Distributed Systems

MTAT.08.009

*eero.vainikko@ut.ee*

Course homepage (http://courses.cs.ut.ee/2016/ds/fall)

Fall 2016
2 Practical information

**Teachers:** Eero Vainikko, Amnir Hadachi, Artjom Lind, Oleg Batrašev

**Tutors:** Careelika Liisi Kuik <clk@ut.ee>, Anders Martoja <martoja.anders@gmail.com>

**Lectures:** WED 14:15, Liivi 2 - 405

**Problem solving classes:** Computer labs at **Paabel** (Ülikooli 17):
1. group **FRI 10:15 room 218** (Amnir Hadachi)
2. group **MON 10:15 room 218** (Artjom Lind)
3. group **MON 12:15 room 218** (Artjom Lind)
4. group **FRI 10:15 room 219** (Artjom Lind)

Lectures: 32h; Computer labs: 32h; Independent work: 92h

**Final grade:**

1. **Homework (40%)**

2. **Exam (40%) = Mid-Term Exam (15%) + Final Exam (25%)**

3. **Active participation** at lectures (10%) and seminars (10%)
Active participation at lectures

How to claim that you were active? 3 ways – the choice is yours:

1. **You ask a good question and/or participated in discussion** during the lecture – come to the lecturer right after the lecture and say your name – activity counted!

2. Answer a quiz question at the end of lecture! (technical details in the end of the lecture) – activity counted after the quiz gets checked!

3. Devize and **insert at least one question** into the online course study-questionary!

   (a) Questions should be about or related to the themes covered **during the last lecture**
4 Practical information

(b) Link to the form for inserting the questions

(c) study-questionary (updated at least in every 5 minutes)

(d) You insert:
   i. Your name (seen only by course teachers)
   ii. Question text
   iii. Hint for an answer
   iv. Answer to the question
   v. optional comment (seen only by course teachers)

(e) Activity counted after the questions got checked by teachers

Active participation at Computer labs
   – described at first lab
0 Introduction

0.1 Syllabus

0.1.1 Lectures:

0. Introduction to the course

• Characterization of distributed systems
• Networking and internetworking
• Interprocessor communication
• System models
• Remote invocation
• Indirect communication

• Operating systems support
• Distributed objects and components
• Web services
• Peer-to-peer systems
• Security
• Distributed files systems
• Name services
• Time and global state; Coordination and agreement; Distributed transactions
• Designing distributed systems: Google case study
• Big Data paradigm

0.1.2 Computer labs

Very important to be present! (for your own sake... :-)

• **Seminars** every second week followed by **seminar tasks** every following week
  – seminars – step-by-step guides
  – seminar tasks – hands-on exercises to be submitted to collect feedback from tutors

• **Bring your laptop!**

• Python 2.7
0.1.3 Homework

- 2 programming tasks with separate deadlines

0.1.4 Exam

- Course materials studied at Lectures and Discussion Seminars

- Dates
  
  - Mid-Term Exam: **19 or 26. October 2016**
  
  - Final Exam:
    
    * **A. ?? December 2016 at 10:15**
    * **B. ?? January 2016 at 10:15**
0.2 Literature

0.2.1 Textbook


0.2.2 Additional reading

- POSIX thread programming
- Pthreads API specification
- Synchronizing threads in Java
- Java tutorial by SUN
- Fundamentals of multithreading
- Introduction to Java threads
9 Introduction

- Flick: The Flexible IDL Compiler Kit
- Java IDL Technology
- ONC+ Developer’s Guide
- Microsoft Interface Definition Language (MIDL)
- Introduction to Java RMI
- Java RMI Tutorial
- Annotated WSDL Example
- The NFS Version 4 Protocol
- Microsoft SMB Protocol and CIFS Protocol Overview

0.2 Literature

- Coda File System
- Remote Filesystems slides
- WebDAV Resources
- Understanding Replication in Databases and Distributed Systems (PDF)
- Linux Virtual Server for Scalable Network Services (PDF)
- NFS Security (PDF)
- Executive Summary: Computer Network Time Synchronization
1 Characterization of distributed systems

1.1 Introduction

What is a Distributed System?

A distributed system is one in which components located at networked computers communicate and coordinate their actions only by passing messages.

