# CSCE–438 / Spring 2018 Homework 3.2

Homework 3.1: A Scalable and Highly Available SNS Service

70 pts

## 1 Overview

The objective of this assignment is to develop the next version of the Tiny SNS service that is scalable to a large number of users with significantly more workload, is fault tolerant and highly available (i.e., failures in the system are handled transparently to the user. You are not required to use the provided client, but you are required to follow the provided gRPC interface. For this assignment the following must be considered:

1. The Tiny SNS Service functionality that was provided in Homework #2 is still required for this assignment. Thus, it might be wise to start with the provided solution for that homework.
2. You have 4 machines available. Your implementation can start at most 8 master/slave servers and 1 routing server. You should have at most 1 master and 1 slave server on each machine.
   1. One of the 4 masters is also Routing server for giving clients ip/port information of an available master.
   2. One of the master servers is an Available master. It is waiting for clients connection and provide SNS service.
   3. The two remaining masters are standing by and one of those will be elected as a new Available server if the current Available server crashes (it will take over the previous available master’s job when crashed).

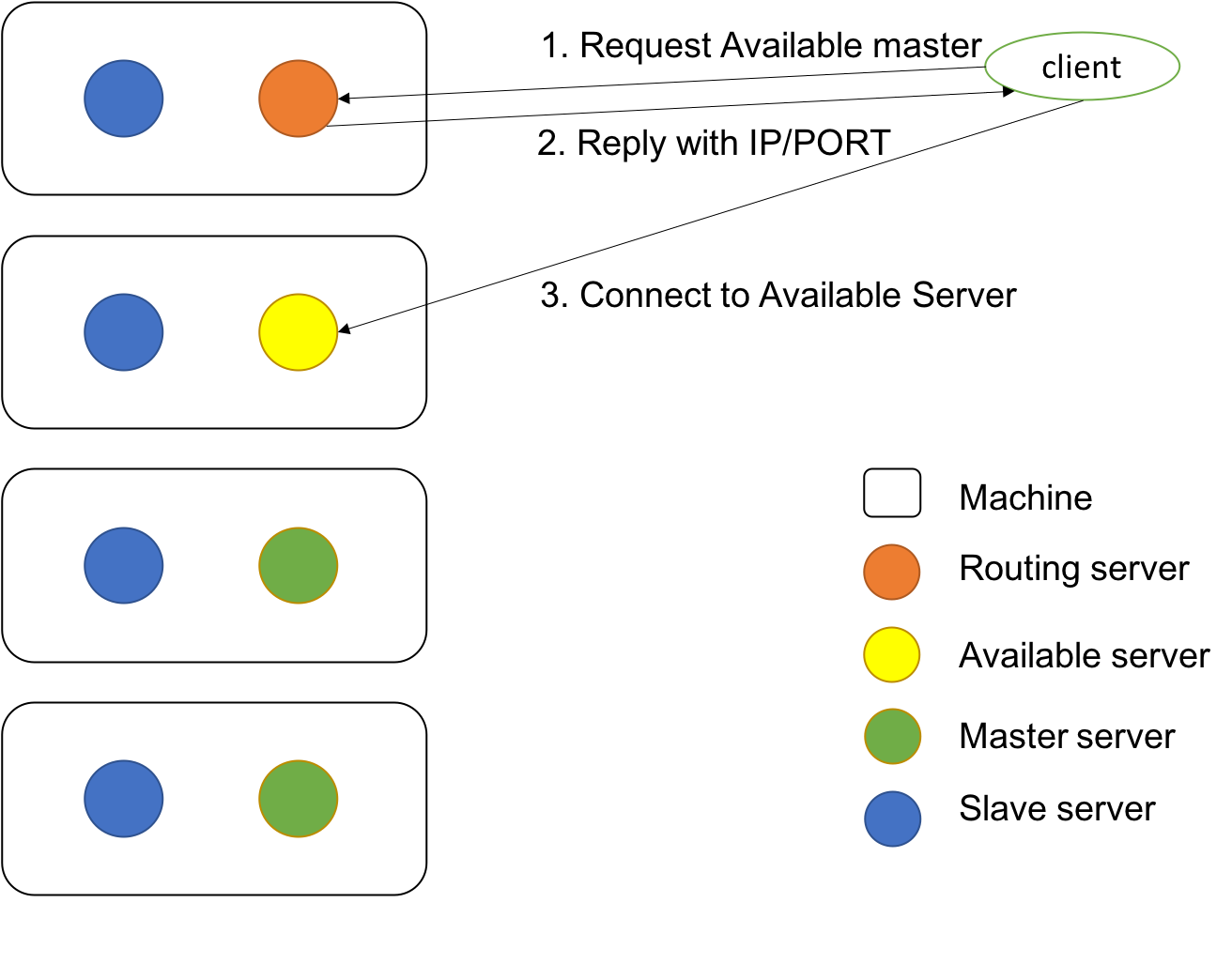
An example deployment is (refer to Figure1)

In Machine1 : 1 routing/master server and its slave are running.

In Machine2 : 1 available master server and its slave are running.

In Machine3: a master server and its slave are running.

In Machine4: a master server and its slave are running.

1. Your system should contain startup scripts that start your system.
2. You can assume that the machine on which the master process runs, is always available, although processes on it may still crash (i.e., killed by us).
3. The failure of ANY process except for Routing Server in the system is possible and you will need to take this into account.
4. We will “kill” at most 1 process within any 30 seconds time window. Thus, we will not attempt to crash a second process immediately after crashing a first process. We will wait 30 seconds. You can use this time interval to restart the crashed process (hint: slave can monitor whether master is crashed or not).
5. Data that has been stored persistently on hard disk will not be corrupted, i.e., you can rely on its accuracy.
   1. When an Available server crashes, the new elected Available server doesn’t need to take over data stored previously by the Available server (this requirement applies ONLY to this homework!).
6. Client should be always available for SNS service. That is, when an Available server crashes, the clients need to reconnect to new elected Available server. (Hint: client can ask the routing server a new available server)
7. The routing server will never crash in this homework

<Figure 1. An example deployment>

## 2 What to Hand In

## 2.1 Design

Start with the provided code for HW2. Based on your design, you may find that significant portions of the provided code are no longer needed. Feel free to start from scratch with coding as well, but abide by the gRPC and protobuf interfaces.

Before you start hacking away, write a design document. The result should be a system level design document, which you hand in along with the source code. Do not get carried away with it, but make sure it convinces the reader that you know how to attack the problem. List and describe the components of the system.

## 2.2 Source code

Hand in the source code, comprising of a makefile, source code files and startup scripts for starting your system on each of the 4 servers your system will run on.

The code should be easy to read (read: well-commented!). The instructors reserve the right to deduct points for code that they consider undecipherable.

## 2.3 Grading criteria

The 70 pts for this assignment are given as follows: 5pts for complete design document, 5pts for compilation, 60 pts for test cases (the test cases have different weights). Refer to provided cases tests which cover most scenarios but these are slightly different with the test cases for grading.