MP 2.2 Design Document

Kevin Zhang

April 10, 2025

1 Overview

In this MP, we explore how explore two new concepts: using RabbitMQ to asynchronously replicate state across multiple clusters and replicating state between master and slave servers. Replicating the state of the SNS has many benefits, include fault tolerance (if a master server goes down, the slave can take over), increase load tolerance, and reduce latency among others.

2 Client

The client functionality remains the same as MP 2.1. From the client's perspective, nothing about the system has changed, which is described as "distribution transparency."

3 Master/Slave Servers

To accomplish master/slave replication, each SNS server is now replicated to form a master/slave pair. For this MP, we assume each cluster has up to 2 servers (one master, one slave).

Only master servers interact with clients and do so via synchronous RPCs. If a slave server exists on the cluster, the master forwards all user requests to the slave; thus, from the slave's perspective, the master looks like a client. As an implementation detail, I configured my SNS servers to lazily check whether they are a master server upon each client request.

To assign server roles, we go by order of registration. The first server on a cluster that registers with the coordinator is given the master role. Any subsequent servers that register on the same cluster become slaves.

For this MP, I updated the core logic of my SNS servers. All user info is persisted in the filesystem: each user has "user_timeline.txt," "user_following.txt," and "user_followers.txt" files that store information about the user's timeline posts, users they follow, and users that follow them, respectively. In addition, each server has an "all_users.txt" file that records the existence of all users currently logged into the SNS.

There also exists an in-memory cache on each server that caches information from the filesystem, providing quick access to the user list and each user's followers. This in-memory cache is frequently updated by a background thread that reads each user's files from the filesystem (every 5 seconds).

Like before, each server has a background thread responsible for sending heartbeats to the coordinator. The first heartbeat registers the server with the coordinator, and subsequent heartbeats maintain the server's "alive" status.

Whenever a master server fails (the coordinator misses 2 consecutive heartbeats), a new server on its cluster (if exists) is appointed as a master. This process is done lazily: the new master does not know it is a master until it serves its next client request. As discussed earlier, an SNS server only discovers whether it is a master before returning from a client RPC.

4 Coordinator

The coordinator now maintains a table of follower synchronizer processes alongside its routing table of servers. Similar to the routing table, this synchronizer table is broken down by cluster; synchronizer processes are assigned to a cluster by the formula $(\text{synchronizerID} - 1) \bmod 3 + 1$, which is the same formula used to assign a client to a cluster.

Follower synchronizers are assigned to SNS servers by order of registration. For example, the follower synchronizer that's registered first is assigned to the SNS server that was assigned first, and so on.

The coordinator has a background thread that checks for heartbeats from all registered servers every 5 seconds. If a server or synchronizer misses 2 consecutive heartbeats (10 seconds), it is declared as dead. If the server was a master for its cluster, then the slave will be promoted with the logic explained in the **Master/Slave Servers** section.

Otherwise, the coordinator remains the same as before.

