COMP28112 – Lecture 1
Distributed Computing

Aims:

Many of the most important and visible uses of computer technology rely on distributed computing. This course unit aims to introduce students to the principles, techniques and methods of distributed computing in sufficient breadth and depth for it to act as a foundation for the exploration of specific topics in more advanced course units. The course unit assumes that students have already a solid understanding of the main principles of computing within a single machine, and that they have a rudimentary understanding of the issues related to machine communication and networking.

Module Organisation

- Lecturing team:
  - *Chris Kirkham*: chris@cs.man.ac.uk
  - ***Rizos Sakellariou***: **rizos@manchester.ac.uk**
- Classes:
  - 16 Lectures + 2 (?) guest lectures + revision+exam
    - (Monday 13:00; Tuesday 16:00)
  - 5 Lab Sessions, week B (plus a marking session)
    - (Tue 11:00-13:00 G23, Wed Fri 11:00-13:00 LF31 – 3 groups)
- webpage:

  http://studentnet.cs.manchester.ac.uk/ugt/COMP28112/

The plan
(not strictly in order - small changes expected)

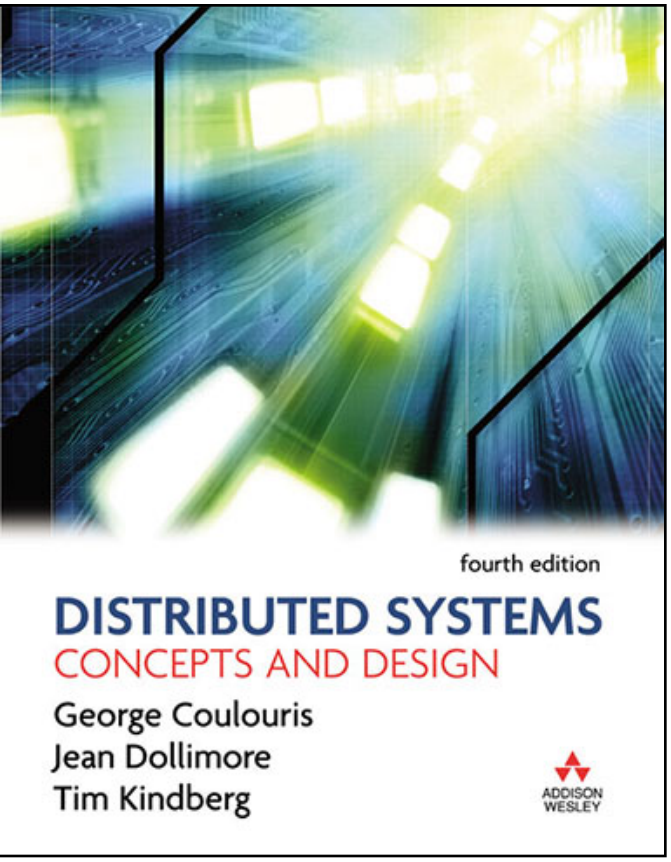
- 01 (Introduction to Distributed Computing)
- 02 (A few words about Parallel Computing)
- 03 (Models and Architectures)
- 04 (RPC+RMI)
- 05 (RPC+RMI - exercises)
- 06 (Intro to lab exercise 2)
- 07 (Name and Directory Servers)
- 08 (Time and Clocks)
- 09 (Coordination and Agreement)
- 10 (Fault Tolerance – Transactions)
- 11 (Distributed Transactions)
- 12 (Byzantine Fault Tolerance)
- 13 (Replication)
- 14 (The Quest for Performance - lab exercise 3)
- 15 (The Integration Game)
- 16 (Grid and Cloud Computing)
- – guest lecture from UBS (?)
- – guest lecture on Cloud Computing (?)
- – some interesting problems, free lectures to prepare the lab, revision and the exam

