
  - Performance
- Dependability issues
  - Correctness
  - Security
  - Fault tolerance

2.3.1: Interaction Model

Performance of communication channels

- Delay (latency)
  includes time for transmission, accessing the network, time by the operation systems
- Bandwidth
  number of bits that can be transmitted in a given time
- Jitter
  variation of the delay

Computer clocks

- clock drift rate
  relative amount that a computer clock differs from a perfect clock

2.3.1: Interaction Model

Synchronous Distributed Systems [Hadzilacos, Toueg, 1994]

- the time to execute each step of a process has known lower and upper bounds
- each message transmitted over a channel is received within a known bounded time
- each process has a local clock whose drift rate has a known bound

Asynchronous Distributed System

No bounds on

- process execution speeds
- message transmission delays
- clock drift rates

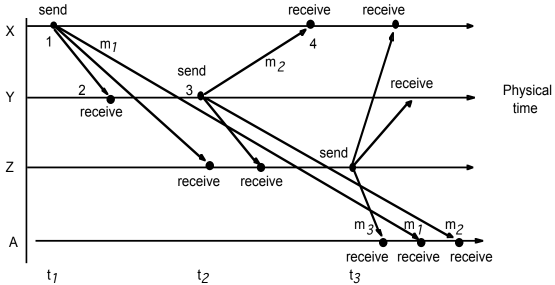
2.3.1: Interaction Model

Event ordering

1. X sends a message with the subject: *Meeting*
2. Y and Z reply, send a message with subject: *Re: Meeting*

User A's inbox:

Item	*From*	*Subject*
23	Z	Re: Meeting
24	X	Meeting
25	Y	Re: Meeting



2.3.2: Failure Model

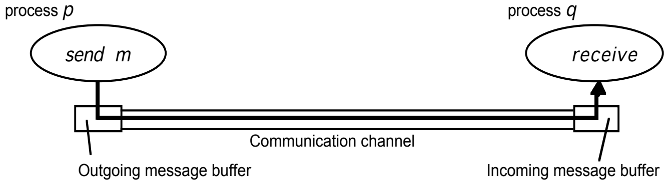
- Process omission failures
  - e.g. crash: can only detected by timeouts
  - e.g. fail-stop: detected crash
- Arbitrary (Byzantine) failures
  - worst possible failure: anything can happen
  - omits steps, takes unintended processing steps, returns wrong values, corrupted messages . . .
  - are rare
  - check sums can detect corrupted messages
  - message sequence number can detect omitted data

2.3.2: Failure Model

- Timing failures
  - internal clock too late or too early
  - process is too slow or to fast
  - messages take longer than wanted
- Masking failures
  - A service masks a failure by hiding it or by converting it into a more acceptable type of failure

2.3.2: Failure Model

- Communication omission failures
  - *dropping messages*: lost messages on the communication channel
  - *send-omission failure*: between send process and outgoing buffer
  - *receive-omission failure*: between incoming buffer and receive process
- Reliability of one-to-one communication
  - validity: any message in the outgoing buffer is eventually delivered o the incoming message buffer
  - integrity: the message received is identical to the one sent, no messages are delivered twice



from *Distributed Systems – Concepts and Design*, Coulouris, Dollimore, Kindberg

2.3.2: Failure Model

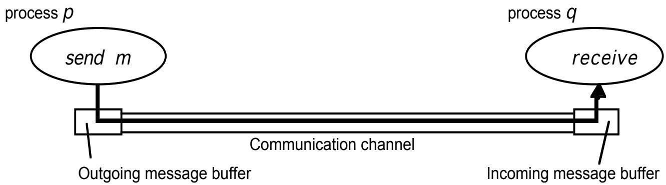
Defeated army problem

- Two confederated armies on two hills separated by the enemy army in the valley
- Dark Blue and Blue communicate via messengers

Problem: In the asynchronous model Dark Blue **cannot distinguish** whether

- Blue has been attacked and defeated by Red **or**
- the messenger with the „*everything is fine*" message from Blue is late.