5 Follower Synchronizer

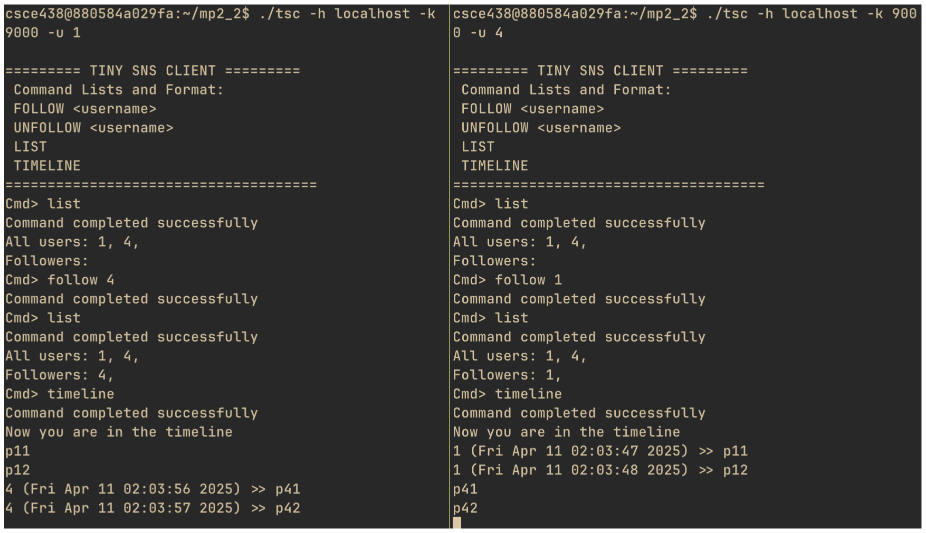
A follower synchronizer is a process that's responsible for publishing the local server's state, making it available for other servers, including servers in other clusters. The intent is for all servers on all clusters to eventually converge to the same state (in their filesystem):

- "all_users.txt" should be the same across all servers
- All user's followers should be the same across all servers. A file that contains "following" information is not strictly necessary.
- The timeline files of a user should be updated if someone they follower makes a post, even if the two users are not on the same server/cluster.
- The state of a master and slave on the same cluster should be exactly the same

To achieve this synchronization, the follower synchronizers use RabbitMQ, a message broker library, to asynchronously publish the server's state. We use a publish-subscribe messaging model: each synchronizer has a producer whose sole purpose is to publish messages to RabbitMQ and a consumer whose sole purpose is to consume messages delivered from RabbitMQ and apply changes accordingly. This model allows us to synchronize server state in the background without affecting the SNS client's time-senstivie synchronous RPCs.

In terms of implementation details, I perform all messaging via RabbitMQ exchanges. Exchanges act as middlemen that can forward any messages they receive to any consumers attached to the exxchange. This allows a producer to easily broadcast a message to any interested parties.
Each synchronizer declares 3 exchanges: "all_users," "relations," and "timelines" that are each responsible for a specific type of message. For example, any message regarding user relationships is sent to the "relations" exchange, which is then forwarded to any consumer that has subscribed to messages from that exchange.

6 Test Cases



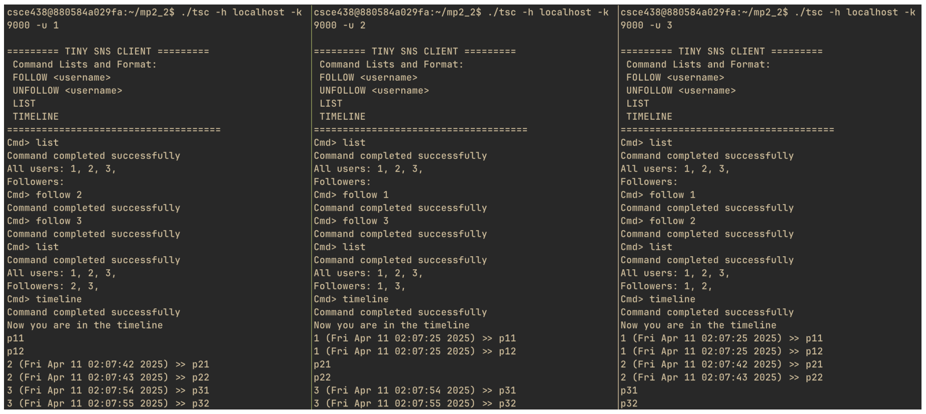
```
csce438@880584a029fa:~/mp2_2$ ./tsc -h localhost -k
9000 -u 1

========= 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 11 02:03:56 2025) >> p41
4 (Fri Apr 11 02:03:57 2025) >> p42
```

```
csce438@880584a029fa:~/mp2_2$ ./tsc -h localhost -k 900
0 -u 4

========= 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 (Fri Apr 11 02:03:47 2025) >> p11
1 (Fri Apr 11 02:03:48 2025) >> p12
p41
p42
```

