Systems, Networks & Concurrency 2017

Uwe R. Zimmer - The Australian National University
Organization & Contents

Uwe R. Zimmer - The Australian National University
what is offered here?

Fundamentals & Overview
as well as perspectives, paths, methods, implementations, and open questions

of/into/for/about

Concurrent & Distributed Systems
who could be interested in this?

anybody who …

… wants to work with real-world scale computer systems

… would like to learn how to analyse and design operational and robust systems

… would like to understand more about the existing trade-off between theory, the real-world, traditions, and pragmatism in computer science

… would like to understand why concurrent systems are an essential basis for most contemporary devices and systems
who are these people? – introductions

This course will be given by

Uwe R. Zimmer & Alistair Rendell

Your individual tutors are

Abigail Thomas, Benjamin Wang,
Liyang (Leon) Guan, Michael Bennett,
Migara Liyanagamage,
Nathan Yong,
Patrick Chieppe
Organization & Contents

how will this all be done?

Lectures:
- 2x 1.5 hours lectures per week ... all the nice stuff
  Monday 16:00, Thursday 17:00 (both in R.N. Robertson - which is: here)

Laboratories:
- 2 hours per week ... all the rough stuff
  time slots: on our web-site – all in CSIT Nxxx laboratories
  - enrolment: https://cs.anu.edu.au/streams/ (will open after first lecture)

Resources:
- Introduced in the lectures and collected on the course page:
  https://cs.anu.edu.au/courses/comp2310/ ... as well as schedules, slides,
  sources, links to forums, etc. pp. ... keep an eye on this page!

Assessment:
- Exam at the end of the course (55%)
  plus one hurdle lab in week 4 (5%)
  plus two assignments (15% + 15%)
  plus one mid-semester exam (10%)
Text book for the course

[Ben-Ari06]
M. Ben-Ari

*Principles of Concurrent and Distributed Programming*

Many algorithms and concepts for the course are in there
– *but not all!*

References for specific aspects of the course are provided
during the course and are found on our web-site.
Organization & Contents

Topics

1. Concurrency [3]
5. Scheduling [2]
7. Architectures for CDS [1]
8. Distributed systems [7]
Topics

1. Concurrency [3]
   1.1. Forms of concurrency [1]
       • Coupled dynamical systems
   1.2. Models and terminology [1]
       • Abstractions
       • Interleaving
       • Atomicity
       • Proofs in concurrent and distributed systems
   1.3. Processes & threads [1]
       • Basic definitions
       • Process states
       • Implementations


5. Scheduling [2]


7. Architectures for CDS [1]

8. Distributed systems [7]
Topics

1. Concurrency [3]

2.1. by shared variables [1]
   - Failure possibilities
   - Dekker’s algorithm

2.2. by test-and-set hardware support [0.5]
   - Minimal hardware support

2.3. by semaphores [0.5]
   - Dijkstra definition
   - OS semaphores


5. Scheduling [2]


7. Architectures for CDS [1]

8. Distributed systems [7]
Topics

1. Concurrency [3]

3.1. Shared memory synchronization [2]
   - Semaphores
   - Cond. variables
   - Conditional critical regions
   - Monitors
   - Protected objects

   - Asynchronous / synchronous
   - Remote invocation / rendezvous
   - Message structure
   - Addressing

5. Scheduling [2]
7. Architectures for CDS [1]
8. Distributed systems [7]
Topics

1. Concurrency [3]

4.1. Correctness under non-determinism [1]
   - Forms of non-determinism
   - Non-determinism in concurrent/distributed systems
   - Is consistency/correctness plus non-determinism a contradiction?

4.2. Select statements [1]
   - Forms of non-deterministic message reception

5. Scheduling [2]
7. Architectures for CDS [1]
8. Distributed systems [7]
Topics

1. Concurrency [3]
5. Scheduling [2]

5.1. Problem definition and design space [1]
   • Which problems are addressed / solved by scheduling?

5.2. Basic scheduling methods [1]
   • Assumptions for basic scheduling
   • Basic methods

7. Architectures for CDS [1]
8. Distributed systems [7]
Topics

1. Concurrency [3]
5. Scheduling [2]

6.1. Safety properties
   - Essential time-independent safety properties

6.2. Livelocks, fairness
   - Forms of livelocks
   - Classification of fairness

6.3. Deadlocks
   - Detection
   - Avoidance
   - Prevention (& recovery)

6.4. Failure modes

6.5. Idempotent & atomic operations
   - Definitions

7. Architectures for CDS [1]
8. Distributed systems [7]
Organization & Contents

Topics

1. Concurrency [3]
5. Scheduling [2]
7. Architectures for CDS [1]
8. Distributed systems [7]

7.1. Hardware architecture

- From switches to registers and adders
- CPU architecture
- Hardware concurrency

