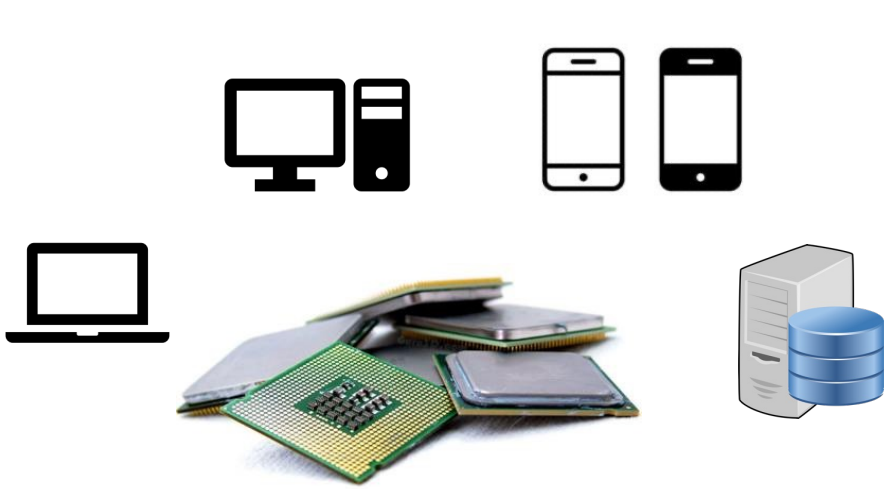
# Analysis of Concurrent and Distributed Programs

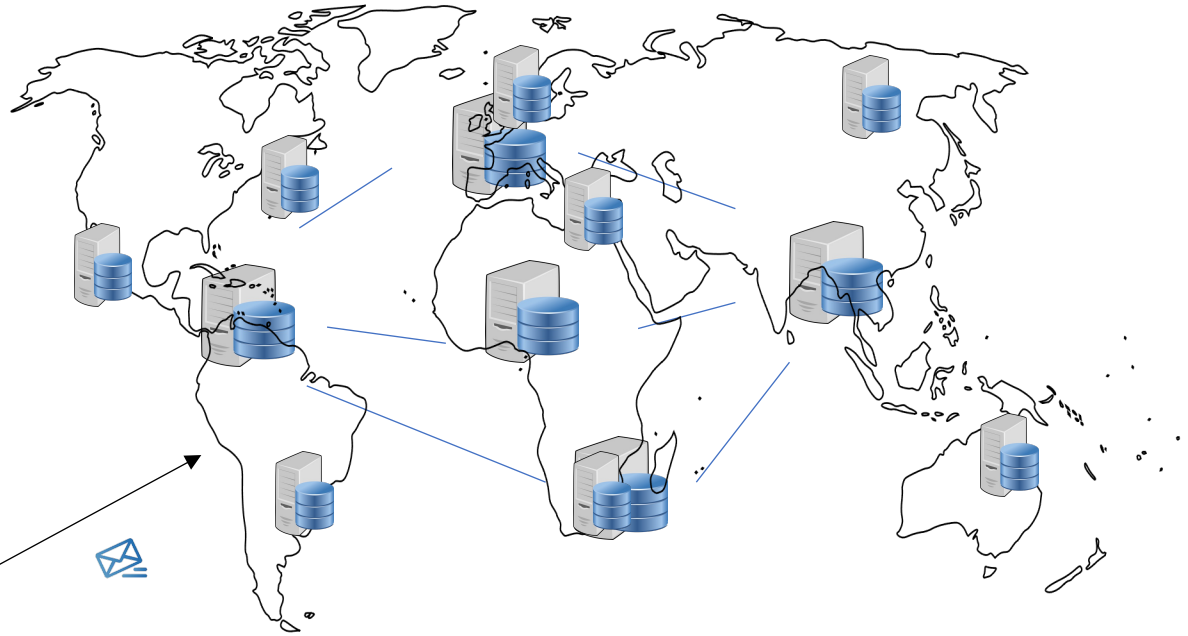
## Introduction

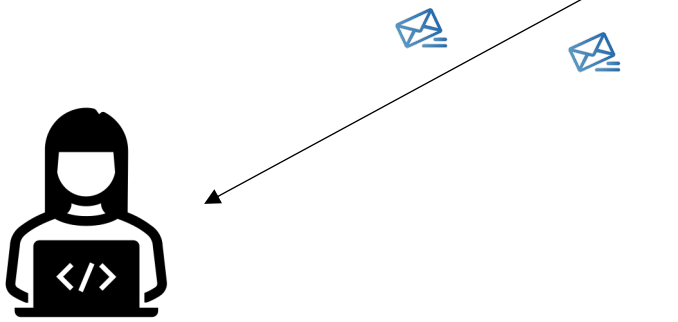
Burcu Kulahcioglu Ozkan, Soham Chakraborty

TU Delft

# Concurrency is everywhere







# Introduction to concurrency

- Running multiple tasks at the same time
- Why write concurrent programs?
  - Performance & scalability
  - Inherent program requirements
- Challenges of writing concurrent programs:
  - Hard to get right
  - Difficult to detect and diagnose concurrency bugs
  - Heisenbugs: Not deterministically reproducible, seems to disappear or change its behavior when one attempts to debug it



# How familiar are you to concurrency?

- What kind of concurrent programs have you worked with?
- Have you encountered *heisenbugs*?
- Have you encountered *flaky tests*?

