YOU'RE ABOUT TO HACK TIME, ARE YOU SURE?
YES NO
Distributed Systems

Day 7: Time and Snapshots [Part 1]
Delete Professor Theo !!!!

I hate Professor Theo!!

But Why?

Delete Professor Theo !!!!

I hate Professor Theo!!

But Why?
Status update: “John had an accident!!!”

Status update: “He’s well now 😊”

Status Comment: “Good News!!”

Status update: “John had an accident!!!”

Status update: “He’s well now 😊”

Status Comment: “Good News!!”
Agenda

• Time
  • Synchronization (NTP, PTP)
  • Challenges and Limitations

• Distributed Systems Notions
  • Total ordering, partial ordering

• Logical Time
  • Logical Clocks
  • Vector Clocks
How do Clocks work?
Clocks and Clock Skew

- Time is a function of a crystal’s oscillations
  - Several oscillations == one second
  - Crystal does not always oscillate at right frequency.
  - So time becomes faster or slower

- Terminology: describing differences in clocks
  - Skew: difference between two clocks!
  - Drift: counting at different frequency than intended

Clock 1  10:00am
Clock 2  10:05am

Skew = 10am – 10:05am
Synchronizing Clocks
PTP: Precision Time Protocol

What time is it?

Time 10 am!

- One client synchronizes with an authoritative clock
  - Through exchange of network messages
How does your phone get its Time?

Clock Synchronization Challenges

• One client synchronizes with an authoritative clock
  • Through exchange of network messages

• Problems:
  • Delay between messages
  • Delay in processing
  • Server may be wrong!!!
Clock Synchronization

• Common protocols: NTP, PTP
  • Cheap and inaccurate across long distances

• More accurate protocols: GPS, atomic clocks
  • Requires expensive hardware
  • Signaling issues when indoors

• Do we really need time precision?
  • What do we care about?

https://www.theverge.com/2012/11/26/3692392/google-spanner-atomic-clocks-GPS
https://www.wired.com/2012/11/google-spanner-time/
https://www.datacenterdynamics.com/analysis/a-wrinkle-in-time/
Clock Synchronization

- Common protocols: NTP, PTP
  - Cheap and inaccurate across long distances

- More accurate protocols: GPS
  - Expensive and hard to retrofit at global scale
  - Signaling issues when indoors

- Do we really need time precision?
  - What do we care about?

https://www.theverge.com/2012/11/26/3692392/google-spanner-atomic-clocks-GPS
https://www.wired.com/2012/11/google-spanner-time/
https://www.datacenterdynamics.com/analysis/a-wrinkle-in-time/
Ordering

• Total ordering
  • Each server can order events (in same order)
  • Possible with a global entity (e.g. global clock)

• Partial ordering: !(total ordering)
  • Some ordering that’s not total ordered
Client sends to server 1 and server 2

Recall “total ordering” – events processed in the same order in all servers.
Linearizability: The Golden Standard...

- Total order: all servers process message in same order
- Linearizability: special case of total order
  - Order is based on real time

Client sends to server 1 and server 2

<table>
<thead>
<tr>
<th>Server 1</th>
<th>Server 2</th>
<th>Total ordered</th>
</tr>
</thead>
<tbody>
<tr>
<td>New Account</td>
<td>Add $50</td>
<td>New Account</td>
</tr>
<tr>
<td>Add $50</td>
<td>Withdraw $100</td>
<td>Add $50</td>
</tr>
<tr>
<td>Withdraw $100</td>
<td>Add $50</td>
<td>New Account</td>
</tr>
<tr>
<td>Add $50</td>
<td>Withdraw $100</td>
<td>Add $50</td>
</tr>
<tr>
<td>Add $50</td>
<td>Withdraw $100</td>
<td>Add $50</td>
</tr>
</tbody>
</table>
**Status update:** “John had an accident!!!”

**Status update:** “He’s well now 😊”

**Status update:** “Cut myself! Headed to the Dr.”
**Status Comment:** “Bummer!!!!”
**Status update:** “I’m fixed!!!!”
**Status Comment:** “Good News!!”

**Status update:** “Cut myself! Headed to the Dr.”
**Status Comment:** “Bummer!!!!”
**Status update:** “I’m fixed!!!!”
**Status Comment:** “Good News!!”

**Total Order:** order based on global time

**FIFO Order:** order based on client time

**All servers process in same order**

**Different servers Can process in different orders but events from clients in same order**

10am

11am

5pm

4pm
How can we implement total/FIFO ordering?

