Introduction to Assignment 3
Distributed Systems Lecture
HS 2010, ETH Zurich

Simon Mayer
simon.mayer@inf.ethz.ch
Today's Menu

• Repetition (lecture slides 189 – 195) + UDP
  • Causality
  • Lamport Time
  • Vector Time [new!]

• Assignment 3
  • Task 1
  • Task 2
  • Task 3.1 and 3.2
Briefly: The User Datagram Protocol

- Simple transmission model
  - No hand-shakes, ordering, data integrity
  - Datagrams delayed (out of order), duplicate, missing

- Common applications
  - DNS (port 53)
  - Streaming
  - VoIP
  - Online gaming
UDP Effects...

"What is the first prime number after 1000000?"

QBot

"P2 answered correctly!"

P1

"1000003!"

"1000003!"

"??!"

P2

"Yeah!"
Causality

• Interesting property of distributed systems...

• Causal Relation '<' (“happened before”):

\[ x < y \iff ( (x, y \text{ on same process, } x \text{ happens before } y) \text{ or } (x \text{ is send and } y \text{ is corresponding receive}) \text{ or } (\text{transitivity}) ) \]
Causality

\[ x < y \text{ iff (} (x, y \text{ on same process, } x \text{ happens before } y) \text{ or } (x \text{ is send and } y \text{ is corresponding receive}) \text{ or } (\text{transitivity}) \) \]
Software Clocks

• *Ideal Real Time*: Transitive, dense, continuous,...

• *Logical Time*: Cheap version of real time
  • *Lamport Timestamps*
  • *Vector Clocks*
  • *Matrix Clocks*
Lamport Time

- Using a single clock value
  - Local Event: *Local clock tick*
  - Send Event: *Attach local clock value*
  - Receive Event: *max(local clock, message clock)*

- Satisfies clock consistency condition: $e < e' \rightarrow C(e) < C(e')$
Lamport Time

- Lamport Time does not satisfy **strong clock consistency condition**

\[ e < e' \iff C(e) < C(e') \]
Vector Time

• Refining Lamport Time: Processes keep one counter per process

• Does satisfy strong clock consistency condition!

\[
e < e' \iff C(e) < C(e')
\]
Vector Time [example]

"What is the first prime number after 1000000?"

"P2 answered correctly!"

"1000003!"

"1000003!"

"Yeah!"

"??!"
Vector Time [example]

QBot

P1

P2
Vector Time

“Process $i$ stores information on what it thinks about the local time of processes $(1,\ldots,n)$.”
Matrix Time [not in the assignment]

- Refining Vector Time: Processes keep n counters per process

"Process i stores information on what it believes that processes (1,...,n) think about the local time of processes (1,...,n)."

```
1 1 0
5 4 5
2 1 2
```

"What is the first prime number after 1000000?"
Today's Menu

- Repetition (lecture slides 189 – 195) + UDP
  - Causality
  - Lamport Time
  - Vector Time [new!]

- Assignment 3
  - Task 1
  - Task 2
  - Task 3.1 and 3.2
A Mobile, Causal, UDP-based Chat-Application

- Task 1: “Getting familiar with Datagrams”
- Task 2: “Starting the Conversation” + Lamport Timestamps
- Task 3: “Vanquishing the Desequencer”
  - 3.1 Vector Clocks
  - 3.2 Additional questions (→ Report)
- Report
1. Getting familiar with Datagrams

- Communicate with server at http://vswot.inf.ethz.ch:3999 using UDP
- Provides “capitalization” service
Side Note: System Setup

- vswot Services
  - (De-)Registration of clients
  - Distributes messages ("Broadcast")
  - De-sequencing “service”

http://vswot.inf.ethz.ch
Side Note: Encoding Time...

- Lamport Time: Need to encode single Timestamp (index 0)
- Vector Time: Need to encode multiple Timestamps
- Marker (\n, newline) to separate message from time vector
2. Starting the Conversation

- UDP chat with server (ports 4000/4001)
- Causality preservation via **Lamport Time**
- Lamport Timestamp stored in 0\textsuperscript{th} time vector index
  - Ignore all other indices while doing this task
Grading – Criteria for getting a 4.0

Task 1
Task 2
Report on that
3.1 Vanquishing the Desequencer

- UDP chat with server (ports 4000/4001)
- Causality preservation via Vector Clocks
- Own Timestamp in \(i^{th}\) time vector index
  - \(i\) assigned by Server on registration

\[
\text{message} \quad n : 0=4::1=3::4=6:
\]
Grading – Criteria for getting a 5.0

- Task 1
- Task 2
- Task 3.1
- Good report
3.2 Vanquishing the Desequencer

- When exactly are two Vector Clocks causally dependent?
  - Does a clock tick happen before or after the sending of a message?
  - How are `receive` events handled? Do they trigger local clock ticks?

- Dynamically Joining / Leaving Clients
  - Read the paper “Dynamic Vector Clocks”
  - Describe the approach taken in the paper
  - What is the difference to our approach?

Cover this in your report (about 1 page for task 3.2)
Send / Receive / Tick policies

- Multiple ways to implement vector clock ticking
  - Tick only when sending, after sending [vs. before sending]
  - Tick when receiving and sending, after sending [vs. before sending]

- QuestionBot's and AnswerBot's policy:
  - Tick only when sending, before sending

  Example: Message from process 2 with timestamp \([4,5,1]\) means:
  - “Before receiving me, you should already have received and delivered 4 messages from process 1, 4 (!) messages from process 2 and 1 message from process 3!”
  - “If you did not receive these, wait before delivering me!”

- What if a message is lost?
Grading – Criteria for getting a 6.0

Task 1
Task 2
Task 3.1
Task 3.2
Excellent & Complete Report
Issues / Considerations

• Maybe try it in pure Java first...
  • Better debugging... (e.g., Exceptions are actually displayed...)
  • Faster & More convenient

• Lots of groups interact via the chat server
  • Potential Problem: Some groups non-compliant
  • Result could be: Everyone's code crashes...

  • Solution: Tag your messages (e.g., using your group number)
    Only consider own messages
Message Delivery / Delay

- Receive chat messages from server
- Delay delivery according to message timestamp (very simple case below...)

Message should not yet be delivered...
The server http://vswot.inf.ethz.ch:4000

- **Registration** (+ notification to all clients)
  
  Send: \texttt{.reg.username}  
  
  Answer OK: \texttt{:reg\_ok:assigned-vector-index:username } \texttt{\backslash n time-vector}  
  
  Answer FAIL: \texttt{:reg\_nok\_name: '.reg.uname'}

- **Deregistration** (+ notification to all clients)
  
  Send: \texttt{.dreg.}  
  
  Answer OK: \texttt{:dreg\_ok:}  

- **Client Information**
  
  Send: \texttt{.clients.}  
  
  Answer: \texttt{:clients:{client\_i\_ID}}
Demo

- Device
- Emulator
  - Port redirect!

```bash
# Example commands

telnet localhost EMULATOR_PORT
redir add udp:4001:4001
redir list
```

- DNS issue (cf. [http://vs.inf.ethz.ch/edu/vs/android/](http://vs.inf.ethz.ch/edu/vs/android/))
  - User server IP instead
Final Remarks

- Make sure that vswot can reach you on port 4001!

- Inside ETH: Use wireless ssid “eth”

- At home: Configure network to forward incoming packets to your machine/phone
That's it... direct all questions to simon.mayer@inf.ethz.ch