.



from *Distributed Systems – Concepts and Design*, Coulouris, Dollimore, Kindberg

2.3.2: Failure Model

Agreement Problem

- Two confederated armies on two hills separated by the enemy army in the valley
- Dark Blue and Blue communicate via messengers.
- Red can delete any message (by killing the messenger)
- Dark Blue and Blue want to agree on whether to attack Red the next morning **or not**

Problem:

Red can prevent Dark Blue and Blue from an agreement by erasing the right messages.

2.3.2: Failure Model: Agreement Problem

2.3.2: Failure Model

Omission and Arbitrary Failures

Class of failure	Affects	Description
Fail-stop	Process	Process halts and remains halted. Other processes may detect this state.
Crash	Process	Process halts and remains halted. Other processes may not be able to detect this state.
Omission	Channel	A message inserted in an outgoing message buffer never arrives at the other end's incoming message buffer
Send-omission	Process	A process completes a *send*, but the message is not put in its outgoing message buffer.
Receive-omission	Process	A message is put in a process's incoming message buffer, but that process does not receive it.
Arbitrary (Byzantine)	Process or channel	exhibits arbitrary behavior: sends/transmits arbitrary message at arbitrary times, omissions, process may stop or may take an incorrect step

2.3.2: Failure Model

Timing Failures

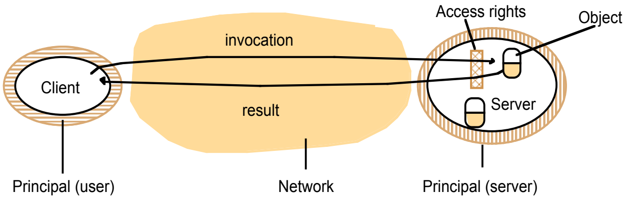
Class of failure	Affects	Description
Clock	Process	Process's local clock exceeds the bounds on its rate of drift from real time
Performance	Process	Process exceeds the bounds on the interval between two steps.
Performance	Channel	A message's transmission takes longer than the stated bound.

2.3.3: Security Model

The security of a distributed system

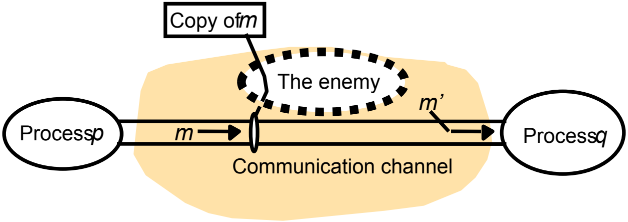
can be achieved by securing the processes and the interaction channels and by protecting the objects they encapsulate against unauthorized access.

- Protecting objects
  - access rights
  - an authority (user or process), called *principal*, grants the access to the objects
- securing processes and interactions
  - messages are exposed to attacks
  - processes expose their interfaces
  - enable invocations



2.3.3: Security Model: The enemy

- threats to processes
  e.g. IP lacks the reliable knowledge of the source of messages
  - Servers, e.g. mail-server delivers e-mail to attacker
  - Clients, e.g. fake GSM radio station captures secret phone calls
- threats to communication channels
  - enemy copies, alters, injects messages
  - enemy saves copies of messages and replays them later
  - such attacks can be defeated by the use of secure channels
- denial of service



from *Distributed Systems – Concepts and Design*, Coulouris, Dollimore, Kindberg

2.3.3: Security Model: Defeating Security Threats

- Cryptography: the science of keeping messages secure
  - symmetric encryption
  - public-key encryption
  - challenge-response protocols
- Authentication
  - shared secrets
  - public-key encryption
- Secure channels
  - process know reliably the identity of the principle
  - ensure privacy and integrity of the data
  - include physical or logical time stamps
- Other threats: denial of service and mobile code

End of Section 2