Summary and examinable material

INF5040/9040 autumn 2009

lecturers: Frank Eliassen, Roman Vitenberg

Learning goals

- Provide a basic understanding of
  - fundamental principles, concepts and state-of-the-art
  - key technologies for realising distributed interactive systems of the future
- Gain practical experience of crafting a distributed application with a state-of-the-art platform
- Provide knowledge about today's challenges to open distributed processing technology, including
  - Communication paradigms
  - Multimedia
  - Mobility
  - Fault-tolerance

Overview examinable material

From Coulouris et al (1/2)

- Chap 1: all
- Chap 2: all
- Chap 4: 4.1, 4.3, 4.4
- Chap 5: all except 5.3.1 (5.5 cursory knowledge only)
- Chap 6: 6.4, 6.5, only 6.4.3 subject to exam
- Chap 10: all
- Chap 11: all except 11.3 and 11.6
- Chap 12: all

From Coulouris et al (2/2)

- Chap 13: 13.1 – 13.4
- Chap 14: 14.1 – 14.4.1:
  (14.5, 14.6 cursory knowledge)
- Chap 15: 15.1 - 15.3
- Chap 16: all except 16.5
- Chap 17: 17.1 – 17.5
- Chap 19.1-19.4, 19.6
- Chap 20: all except 20.2.6, 20.3.1-20.3.4, and details of code examples

From Tanenbaum and van Steen

- Chapter 4: 4.1.2, 4.3-4.5
- Chapter 6: All except 6.1 and 6.4
- Chapter 7: All except 7.3
- Chapter 8: 8.4 (except 8.4.2), 8.5 (except 8.5.2)
- Chapter 10.2.1, 10.3.1
- Chapter 12: 12.1-12.4, 12.6
- Chapter 13: 13.4.1

Other ...

- G. T. Heineman, W.T. Councill: *Component-based Software Engneering - Putting the Pieces Together*, 2001. Addison Wesley. Ch. 1 & 3.
- P. Eugster et. al.: *The Many Faces of Publish/Subscribe*.
- P. Eugster et. al.: *Epidemic Information Dissemination in Distributed Systems*
- All lectures and lecture slides

Mandatory programming assignments

- Knowledge related to the mandatory programming assigments *can* be tested during the exam

On examination 14. and 15. December

- The exam will be conducted in the following way:
  - Each student will be assigned a time and place (room) to meet.
  - All students are required to meet at 08:00 on their exam day in front of the IFI-reception, for name checking and receiving the exact times for the exam
  - You will be given a set of written questions that you have 30 minutes to prepare answers to (e.g., by making notes, diagrams, keywords or similar). You may not use any kind of aid (written or otherwise).
  - After 30 minutes you will be brought to the examination room, where you will be allowed 15 minutes to present your answers to the written questions, followed by an additional 15 minutes of questioning from other parts of the examinable material.
  - Bring picture id!
  - Ifiadm will prepare a schedule for the exam (time and place for each student).
    - If you have particular wishes wrt your scheduling, contact IFI studadm (Mozhdeh Sheibani Harat)

Summary of lectures

Introduction and design

- What is a distributed system
- Implications of distributed systems:
  - partial failures, unreliable and unsecure comm, ...
- Requirements leading to distributed systems:
  - resource sharing, scalability, fault tolerance, ...
- Distribution transparency
- Distributed systems organized as middleware
- Design of distributed objects is different from design of programs where all objects are in the same process (pitfalls)

System models

- Two types of system models
- **Arcitecture models**: defines the components of the system, the way they interact, and the way they are deployed in a network of computers
  - architectural styles
  - client-server models (many variants)
  - peer processes (P2P)
  - spontaneous networks (mobility)
- **Fundamental models**: formal description of the properties that are common to all architecture models
  - interaction models (synchronous vs asynchronous, event ordering)
  - failure models
  - security models (not covered in this course, but see e.g., INF3190)

Distributed objects and object-based middleware

- Distributed objects execute in different processes.
  - remote interfaces allow an object in one process to invoke methods of objects in other processes located on the same or on other machines using Remote Object References
- Object-based distribution middleware:
  - middleware that models a distributed application as a collection of interacting distributed objects (e.g., CORBA, Java RMI)
  - some middleware (as CORBA) allow objects in the same application to be implemented in different programming languages based on a Common Object Model, a separate ("programming language indenpendent") IDL, and a set of language bindings.

