Advanced Distributed Algorithms and Data Structures

Christian Scheideler
Institut für Informatik
Universität Paderborn
Advanced Distributed Algorithms and Data Structures

Lecture: Thu 2-4 pm, F2.211
Tutorial: Thu 10-11 am, F2.211 (starts 3rd week)

Website:

Modules and Grading:
• MuA Modules III.2.1, III.2.2 and III.2.4
• 50%: oral exam, 50%: software project
  both parts must be passed to pass the course

Prerequisites:
• basic knowledge in algorithms and data structures
• recommended: distributed algorithms and data structures course
Advanced Distributed Algorithms and Data Structures

Homework assignments:
• Weekly assignments each Thursday on the website (starting with next week)
• Theoretical and implementations

Slides and assignments: course website
Book recommendations: no book available (lecture is based on newest results)
Embedding into CS Curriculum

Bachelor I

Data Structures and Algorithms

Bachelor II

Distributed Algorithms and Data Structures

Master

Advanced Distributed Algorithms and Data Structures
Embedding into CS Curriculum

Advanced Distributed Algorithms and Data Structures

Master

Project Group: Design of a Trusted Communication Module
Advanced Distributed Algorithms and Data Structures

Goals:

1. Introduction to advanced concepts in distributed algorithms and data structures.
2. Introduction to important design methods.
3. Introduction to important analytical methods.
Introduction

Sequential Algorithms and Data Structures

Distributed Algorithms and Data Structures
Introduction

What are the basic problems for distributed algorithms and data structures?
Definition 1.1: A data structure is a certain way to organize data in a computer so that operations like, for example, search, insert, and delete are simple and effective to realize.

Simple examples:
- Lists

![Lists Diagram]
- Arrays

![Arrays Diagram]
Introduction

Basic view:

Data Structure

Operation 1
Operation 2
Operation 3
Introduction

Classical case: computer with one processor

Operations

Data structure

Storage
Introduction

Computer with several processors/cores:

![Diagram showing multiple processors connected to a data structure and storage]

Data structure

Storage
Introduction

Computer with several processors/cores:

Overlaps:
- access conflicts (correctness)
- performance problems (efficiency)
Introduction

Multiple computers:

Data structure

**Problem**: distribution of DS among computers
Introduction

Multiple computers:

Problem: distribution of DS among computers
Introduction

Multiple computers:

Basic problems:
• How to interconnect the computers?
• How to coordinate the management of the DS among the computers?
How to represent the connections between the computers?

A knows (IP address, MAC address, ... of) resp. has access authorization for B: network can send message from A to B

High-level view:
A knows B $\Rightarrow$ overlay edge $(A,B)$ from A to B (A $\rightarrow$ B)

Set of all overlay edges forms directed graph known as overlay network.
Introduction

**Problem:** find suitable overlay network for the computers / processes

Best topology depends on problem and context.
Still many problems left: link management, access control, synchronization, communication primitives, transactions, and various applications.
Introduction

Distributed Algorithms and Data Structures

Advanced Distributed Algorithms and Data Structures

Extensive study of overlay networks

Assumption: processes form a clique

Focus: strategies that are scalable, robust and secure.
Focus: strategies that are scalable, robust and secure (because participants might be faulty or adversarial, or might get attacked from outside!)
Central properties a system should satisfy:

- **Robustness**: availability
- **Security**: integrity and confidentiality
Introduction

Four Commandments:
1. You shall not sleep
2. You shall not fail
3. You shall not lie
4. You shall not leak
Introduction

Four Commandments:
1. You shall not sleep
2. You shall not fail
3. You shall not lie
4. You shall not leak

Measures against violations:
Quite challenging…

\{ availability, integrity, confidentiality \}
Introduction

Examples of violations:
1. You shall not sleep
   → asynchronicity
2. You shall not fail
   → message loss, process/link failures, Denial-of-Service attacks
3. You shall not lie
   → corrupted information/messages/protocols
4. You shall not leak
   → outsider (e.g., man-in-the-middle) or insider (e.g., through viruses) attacks
Introduction

Measures to be robust to asynchronicity:

Decoupling of time and flow

• **Time decoupling:** interacting processes do not need to be available at the same time (usually needs mediator)

• **Flow decoupling:** the execution of an action within a process should not depend on the availability of another process (no remote calls asking for immediate return values)
Introduction

Measures against message loss and failures:

• **Reactive approaches:** recovery from any illegal state (such systems known as self-healing / self-stabilizing)
• **Proactive approaches:** maintain availability even in faulty states (requires redundancy)

Measures against Denial-of-Service attacks:

• **Standard approach:** enforce confidentiality to make targeted DoS attacks hard
Introduction

Measures against corrupted messages and information:

• Algorithmic approach: use redundancy
• Cryptographic approach: integrity measures (encode messages and information so that correctness can be checked)

Measures against corrupted protocols:

• Hard and expensive (→ Byzantine general models)
Introduction

Measures against leaking:

• Algorithmic approach: secret sharing
• Cryptographic approach: use encryption

In the real world, leaking is hard to avoid.
Main problem: exposure!
Introduction

• **SPAM:**
  Anyone can send you an email and disseminate your email address.