A distributed system consists of a collection of autonomous computers linked by a computer network and equipped with distributed system software. This software enables computers to coordinate their activities and to share the resources of the system hardware, software, and data.

---

1 Textbook Chapter 1
How to characterize a distributed system?

- concurrency of components
- lack of global clock
- independent failures of components

Leslie Lamport :-)  
*You know you have a distributed system when the crash of a computer you’ve never heard of stops you from getting any work done!*

Prime motivation: **to share resources**
What are the challenges?

- heterogeneity of components
- openness
- security
- scalability – the ability to work well when the load or the number of users increases
- failure handling
- concurrency of components
- transparency
- providing quality of service
### 1.2 Examples of distributed systems

**Distributed Systems application domains connected with networking:**

<table>
<thead>
<tr>
<th>Domain</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Finance and commerce</td>
<td>eCommerce e.g. Amazon and eBay, PayPal, online banking and trading</td>
</tr>
<tr>
<td>The information society</td>
<td>Web information and search engines, ebooks, Wikipedia; social networking: Facebook and MySpace</td>
</tr>
<tr>
<td>Creative industries and entertainment</td>
<td>online gaming, music and film in the home, user-generated content, e.g. YouTube, Flickr</td>
</tr>
<tr>
<td>Healthcare</td>
<td>health informatics, on online patient records, monitoring patients</td>
</tr>
<tr>
<td>Education</td>
<td>e-learning, virtual learning environments; distance learning</td>
</tr>
<tr>
<td>Transport and logistics</td>
<td>GPS in route finding systems, map services: Google Maps, Google Earth</td>
</tr>
<tr>
<td>Science</td>
<td>The Grid as an enabling technology for collaboration between scientists</td>
</tr>
<tr>
<td>Environmental management</td>
<td>sensor technology to monitor earthquakes, floods or tsunamis</td>
</tr>
</tbody>
</table>
1.2 Examples of distributed systems

1.2.1 Web search

**An example:** Google

Highlights of this infrastructure:

- physical infrastructure
- distributed file system
- structured distributed storage system
- lock service
- programming model

1.2.2 Massively multiplayer online games (MMOGs)

**Examples**

- EVE online – *client-server architecture*
- EverQuest – more distributed architecture
- Research on completely decentralized approaches based on *peer-to-peer (P2P) technology*
15 Characterization of Distributed Systems

1.2 Examples of distributed systems

1.2.3 Financial trading

- distributed event-based systems

- Reuters market data events

- FIX events (events following the specific format of the Financial Information eXchange protocol)
1.3 Trends in distributed systems

- emergence of pervasive networking technology
- emergence of ubiquitous computing coupled with the desire to support user mobility
- multimedia services
- distributed systems as utility

1.3.1 Pervasive networking and the modern Internet

*networking has become a pervasive resource and devices can be conected at any time and any place*
A typical portion of the Internet:
1.3.2 Mobile and ubiquitous computing

- laptop computers

- handheld devices (mobile phones, smart phones, tablets, GPS-enabled devices, PDAs, video and digital cameras)

- wearable devices (smart watches, glasses, etc.)

- devices embedded in appliances (washing machines, refrigerators, cars, etc.)
Portable and handheld devices in a distributed system

- mobile computing
- location/context-aware computing
- ubiquitous computing
- spontaneous interoperation
- service discovery
1.3.3 Distributed multimedia systems

- live or pre-ordered television broadcasts
- video-on-demand
- music libraries
- audio and video conferencing
1.3.4 Distributed computing as a utility

- Cluster computing
- Grid computing
- Cloud computing
1.4 Sharing resources

What are the resources?

- Hardware
  - Not every single resource is for sharing
- Data
  - Databases
  - Proprietary software
  - Software production
  - Collaboration
Sharing Resources

- Different resources are handled in different ways, there are however some generic requirements:
  - Namespace for identification
  - Name translation to network address
  - Synchronization of multiple access
1.5 Challenges

1.5.1 Heterogeneity

Heterogeneity – variety and difference in:

- networks
- computer hardware
- OS
- programming languages
- implementations by different developers
Middleware

- *middleware* – software layer providing:
  - programming abstraction
  - masking heterogeneity of:
    * underlying networks
    * hardware
    * operating systems

Heterogeneity and mobile code

*Mobile code* – programming code that can be transferred from one computer to another and run at the destination (Example: think Java applets)

*Virtual machine approach* – way of making code executable on a variety of host computers – the compiler for a particular language generates code for a virtual machine instead of a particular hardware order code.
1.5.2 Openness

**OPENNESS** of a:

- **open computer system** - can the system be extended and reimplemented in various ways?