# CS4405 Course learning objectives

**By the end of this course, you should be able to:**

- Describe the fundamental concurrency models in multicore and distributed systems
- Explain the concurrency nondeterminism in the executions of concurrent and distributed programs
- Discover concurrency bugs in multicore and distributed programs by program analysis and testing techniques
- Evaluate the pros and cons of program analysis and testing techniques for specific multicore and distributed programs

- Disclaimer: The course does **not** teach coding concurrent programs, rather it teaches how to reason about concurrency in shared-memory and distributed programs.

*"People confuse programming with coding. Coding is to programming, what typing is to writing."* — Leslie Lamport



# Course organization

- **5 ECTS:** You need to devote at least 140 hours of study for this course.
- **Lectures:** The course consists of 10 2-hour meetings. You are not required, but you are strongly encouraged, to attend.
- **Project (100%):** Project implementation, report, and a presentation
- **Teams (2-4 students):** The students are responsible to form teams and communicate them to the course TAs.
- **TAs:**
  - Ege Berkay Gulcan
  - Daan de Graaf



# Many flavours of concurrency

- Concurrent programming
  - Multiple tasks can be in progress at any instant
- Parallel programming
  - Utilizing more than one processors for running the program
- Asynchronous programming
  - Programming with non-blocking requests/method calls
- Event-driven programming
  - The flow of the execution is determined by the (possibly concurrent) events
- Distributed programming
  - Multiple computers run as a single system


*Many authors consider shared-memory programs to be "parallel" and distributed-memory programs to be "distributed"*




# Shared-memory concurrency:

- An executin module from Deep Space-1 spacecraft (launched in October 1998)

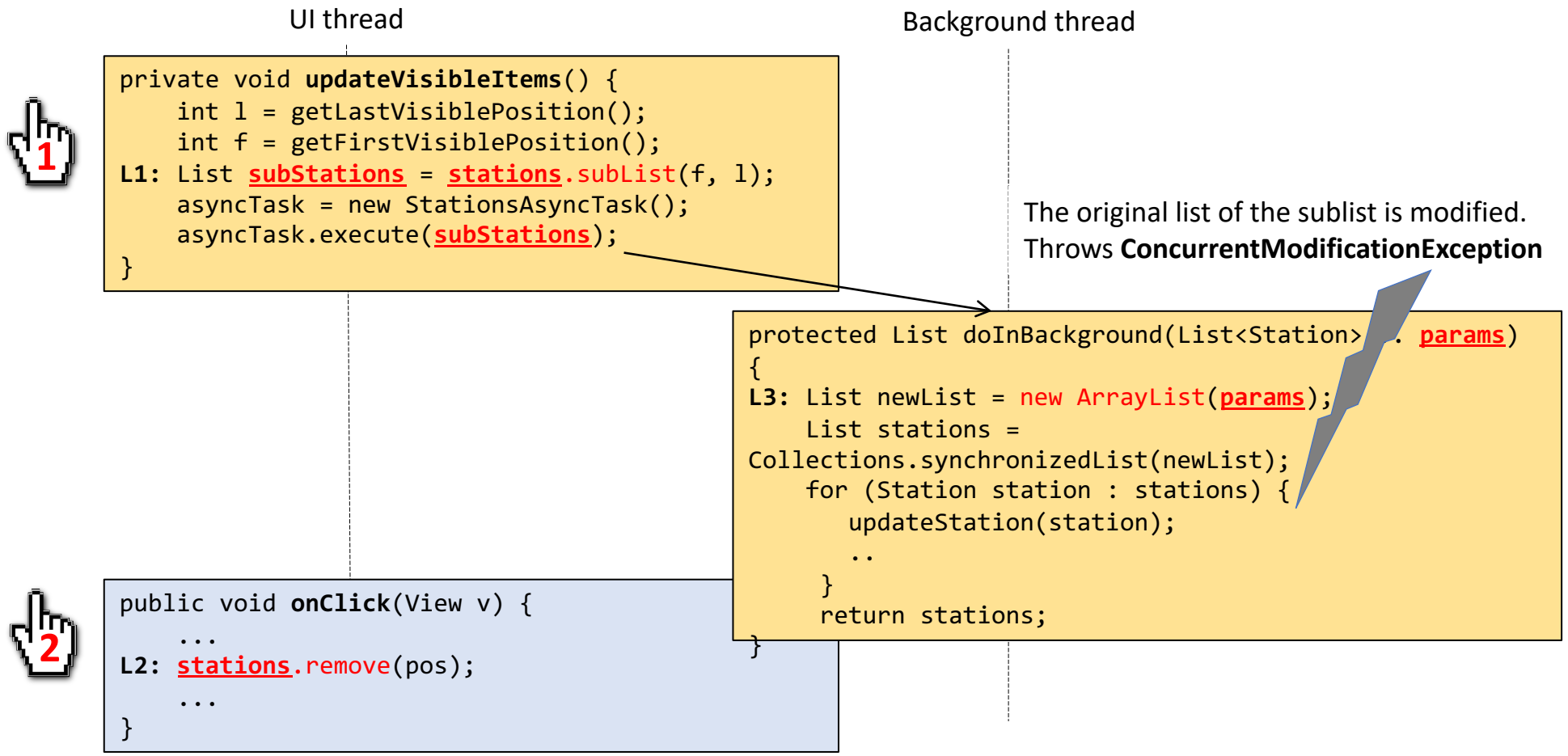
- proc Inc = while true { if x < 200 then x := x + 1 }
- proc Dec = while true { if x > 0 then x := x − 1 }
- proc Reset = while true { if x = 200 then x := 0 }

