Introduction to Assignment 3
Distributed Systems Lecture
HS 2011, ETH Zurich

Wilhelm Kleiminger
kleiminger@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 → P1

"1000003!"

P1 → P2

"1000003!"

P2 → P1

"P2 answered correctly!"

P1 → QBot

"??!"

QBot → P2

"Yeah!"
Causality

- Interesting property of distributed systems...
- Causal Relation '<' ("happened before"):

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

\[ x < y \iff (x, y \text{ on same process, } x \text{ happens before } y) \text{ or } (x \text{ is send and } y \text{ is corresponding receive}) \text{ or } (\text{transitivity}) \]
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: $\max(\text{local clock, message clock})$

• 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?"

QBot

"1000003!"

"1000003!"

"P2 answered correctly!"

P1

"?? !"

"Yeah!"

P2
Vector Time [example]
Vector Time

“Process $i$ stores information on what it thinks about the local time of processes (1,...,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 $(1,\ldots,n)$ think about the local time of processes $(1,\ldots,n)$.”

```
QBot
```

```
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: “Overcoming the Desequencer”
  - 3.1 Vector Clocks
  - 3.2 Additional questions (→ Report)
- Report
1. Getting familiar with Datagrams

- Communicate with server at \texttt{vswot.inf.ethz.ch:3999} using UDP
- Provides “capitalization” service
Side Note: Encoding Time...

- Lamport Time: Need to encode single Timestamp
- Vector Time: Need to encode multiple Timestamps

We use a Map<int, int> or dictionary to identify timestamps.

The key or index “0” always corresponds to Lamport time
Index i is associated to one of the clients and issued when registering!
Side Note: System Setup

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

Port 4000
Port 4001

vswot.inf.ethz.ch

AnswerBot

QuestionBot
The server **http://vswot.inf.ethz.ch:4000**

**JSON Protocol:**

--> {"cmd":"register","user":"willi"}

<-- {"index":3,"time_vector":{"3":0,"2":70,"1":71,"0":74},"success":"reg_ok"}

--> {"cmd":"get_clients"}

<-- {"clients":{"/129.132.75.130":"QuestionBot","/129.132.252.221":"AnswerBot","/77.58.228.17":"willi"}}

--> {"cmd":"info"}

<-- {"info":"I am an advanced UDP server that is running at port 4000 to provide a de-sequencing service for Android UDP chatting programs..."}

--> {"text":"hallo","cmd":"message","time_vector":{"3":1,"2":70,"1":71,"0":75}}

--> {"cmd":"deregister"}

<-- {"success":"dreg_ok"}
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
    - So: Only consider this index when doing task 2...
3.1 Overcoming the Desequencer

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

- When exactly are two Vector Clocks causally dependent?
  - Does your application allow “purely local” events? Do they trigger a clock tick?
  - Does a local 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 there

Cover this in your report!
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?
Issues / Considerations

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

- Use VPN when not in ETH network!
- 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
That's it...

Copyright © 2003 United Feature Syndicate, Inc.