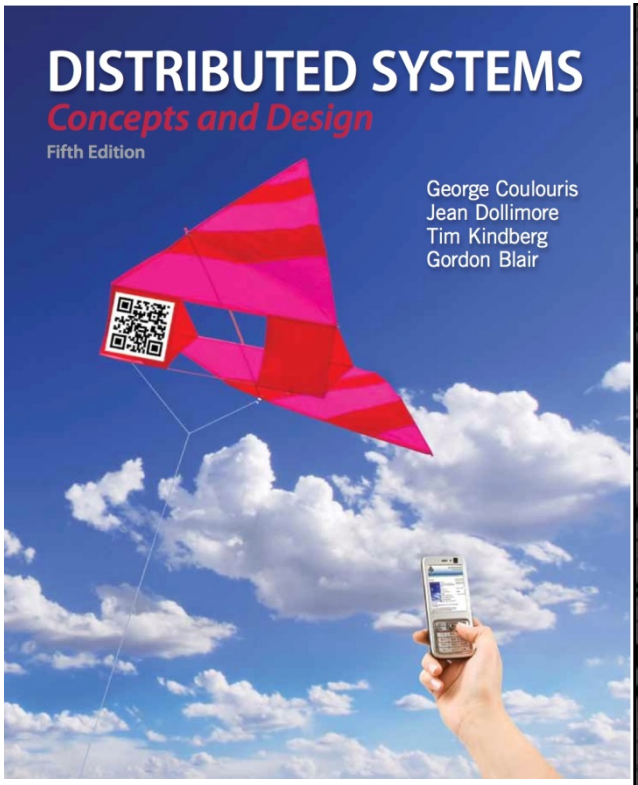
The Lab Assignments

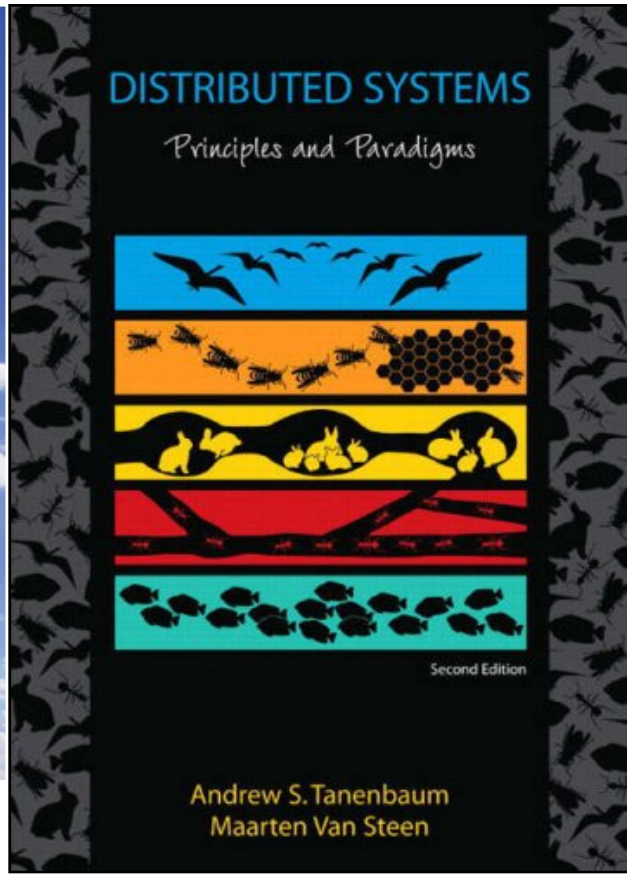
- 1st exercise: Servers & Clients (1 lab session)
- 2nd exercise: Wedding Planner (3 lab sessions)
- 3rd exercise: The quest for performance (1 lab session)
- Warnings:
  - These will require some hundred lines of code - **make sure your Java (or …) skills are up-to-date**!
  - There is flexibility: you can use other languages (2nd, 3rd ex.).
  - Code may be marked using automatic tests!
  - You need to produce fully working software!

