

Seminar Report: Paxy

Maria Gabriela Valdes and Victoria Beleuta

December 9, 2014

1 Introduction

This assignment gave us the opportunity to learn and implement the *Paxos* algorithm. This algorithm is a protocol for solving consensus between processes in a distributed system. Consensus consists in agreeing on a proposed value among a group of participants. In this particular case, we have a set of *Proposers* that propose different values, to a set of *Acceptors* until a single and unique value is agreed between all *Proposers* and *Acceptors*.

2 Work done

We have completed the code files provided to correctly implement the algorithm. Our source code consists of two folders: *src* and *src-sep*. The first one contains the erlang files implementig the Paxos algorithm with both proposers and acceptors starting in the same instance. The folder *src-sep* we have modules to give the possibility to start the proposers in one machine and the acceptors in a different machine.

3 Experiments

First we run an experiment in the same machine with 3 proposers and 5 acceptors. After compiling the code files located in *src* and running the paxy module `pxy:start([200,200,200])`. Proposer 0, *willard* proposes color *blue*, and in round 2, the acceptors voted for *kilgore* and decided on color *blue*.

The screenshot shows the Paxos Algorithm simulation interface. On the left, there are three Proposers (1, 2, 3) and five Acceptors (1-5). Each Proposer has a box for Round, Proposal, and Proposal. Each Acceptor has a box for Voted and Promised. The log on the right shows the sequence of messages: Phase 1, Phase 2, and Phase 3. The log indicates that the algorithm terminates with a decision on round 2, proposal 0, color 255.

i) In this experiment we delay *prepare* messages and we see that the algorithm terminates even with the delays shown in the trace.

The screenshot shows the Paxos Algorithm simulation interface. On the left, there are three Proposers (1, 2, 3) and five Acceptors (1-5). Each Proposer has a box for Round, Proposal, and Proposal. Each Acceptor has a box for Voted and Promised. The log on the right shows the sequence of messages: Phase 1, Phase 2, and Phase 3. The log indicates that the algorithm terminates with a decision on round 2, proposal 0, color 255. The log also shows delays in the prepare messages.

Now we add delays before the *accept* messages, the algorithm terminates voting in round 2 for *kilgore* and color *blue*.

The screenshot shows the Paxos Algorithm simulation interface. On the left, there are two panels: 'Proposers' and 'Acceptors'. The 'Proposers' panel shows three proposers: Proposer 1 (Round: {0,willard}, Proposal: {0,0,255}), Proposer 2 (Round: {2,kilgore}, Proposal: {0,0,255}), and Proposer 3 (Round: {1,kurtz}, Proposal: {0,0,255}). The 'Acceptors' panel shows five acceptors: Acceptor 1 (Voted: {2,kilgore}, Promised: {2,kilgore}), Acceptor 2 (Voted: {2,kilgore}, Promised: {2,kilgore}), Acceptor 3 (Voted: {2,kilgore}, Promised: {2,kilgore}), Acceptor 4 (Voted: {2,kilgore}, Promised: {2,kilgore}), and Acceptor 5 (Voted: {2,kilgore}, Promised: {2,kilgore}). On the right, there is a log of messages showing the progress of the algorithm, including votes, promises, and delays.

Lastly, we add delays both before *prepare* and *accept* messages and the acceptors reach the same consensus again.

The screenshot shows the Paxos Algorithm simulation interface with delays added. The 'Proposers' and 'Acceptors' panels are the same as in the previous screenshot. The log on the right shows the progress of the algorithm, including votes, promises, and delays. The log shows that the acceptors have reached the same consensus again, with the same round and proposal being accepted.

ii) We stop sending *sorry* messages to the processor and we run the experiment to try to reach a consensus.

The screenshot shows the Paxos Algorithm simulation interface. On the left, there are three Proposers and five Acceptors. Proposer 1 has Round: {0,willard} and Proposal: {0,0,255}. Proposer 2 has Round: {2,kilgore} and Proposal: {0,0,255}. Proposer 3 has Round: {1,kurtz} and Proposal: {0,0,255}. Acceptors 1 through 5 show their current state: Voted: {2,kilgore} and Promised: {2,kilgore} for Acceptors 1, 2, 3, and 4; and Voted: {2,kilgore} and Promised: {2,kilgore} for Acceptor 5. On the right, a log window titled '1. beam.smp' shows a sequence of messages including Phase 1 votes, Phase 2 proposals, and Phase 2 votes, all leading to a consensus on the color blue (kilgore).

iii) In this experiment, we randomly drop *promise* messages, and we look out for the consensus that the acceptors arrive to. Even though a promise message in round 2 is dropped, they still agree on the color blue.

