

MP2.2: A Highly Available and Scalable Tiny SNS

Features:

- Coordinator that assigns newly connected clients to an available server cluster
- Servers replicate their cluster's data on a slave so that the cluster is fault tolerant, and clients can re-login after a master failure
- Clients do not know implementation; only connect to coordinator once
- Follower synchronizers that propagate posts, followers across the system, and follower-following relationships so that all users across clusters can interact

Architecture:

- Coordinator
 - Serves both SNS servers, SNS clients, and Follower Synchronizers
 - Coordinator stores metadata of all servers and synchronizers on all clusters as a "routing table"
 - This table is updated on every heartbeat (updating current metadata) or when a new server/synchronizer joins the SNS (adding new entry to table)
 - Servers and synchronizers register themselves with a heartbeat to determine their role
 - Coordinator periodically checks for "dead" or inactive servers by seeing if their last heartbeat was more than 10 seconds ago
 - Marks a dead server as having missed a heartbeat
 - If a master dies, its slave becomes the new master of that cluster
 - Additionally, the synchronizer serving that master machine becomes orphaned, and the slave synchronizer becomes the new master synchronizer for that cluster
 - **In this assignment, synchronizers do not fail**
- Servers
 - Each cluster has a Master server - Slave server pair
 - Server opens gRPC channel as a client of the coordinator
 - The coordinator informs the server as to whether it is a slave or a master server
 - Master server opens a gRPC channel as client of its slave server (if it exists), where client requests are forwarded so that the state on the Master's filesystem is replicated on the slave's filesystem
 - Server sends "Heartbeats" to tell the coordinator that it is still "alive" or active
 - The coordinator's responses indicate whether the server is a master or a slave. If it is a master, it takes on the master role

- Synchronizers
 - Each cluster has a master synchronizer and a slave synchronizer
 - Master synchronizer
 - Monitors the data in its own filesystem and publishes this data to all of the other synchronizers
 - Continuously publishes all of the clients/users it knows of
 - Checks each of its native clients' timeline files and publishes updates to the rest of the network
 - Continuously publishes the following relationships it sees on its cluster
 - Consumes data given to it by other synchronizers and updates the data on its filesystem accordingly
 - Adds users it has not seen before
 - Adds following relationships that did not already exist
 - Updates the "following" timelines (posts from users that a particular client is following) with the posts consumed from the rest of the network
 - Appending the information of other filesystems to its own creates an accurate replica at every cluster eventually, making it so that all users on all clusters can interact with each other without knowing they are on different machines
 - Slave synchronizer
 - DOES NOT publish the data in its filesystem; it serves to maintain an up-to-date replica of the master's filesystem
 - Consumes data given to it by other synchronizers and updates the data on its filesystem accordingly
- Clients
 - Client opens gRPC channel as client of the coordinator
 - Client requests server from the coordinator
 - Coordinator assigns client a server, client connects and begins to interact with SNS as normal

Implementation:

- Server
 - Initialize a server info struct for communication with coordinator in accordance with coordinator service protobuf definition
 - Track the size of users' "following" files to know if the synchronizer has added something the user has not seen yet, and write it to their timeline stream
 - Create stub on coordinator channel
 - Create a heartbeat handler for a thread that will send heartbeat every 5 seconds with `thread::sleep_for` and `chrono::seconds`
 - In thread, use stub to send heartbeats to coordinator as rpc's
 - First heartbeat should be a "registration" heartbeat to create the data for this server on the coordinator's end

- Create update file handler for thread that checks if the synchronizer has changed anything in the filesystem, and update in-memory data structures accordingly
 - Update clientdb vector with new clients and following relationships
 - Use the reader/writer streams of clients in timeline mode to update their timeline streams if their “following” file gets updated
- Client
 - Create stub on coordinator channel
 - Use stub to request a server
 - Use returned server info to create normal server channel and corresponding stub
- Coordinator
 - Hold metadata table as vector of vectors: [clusterid][serverid]
 - Assign clients to clusters by $(clientid - 1) \bmod 3 + 1$
 - Assign synchronizers the same way
 - Make the first synchronizer to connect the master synchronizer
 - Use mutex when writing/reading server metadata so as to not create race condition between threads
 - Periodically checks to see if any servers are dead by looping through metadata files in a separate thread
- Synchronizer
 - Creates a stub for communication with the coordinator
 - Use server info struct to hold information about itself
 - Use named semaphores to avoid read/write conflicts with the server it serves
 - Create heartbeat thread and handler to update master/slave role and adjust accordingly

Test 1:

```
csce438@cf0a93b6160:~/mp2_2$ ./tsc -k 9000 -u 1
logging Initialized. Client starting...connect to this server: 127.0.0.1:10000
login status:

===== TINY SNS CLIENT =====
Command Lists and Format:
FOLLOW <username>
UNFOLLOW <username>
LIST
TIMELINE
=====
Cmd> list
Command completed successfully
All users: 1, 4,
Followers:
Cmd> follow 4
Command completed successfully
Cmd> list
Command completed successfully
All users: 1, 4,
Followers: 4,
Cmd> timeline
Command completed successfully
Now you are in the timeline
p11
p12
4 (Fri Apr 18 05:33:32 2025) >> p41
4 (Fri Apr 18 05:33:34 2025) >> p42
□

p42
^C
csce438@cf0a93b6160:~/mp2_2$ ./tsc -k 9000 -u 4
logging Initialized. Client starting...connect to this server: 127.0.0.1:10000
login status:

===== TINY SNS CLIENT =====
Command Lists and Format:
FOLLOW <username>
UNFOLLOW <username>
LIST
TIMELINE
=====
Cmd> list
Command completed successfully
All users: 1, 4,
Followers:
Cmd> follow 1
Command completed successfully
Cmd> list
Command completed successfully
All users: 1, 4,
Followers: 1,
Cmd> timeline
Command completed successfully
Now you are in the timeline
1 (Thu Jan 1 00:00:00 1970) >> p11
1 (Thu Jan 1 00:00:00 1970) >> p12
p41
p42
□
```

