Agenda

• Last time
  • Finish P2P
  • Assignment #3 due
• This time (D222 Thurs 5-6:15pm, D223 Fri 3-4:15pm)
  • Chapter 11 (time and global states)
• Next time (Tuesday Apr 3)
  • More Chapter 11 Time and global states

Before we start: Assignment #4 and #5

• Old
  • Assignment #4 (no book): due Tues Apr 10
  • Assignment #5 (no book): out Tues April 3, proposal due Thurs April 12, due April Tues 24
• New
  • Assignment #4 (no book):
    • P2P out now, due Tues April 17 OR
    • “comparable” project proposal due Thurs Apr 5, due Tues Apr 17
  • Assignment #5 (only book): out April 5, due April Tues 24

Before we start: Schedule

<table>
<thead>
<tr>
<th>Sun</th>
<th>Tues</th>
<th>Thurs</th>
<th>Fri</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNS (10)</td>
<td>27</td>
<td>P2P, #3 due, #4 out</td>
<td>Time/global (11)</td>
</tr>
<tr>
<td>Coordination/agreement</td>
<td>3</td>
<td>Transactions (13)</td>
<td>12</td>
</tr>
<tr>
<td>Assignment #5 Out</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Transactions (13)</td>
<td>17</td>
<td>Dias transactions (14): RAM due</td>
<td></td>
</tr>
<tr>
<td>Dias transactions (14): RAM due</td>
<td>17</td>
<td>Dias transactions (14): Replication (15)</td>
<td></td>
</tr>
<tr>
<td>Replication (15)</td>
<td>24</td>
<td>Class WrapUp</td>
<td></td>
</tr>
</tbody>
</table>

Before we start: “comparable project”

• I think the P2P file system is very beneficial and fun, so I think you should do that
• However, you ARE allowed to pursue a different project
  • Which project? Let’s start a dialogue. Contact me.

Chapter 11: Time and Global States: Roadmap

• Problem definition: what would happen if there’s no attempt to synchronize clocks in a distributed system?
• Ground rules: Introduce a general model of computing (process, state, “happens before”, HW clock, etc.)
• Approximate synchronization of clocks
  • (internal) Synchronization in a synchronous system
  • Christian’s method (external)
  • Berkeley alg. (internal)
  • NTP (external)
• Logical time and logical clocks
  • Happened before
  • Logical clocks
• Global states

Introduction

• Time: important issue in DSs, must be measured accurately
  • To know when a particular event occurred
• No global clock
• Synchronization algorithms developed for:
  • Maintaining the consistency of distributed data
  • Eliminating the processing of duplicate updates
  • Checking the authenticity of a request sent to a server
• …
Introduction

- Distributed system = collection of N processes P_i, i=1,2,...,N: Independent, No shared memory
- S_i: state of process P_i (values of variables or objects in its local operating system environment, e.g., files)
- Process communication: done only by message passing

<table>
<thead>
<tr>
<th>Process P_i:</th>
<th>Set of actions that can modify the state of a process:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Send</td>
<td>S_i → S_i' Operations that transform the state S_i</td>
</tr>
</tbody>
</table>

Event = Occurrence of an action
- Sequence of events in a single process P_i: unique total ordering denoted by the relation ⇒
  \[ e ⇒ e' \text{ if event } e \text{ occurs before } e' \text{ at } P_i \]
- History of process P_i = series of events that take place within it, ordered by the relation ⇒
  \[ \text{History}(P_i) = h_i = \langle e_0, e_1, e_2, ... \rangle \]
- Software clock C_i(t)
  \[ C_i(t) = \alpha H_i(t) + \beta \]
- Physical clock: hardware clock value of process P_i

Synchronization of Physical Clocks

- External synchronization: clock C_i synchronized with an external source of time S (accurate to within bound D')
  \[ |S(t) - C(t)| < D, i=1,2,...,N \]
- Internal synchronization: clocks C_i synchronized with each other (agree within the bound D)
  \[ C_i(t) - C(t) < D, i=1,2,...,N \]

