Distributed Systems (5DV147)

Introduction

Fall 2014

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

Goals of the course (1)

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.

Goals of the course (2)

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.

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

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

Topics

	Tuesday		Friday	
	(10 - 11)	(11 - 12)	(10 - 11)	(11 - 12)
V36 (2, 5 Sep)	Introduction to the course	Introduction to the course	Time	Global States
V37 (9,12 Sep)	Mutual Exclusion	Elections	Group Communication	Consensus (1)
V38 (16,19 Sep)	Consensus (2)	GCom Design	Replication (1)	Replication (1)
V39 (23,26 Nov)	Replication (2)	Consistency	Cassandra	Cassandra (Hand on)
V40 (30 Sep)	Systems Performance	Systems Performance (Hands on)	Hacking day	Hacking day
V41 (7,10 Oct)	Transactions	Transactions	Concurrency Control	Concurrency Control
V42 (14,17 Oct)	Persistence GCom explanation	Persistence GCom explanation	Peer-to-peer	Peer-to-peer
V43 (21 Oct)	Security	Security		

About the book…

We will not use a textbook

Readings about specific topics (in Cambro)

https://www.cambro.umu.se/portal/site/57202HT14-1

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

What to learn?

- Understand problems and solutions
  - Learn the general ideas of algorithms and how/why they work, not every single detail
  - Do the written assignments as topics are introduced
- Definitions are very important
- Focus on referenced readings and lectures

Lessons from last year

What was positive about the course and should be retained?

- The structure of the course was good
- Having written assignments was good as they helped understanding the theory behind some parts needed for the labs.
- The practical sessions (Cassandra, Performance, and Security) were good
- The programming projects were interesting and it was good that the GCom project (second project) was to be finished before Christmas
- The choice to either do written assignments or write a comprehensive exam was positive
- The slides were nice

What can be improved?

- To return graded written assignments faster so that they can know how they are doing
- They liked the practical sessions (like Cassandra's and performance) and would like to have more of those as part of the course
- One student mentioned that the course was time consuming, perhaps too much at times, e.g., the programming project required, for that student, 70-80% of the course time when it should have required 50% of the time
- Several students dislike the course book. One complaint was that the book covers a lot of material that is not part of the lectures, so it is really hard to find out what needs to be read and what shouldn't. There were some suggestions to change the book or to give additional references as well
- Students also agreed that the practical part of the course corresponds well to the theoretical part since the theoretical part was used for implementing the labs

This year

- No official course book
- Re-organization of topics to facilitate work on programming projects

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 knowledge 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

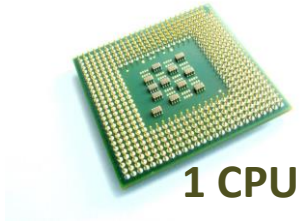
- Apply concepts from theory
  - Vector clocks, group handling, message ordering, reliable multicast, replication, …
- JavaRMI (just to get you started)
- GCom (group communication middleware)
- Persistent chat (a taste of a large system)
- No security, however
- Late submission policy
  - Late submissions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

What is a distributed system?

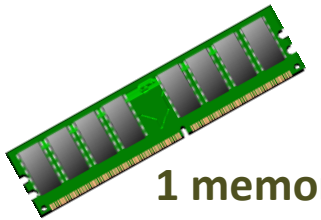
…basics

Single process

```
int i;
i=i+1;
  …
```



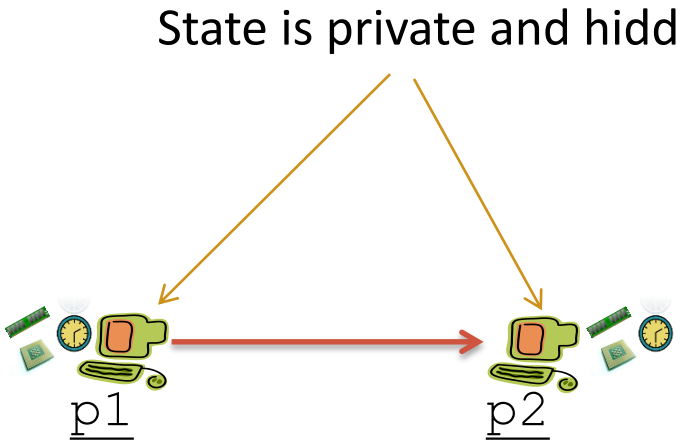
1 CPU



1 memory

- Steps are strictly sequential
- Program behavior & variables' state determined by sequence of operations

