Distributed Systems (5DV147)

Introduction

Fall 2013

about the course

Staff presentation

Francisco Hernandez-Rodriguez

francisco@cs.umu.se

Ewnetu Bayuh Lakew

ewnetu@cs.umu.se

Cristian Klein

cklein@cs.umu.se

- Questions about the assignment?
  - Send to 5dv147-staff@cs.umu.se
- Questions about lectures?
  - Send email to the appropriate teacher!
- Ewentu’s and Cristian’s office hours: Monday, Tuesdays, and Thursday between 14:00 and 15:00, MIT-huset, D447 and D444
  - Priority / FIFO queue

Course presentation

The goal of this course is to **introduce basic knowledge** to **understand** how modern distributed systems operate. Our focus will be on distributed algorithms and on practical aspects that should be considered when designing and implementing real systems.

Although students will need to learn various distributed algorithms, this is **not only** a **theoretical** course. Thus, computer based assignments will be used extensively so that students will gain **practical experience** designing and implementing real systems.

About the book…

Distributed Systems, 5th ed. Coulouris, Dollimore, Kindberg, and Blair, Addison-Wesley/Pearson Education

1. Buy the book.
2. No, seriously. Buy it!
3. 5th edition

The course covers:

- Fundamental models of distributed systems
- System performance
- Replication, transactions, transparency, peer-to-pear, consistency, coordination and synchronization …
- Computer security in a broad perspective

	Monday		Thursday	
	(10 - 11)	(11 - 12)	(10 - 11)	(11 - 12)
V45 (7 Nov)			(1) Introduction to the course	(2) Introduction to the course
V46 (11,14 Nov)	(3) Fundamental models	(4) Fundamental models	(5) Security	(6) Security
V47 (18,21 Nov)	(7) System Performance Lecture	(8) System Performance Practical	(9) Indirect communication	(10) Explanation Java RMI
V48 (25,28 Nov)	(11) Logical time	(12) Global states	(13) Mutual exclusion	(14) Elections
V49 (2,5 Dec)	(15) Group communication	(16) GCoM design	(17) Consensus	(18) Consensus
V50 (9,12 Dec)	(19) Replication	(20) Consistency	(21) Cassandra Lecture	(22) Cassandra practical
V51 (16,19 Dec)	(23) Transactions	(24) Distributed transactions	(25) Peer-to-peer	(26) Persistent GCom design
V2 (6,9 Jan)	Hacking day	(28) Hacking day	(29) Course summary	(30) Questions regarding third project

What to learn?

- Book is dense with information
  - See reading guide on web page but consider it a guide not a guarantee
  - Not an easy read
    - Start now! You *will* be busy later...
- Understand the problems and solutions
  - Learn the general ideas of algorithms and how/why they work, not every minute step
- Definitions are very important

Lessons from last year

What was positive about the course and should be retained?

- The course content as a whole
- The examples presented on the board
- The slide presentations
- The project assignment

What can be improved?

- Faster feedback on the first deliverable of the project
- Teach lectures at a slower pace (talk slower)
- Give examples of distributed systems before having to design the project
- Suggestions about the slides- avoid too much text and keep titles at the top of each slide
- Present examples on message orderings: e.g. message ordering such as causal is a bit difficult to understand theoretically
- It is a bit unclear what is expected from assignment one

Do you think the programming assignments utilized or build on the topics covered in the lectures?

- Most of the students answered that the practical part is in line with the theory presented during lectures. They claim that this is one of the best course in doing this
- One student feels that the project gives too much focus to message orderings

This year

Learning outcomes

- Explain general properties, challenges, and characteristics of distributed systems
- Explain the notions of causality and time in light of the design and implementation of distributed systems
- Explain general distributed algorithms for synchronization and concurrency, coordination, transactions, and replication
- Compare replication schemes with respect to performance, availability, and consistency concerns
- Explain practical issues that need to be considered when designing, implementing, and debugging distributed systems

this year

Skills and abilities

- Design, implement, and debug distributed systems
- Employ fundamental distributed algorithms to solve problems that arise when engineering distributed systems
- Explain the inner mechanisms of current production distributed systems

Evaluation

Credit points (7.5 ECTS)

- Theoretical part 50% of final grade
- Practical part 50% of final grade
- Both parts will be evaluated through mandatory assignments
- Optional comprehensive examinations for those that fail the theoretical part
- You need to score at least 50% of the points on each part to pass the course