Figure 1: sanity check



```
csce438@880584a029fa:~/mp2_2$ ./tsc -h localhost -k
9000 -u 1

========= TINY SNS CLIENT =========
 Command Lists and Format:
 FOLLOW <username>
 UNFOLLOW <username>
 LIST
 TIMELINE
=====================================
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 (Fri Apr 11 02:07:42 2025) >> p21
2 (Fri Apr 11 02:07:43 2025) >> p22
3 (Fri Apr 11 02:07:54 2025) >> p31
3 (Fri Apr 11 02:07:55 2025) >> p32
```

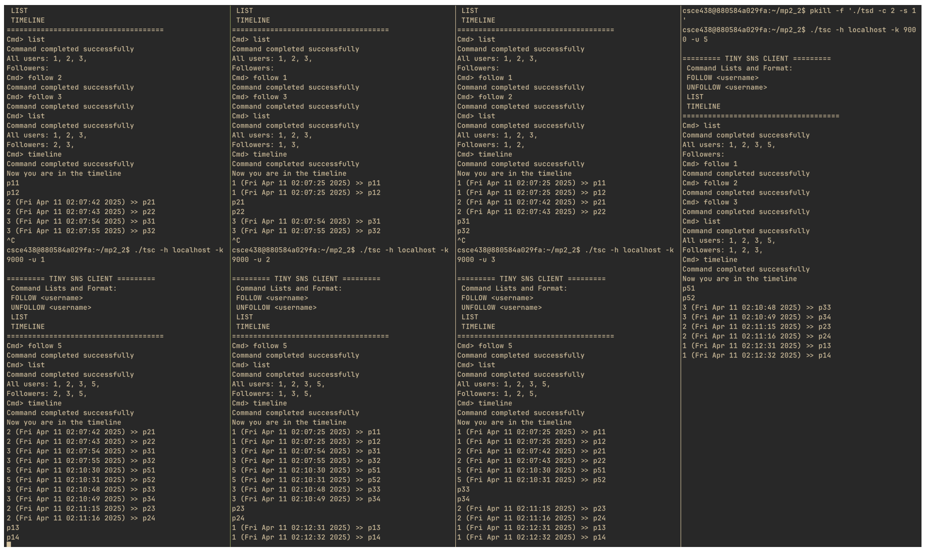
```
csce438@880584a029fa:~/mp2_2$ ./tsc -h localhost -k
9000 -u 2

========= TINY SNS CLIENT =========
 Command Lists and Format:
 FOLLOW <username>
 UNFOLLOW <username>
 LIST
 TIMELINE
=====================================
Cmd> list
Command completed successfully
All users: 1, 2, 3,
Followers:
Cmd> follow 1
Command completed successfully
Cmd> follow 3
Command completed successfully
Cmd> list
Command completed successfully
All users: 1, 2, 3,
Followers: 1, 3,
Cmd> timeline
Command completed successfully
Now you are in the timeline
1 (Fri Apr 11 02:07:25 2025) >> p11
1 (Fri Apr 11 02:07:25 2025) >> p12
p21
p22
3 (Fri Apr 11 02:07:54 2025) >> p31
3 (Fri Apr 11 02:07:55 2025) >> p32
```

```
csce438@880584a029fa:~/mp2_2$ ./tsc -h localhost -k
9000 -u 3

========= TINY SNS CLIENT =========
 Command Lists and Format:
 FOLLOW <username>
 UNFOLLOW <username>
 LIST
 TIMELINE
=====================================
Cmd> list
Command completed successfully
All users: 1, 2, 3,
Followers:
Cmd> follow 1
Command completed successfully
Cmd> follow 2
Command completed successfully
Cmd> list
Command completed successfully
All users: 1, 2, 3,
Followers: 1, 2,
Cmd> timeline
Command completed successfully
Now you are in the timeline
1 (Fri Apr 11 02:07:25 2025) >> p11
1 (Fri Apr 11 02:07:25 2025) >> p12
2 (Fri Apr 11 02:07:42 2025) >> p21
2 (Fri Apr 11 02:07:43 2025) >> p22
p31
p32
```