- **open distributed system** - can new resource-sharing services be added and made available for use by variety of client programs?
An open system – key interfaces need to be published!

An open distributed system has:

- uniform communication mechanism
- published interfaces to shared resources

Open DS - heterogeneous hardware and software, possibly from different vendors, but **conformance of each component to published standard** must be tested and verified for the system to work correctly.
1.5.3 Security

1. *Confidentiality* – protection against disclosure to unauthorized individuals

2. *Integrity* – protection against alteration or corruption

3. *Availability* – protection against interference with the means to access the resources

Security challenges not yet fully met:

- *denial of service attacks*
- *security of mobile code*
1.5.4 Scalability

– the ability to work well when the system load or the number of users increases

Challenges with building scalable distributed systems:

• Controlling the cost of physical resources

• Controlling the performance loss

• Preventing software resources running out (like 32-bit internet addresses, which are being replaced by 128 bits)

• Avoiding performance bottlenecks

  – Example: some web-pages accessed very frequently – remedy: caching and replication
1.5.5 Failure handling

Techniques for dealing with failures

- Detecting failures

- Masking failures
  1. messages can be retransmitted
  2. disks can be replicated in a synchronous action

- Tolerating failures

- Recovery from failures
• Redundancy

  – redundant components

    1. at least two different routes
    2. like in DNS every name table replicated in at least two different servers
    3. database can be replicated in several servers

Main goal: **High availability** – measure of the proportion of time that it is available for use
1.5.6 Concurrency

Example: Several clients trying to access shared resource at the same time

Any object with shared resources in a DS must be responsible that it operates correctly in a concurrent environment

Discussed in Chapters 7 and 17 in the book

1.5.7 Transparency

*Transparency* – concealment from the user and the application programmer of the separation of components in a Distributed System for the system to be perceived as a whole rather than a collection of independent components
Access transparency – access to local and remote resources identical

Location transparency – resources accessed without knowing their physical or network location

Concurrency transparency – concurrent operation of processes using shared resources without interference between them

Replication transparency – multiple instances seem like one

Failure transparency – fault concealment

Mobility transparency – movement of resources/clients within a system without affecting the operation of users or programs

Access and Location transparency – together called also Network transparency
1.5.8 Quality of service

Main nonfunctional properties of systems that affect *Quality of Service (QoS)*:

- reliability
- security
- performance

Time-critical data transfers

Additional property to meet changing system configuration and resource availability:

- adaptability
1.6 Case study: The World Wide Web

CERN 1989

*hypertext* structure, *hyperlinks*

- Web is an open system
- Content standards freely published and widely implemented
- Web is open with respect to types

Figure 1.7 Web servers and web browsers
1.6 Characterization of Distributed Systems

**Case study: The World Wide Web**

**HTML**

HyperText Markup Language  [www.w3.org](http://www.w3.org)

**URL-s**

Uniform Resource Locators (also known as URI-s - Uniform Resource Identifiers)

http://servername[:port][/pathName][?query][#fragment]

**HTTP**

- Request-reply interactions
- Content types
- One resource per request
- Simple access control
- Dynamic pages
Web services

HTML – limited – not extensible to applications beyond information browsing

The Extensible Markup language (XML) designed to represent data in standard, structured, application-specific way

XML data can be transmitted by POST and GET operations

- Semantic web – web of linked metadata resources

Web as a system – main problem – the problem of scale
NB! Home task for next week: Read / work through the Textbook\textsuperscript{2} Chapter 3 – Networking and internetworking

Some questions to address:

- What is the difference between streaming and packet transmission?
- What are the Internet protocols?
- TCP and UDP
  - what are UDP features and functionalities?
  - what are TCP features and functionalities?
  - when to use UDP and when to use TCP?