7.2. Language architecture

- Chapel
- Occam
- Rust
- Ada
- C++
Topics

1. Concurrency [3]
5. Scheduling [2]
7. Architectures for CDS [1]
8. Distributed systems [7]

8.1. Networks [1]
    - OSI model
    - Network implementations

8.2. Global times [1]
    - synchronized clocks
    - logical clocks

8.3. Distributed states [1]
    - Consistency
    - Snapshots
    - Termination

8.4. Distributed communication [1]
    - Name spaces
    - Multi-casts
    - Elections
    - Network identification
    - Dynamical groups

8.5. Distributed safety and liveness [1]
    - Distributed deadlock detection

8.6. Forms of distribution/redundancy [1]
    - computation
    - memory
    - operations

8.7. Transactions [2]
# 24 Lectures

## 1. Concurrency [3]

1.1. Forms of concurrency [1]
- Coupled dynamical systems

1.2. Models and terminology [1]
- Abstractions
- Interleaving
- Atomicity
- Proofs in concurrent and distributed systems

1.3. Processes & threads [1]
- Basic definitions
- Process states
- Implementations


2.1. by shared variables [1]
- Failure possibilities
- Dekker's algorithm

2.2. by test-and-set hardware support [0.5]
- Minimal hardware support

2.3. by semaphores [0.5]
- Dijkstra definition
- OS semaphores


3.1. Shared memory synchronization [2]
- Semaphores


4.1. Correctness under non-determinism [1]
- Forms of non-determinism
- Non-determinism in concurrent/distributed systems
- Is consistency/correctness plus non-determinism a contradiction?

4.2. Select statements [1]
- Forms of non-deterministic message reception

## 5. Scheduling [2]

5.1. Problem definition and design space [1]
- Which problems are addressed / solved by scheduling?

5.2. Basic scheduling methods [1]
- Assumptions for basic scheduling
- Basic methods

## 6. Safety and liveness [3]

6.1. Safety properties
- Essential time-independent safety properties

6.2. Livelocks, fairness
- Forms of livelocks
- Classification of fairness

6.3. Deadlocks
- Detection
- Avoidance
- Prevention (& recovery)

6.4. Failure modes

6.5. Idempotent & atomic operations
- Definitions

## 7. Architectures for CDS [1]

7.1. Hardware architecture
- From switches to registers and adders
- CPU architecture
- Hardware concurrency

7.2. Language architecture
- Chapel
- Occam
- Ada

## 8. Distributed systems [7]

8.1. Networks [1]
- OSI model
- Network implementations

8.2. Global times [1]
- Synchronized clocks
- Logical clocks

8.3. Distributed states [1]
- Consistency
- Snapshots
- Termination

8.4. Distributed communication [1]
- Name spaces
- Multi-casts
- Elections
- Network identification
- Dynamical groups

8.5. Distributed safety and liveness [1]
- Distributed deadlock detection

8.6. Forms of distribution/redundancy [1]
- Computation
- Memory
- Operations

8.7. Transactions [2]
Laboratories & Assignments

1. Concurrency language support basics (in Ada) [3]
   1.1. Structured, strongly typed programming
       • Program structures
       • Data structures
   1.2. Generic, re-usable programming
       • Generics
       • Abstract types
   1.3. Concurrent processes:
       • Creation
       • Termination
       • Rendezvous

2. Concurrent programming [3]
   2.1. Synchronization
       • Protected objects
   2.2. Remote invocation
       • Extended rendezvous
   2.3. Client-Server architectures
       • Entry families
       • Requeue facility

   3.1. Multi-core process creation, termination
   3.2. Multi-core process communication

Assignments

1. Concurrent programming [15%]
   Ada programming task involving:
   • Mutual exclusion
   • Synchronization
   • Message passing

2. Concurrent programming in multi-core systems [15%]
   Multi-core programming task involving:
   • Process communication

Examinations

1. Mid-term check [10%]
   • Test question set [not marked]

2. Final exam [55%]
   • Examining the complete lecture

Marking

The final mark is based on the assignments [30%]
plus the examinations [65%]
plus the lab mark [5%]