Object interaction using RMI

- Request-response protocols (failure model, recovery)
- Reliability semantics of RMI (under partial failure)
  - maybe, at-least-once, at-most-once
  - Reliability of RMI is at best “at-most-once”
- Implementation of RMI
  - proxies, skeletons, dispatcher
  - interface processing, object name resolution, object binding and locating, activation
- Object servers and multi-threading
  - Object servers are tailored to support distributed objects (activation policies).
  - Muti-threading can in some cases be used to increase the throughput (method calls/time unit) if, e.g., I/O is the bottleneck, and provide better resource utilization.
- Principles of CORBA
  - Clients may invoke methods of remote objects without worrying about: object location, programming language, operating system platform, communication protocols, or hardware.
- Principles of Java RMI
  - Similar to CORBA but limited to a Java environment

Software components and distributed systems

- Rationale for components (reuse, programming by assembly)
- Components
  - Programming according to LEGO-principle
  - Contractually specified interfaces and composition
  - Support for connection oriented programming
- Implicit and explicit middleware (separation of concerns)
- Component architecture
  - Unit of deployment and composition (reuse), explicit dependencies
  - Contractually specified interfaces between components and application servers
  - Realizes ”implicit middleware” (containers)
  - Java: EJB, CORBA: CCM, Microsoft: COM+/.NET
- Basis for third generation middleware (& self-adaptive software)

Communication paradigms

- Communication properties
  - Persistence, synchrony, time and space decoupling
- MPI
  - Motivation, main ideas, and uses
- MOM
  - Architecture, routing, applications
- Publish-subscribe
  - Architecture and properties
  - Subscription semantics and content-based routing
- Overview of overlay-based multicast
  - Desired properties
  - Small-world phenomenon
- Epidemic-based dissemination
  - Concepts and ideas
  - Push-based and pull-based mechanisms
  - Applications

Time and coordination

- Logical time
  - Implementations: logical clocks & vector clocks
- The snapshot problem
  - Consistent global states
  - [Chandy,Lamport] protocol
- Distributed consensus
  - Known impossibility results
- Distributed mutual exclusion
  - Requirements
  - 3 Algorithms: centralised, ring-based, and [Ricart,Agrawala]
- Distributed leader election
  - Bully and ring-based algorithms

Distributed transactions

- Transactions
  - Motivation, interface, etc.
  - Desirable properties of a transaction (ACID)
- Distributed transactions
  - Atomic commitment problem
  - 2PC protocol
    - State diagram
    - Recovery
  - 3PC protocol
    - State diagram
    - Recovery

Replication

- Motivation and objectives
- Placement of replicas
- Propagation of updates among the replicas
  - Consistency guarantees
- Replication schemes
  - Active and passive replication
- Group communication
  - Group membership, view synchrony, ordered message delivery

Peer-to-peer

- P2P systems distribute processing load and network traffic between all nodes that participate in an distributed information system
- P2P systems are self-organising and independent of any central entity
- The efficiency critically depends on algorithms for distributing the data among a high number of nodes and subsequently locating the data
- P2P middleware is an application-independent software layer that implements a ”routing overlay”
- Case study and evaluation: Pastry

Mobile and ubiquitous computing

- Mobile and ubiquitous systems are volatile
- Applications in mobile and ubiquitous environments are
  - Integrated with the physical world through sensing and context awareness
  - Adaptive to changes in the physical circumstances by changing behavior
- Common system model for mobile and ubiquitous computing (and their subfields). Elements of the model
  - smart spaces
  - device model
  - volatile connectivity
  - spontaneous interoperation
    - role of discovery services,
    - interoperability: function-oriented vs data-oriented interfaces
- Sensors and sensor software architectures, wireless sensor networks, directed diffusion

Distributed multimedia systems

- Multimedia applications require mechanisms that enable them to handle large amounts of time dependent data
- Most important mechanisms: media synchronization and QoS/resource management
- QoS driven resource management
  - admission control (by mapping QoS requirements to resource needs)
  - scheduling function (making resources available (CPU, memory, bw, ..) when needed)
  - QoS models for streaming
- Streaming over the Internet
  - Compensating for quality degradation
    - Jitter-compensation (different buffering strategies)
    - Compression
    - Traffic shaping (leaky bucket, token bucket)
    - Media scaling (stream adaptation)
  - Continuous media distribution services
    - overlay vs P2P
    - in-network filtering

Web-based systems

- Communication architecture
  - Web proxies
  - Proxy-based caching (hierarchical and cooperative)
  - Server-side backend
    - Content-distribution networks
- Web Services
  - Applications
  - Architecture
    - URIs, WSDL, SOAP
  - Comparison with CORBA