Written assignments (WA) total 100 points

- Written assignment 1 – 20 points
- Written assignment 2 – 40 points
- Written assignment 3 – 40 points

Programming projects (PP) total 100 points

- Java RMI – 10 points
- GCom – 60 points
- Persistent chat – 30 points

Final score = (WA + PP)/2

- Final score ≥ 80 → 5
- 65 ≤ Final score < 80 → 4
- 50 ≤ Final score < 65 → 3
- Final score < 50 → U (Fail)

Written assignments

- Test handling of theoretical concepts
- To do at home – two working weeks
- Individually
- All normal rules apply
  - Thou shall not cheat, …

  http://www.student.umu.se/english/code-of-rules/
  http://www8.cs.umu.se/information/hederskodex_eng.html
- Comprehensive examination if you failed to obtain 50 points

Programming projects

- Apply concepts from theory
  - Vector clocks, group handling, message ordering, reliable multicast, replication, …
- JavaRMI (just to get you started)
- GCom (group communication middleware)
- Persistent chat (a taste of a large system)
- No security, however
- Late submission policy
  - Late submissions will be penalized by reducing 10% of the final score for each late working day
- More specifics at the end of the lecture

- Course evaluation

What is a distributed system?

“A distributed system is one in which components located at networked computers **communicate and coordinate their actions only by passing messages**.”

Coulouris, Dollimore, Kindberg, and Blair, 2012

“A distributed system is one in which the **failure of computer you didn't even know existed** can render your own computer unusable.”

Leslie Lamport, 1987

Why distributed systems?

BANK
$
$
ATM

Figure 1.1 - Selected application domains and associated networked applications

Finance and commerce	eCommerce e.g. Amazon and eBay, PayPal, online banking and trading
The information society	Web information and search engines, ebooks, Wikipedia; social networking: Facebook and MySpace.
Creative industries and entertainment	online gaming, music and film in the home, user-generated content, e.g. YouTube, Flickr
Healthcare	health informatics, on online patient records, monitoring patients
Education	e-learning, virtual learning environments; distance learning
Transport and logistics	GPS in route finding systems, map services: Google Maps, Google Earth
Science	The Grid as an enabling technology for collaboration between scientists
Environmental management	sensor technology to monitor earthquakes, floods or tsunamis

why distributed systems?

Asociación TulaSalud, ONG
Atencion de salud con el uso eficiente de TICs.
www.tulasalud.org

Tele Medicina en Alta Verapaz
Traslado de paciente de comunidad en mecapal
www.tulasalud.org

Resource sharing

Cloud computing

http://opencompute.org/wp/wp-content/uploads/2011/07/Freedom-PRN1-02-1024x569.jpg