• **DoS attacks:**
  Anyone who has your IP address can send you a message and freely disseminate your IP address.

• **Viruses:**
  Once a virus is in your system, it has access to all information in it.
Introduction

Owner consent and control:
• Clearly defined responsibilities
• Complete control over own info & resources

Least exposure:
• Not more knowledge than necessary
• Complete control over information flow

Self-recovery:
• Recovery from every possible state (as long as underlying layer is still in a legal state)

Decoupling:
• No synchronization necessary for primitives

Bachelor course: overlays
Introduction

Owner consent and control:
• Clearly defined responsibilities
• Complete control over own info & resources

Least exposure:
• Not more knowledge than necessary
• Complete control over information flow

Problem: current systems cannot enforce these rules (because they are too open, too complex)
Introduction

Owner consent and control:
• Clearly defined responsibilities
• Complete control over own info & resources

Least exposure:
• Not more knowledge than necessary
• Complete control over information flow

Not even suitable models and primitives available to rigorously study guidelines in the algorithms community. So viruses, trojans, and DoS attacks are usually ignored.

In this lecture: new approach
New approach: Trusted Communication Model (TCM)

- AL (Application Layer): large storage capacity and computational power, but potentially insecure
- TCL (Trusted Communication Layer): low storage capacity and computational power but can securely manage ports and keys and can securely execute basic primitives
Introduction

New approach: Trusted Communication Model (TCM)

- **AL**: can be invaded
- **TCL**: cannot be invaded or inspected

**Goal of TCL**: support AL in ensuring availability, integrity, and confidentiality
Introduction

ISO OSI-Model:

<table>
<thead>
<tr>
<th>Layer</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application Layer</td>
<td>HTTP</td>
</tr>
<tr>
<td>Presentation Layer</td>
<td>SMTP</td>
</tr>
<tr>
<td>Session Layer</td>
<td>NCP</td>
</tr>
<tr>
<td>Transport Layer</td>
<td>TCP/UDP</td>
</tr>
<tr>
<td>Network Layer</td>
<td>IP</td>
</tr>
<tr>
<td>Data Link Layer</td>
<td>Ethernet</td>
</tr>
<tr>
<td>Physical Layer</td>
<td>MAC</td>
</tr>
</tbody>
</table>

AL

TCL
Advanced distributed algorithms and data structures

Contents:
1. Introduction
2. Graph theory
3. Probability theory
4. Link primitives
5. TCM model and programming environment
6. Congestion control
7. Clock synchronization
8. Logical Clocks
9. Dynamic Overlay Networks
10. Broadcasting and Anycasting
11. Distributed Commit
12. Applications
Graph Theory

• Introduction to fundamental topologies

• Basic graph parameters
  (degree, diameter, expansion, …)
Contents:
1. Introduction
2. Graph theory
3. Probability theory
4. Link primitives
5. TCM model and programming environment
6. Congestion control
7. Clock synchronization
8. Logical Clocks
9. Dynamic Overlay Networks
10. Broadcasting and Anycasting
11. Distributed Commit
12. Applications
Probability Theory

- Random variable
- Expectation
- Variance
- Markov inequality
- Chernoff bounds
Advanced distributed algorithms and data structures

Contents:
1. Introduction
2. Graph theory
3. Probability theory
4. Link primitives
5. TCM model and programming environment
6. Congestion control
7. Clock synchronization
8. Logical Clocks
9. Dynamic Overlay Networks
10. Broadcasting and Anycasting
11. Distributed Commit
12. Applications
Link Primitives

Safe forms of

Introduction

Delegation

Fusion

Reversal
Advanced distributed algorithms and data structures

Contents:
1. Introduction
2. Graph theory
3. Probability theory
4. Link primitives
5. TCM model and programming environment
6. Congestion control
7. Clock synchronization
8. Logical Clocks
9. Dynamic Overlay Networks
10. Broadcasting and Anycasting
11. Distributed Commit
12. Applications
Processes can interconnect (using link primitives) and execute actions.

