The Part-Time Parliament

Leslie Lamport, 1998

Carlos Eduardo B. Bezerra

Faculty of Informatics
Università della Svizzera italiana

May 25th, 2011

Outline

1. Introduction
   - A long way to Paxos
   - The consensus problem
2. The Paxos Algorithm
   - Notations used
   - Consistency conditions
   - Proof sketch

Introduction

Interesting information about 'The Part-Time Parliament':

- the algorithm was first described in 1989 as a technical report [1]
- the paper was submitted in 1990...and accepted in 1998
- several other publications, such as [2, 3, 4], had explaining it as their main purpose...

The paper mostly uses an analogy with the fictional ancient island of Paxos, where:

- The parliament's (group's) legislators (processes) and their messengers (communication channels) may be unpredictably absent;
- There may be several ballots for each decree to be passed (rounds in each consensus instance);
- Legislators – and messengers – may be lazy or absent, but not dishonest (non-Byzantine failure model*)

* The algorithm was extended later, however, to tolerate Byzantine failures [5]

Definition

Consensus is the process of agreeing on one single result among a group of participants. Assuming a collection of processes that can propose values, a consensus algorithm ensures that:

- only a single value among the proposed ones may be chosen;
- if no value has been proposed, then no value is chosen;
- a process never learns that a value has been chosen unless it really has been.

The Paxos algorithm

Some notations used

A value is chosen after a series of numbered **ballots**. For each ballot *B*, we have that:

- B_{bal} is its ballot number;
- B_{qrm} is a nonempty finite set of processes (quorum);
- B_{vot} is the set of processes who voted in B – each process may decide to participate (vote) or not in B;
- B_{val} is the value being voted on in that ballot.

A ballot is said to be *successful* when $B_{qrm} \subseteq B_{vot}$ – which means that every quorum member voted.

The Paxos algorithm

Some more notations

In each ballot, each process can cast a vote (accept or not a given proposed value). A vote v has the components:

- v_p, which is the process p who cast it;
- v_{bal}, the number of the ballot for which it was cast;
- v_{val}, a value voted on.

For the set β of ballots, the set $Votes(\beta)$ is defined as $\{v : \exists B \in \beta \mid v_p \in B_{vot} \wedge v_{bal} = B_{bal} \wedge v_{val} = B_{val}\}$

The Paxos algorithm
Still more notations...

We then define *MaxVote(b, p, β)* as the vote v with highest v_{bal} in $\{v \in Votes(\beta) : (v_p = p) \wedge (v_{bal} < b)\} \cup \{null\}$,
where *null* means that p didn't vote in any $B \in \beta$ ($null_{bal} = -\infty$)

We also define *MaxVote(b, Q, β)* as the vote v, such that v_{bal} is the highest among all *MaxVote(b, p, β)* with $p \in Q$

With such notations, we can specify the conditions which ensure consistency and allow for progress.

Consistency conditions

Consistency is guaranteed if the following conditions are ensured:

- $B1(\beta)$
  $\forall B, B' \in \beta : (B \neq B') \Rightarrow (B_{bal} \neq B'_{bal})$
- $B2(\beta)$
  $\forall B, B' \in \beta : B_{qrm} \cap B'_{qrm} \neq \emptyset)$
- $B3(\beta)$
  $\forall B, B' \in \beta :$
  $(MaxVote(B_{bal}, B_{qrm}, \beta)_{bal} \neq -\infty) \Rightarrow$
  $\Rightarrow (B_{val} = MaxVote(B_{bal}, B_{qrm}, \beta)_{val})$

Proof sketch

LEMMA. If $B1(\beta)$, $B2(\beta)$ and $B3(\beta)$ hold, then

$$((B_{qrm} \subseteq B_{vot}) \wedge (B'_{bal} > B_{bal})) \Rightarrow (B'_{val} = B_{val})$$

for any B, B' in β.

PROOF OF LEMMA. Let $\psi(B, \beta)$ be defined as:

$$\psi(B, \beta) = \{B' \in \beta : (B'_{bal} > B_{bal}) \wedge (B'_{val} \neq B_{val})\}$$

Assume that $B_{qrm} \subseteq B_{vot}$. By contradiction, assume that $\psi(B, \beta) \neq \emptyset$.

(1) As $\psi(B, \beta) \neq \emptyset$, we have that $\exists C : C_{bal} = min\{B'_{bal} : B' \in \psi(B, \beta)\}$
(2) From (1) and from the definition of $\psi(B, \beta)$, we have that
$C_{bal} > B_{bal}$

Proof sketch

continuing…

(3) As $B_{qrm} \subseteq B_{vot}$ and $\forall B, B' \in \beta : B_{qrm} \cap B'_{qrm} \neq \emptyset$, we have that $B_{vot} \cap C_{qrm} \neq \emptyset$

(4) As $C_{bal} > B_{bal}$ and $B_{vot} \cap C_{qrm} \neq \emptyset$, we have that $MaxVote(C_{bal}, C_{qrm}, \beta)_{bal} \geq B_{bal}$

(5) From (4) and the definition of $MaxVote(C_{bal}, C_{qrm}, \beta)$, we know that such vote exists in $Votes(\beta)$ (i.e. it is not *null*).

(6) From (5) and $B3(\beta)$, we have that $MaxVote(C_{bal}, C_{qrm}, \beta)_{val} = C_{val}$

(7) From (6), and as $C_{val} \neq B_{val}$ (from $\psi(B, \beta)$), we have that $MaxVote(C_{bal}, C_{qrm}, \beta)_{val} \neq B_{val}$

- using the `pause` command:
  - First item.
  - Second item.
- using overlay specifications:
  - First item.
  - Second item.
- using the general `uncover` command:
  - First item.
  - Second item.

- using the `pause` command:
  - First item.
  - Second item.
- using overlay specifications:
  - First item.
  - Second item.
- using the general `uncover` command:
  - First item.
  - Second item.

- using the `pause` command:
  - First item.
  - Second item.
- using overlay specifications:
  - First item.
  - Second item.
- using the general `uncover` command:
  - First item.
  - Second item.

- using the `pause` command:
  - First item.
  - Second item.
- using overlay specifications:
  - First item.
  - Second item.
- using the general `uncover` command:
  - First item.
  - Second item.

- using the `pause` command:
  - First item.
  - Second item.
- using overlay specifications:
  - First item.
  - Second item.
- using the general `uncover` command:
  - First item.
  - Second item.

- using the `pause` command:
  - First item.
  - Second item.
- using overlay specifications:
  - First item.
  - Second item.
- using the general `uncover` command:
  - First item.
  - Second item.

Summary

- The **first main message** of your talk in one or two lines.
- Outlook
  - Something you haven't solved.
  - Something else you haven't solved.

References

[1] Lamport, L. **The part-time parliament**, Technical Report 49, Systems Research Center, Digital Equipment Corp., 1989

[2] Lamport, L. **Paxos Made Simple**, ACM SIGACT News, 2001

[3] De Prisco, R., Lampson, B., Lynch, N. **Revisiting the Paxos Algorithm**, Distributed Algorithms, 1997

[4] Lampson, B. **How to build a highly available system using consensus**, Distributed Algorithms, 1996

[5] Castro, M., Liskov, B., **Practical Byzantine Fault Tolerance**, Operating Systems Design and Implementation, 1999