Faulty clocks (crash, arbitrary failure): don't satisfy these properties

Fault tolerance techniques can mask such failures

| Synchronization of Physical Clocks | Synchronization of a Synchronous System |
---|---|
- Synchronization of Physical Clocks
- Synchronization of a Synchronous System
- Network Time Protocol
Synchronization of a Synchronous System (1)

- **Simple case**: (internal) synchronization between two processes in a synchronous system
  - (definition of sync system: bounds are known for drift rate, max message transmission, time to execute each step)

If \( T_{\text{trans}} \) is constant:

\[
\text{Set Clock}(t + T_{\text{trans}})
\]

Internal synchronization reached

Synchronization of a Synchronous System (2)

- **Problem**: \( T_{\text{trans}} \) subject to variation and is unknown
- **Solution**: let \( \text{MAX} \) be the maximum possible value for one message delay, and \( \text{MIN} \) be the smallest

\[
\begin{array}{c|c|c}
\text{Scenario} & \text{Receiver assigns to its clock} & \text{Maximal skew of the clock} \\
\hline
1 & t + \text{MIN} & \text{MAX-MIN} \\
2 & t + \text{MAX} & \text{MAX-MIN} \\
3 & t + (\text{MAX} + \text{MIN}) / 2 & (\text{MAX-MIN}) / 2 \\
\end{array}
\]

Synchronization of a Synchronous System (3)

- **General case**: optimum bound that can be achieved when synchronizing \( N \) clocks = \( \Omega(1 - 1/N) \)
- **Practical distributed systems**: asynchronous, there is no upper bound \( \text{MAX} \)
  \[
  T_{\text{trans}} = \text{MIN} + x, \quad x > 0 \text{ and value of } x \text{ unknown}
  \]

Synchronization of a Synchronous System (4)

- **Cristian's method (1989)**: external synchronization
  - Use of a time server: connected to a device that receives signals from a source of UTC
  - Limit: server may fail
    - **Solution**: time should be provided by a group of synchronized servers

Synchronization of a Synchronous System (5)

- **Berkeley Algorithm (1989)**: internal synchronization
  - Developed for collections of computers running Berkeley UNIX
  - **Coordinator computer (master)**: polls the other computers whose clocks are to be synchronized (slaves)
    - The slaves send back their clock values
    - The master estimates their local clock times by observing the round-trip times; averages the values obtained including its own clock's reading
    - The master sends the amount by which each individual slave's clock requires adjustment
    - **Master fails**: another is elected to take over and function exactly as its predecessor; also: a "fault-tolerant average": don't consider outliers
Network Time Protocol

- Cristian’s method & Berkeley algorithm: designed for use within intranets
- Network Time Protocol:
  - Architecture for time service and a protocol to distribute time information over the Internet (potentially large delays)
  - Service NTP: provided by a network of servers located across the Internet
    - Primary servers: connected directly to an UTC time source (radio, satellite)
    - Secondary servers: synchronized with primary servers

Network Time Protocol

- NTP servers synchronize with each other (all via UDP):
  - Multicast:
    - Used on high-speed LAN
    - One or more servers periodically multicasts the time to the servers running in other computers connected to the LAN
    - Receiving computers set their clocks assuming a small delay
      - This mode can achieve only relatively low accuracies

Network Time Protocol

- Synchronization modes (cont’d):
  - Synchronization modes (cont’d):
    - Procedure-call: (similar to Christian’s algorithm)
      - One server accepts requests from other computers
      - Replies by sending its timestamp (current clock reading)
        - This mode is suitable where higher accuracies are required

Network Time Protocol

- Synchronization modes (cont’d):
  - Symmetric mode:
    - A pair of processes operate in symmetric mode and exchange messages bearing timing information