Distributed systems

Characteristics

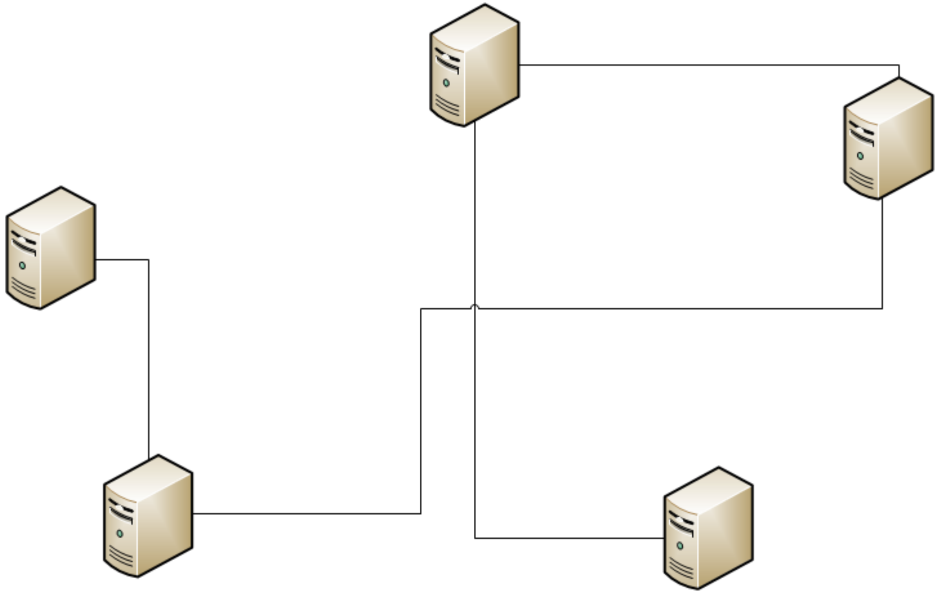
Concurrency of components

http://en.wikipedia.org/wiki/Dining_philosophers_problem

Absence of shared clock
+ Absence of shared memory

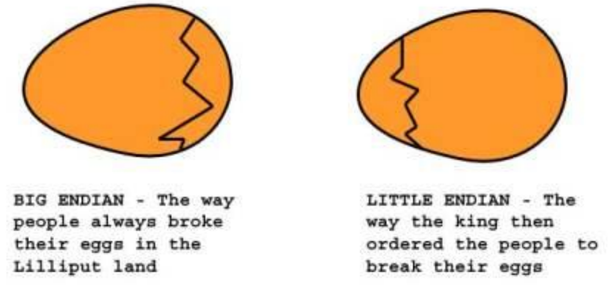
Impossible to know the global state of the system

Independent failures of components



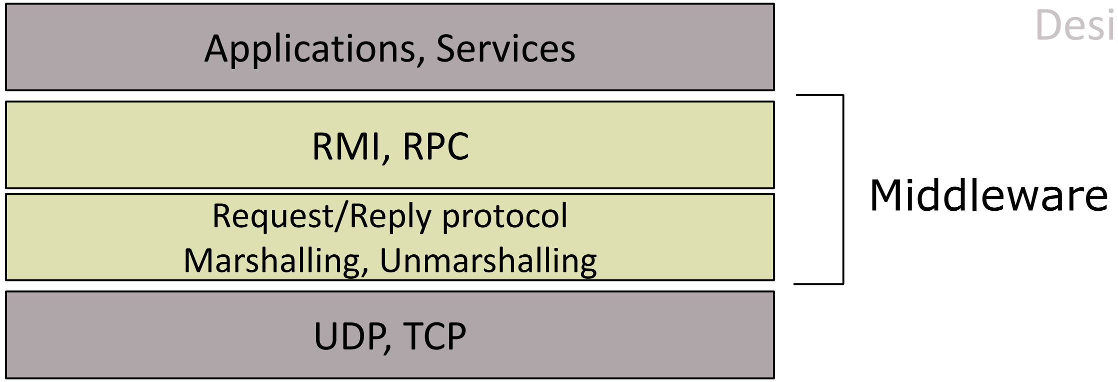
Design challenges

Heterogeneity


BIG ENDIAN - The way people always broke their eggs in the Lilliput land
LITTLE ENDIAN - The way the king then ordered the people to break their eggs

Middleware & Virtualization

Image from: *http://cyriacgeorge.files.wordpress.com/2012/05/big-endian-little-endian.jpg?w=500&h=250*



- Distributed systems often utilize middleware to aid development
- Offers layer of abstraction
- Extends upon traditional programming models:
  - Local procedure call → Remote procedure call
  - OOP → Remote Method Invocation
  - Event-based programming model

Openness

Published interfaces and/or according to standards

Security

authentication and authorization

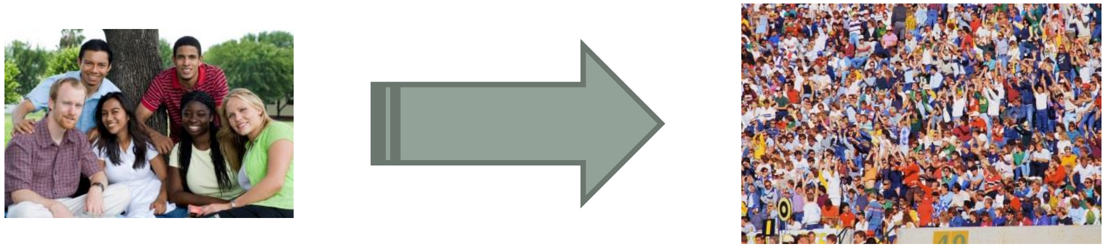
denial of service attacks

security of mobile code

Server must be able to prove that something has been executed

Non repudiation: it should not be possible to claim that something did not happen if it did

