# Assignment 3

---

**Due**  26 Oct by 16:59        **Points**  100

---

## Assignment 3 - Paxos

### Marks

For undergraduates, the marks awarded for this assignment are worth 30% of the total mark for DS. For postgraduate students, the marks awarded for this assignment are worth 20% of the total mark for DS.

**PLEASE NOTE:** If your code does not compile and run **the awarded mark is an automatic zero**. Our expectation is that you are able to code and submit a tested and working copy of your code. There will not be any partial marks and you are expected to debug and fix errors and test that your code works on the Linux image at University (CATS machines or ssh remotely to `uss.cs.adelaide.edu.au`).

### Assignment Description

#### Objective

To gain an understanding of consensus and voting protocols in the presence of failures of one or more of the participants.

# Welcome to the Suburbs Council Election!

This year, Suburbs Council is holding elections for council president. Any member of its nine person council is eligible to become council president.

**Member M1** – M1 has wanted to be council president for a very long time. M1 is very chatty over social media and responds to emails/texts/calls almost instantly. It is as if M1 has an in-brain connection with their mobile phone!

**Member M2** – M2 has also wanted to be council president for a very long time, except their very long time is longer than everybody else's. M2 lives in a remote part of the Suburbs and thus their internet connection is really poor, almost non-existent. Responses to emails come in very late, and sometimes only to one of the emails in the email thread, so it is unclear whether M2 has read/understood them all. However, M2 sometimes likes to work at Café @ Bottom of the Hill. When that happens, their responses are instant and M2 replies to all emails.

**Member M3** – M3 has also wanted to be council president. M3 is not as responsive as M1, nor as late as M2, however sometimes emails completely do not get to M3. The other councilors suspect that it's because sometimes M3 goes on retreats in the woods at the top of the Suburbs, completely disconnected from the world.

**Members M4-M9** have no particular ambitions about council presidency and no particular preferences or animosities, so they will try to vote fairly. Their jobs keep them fairly busy and as such their response times will vary.

How does voting happen: On the day of the vote, one of the councilors will send out an email/message to all councilors with a proposal for a president. A majority (half+1) is required for somebody to be elected president.

# YOUR TASK:

Write a program that implements a Paxos voting protocol for Suburbs Council President that is fault tolerant and resilient to various failure types, some of which are shown in the above. Communication happens strictly via sockets. You are responsible for the message design.

## Assessment

Your assignment will be marked out of 100 points, as following:

- 10 points - Paxos implementation works when two councillors send voting proposals at the same time

- 30 points – Paxos implementation works in the case where all M1-M9 have immediate responses to voting queries

- 30 points – Paxos implementation works when M1 – M9 have responses to voting queries suggested by the profiles above, including when M2 or M3 propose and then go offline

- 20 points – Testing harness for the above scenarios + evidence that they work (in the form of printouts)

- 10 points for the quality of your code:

**Code Quality Checklist**

**Do!**

```
o write comments above the header of each of your methods, describing
o what the method is doing, what are its inputs and expected outputs
o describe in the comments any special cases
o create modular code, following cohesion and coupling principles
```

**Don't!**

```
o use magic numbers
o use comments as structural elements (see video)
o mis-spell your comments
o use incomprehensible variable names
o have long methods (not more than 80 lines)
o allow TODO blocks
```

# Bonus

- 10 points – Paxos implementation works with a number 'n' of councilors with four profiles of response times: immediate;  medium; late; never
- 50 points – (you can use these points in this assignment, or in any other subsequent assignment) – Fast Byzantine Paxos implementation that works when councilors lie, collude, or intentionally do not participate in some voting queries but participate in others.

## Handin

Use the Websubmission system.

**IMPORTANT** If your code does not compile and run **the awarded mark is 0**. For details on how to submit this exercise read through the steps of this assignment.

## Procedure for this Assignment

### Step 0: Getting to know Subversion

Subversion, also known as svn, is a powerful version control system to help maintain a coherent copy of a project that can be worked on from multiple locations. We will use Subversion as the handin mechanism throughout this course. Click **here** **(http://www.cs.adelaide.edu.au/docs/svn-instr.pdf)** to learn about the features of interest to us.

### Step 1: Creating the assignment directory in your svn repository

**Current versions of the svn client program.**

Open a terminal window on your machine, and cut-and-paste the following command:

```
svn mkdir --parents -m "assignments" https://version-control.adelaide.edu.au/svn/aXXXXXX/YYYY/s2/ds/assignment3  (https://version-control.adelaide.edu.au/svn/aXXXXXX/YYYY/s2/ds/assign1)
```

This command will create an empty directory named YYYY/s2/ds/assignment3 in an svn repository that we have set up for you on the machine version-control.adelaide.edu.au. Your svn repository is called https://version-control.adelaide.edu.au/svn/aXXXXXXX/

Note: Replace XXXXXXX with your student ID and YYYY with the four digits representing the year we are in.