Distributed processes




```
int i;          int a;
i=i+1;          …
send(i,p2);     receive(p1,a);
…               …
```

Definition of steps of each process including transmission of messages between processes (computation, send, receive)

Can't predict the rate at which each process executes nor the timing of transmissions

Progress depends on internal local state and of messages received

Impossible to describe all states of a distributed algorithm because of failure of any of the processes or of message transmissions

"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

Message-passing model

Characteristics:

- Synchrony of the system
- Type of communication network
  - point-to-point or broadcast channel
- Model of process and communication failures that may occur

Synchronous systems

- There is a known upper bound on the time required by any process to execute a step
- Every process has a clock with known bound rate of drift with respect to real time
- There is a known bound on message delay – time to send, transport, and receive a message over a link

(Hadzilacos and Toueg, 1994)

Asynchronous systems

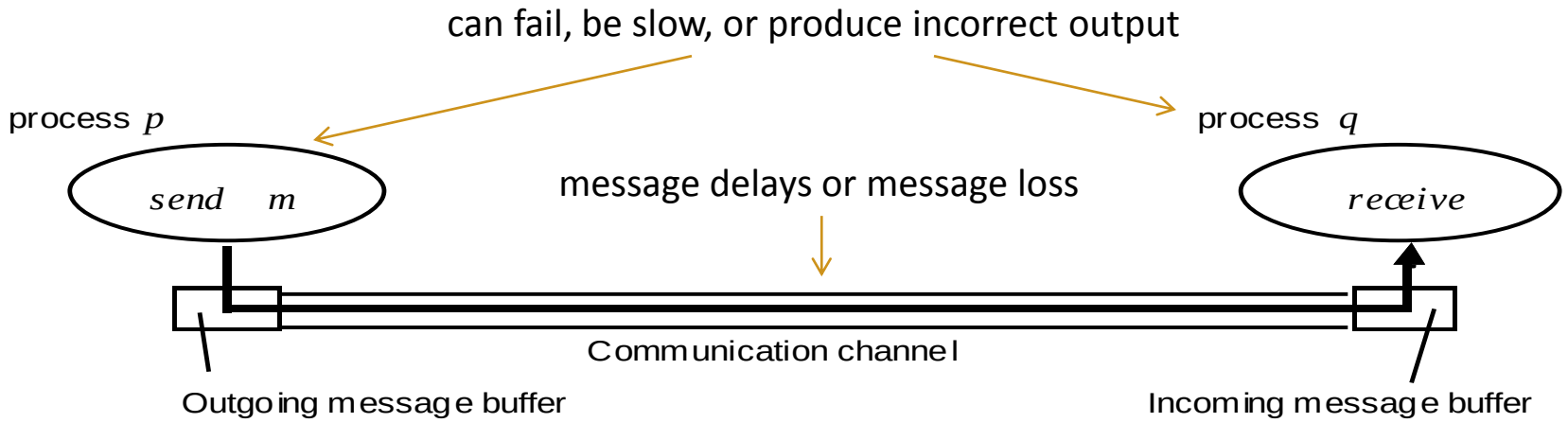
No timing assumptions, in particular on the maximum message delay, clock drift, or the time needed to execute a step

(Hadzilacos and Toueg, 1994)

Attractive because of:

- Simple semantics
- Applications are easier to port
- Variable or unexpected workloads are sources of asynchrony
- Processes share resources and communications channels share the network

Point-to-point networks



Two assumptions:

1. If p sends a message m to q then q eventually receives m
2. Every process executes an infinite sequence of steps

Figure adapted from Instructor's Guide for Coulouris, Dollimore, Kindberg and Blair, Distributed Systems: Concepts and Design Edn. 5 © Pearson Education 2012 – **Figure 2.14**