Course Textbooks

I'll try to provide pointers to the following books
Advice: Get hold of a copy!







How to Study

- This is complex stuff, so you need to keep a tight grip on it:
  - attend lectures, and:
    - make your own notes, listen, understand, jot down, reflect, ... (lecture notes will contain essential information)
  - read the book (even if you don't have your own copy)
    - there is an abundance of books and material on distributed computing; consulting different sources helps!
  - attend the lab
  - ask questions when you don't understand
  - don't get behind!

Style

- Be flexible, keep an open mind, etc...
- We're doing **engineering** :
  - not an exact science...
    - but, basic exact science skills are essential (e.g., how long will it take to transmit a message of size 4MB over a network link with speed 256KB/sec?)
  - constraints, optimisations, ...
  - unreasonable (or infinite) demands, ...
  - imperfections, trade-offs, ...

  Distributed Systems typically encompass a number of such trade-offs!

Why?

- We have smartphones, computers everywhere, remote storage, massive online games, apps, tons of data…
- The future of computing is distributed
  - (be careful with predictions: "*I think there is a world market for maybe five computers*")
- Technological innovation will have a profound impact in the future of jobs

http://www.economist.com/news/briefing/21594264-previous-technological-innovation-has-always-delivered-more-long-run-employment-not-less

Definitions

- **System**: "A complex whole; a set of connected parts; an organized assembly of resources and procedures (collection of …) united and regulated by interaction or interdependence to accomplish a set of specific functions."

- **Distributed System** definitions:
  - A collection of independent computers that appears to its users as a single coherent system.
  - A system in which hardware and software components of networked computers communicate and coordinate their activity only by passing messages.

Distributed System definition

A computing platform built with many computers that:

- Operate concurrently;
- Are physically distributed; (have their own failure modes)
- Are linked by a network;
- Have independent clocks

"*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.*"

Leslie Lamport

(http://en.wikipedia.org/wiki/Leslie_Lamport)

Consequences

- Concurrent execution of processes:
  - Non-determinism, race conditions, synchronisation, deadlocks, …
- No global clock
  - Coordination is done by message exchange
  - No single global notion of the correct time
- No global state
  - No process has a knowledge of the current global state of the system.
- Units may fail independently
  - Network faults may isolate computers that are still running
  - System failures may not be immediately known

Why do we have distributed systems?

- People are distributed but need to work together…
- Hardware needs to be physically close to people (who are distributed)…
- Information is distributed but needs to be shared (trustworthily)…
- Hardware can be shared (increases computing power by doing work in parallel; more efficient resource utilisation)…

Examples of distributed systems…

- Intra-nets, Inter-net, WWW, email, …
- DNS (Domain Name System)
  - Hierarchical distributed database
- Distributed supercomputers, Grid/Cloud computing
- Electronic banking
- Airline reservation systems
- Peer-to-peer networks
- Sensor networks
- Mobile and Pervasive Computing

Evolution

- Parallel Computing was a hot topic in the 70s and 80s. (the vision existed since the 1920s)
  - Cluster computers started dominating in the 1990s.
- Early distributed systems:
  - Airline reservation systems
  - Banking systems
- The real proliferation came with developments in **network technology** and the **WWW** (early 90s)

The 8 fallacies of distributed computing

- It is a common mistake for programmers, when they first build a distributed application, to make the following 8 assumptions. All prove to be ***false*** in the long run and all cause ***big trouble*** and ***painful learning experiences***: (http://www.rgoarchitects.com/Files/fallacies.pdf)

1. **The network is reliable**
2. **Latency is zero**
3. **Bandwidth is infinite**
4. **The network is secure**
5. **Topology doesn't change**
6. **There is one administrator**
7. **Transport cost is zero**
8. **The network is homogeneous**

Peter Deutsch, a SUN fellow is credited with the first seven (1994); around 1997, *James Gosling* added the 8th fallacy.
Lots of information can be found through google.

