Distributed Systems
Lecture notes

Prof. Dr. Katinka Wolter
Freie Universität Berlin
Institut für Informatik
Takustraße 9
14195 Berlin, Germany

16. April 2015

1 Introduction (week 1)

Organisational issues:

- 2 SWS lecture, 2 SWS tutorial, 5 ECTS
- planned is extension with practical part to 10 ECTS
- every group has to implement one distributed algorithm
- programming language TBD

Literature: We will use the following books

- Tanenbaum, van Steen Distributed Systems for the systems part
- Vijay Garg Elements of Distributed Computing for the distributed algorithms
- Coulouris, Distributed Systems as accompanying book.

Examples of distributed systems:

- web search, google
- finance and commerce
- information systems, flickr, youtuve, google scholar
- health
- education
- transport and logistics
- science
- environmental management (sensor networks, desaster warning systems)

Trends in distributed systems

- small nodes, energy awareness (sensor networks)
- mobile wireless networks
- distributed multi media (streaming)
- Cloud-utility computing
- sharing of
  - application
  - storage
  - computation

Resource sharing:

services manage resources, present them to users, perform access control, etc.

client invokes an operation on the server

⇒ remote invokation

Client — server, are both processes

active versus passive

server can be client to the server at the next stage

Many distributed systems fit the client-server model (www, email, network printers). Others are rather P2P systems (sensor networks,...).

Challenges when building distributed systems:

1. Heterogeneity, variety and difference. Applies to
   - network
   - computer hardware
   - operating systems
   - programming languages
   - implementation by different developers.
2. Openness
   - system is characterised by its interfaces
   - provision of uniform communication
   - heterogeneous hardware, conformity in interaction
3. Security, very important issue, not part of this course. Not solved are problems of
   - denial of service attack
   - security of mobile code, and others.
4. Scalability
   - controlling hardware costs
   - controlling performance loss

  - preventing software resources from running out
  - avoiding performance bottleneck

5. Failure handling (Fault-Tolerance)
   - Detecting failures
   - Masking failures (retransmit, redundancy)
   - Tolerating failures (design for FT)
   - Recovery from failures
   - Redundancy (routing, duplication, data replication)
   - Availability must be high!
6. Concurrency ⇔ Consistency under concurrent requests
7. Quality of Service, what does the user see?

Definition 1 *Distributed System A distributed system is a collection of* independent *components (computers) that appears to its users as a* single*, coherent system.*

Characteristics are:

- autonomous components
- communication is hidden
- organisation is hidden

A distributed system can be a high-performance mainframe or a sensor network. It is a heterogeneous system that offers a homogeneous look and interfaces.

Objectives of a distributed system are to provide resources to its users. It

- shares resources in a controlled, fair and efficient way (printer, storage, computing, ..)
- grants access to the resources.

The distributed system connects users and resources.

1.1 Transparency

Transparency is an important property through which the quality of a distributed system can be assessed. It hides the fact that processes and resources are physically distributed.

Types of transparency:

- Access, hide difference in representation and how a resource is accessed.
- Location, hide resource location.
- Migration, hide movement of resource.
- Relocation, hide movement of resource during usage.
- Replication, hide that resource is being replicated.
- Concurrency, hide that resource is being used by several users.
- Failure, hide that resource has failed and is being repaired.

Transparency is desirable, but not always possible. There is a tradeoff between transparency and performance (masking failure through retry slows a resource down). Location transparency for printer not really desirable.

1.2 Scalability

Scalability is an important property of a distributed system.

It must be

- scalable in size (no. of nodes) (easy)
- scalable in geographical spread (more difficult)
- scalable in administration (most difficult) - (administration by several organisations).

It comes at a performance cost.

Problems for scalability:

- centralised services
- centralised data
- centralised algorithms.

Properties of decentralised algorithms:

1. no machine has global knowledge
2. machines decide based on local knowledge
3. failure of machine does not ruin the algorithm
4. no assumption of global clock.

Scaling techniques:

- only use asynchronous communication, shift code from server to client (Java applets, Java script)
- distribution, split components (DNS for subdomains)
- replication of components.

Pitfalls (wrong assumptions)

1. reliable network
2. secure network
3. homogeneous network
4. constant topology
5. zero latency
6. infinite bandwidth
7. zero transport cost
8. one administrator!

2 Distributed System Architectures (week 2)