- Is the value of x always between (and including) 0 and 200?

*Example from "Principles of Model Checking". Baier&Katoen. 2008.*



# Asynchronous event-driven concurrency



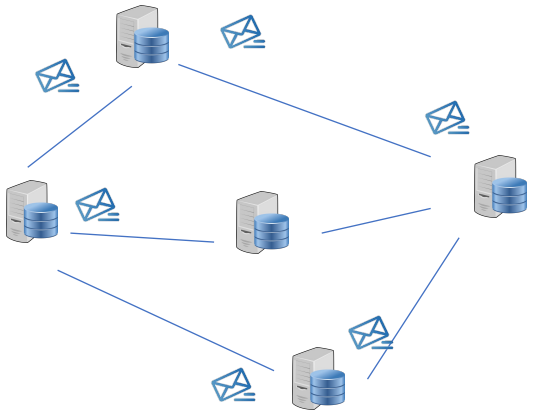

ojacquemart / vilIeChecker

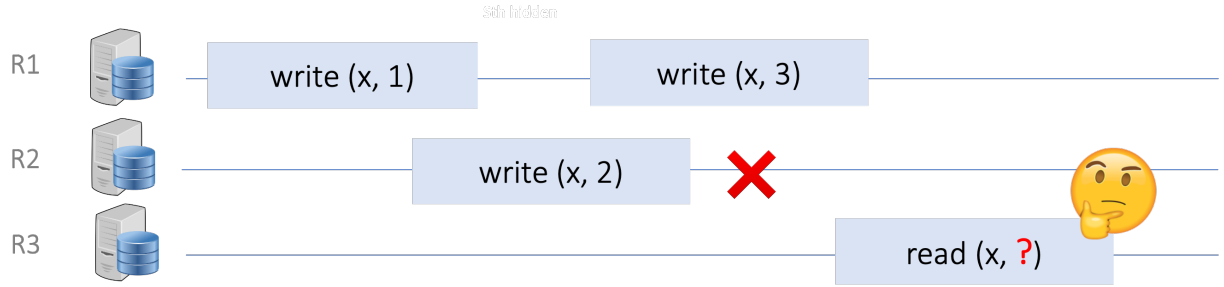
Fix java.util.ConcurrentModificationException #60

Closed ojacquemart opened this issue on Sep 15, 2013 · 0 comments

*Example from "Systematic Asynchrony Exploration of Android Apps". K. Ozkan, Emmi, Tasiran. 2015.*

# Concurrency + Distributed systems + fault tolerance






Consistency model is a contract between programmer and replicated system, i.e., it specifies the consistency between replicas and what can be observed as possible results of operations.

It answers questions such as:

- What can be possible results of a read query?
- Can a client see its own updates?
- …

# Takeaways

- Nondeterminism in the execution
- Know the execution model & programming model
- Hard to reason about concurrency

- We need methods and tools for their analysis!



# Course contents and schedule

- Feb 13 - Introduction to concurrency

- Feb 15 - Concurrency primitives and bugs
- Feb 20 - Concurrency analysis for multithreaded programs
- Feb 22 - Weak-memory consistency - I
- Feb 27 - Weak-memory consistency – II

- Mar 01 - Distributed concurrency
- Mar 06 - Concurrency analysis for distributed systems
- Mar 08 - Strong consistency: SC and Linearizability
- Mar 22 - CAP theorem & weak consistency and isolation

- Apr 03-05 - In-progress project evaluation
- Apr 17-19 - Final project presentations



# Course Projects

- Data race detection in C/C++ weak memory programs
  - How to detect non-atomic-races, relaxed-races, RA-races in shared memory accesses?
- Concurrency testing of weak memory programs
  - How do the existing testing algorithms for distributed systems apply to testing weak memory concurrency?
- Detecting concurrency-related flaky tests in event-driven applications
  - How do the existing testing algorithms for flaky test detection methods apply to different Android applications?
- Checking consistency & isolation of distributed database executions
  - How to check linearizability, sequential consistency, snapshot isolation, etc?
- Interested in another project? Contact us with your project proposal ☺