This screenshot is identical to the one above, showing the Paxos Algorithm simulation interface. It displays the same Proposers, Acceptors, and message log, illustrating a scenario where a promise message is dropped but consensus is still reached on the color blue.

Now, we run an experiment dropping *vote* messages. Again, the acceptors manage to agree on the proposed color blue, even with a *vote* message dropped because the other messages still amount to the majority of acceptors.

Paxos Algorithm

Proposers

- Proposer 1: Round: {0,willard} Proposal: {0,0,255}
- Proposer 2: Round: {2,kilgore} Proposal: {0,0,255}
- Proposer 3: Round: {1,kurtz} Proposal: {0,0,255}

Acceptors

- Acceptor 1: Voted: {2,kilgore} Promised: {2,kilgore}
- Acceptor 2: Voted: {2,kilgore} Promised: {2,kilgore}
- Acceptor 3: Voted: {2,kilgore} Promised: {2,kilgore}
- Acceptor 4: Voted: {2,kilgore} Promised: {2,kilgore}
- Acceptor 5: Voted: {2,kilgore} Promised: {2,kilgore}

1. beam.smp

```
[Proposer kilgore] aborting collection of PROMISES in round {0,kilgore}
[Proposer kurtz] Phase 1: round {1,kurtz} proposal {255,0,0}
[Acceptor a] Phase 1: voted {0,willard} promised {1,kurtz} colour {0,0,255}
[Acceptor b] Phase 1: voted {0,willard} promised {1,kurtz} colour {0,0,255}
[Acceptor c] Phase 1: voted {0,willard} promised {1,kurtz} colour {0,0,255}
[Acceptor d] Phase 1: voted {0,willard} promised {1,kurtz} colour {0,0,255}
[Acceptor e] Phase 1: voted {0,willard} promised {1,kurtz} colour {0,0,255}
[Proposer kurtz] [b,c,d] acceptors promised {0,0,255} in round {1,kurtz}
[Proposer kilgore] Phase 1: round {1,kilgore} proposal {0,255,0}
[Proposer kurtz] Phase 2: round {1,kurtz} proposal {0,0,255}
[Acceptor a] Phase 2: voted {1,kurtz} promised {1,kurtz} colour {0,0,255}
[Acceptor b] Phase 2: voted {1,kurtz} promised {1,kurtz} colour {0,0,255}
[Proposer kurtz] [a,a,b] acceptors voted {0,0,255} in round {1,kurtz}
-----[Acceptor c] Phase 1: VOTE message dropped in round {1,kurtz}
[Acceptor d] Phase 2: voted {1,kurtz} promised {1,kurtz} colour {0,0,255}
[Acceptor e] Phase 2: voted {1,kurtz} promised {1,kurtz} colour {0,0,255}
[Acceptor c] Phase 2: voted {1,kurtz} promised {1,kurtz} colour {0,0,255}
[Proposer kilgore] aborting collection of PROMISES in round {1,kilgore}
[Proposer kilgore] Phase 1: round {2,kilgore} proposal {0,255,0}
[Acceptor a] Phase 1: voted {1,kurtz} promised {2,kilgore} colour {0,0,255}
[Acceptor c] Phase 1: voted {1,kurtz} promised {2,kilgore} colour {0,0,255}
[Acceptor b] Phase 1: voted {1,kurtz} promised {2,kilgore} colour {0,0,255}
[Acceptor d] Phase 1: voted {1,kurtz} promised {2,kilgore} colour {0,0,255}
[Acceptor e] Phase 1: voted {1,kurtz} promised {2,kilgore} colour {0,0,255}
[Proposer kilgore] [c,b,d] acceptors promised {0,0,255} in round {2,kilgore}
[Proposer kilgore] Phase 2: round {2,kilgore} proposal {0,0,255}
[Acceptor b] Phase 2: voted {2,kilgore} promised {2,kilgore} colour {0,0,255}
[Acceptor c] Phase 2: voted {2,kilgore} promised {2,kilgore} colour {0,0,255}
[Proposer kilgore] [b,b,c] acceptors voted {0,0,255} in round {2,kilgore}
[Acceptor d] Phase 2: voted {2,kilgore} promised {2,kilgore} colour {0,0,255}
[Acceptor e] Phase 2: voted {2,kilgore} promised {2,kilgore} colour {0,0,255}
[Acceptor a] Phase 2: voted {2,kilgore} promised {2,kilgore} colour {0,0,255}
3> |
```

When we run the experiment dropping both *promise* and *vote* messages, we see that the algorithm still terminates successfully due to the fact that the majority of acceptors whose messages were not dropped agreed.