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

© Pearson Education 2012


TulaSalud


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


NEWS
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

Characteristics

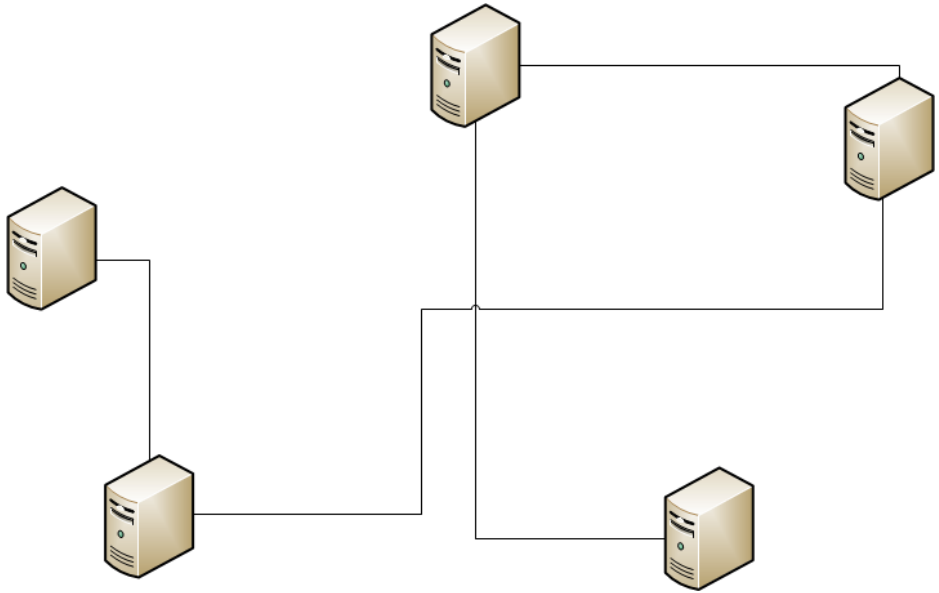
Concurrency of components

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

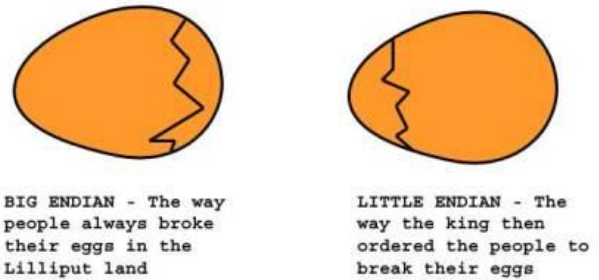


"A distributed system is one in which the **failure of computer you didn't even know existed** can render your own computer unusable."

Leslie Lamport, 1987

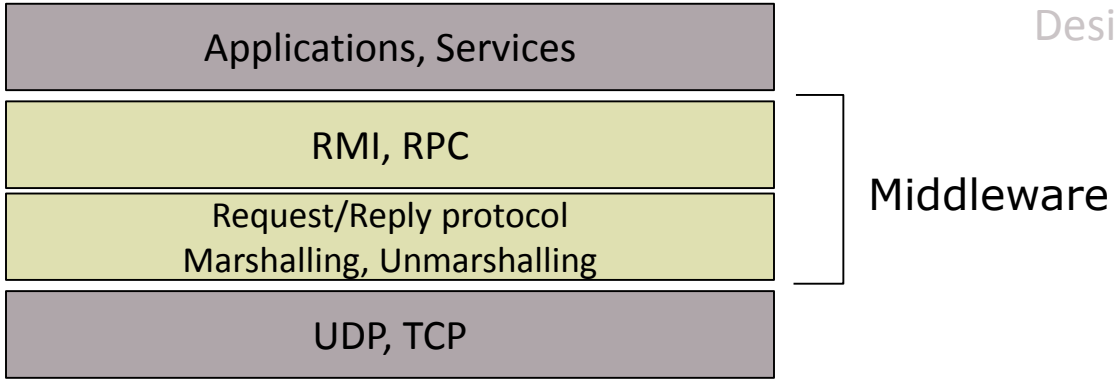
Design challenges

Heterogeneity



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

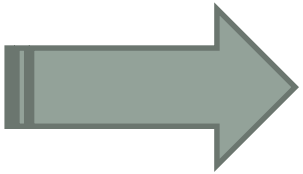
Published interfaces and/or according to standards