System Assumptions
- No server (node) failure.
- Network is reliable:
  - Messages are delivered eventually
  - Messages are delivered in-order
Implementing Total Ordering

• Key insight: need a third party to assign IDs

• Process Pseudocode
  • Before sending message, m, ask for an ID
  • Include ID with messages
  • Send to all Servers!!!

• Global Sequencer Pseudocode
  • Start ID = 0
  • On receive ID request,
    • Increment ID (so, ID++)
    • Assign ID to Process

• Server Pseudocode
  • On receive message place in queue
  • Order Queue by ID – process Queue in order of ID
Implementing FIFO Ordering

- Local sequencer: Process locally maintain ID
  - ID is monotonic

- Client Pseudocode
  - Increment ID
  - Assign ID to message
  - Send message to all servers

- Server Pseudocode
  - Maintain a queue for each client
  - Order message by ID -- process in order

This is FIFO Order

Client maintain different IDs: local and monotonic
How can we implement total/FIFO ordering?

**System Assumptions**

**Current:**
- No server (node) failure.
- Network is reliable:
  - Messages are delivered eventually

**New:**
- Synchronous failures.
- Network is unreliable:
  - Messages can be lost
  - Messages can be duplicated
Do We Always Need Total Ordering?

• For some applications: yes!!!
  • My back account

• For others: ``causal ordering’’
  • Dependency between events are preserved
  • A → B: on all servers A is processed before B.

• How do I capture and enforce causal ordering???
Logical Clocks

• Each process maintains a ID – i.e., Clock
  • Initially set to 0

• Rules for updating Clock
  • Each event increments the ID
    • Event: Send msg, recv msg, or run function
  • Include ID in every message
  • On Receiving a message:
    • ID = max (my_ID, ID-in-msg) ++

• Only way to exchange information is through message exchange
  • IDs are only exchanged through message exchange

If x->z, then clock(z) > clock(x)
Ordering with Logical Clocks

• Which ordering do you get?
  • Total? FIFO? Causal?

• Is B before or after A?
• Is C before or after D?
• Is F before or after E?

If e -> e’, then clock(e’) > clock(e)
Logical clocks do help with inferring ordering: can confirm
Vector Clocks

- Each process maintains an array of ID
  - Array is of size N (N= # of processes)
  - All entries initialized to 0
  - Array is called a Vector clock (VC)

- Size (VC\_i) == N
- VC\_i[i]
  - Number of events at P\_i
- VC\_i[h] = K
  - Process P\_i is aware of the first k events at P\_h

If VC\_x <= VC\_z then x->z (x happens before z)
Vector Clocks

- Each process maintains an array of ID
  - Array is of size N (N = # of processes)
  - All entries initialized to 0

- Rules for updating Vector Clock for process Z
  - Each event increments \( VC_z[z] \)
    - So each event increments processes clock in vector
    - Event: Send msg, recv msg, or run function
  - Include ENTIRE vector in every message
  - On Receiving a message from process X –
    - This message will have \( VC_x[i] \) – X’s vector clock
    - Z updates its clock for each entry: \( VC_z[k] = \max(VC_z[k], VC_x[k]) + 1 \)

- Only way to exchange information is through message exchange
  - IDs are only exchanged through message exchange

If \( VC_x \leq VC_z \) then \( x \rightarrow z \) (\( x \) happens before \( z \))
Ordering with Vector Clocks

- Which ordering do you get?
  - Total? FIFO? Causal?

- Is B before or after A?
- Is C before or after D?
- Is F before or after E?

If $VC_x \leq VC_z$ then $x \rightarrow z$ (x happens before z)
Vector clocks are enough for FIFO and Causal ordering
Terminology Review

• **Total order**: All servers process events in same order
  - Linearizability: total order according to global time
  - Requires: magical global time or a global mutex

• **FIFO order**: all servers process events in order generated by client
  - More practical: you can trust client clock to be monotonic

• **Logical clock**: counters

• **Vector clock**: a vector of logical clocks
Logical Time

- Vector clocks provider better ordering than logical clocks
  - HOWEVER!! Expensive – scales with # of processes
    - Network message: include vector clocks
    - Processes: must maintain vector clocks

- Total ordering is a close second but too costly
  - Achievable with global sequencer
  - Ideal is Linearizability but impractical
- FIFO ordering: easy but less meaningful
- Causal ordering: goal for most distributed systems
  - Achievable with vector clocks