Scalability



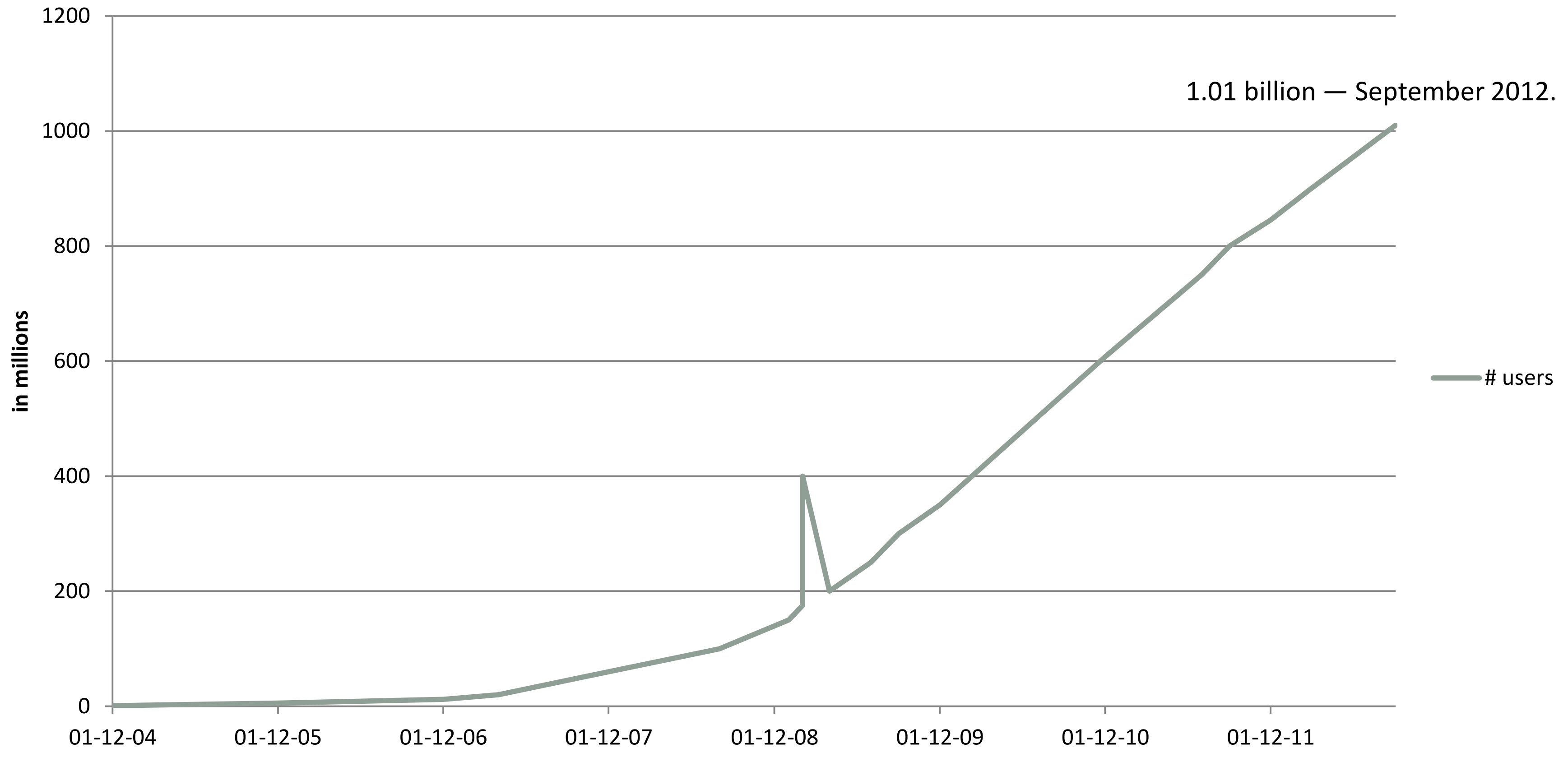
Controlling the cost of physical resources

Controlling the performance loss

Preventing resources from running out

Avoid performance bottlenecks

Number of facebook users



Failure handling

Detecting failures, masking failures, tolerating failures, recovery from failures, redundancy

Availability in the face of hardware faults

Transparency

Access, location, concurrency, replication, failure, mobility, performance, scaling

Users shouldn't need to know!