Fallacy 1: The Network is Reliable

- Hardware may fail!
  - Power failures; Switches have a mean time between failures.
    (e.g., a router between you and the server you get data from)
- The implications:
  - Hardware: weigh the risks of failure versus the required investment to build **redundancy** (yet another trade-off!).
  - Software: we need reliable messaging: be prepared to retry messages, acknowledge messages, reorder messages (do not depend on message order), verify message integrity, and so on.

Fallacy 2: Latency is zero

Latency (not bandwidth): how much time it takes for data to move from one place to another: measured in time.

- The minimum round-trip time between two points on earth is determined by the maximum speed of information transmission: the speed of light. At 300,000 km/sec, it will take at least 30msec to send a ping from Europe to the USA and back.
- The implications:
  - You may think all is ok if you deploy your application on LANs, but you should strive to make as few calls over the network as possible (and transfer as much data out in each of these calls).
- **Read**: http://blogs.msdn.com/oldnewthing/archive/2006/04/07/570801.aspx
- **Exercise**: 100MB file, latency 1sec or 0.001sec, bandwidth 100MB/sec, at once or not?

Fallacy 3: Bandwidth is infinite

- **Bandwidth**: how much data you can transfer over a period of time (may be measured in bits/second)
- It constantly grows, but so does the amount of information we are trying to squeeze through it! (VoIP, videos, verbose formats such as XML, …)
- Bandwidth may be lowered by packet loss (usually small in a LAN): we may want to use larger packet sizes.
- The implications:
  - Compression; try to simulate the production environment to get an estimate for your needs.

Fallacy 4: The Network is Secure

"*In case you landed from another planet, the network is far from being secured*"

(common wisdom)

- The Implications:
  - You may need to build security into your applications from Day 1.
  - As a result of security considerations, you might not be able to access networked resources, different user accounts may have different privileges, and so on…

Fallacy 5: Topology doesn't change

- The topology doesn't change as long as we stay in the lab.
- In the wild, servers may be added and removed often, clients (laptops, wireless ad hoc networks) are coming and going: the topology is changing constantly.
- The implications:
  - Do not rely on specific endpoints or routes.
  - Abstract the physical structure of the network: the most obvious example is DNS names as opposed to IP addresses.

    (refresh your memory about the Internet Domain Name System – DNS)

Fallacy 6: There is one administrator

- Unless we refer to a small LAN, there will be different administrators associated with the network with different degrees of expertise.
- Might make it difficult to locate problems (is it their problem or ours?)
- Coordination of upgrades: will the new version of MySql work as before with Ruby on Rails?
- **Don't underestimate the 'human' ('social') factor!**

Fallacy 7: Transport Cost is Zero

- Going from the application layer to the transport layer (2nd highest in the five layer TCP/IP reference model) is not free:
- Information needs to be serialised (marshalling) to get data onto the wire.
- The cost (in terms of money) for setting and running the network is not zero. Have we leased, for instance, the necessary bandwidth?

Fallacy 8: The Network is Homogeneous

- (homogeneous = of the same kind; uniform).
- Even a home network may connect a Linux PC and a Windows PC. A homogeneous network today is the exception, not the rule!
- Implications:
  - Interoperability will be needed.
  - Use standard technologies (not proprietary protocols), such as XML (a W3C recommended general-purpose markup language – a markup language combines text and extra information about the text – designed to facilitate the sharing of data across different information systems. Its drawback? It's slow…)

Summary

- COMP28112 synopsis:
  - Basic principles of distributed systems

- In distributed systems as opposed to centralised systems:
  - There is concurrency
  - There is no global clock/state
  - Systems may fail independently

- Reading:
  - Coulouris, 1.1 to 1.3 (pages 1-15); Tanenbaum 1.1 and 1.3
- **Read lab exercise 1**.
- Next: Challenges – parallel computing.