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

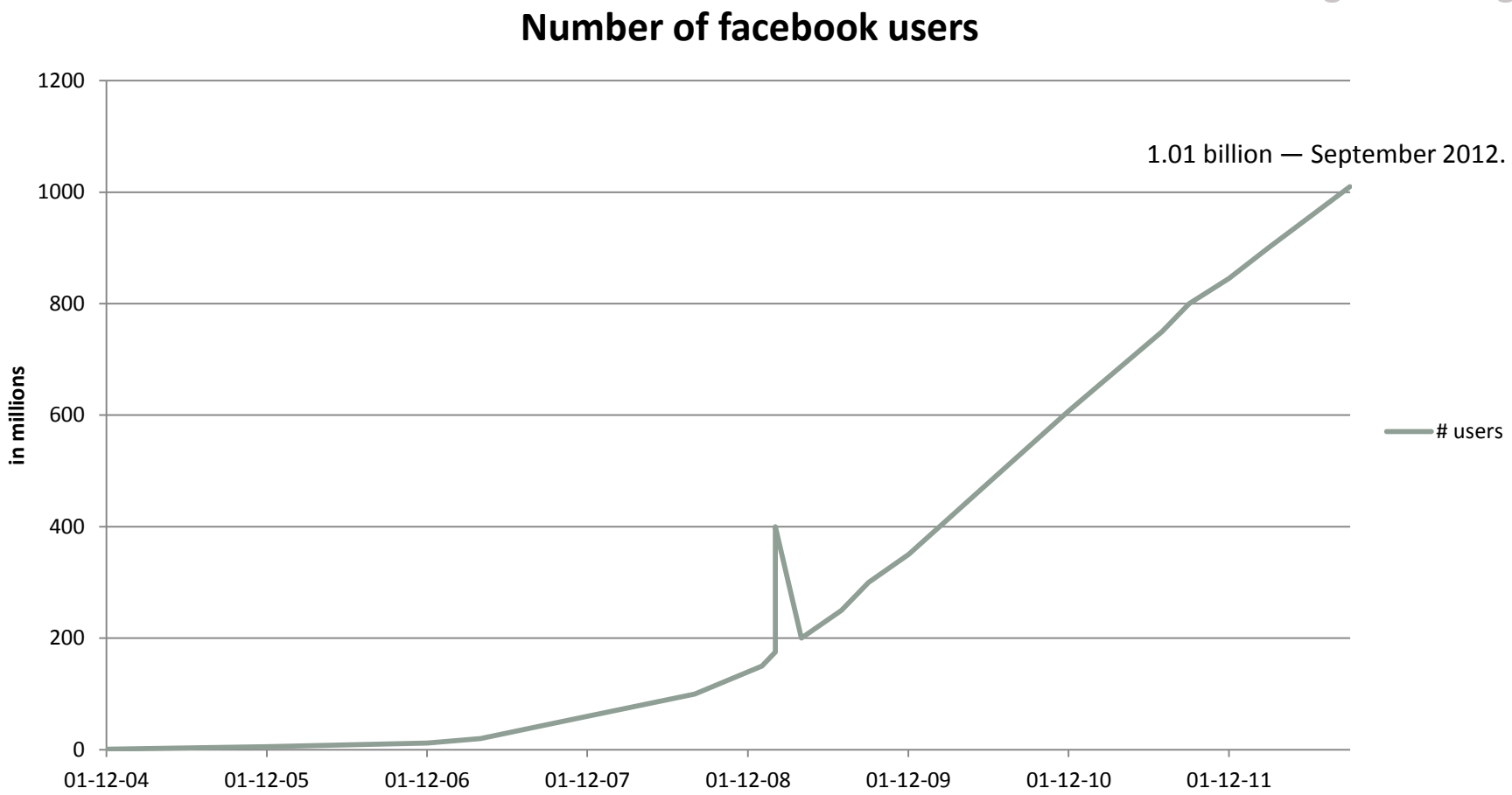


Controlling the cost of physical resources

Controlling the performance loss

Preventing resources from running out

Avoid performance bottlenecks


Number of facebook users
1200
1000
800
600
400
200
0
in millions
01-12-04
01-12-05
01-12-06
01-12-07
01-12-08
01-12-09
01-12-10
01-12-11
1.01 billion — September 2012.
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

Reliability

Privacy

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 first set of lectures
  http://www8.cs.umu.se/kurser/5DV147/HT14/
- 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 – September 19 (Optional)
  - Your interpretation of the assignment
  - Requirement analysis
  - Project and time plan
  - Basic design of the system

- Report – October 13
  - 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 second set of lectures
  http://www8.cs.umu.se/kurser/5DV147/HT14/
- 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 – October 15
  - Your interpretation of the assignment
  - Requirement analysis
  - Project and time plan
  - Basic design of the system

- Report – October 31
  - 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

Time and Global states