Quality of Service (QoS)

Reliability, security, performance, availability, adaptability, timeliness guarantees

Distributed systems, a mess!

Availability

Fault tolerant

Scalability

Consistency

Privacy

Reliability

Low latency

Security

Communication performance variations:
Latency (delay), bandwidth (throughput), jitter (variation in time)

Clocks and timing: Clock drift

Interaction models: asynchronous, synchronous

Event ordering: Delays cause replies to arrive to some process before the request

Failures: Distributed systems are much more likely to fail unexpectedly due to e.g., lost packets, bit errors, local failures, no response, method does not exist, etc. …

If you can write stable programs in spite of these difficulties, you are a great programmer!

Projects

GCom

- Middleware for group communication
  - Vector clocks
  - Handling of group membership
    - Dynamic groups
  - Leader election
  - (Reliable) Multicast communication
  - Message ordering guarantees
    - Causal ordering
  - Debugging functionality
  - Client → chat application
- Theory from the second set of lectures
  http://www8.cs.umu.se/kurser/5DV147/HT13/
- Presentation of working implementation

Rules and grading

- Solved in pairs – select your partner ASAP
- Bonus points
  - Tree-based reliable multicast (20p bonus)
  - Need at least 50 points on the written assignments valid only for this year

Constraints

- May use any programming language and any tools you like
  - … as long as they don't provide too much advantage (check with us)
  - We will only help with Java RMI and Python

    You may absolutely not use plain sockets
- All normal rules apply
  - Thou shall not cheat, …

  http://www.student.umu.se/english/code-of-rules/

  http://www8.cs.umu.se/information/hederskodex_eng.html

What do you need to turn in?

- Test application
  - A chat client that shows the functionality of the system
- Debug application
  - Used to demonstrate the correctness of your implementation

These programs can be the same but make the debug parts non-essential to use the application, and they **must be** GUI applications!

- Project plan – December 5 (Optional)
  - Your interpretation of the assignment
  - Requirement analysis
  - Project and time plan
  - Basic design of the system

- Report – December 23
  - Describe your system
  - … the usual
  - More information when the text of the assignment is posted
  - Make something to be proud of!

One of your biggest projects during your time here at CS

- Written test protocol to demonstrate your system

```
Two nodes – A,B
1. A sends message M1 to B
2. A sends message …
3. Message M1 is delayed on B
```

Persistent chat

- Make GCom persistent
  - Clients can disconnect
    - Retrieve messages sent during disconnected periods
  - Save messages maintaining the ordering
    - Causal ordering
  - Use Cassandra for storing data
  - Debugging functionality
  - Fault tolerant
- Theory from the third set of lectures
  http://www8.cs.umu.se/kurser/5DV147/HT13/
- Presentation of working implementation

Rules and grading

- Solved in pairs – same pairs as for GCom
- Bonus points
  - Surprise us (10p bonus-subjective)
    - Search functionality, statistical information, performance evaluation, only display or highlight unread messages, …
  - Need at least 50 points on the written assignments valid only for this year

Constraints

- May use any programming language and any tools you like
  - … as long as they don’t provide too much advantage (check with us)
  - We will only help with Java RMI and Python

    You may absolutely not use plain sockets
- All normal rules apply
  - Thou shall not cheat, …

http://www.student.umu.se/english/code-of-rules/

http://www8.cs.umu.se/information/hederskodex_eng.html

What do you need to turn in?

- Test application
  - A chat client that shows the functionality of the system
- Debug application
  - Used to demonstrate the correctness of your implementation

These programs can be the same but make the debug parts non-essential to use the application, and they **must be** GUI applications!

- Project plan – December 17
  - Your interpretation of the assignment
  - Requirement analysis
  - Project and time plan
  - Basic design of the system

- Report – January 20
  - Describe your system
  - … the usual
  - More information when the text of the assignment is posted
  - Make something to be proud of!

- Written test protocol to demonstrate your system

```
Two nodes – A,B,C
1. A sends message M1
2. A sends message …
3. B replies to message M1
4. C connects and receives messages
5. C replies to message M2
6. A disconnects
```

Good luck!

- Students have done this before and succeeded
  - It is certainly not easy
  - Hard work, big payoff
  - All students that attempted the entire assignment passed!

… remember
  - Start on time
  - Read the whole specification (it's long but it helps)

Next Lecture

Fundamental properties of distributed systems