Test 2:

```
csce439@cs-f0a93b6160:~/mp2_2$ ./tsc -k 9000 -u 1
Logging Initialized. Client starting...connect to this server: 127.0.0.1:10000
login status:

=====
Cmd> list
Command completed successfully
All users: 1, 2, 3,
Followers:
Cmd> follow 2
Command completed successfully
Cmd> follow 3
Command completed successfully
Cmd> list
Command completed successfully
All users: 1, 2, 3,
Followers: 2, 3,
Cmd> timeline
Command completed successfully
Now you are in the timeline
p11
p12
2 (Thu Jan 1 00:00:00 1970) >> p21
2 (Thu Jan 1 00:00:00 1970) >> p22
3 (Thu Jan 1 00:00:00 1970) >> p31
3 (Thu Jan 1 00:00:00 1970) >> p32

csce439@cs-f0a93b6160:~/mp2_2$ ./tsc -k 9000 -u 2
Logging Initialized. Client starting...connect to this server: 127.0.0.1:20000
login status:

=====
Cmd> list
Command completed successfully
All users: 2, 1, 3,
Followers:
Cmd> follow 1
Command completed successfully
Cmd> follow 3
Command completed successfully
Cmd> list
Command completed successfully
All users: 2, 1, 3,
Followers: 1, 3,
Cmd> timeline
Command completed successfully
Now you are in the timeline
1 (Thu Jan 1 00:00:00 1970) >> p11
1 (Thu Jan 1 00:00:00 1970) >> p12
p21
p22
3 (Thu Jan 1 00:00:00 1970) >> p31
3 (Thu Jan 1 00:00:00 1970) >> p32

csce439@cs-f0a93b6160:~/mp2_2$ ./tsc -k 9000 -u 3
Logging Initialized. Client starting...connect to this server: 127.0.0.1:30000
login status:

=====
Cmd> list
Command completed successfully
All users: 3, 1, 2,
Followers:
Cmd> follow 1
Command completed successfully
Cmd> follow 2
Command completed successfully
Cmd> list
Command completed successfully
All users: 3, 1, 2,
Followers: 1, 2,
Cmd> timeline
Command completed successfully
Now you are in the timeline
1 (Thu Jan 1 00:00:00 1970) >> p11
1 (Thu Jan 1 00:00:00 1970) >> p12
2 (Thu Jan 1 00:00:00 1970) >> p21
2 (Thu Jan 1 00:00:00 1970) >> p22
p31
p32
```

Test 3:

Servers- (all masters dead)

[illegible]

Clients-

```

Cmd> follow 5
Command completed successfully
Cmd> list
Command completed successfully
All users: 1, 2, 3, 5,
Followers: 3, 2, 5,
Cmd> timeline
Command completed successfully
Now you are in the timeline
2 (Thu Jan 1 00:00:00 1970) >> p21
2 (Thu Jan 1 00:00:00 1970) >> p22
3 (Thu Jan 1 00:00:00 1970) >> p31
3 (Thu Jan 1 00:00:00 1970) >> p32
5 (Thu Jan 1 00:00:00 1970) >> p5
5 (Thu Jan 1 00:00:00 1970) >> p52
3 (Thu Jan 1 00:00:00 1970) >> p33
3 (Thu Jan 1 00:00:00 1970) >> p34
3 (Thu Jan 1 00:00:00 1970) >> p32
3 (Thu Jan 1 00:00:00 1970) >> p31
5 (Thu Jan 1 00:00:00 1970) >> p5
5 (Thu Jan 1 00:00:00 1970) >> p52
3 (Thu Jan 1 00:00:00 1970) >> p33
3 (Thu Jan 1 00:00:00 1970) >> p34
p23
p24
1 (Thu Jan 1 00:00:00 1970) >> p13
1 (Thu Jan 1 00:00:00 1970) >> p14
p13
p14

Cmd> follow 5
Command completed successfully
Cmd> list
Command completed successfully
All users: 3, 1, 2, 5,
Followers: 1, 2, 5,
Cmd> timeline
Command completed successfully
Now you are in the timeline
1 (Thu Jan 1 00:00:00 1970) >> p11
1 (Thu Jan 1 00:00:00 1970) >> p12
2 (Thu Jan 1 00:00:00 1970) >> p21
2 (Thu Jan 1 00:00:00 1970) >> p22
5 (Thu Jan 1 00:00:00 1970) >> p5
5 (Thu Jan 1 00:00:00 1970) >> p52
p33
p34
2 (Thu Jan 1 00:00:00 1970) >> p23
2 (Thu Jan 1 00:00:00 1970) >> p24
1 (Thu Jan 1 00:00:00 1970) >> p13
1 (Thu Jan 1 00:00:00 1970) >> p14

Cmd> follow 2
Command completed successfully
Cmd> follow 3
Command completed successfully
Cmd> list
Command completed successfully
All users: 2, 5, 1, 3,
Followers: 2, 3, 1,
Cmd> timeline
Command completed successfully
Now you are in the timeline
3 (Thu Jan 1 00:00:00 1970) >> p33
3 (Thu Jan 1 00:00:00 1970) >> p34
2 (Fri Apr 18 06:03:36 2025) >> p23
p34
2 (Fri Apr 18 06:03:37 2025) >> p24
p23
p24
1 (Thu Jan 1 00:00:00 1970) >> p13
1 (Thu Jan 1 00:00:00 1970) >> p14

```