Paxos Algorithm

Proposers

- Proposer 1: Round: {0,willard} Proposal: {0,0,255}
- Proposer 2: Round: {2,kilgore} Proposal: {0,0,255}
- Proposer 3: Round: {1,kurtz} Proposal: {0,0,255}

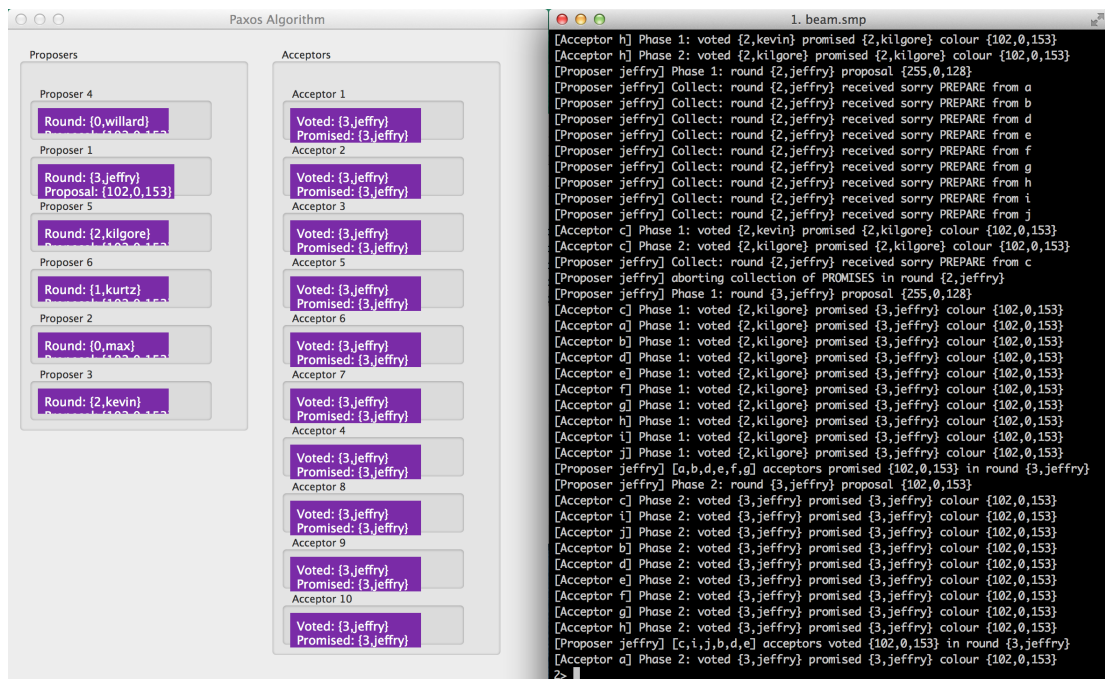
Acceptors

- Acceptor 1: Voted: {2,kilgore} Promised: {2,kilgore}
- Acceptor 2: Voted: {2,kilgore} Promised: {2,kilgore}
- Acceptor 3: Voted: {2,kilgore} Promised: {2,kilgore}
- Acceptor 4: Voted: {2,kilgore} Promised: {2,kilgore}
- Acceptor 5: Voted: {2,kilgore} Promised: {2,kilgore}

