INTRODUCTION TO CONSENSUS

George Porter
May 14, 2018

ATTRIBUTION

- These slides are released under an Attribution-NonCommercial-ShareAlike 3.0 Unported (CC BY-NC-SA 3.0) Creative Commons license
- These slides incorporate material from:
  - Tanenbaum and Van Steen, Dist. Systems: Principles and Paradigms
  - Kyle Jamieson, Princeton University (also under a CC BY-NC-SA 3.0 Creative Commons license)
ANNOUNCEMENTS
HW 3 posted

RECALL OUR 2PC COMMIT PROBLEM

1. C → TC: “go!”

2. TC → A, B: “prepare!”

3. A, B → P: “yes” or “no”

4. TC → A, B: “commit!” or “abort!”
RECALL OUR 2PC COMMIT PROBLEM

- Who acts as TC?
- Which server(s) own the account of A? B?
- Who takes over if TC fails? What about if A or B fail?

DOING FAILOVER “CORRECTLY” ISN’T EASY

Which node takes over as backup?
DOING FAILOVER "CORRECTLY" ISN’T EASY

Okay, so specify some ordering
(manually, using some identifier)

DOING FAILOVER "CORRECTLY" ISN’T EASY

But who determines if 1 failed?

But who determines if 1 failed?
DOING FAILOVER “CORRECTLY” ISN’T EASY

Easy, right? Just ping and timeout!

Is the server or the network actually dead/slow?
WHAT CAN GO WRONG?

Two nodes think they are TC: “Split brain” scenario
WHAT CAN GO WRONG?

Safety invariant:
Only 1 node is TC at any single time

Another problem:
A and B need to know (and agree upon) who the TC is...

CONSENSUS (NOUN):

1. A general agreement about something
2. An idea or opinion that is shared by all the people in a group
CONSENSUS

• Given a set of processors, each with an initial value:

• **Termination**: All non-faulty processes eventually decide on a value

• **Agreement**: All processes that decide do so on the same value

• **Validity**: The value that has been decided must have proposed by some process

CONSENSUS USED IN SYSTEMS

Group of servers attempting:

• Make sure all servers in group receive the same updates in the same order as each other

• Maintain own lists (views) on who is a current member of the group, and update lists when somebody leaves/fails

• Elect a leader in group, and inform everybody

• Ensure mutually exclusive (one process at a time only) access to a critical resource like a file
### STEP ONE: DEFINE YOUR SYSTEM MODEL

<table>
<thead>
<tr>
<th><strong>Network model:</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>• Synchronous (time-bounded delay) or asynchronous (arbitrary delay)</td>
</tr>
<tr>
<td>• Reliable or unreliable communication</td>
</tr>
<tr>
<td>• Unicast or multicast communication</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Node failures:</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>• Fail-stop (correct/dead) or Byzantine (arbitrary)</td>
</tr>
</tbody>
</table>

---

### STEP ONE: DEFINE YOUR SYSTEM MODEL

<table>
<thead>
<tr>
<th><strong>Network model:</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>• Synchronous (time-bounded delay) or asynchronous (arbitrary delay)</td>
</tr>
<tr>
<td>• Reliable or unreliable communication</td>
</tr>
<tr>
<td>• Unicast or multicast communication</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Node failures:</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>• Fail-stop (correct/dead) or Byzantine (arbitrary)</td>
</tr>
</tbody>
</table>
IMPORTANCE OF THIS STUFF

• Werner Vogels, Amazon CTO

• Job openings in my group

• What kind of things am I looking for in you?

• “You know your distributed systems theory: You know about logical time, snapshots, stability, message ordering, but also acid and multi-level transactions. You have heard about the FLP impossibility argument. You know why failure detectors can solve it (but you do not have to remember which one diamond-w was). You have at least once tried to understand Paxos by reading the original paper.”

UC San Diego