Distributed Systems - HS 2013
Assignment 3

Hông-Ân Cao
hong-an.cao@inf.ethz.ch
Contact format

- If you need to contact me by email, please respect the following format:
  - Subject should be [VS HS2013] nethz – Description of your problem
  - Where nethz is the nethz of the group's leader
  - This way your email doesn't get lost in the flow of emails and if I need to check logs on the server, I can check your entries more efficiently.

- Please remember that if you contact me shortly before the deadline, a timely reply cannot be expected.
Outline

- Review of logical time and UDP
  - Causality
  - Lamport Timestamps
  - Vector Times

- Assignment 3
  - Task 1
  - Task 2
  - Task 3.1 and 3.2
The User Datagram Protocol

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

The User Datagram Protocol
UDP Effects

"What is the first prime number after 1000000?"

"P2 answered correctly!"

"1000003!"

"?? !"

"Yeah!"
Causality

- Interesting property of distributed systems
- Causal relationship $\prec$ ("happened before")

$x \prec y$ iff 
(x, y on same process, x happens before y) or
(x is send and y is correspondingly received) or
(transitivity

I USED TO THINK
CORRELATION IMPLIED
CAUSATION.

THEN I TOOK A
STATISTICS CLASS.
NOW I DON'T.

SOUNDS LIKE THE
CLASS HELPED.

WELL, MAYBE.
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 $\rightarrow$ Transitive, dense, continuous, etc.

- Logical time $\rightarrow$ 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
  
  \[ e < e' \iff C(e) < C(e') \]

---

**Diagram:**

- Nodes labeled 1, 2, 3, 4, 5
- Arrows indicating order of events
- Node 1 on \( P_1 \), node 2 on \( P_2 \), node 3 on \( P_3 \)

---

**Notes:**

- Lamport Time is a model for ordering events in a distributed system.
- It is based on the concept of a local clock at each node.
- Events are ordered based on the local clock values.
- However, it does not satisfy the strong clock consistency condition, which is necessary for certain types of distributed systems.
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

"What is the first prime number after 1000000?"

"P2 answered correctly!"

"1000003!"

"1000003!"

"??!"

"1000003!"

"Yeah!"
Vector Time

Distributed Systems - HS 2013
Introduction Assignment 3
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,...,n) think about the local time of processes (1,...,n).”

QBot

"What is the first prime number after 1000000?"

1 1 0
5 4 5
2 1 2
...
Outline

- Review of logical time and UDP
  - Causality
  - Lamport Time
  - Vector Time

- 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 vslab.inf.ethz.ch:4000 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 Map<int, int> or dictionary to identify vector times.
- An int is associated to the lamport time.
Side Note: System Setup

vslab services:
- (De-) Registration of clients
- Distributes messages ("Broadcast")
- De-sequencing "service"

Port 5000
vslab.inf.ethz.ch
JSON Protocol on vslab.inf.ethz.ch:5000

→ {"cmd": "register", "user": "caoh1"}

← {"index": 2, "init_time_vector": {"2": 0, "1": 70, "0": 71}, "init_lamport": 74,"success": "reg_ok"}

→ {"cmd": "get_clients"}

← {"clients": {"0": "questionbot", "1": "answerbot", "2": "caoh1"}}

→ {"cmd": "info"}

← {"info": "I am an advanced UDP server that is running at port 5000 to provide a de-sequencing service for Android UDP chatting programs..."}

→ {"text": "hallo","cmd": "message", "time_vector": {"2": 1, "1": 70, "0": 71}, "lamport": 75}

→ {"cmd": "deregister"}

← {"success": "dreg_ok"}

Everyone else receives:

← {"cmd": "message", "text": "caoh1 has left (index 2)"}
2. Starting the conversation

- UDP chat with server port 5000
- Causality preservation via Lamport Timestamps
- Lamport Timestamp stored in integer in field "Lamport"
  - So, only consider this value when doing task
3.1 Overcoming the desequencer

- UDP chat with server on port 5000
- Causality preservation via Vector Clocks
- Own timestamp in $i^{th}$ time vector index
  - $i$ assigned by server upon registration
3.2 Overcoming the desequencer

- When exactly are 2 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 sending 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 (!) 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 and more convenient
- **Forward port to emulator**
  [Link](http://stackoverflow.com/questions/5064304/how-can-i-forward-my-localhost-ip-address-to-an-android-emulator)
- **Use VPN when not in ETH network!**
- Lots of groups interact via the chat server
  - Potential problem → some groups non-compliant
  - Results could be → Everyone's code crashes…
  - Solution → Tag your messages (e.g. using your group's number) and/or only consider your own messages
The End