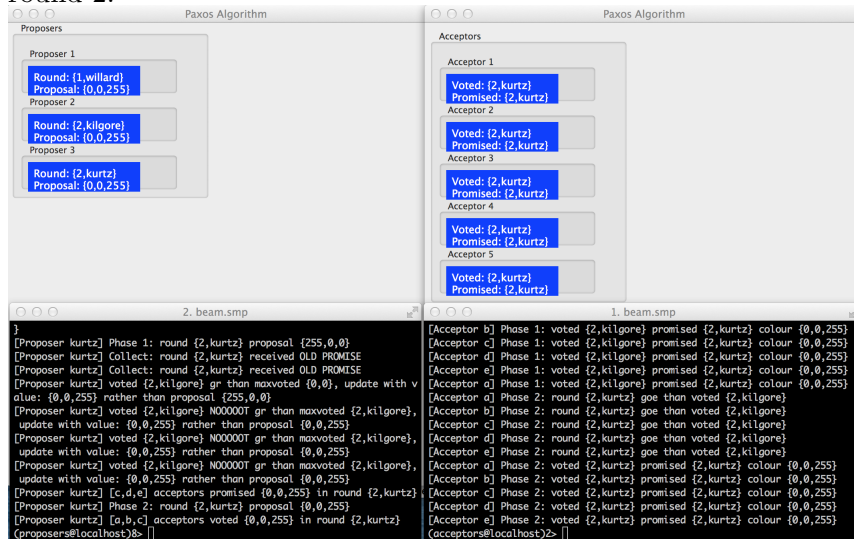
1. beam.smp

```
[Acceptor e] Phase 1: voted {0,0} promised {0,willard} colour na
[Proposer willard] Phase 2: round {0,willard} proposal {0,0,255}
[Acceptor e] Phase 2: voted {0,willard} promised {0,willard} colour {0,0,255}
[Acceptor a] Phase 2: voted {0,willard} promised {0,willard} colour {0,0,255}
[Acceptor b] Phase 2: voted {0,willard} promised {0,willard} colour {0,0,255}
-----[Acceptor c] Phase 1: VOTE message dropped in round {0,willard}
[Acceptor d] Phase 2: voted {0,willard} promised {0,willard} colour {0,0,255}
[Proposer willard] [e,e,a] acceptors voted {0,0,255} in round {0,willard}
[Acceptor c] Phase 2: voted {0,willard} promised {0,willard} colour {0,0,255}
[Proposer kurtz] aborting collection of PROMISES in round {0,kurtz}
[Proposer kilgore] aborting collection of PROMISES in round {0,kilgore}
[Proposer kurtz] Phase 1: round {1,kurtz} proposal {255,0,0}
[Acceptor d] Phase 1: voted {0,willard} promised {1,kilgore} colour {0,0,255}
[Acceptor e] Phase 1: voted {0,willard} promised {1,kilgore} colour {0,0,255}
[Proposer kilgore] [d,e,e] acceptors promised {0,0,255} in round {1,kilgore}
[Acceptor a] Phase 1: voted {0,willard} promised {1,kilgore} colour {0,0,255}
[Acceptor b] Phase 1: voted {0,willard} promised {1,kilgore} colour {0,0,255}
[Acceptor c] Phase 1: voted {0,willard} promised {1,kilgore} colour {0,0,255}
[Proposer kilgore] Phase 2: round {1,kilgore} proposal {0,0,255}
[Acceptor d] Phase 1: voted {0,willard} promised {1,kurtz} colour {0,0,255}
-----[Acceptor e] Phase 1: PROMISE message dropped in round {1,kurtz}
-----[Acceptor a] Phase 1: PROMISE message dropped in round {1,kurtz}
[Acceptor b] Phase 1: voted {0,willard} promised {1,kurtz} colour {0,0,255}
[Proposer kurtz] [d,b,b] acceptors promised {0,0,255} in round {1,kurtz}
[Acceptor e] Phase 1: voted {0,willard} promised {1,kurtz} colour {0,0,255}
[Proposer kurtz] Phase 2: round {1,kurtz} proposal {0,0,255}
[Acceptor d] Phase 2: voted {1,kurtz} promised {1,kurtz} colour {0,0,255}
[Acceptor e] Phase 2: voted {1,kurtz} promised {1,kurtz} colour {0,0,255}
[Proposer kurtz] [d,d,e] acceptors voted {0,0,255} in round {1,kurtz}
```

iv) We increase the number of acceptors to 10 and the number of proposers to 6. The acceptors still get to a consensus in round 3 on color *purple*.



v) In this experiment we changed the modules to adjust them such that we can run the proposers in one machine and the acceptors in a different one and as they communicate they reach an agreement on the color *blue* in round 2.



4 Open questions

Try to answer all the open questions in the documentation. When possible, do experiments to support your answers.

5 Personal opinion

Provide your personal opinion of the seminar, including whether it should be included in next year's course or not.