Types of actions:
- Triggered by a local/remote call:
  \[ \langle \text{name} \rangle(\langle \text{parameters} \rangle) \rightarrow \langle \text{commands} \rangle \]
- Triggered by a local state:
  \[ \langle \text{name} \rangle: \langle \text{predicate} \rangle \rightarrow \langle \text{commands} \rangle \]

All messages are remote action calls.

Example:

\[
\text{minimum}(x,y) \rightarrow \\
\quad \text{if } x < y \text{ then } m := x \text{ else } m := y \\
\quad \text{print}(m) \quad \text{no return command!}
\]

Action „minimum“ is executed whenever a request to call \text{minimum}(x,y) is processed.
TCM Model and Programming Environment

Processes can interconnect (using link primitives) and execute actions.

Types of actions:
- Triggered by a local/remote call:
  \[ \langle \text{name} \rangle (\langle \text{parameters} \rangle) \rightarrow \langle \text{commands} \rangle \]
- Triggered by a local state:
  \[ \langle \text{name} \rangle : \langle \text{predicate} \rangle \rightarrow \langle \text{commands} \rangle \]

All messages are remote action calls.

Example:

\[ \text{timeout}: \text{true} \rightarrow \]
\[ \text{print(} \text{"I am still alive!"} \text{)} \]

"true" ensures that action timeout is periodically executed by the given peer (length of period handled by access control).
Advanced distributed algorithms and data structures

Contents:
1. Introduction
2. Graph theory
3. Probability theory
4. Link primitives
5. TCM model and programming environment
6. Congestion control
7. Clock synchronization
8. Logical Clocks
9. Dynamic Overlay Networks
10. Broadcasting and Anycasting
11. Distributed Commit
12. Applications
Congestion Control

**Problem:** nodes need to periodically contact their neighbors (synchronization, failure detection, …) without overwhelming them
Advanced distributed algorithms and data structures

Contents:
1. Introduction
2. Graph theory
3. Probability theory
4. Link primitives
5. TCM model and programming environment
6. Congestion control
7. Clock synchronization
8. Logical Clocks
9. Dynamic Overlay Networks
10. Broadcasting and Anycasting
11. Distributed Commit
12. Applications

WS 2016  Chapter 1  48
Clock Synchronization

The physical clocks of the processes should show the same time.
Advanced distributed algorithms and data structures

Contents:

1. Introduction
2. Graph theory
3. Probability theory
4. Link primitives
5. TCM model and programming environment
6. Congestion control
7. Clock synchronization
8. Logical Clocks
9. Dynamic Overlay Networks
10. Broadcasting and Anycasting
11. Distributed Commit
12. Applications
Logical Clocks

Each request obtains a logical time stamp when processed so that requests can be topologically sorted.

→ Important for transactions, snapshots, …
Advanced distributed algorithms and data structures

Contents:
1. Introduction
2. Graph theory
3. Probability theory
4. Link primitives
5. TCM model and programming environment
6. Congestion control
7. Clock synchronization
8. Logical Clocks
9. Dynamic Overlay Networks
10. Broadcasting and Anycasting
11. Distributed Commit
12. Applications
Dynamic Overlay Networks

- Self-stabilizing clique and 2-clique (diameter 2)

- Other topologies: see Bachelor lecture
Advanced distributed algorithms and data structures

Contents:
1. Introduction
2. Graph theory
3. Probability theory
4. Link primitives
5. TCM model and programming environment
6. Congestion control
7. Clock synchronization
8. Logical Clocks
9. Dynamic Overlay Networks
10. Broadcasting and Anycasting
11. Distributed Commit
12. Applications
Broadcasting and Anycasting

Broadcasting: send a message to all nodes

Anycasting: send a message to a random node
Advanced distributed algorithms and data structures

Contents:
1. Introduction
2. Graph theory
3. Probability theory
4. Link primitives
5. TCM model and programming environment
6. Congestion control
7. Clock synchronization
8. Logical Clocks
9. Dynamic Overlay Networks
10. Broadcasting and Anycasting
11. Distributed Commit
12. Applications
Distributed Commit

commit 2

commit 4

2

2

2

2

2

2
Advanced distributed algorithms and data structures

Contents:
1. Introduction
2. Graph theory
3. Probability theory
4. Link primitives
5. TCM model and programming environment
6. Congestion control
7. Clock synchronization
8. Logical Clocks
9. Dynamic Overlay Networks
10. Broadcasting and Anycasting
11. Distributed Commit
12. Applications
Applications

• Authenticated information system
• Crypto currencies
• Publish / subscribe systems

(may still change)
Questions?