**Old versions of the svn client program.**

NOTE Older versions of the svn client, (pre 1.5), do not support the --parents option. If you are using an older client, the following explicit steps must be used to create the YYYY/s2/ds/assignment3 directory in your svn repository. If a directory already exists and you attempt to recreate it, you will receive an error message complaining about MKCOL.

Create the YYYY directory if it does not exist.

```
svn mkdir -m "assignment 0" https://version-control.adelaide.edu.au/svn/aXXXXXXX/YYYY ➦ (https://version-c
ontrol.adelaide.edu.au/svn/aXXXXXXX/YYYY)
```

➦ **(https://version-control.adelaide.edu.au/svn/aXXXXXXX/YYYY)** Create the YYYY/s2 directory if it does not exist.

```
svn mkdir -m "assignment 0" https://version-control.adelaide.edu.au/svn/aXXXXXXX/YYYY/s2 ➦ (https://versio
n-control.adelaide.edu.au/svn/aXXXXXXX/YYYY/s2)
```

Create the YYYY/s2/ds directory if it does not exist.

```
svn mkdir -m "assignment 0" https://version-control.adelaide.edu.au/svn/aXXXXXXX/YYYY/s2/ds ➦ (https://ve
rsion-control.adelaide.edu.au/svn/aXXXXXXX/YYYY/s2/ds)
```

Create the YYYY/s2/ds/assignment3 directory.

```
svn mkdir -m "assignment 0" https://version-control.adelaide.edu.au/svn/aXXXXXXX/YYYY/s2/ds/assignment3
➦ (https://version-control.adelaide.edu.au/svn/aXXXXXXX/YYYY/s2/ds/assign1)
```

## Step 2: Checking out a working version of your assignment.

In your home account, change to a directory above the one where you want to place your working files for this assignment. Then type:

```
svn checkout https://version-control.adelaide.edu.au/svn/aXXXXXXX/YYYY/s2/ds/assignment3 ds-YY-s2-assign3
```

A new directory called ds-YY-s2-assign3 will now be created in your current working directory. The contents of this now become a working-copy of the files for all assignments for this offering of Distributed Systems. Note that this working copy contains hidden sub-directories needed by svn to operate correctly.

Note that you can have more than one working copy of a project. For example you could have one working copy at home and one at university and you can use svn to help keep them in sync. See the **svn documentation** ➦ **(http://www.cs.adelaide.edu.au/docs/svn-instr.pdf)** for details on how this can be done. However, for now, we will assume you have just the one working copy.

## Step 3: Saving your files in your repository

The assignments require you to edit and create files. If you create a new file or directory that is part of your assignment you must add it to the list of files managed by svn. You can add a file, Test.java, to the svn repository by typing:

```
svn add Test.java
```

Note that the directory containing the new file must already have been added to svn.

As you work on your assignment, you should commit your changes early and often. That is, you should regularly ask svn to copy changed files in your working copy to your repository. To commit

your files to the repository you change to the directory containing the working copy of your files and type:

```
svn commit -m "meaningful message about what I just changed."
```

Note, that the message after the -m in the line above can be anything. However, it should be relevant to the state of the project at this moment. Also note that running commit is the only way your repository will get to know about the latest versions of your files. When we mark your submission we will look at what has been committed to your repository and, also, how often and what times you committed to your repository.

There are two major benefits to be gained by maintaining an up to date copy of your assignment in the svn repository. Firstly, svn can act as a backup mechanism. If you accidentally lose a file you can retrieve it from the svn repository. Secondly, when you post questions on the forums the lecturer(s) can access your svn repository and give you more helpful answers.

## Step 4: Assignment Submission

The next step is to submit your assignment using the Computer Science **Web Submission System** ⬀ **(https://cs.adelaide.edu.au/services/websubmission/)** .

You will need to submit:

- Your code
- Your testing harness + evidence that the above tests work
- A readme file containing instructions about how to compile and run your code

By now you have written your assignment, committed your changed files to the repository along the way and thoroughly tested them. The Web Submission System will not mark your assignment but will perform some limited checks using the current versions of your files in your repository. When you attempt to make a new submission you will be presented with an assessment cover sheet. This includes a link to your repository so you can check what files you are submitting. The cover sheet also includes a declaration that you are submitting your own work.

Note that, if the testing highlights some errors then you can always edit your files, commit them and make a new submission using the Web Submission System.