Figure 2: testing multiple servers



```
 LIST
 TIMELINE
=====================================
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 (Fri Apr 11 02:07:42 2025) >> p21
2 (Fri Apr 11 02:07:43 2025) >> p22
3 (Fri Apr 11 02:07:54 2025) >> p31
3 (Fri Apr 11 02:07:55 2025) >> p32
^C
csce438@880584a029fa:~/mp2_2$ ./tsc -h localhost -k
9000 -u 1

========= TINY SNS CLIENT =========
 Command Lists and Format:
 FOLLOW <username>
 UNFOLLOW <username>
 LIST
 TIMELINE
=====================================
Cmd> follow 5
Command completed successfully
Cmd> list
Command completed successfully
All users: 1, 2, 3, 5,
Followers: 2, 3, 5,
Cmd> timeline
Command completed successfully
Now you are in the timeline
2 (Fri Apr 11 02:07:42 2025) >> p21
2 (Fri Apr 11 02:07:43 2025) >> p22
3 (Fri Apr 11 02:07:54 2025) >> p31
3 (Fri Apr 11 02:07:55 2025) >> p32
5 (Fri Apr 11 02:10:30 2025) >> p51
5 (Fri Apr 11 02:10:31 2025) >> p52
3 (Fri Apr 11 02:10:48 2025) >> p33
3 (Fri Apr 11 02:10:49 2025) >> p34
2 (Fri Apr 11 02:11:15 2025) >> p23
2 (Fri Apr 11 02:11:16 2025) >> p24
p13
p14
```

```
 LIST
 TIMELINE
=====================================
Cmd> list
Command completed successfully
All users: 1, 2, 3,
Followers:
Cmd> follow 1
Command completed successfully
Cmd> follow 3
Command completed successfully
Cmd> list
Command completed successfully
All users: 1, 2, 3,
Followers: 1, 3,
Cmd> timeline
Command completed successfully
Now you are in the timeline
1 (Fri Apr 11 02:07:25 2025) >> p11
1 (Fri Apr 11 02:07:25 2025) >> p12
p21
p22
3 (Fri Apr 11 02:07:54 2025) >> p31
3 (Fri Apr 11 02:07:55 2025) >> p32
^C
csce438@880584a029fa:~/mp2_2$ ./tsc -h localhost -k
9000 -u 2

========= TINY SNS CLIENT =========
 Command Lists and Format:
 FOLLOW <username>
 UNFOLLOW <username>
 LIST
 TIMELINE
=====================================
Cmd> follow 5
Command completed successfully
Cmd> list
Command completed successfully
All users: 1, 2, 3, 5,
Followers: 1, 3, 5,
Cmd> timeline
Command completed successfully
Now you are in the timeline
1 (Fri Apr 11 02:07:25 2025) >> p11
1 (Fri Apr 11 02:07:25 2025) >> p12
3 (Fri Apr 11 02:07:54 2025) >> p31
3 (Fri Apr 11 02:07:55 2025) >> p32
5 (Fri Apr 11 02:10:30 2025) >> p51
5 (Fri Apr 11 02:10:31 2025) >> p52
3 (Fri Apr 11 02:10:48 2025) >> p33
3 (Fri Apr 11 02:10:49 2025) >> p34
p23
p24
1 (Fri Apr 11 02:12:31 2025) >> p13
1 (Fri Apr 11 02:12:32 2025) >> p14
```

```
 LIST
 TIMELINE
=====================================
Cmd> list
Command completed successfully
All users: 1, 2, 3,
Followers:
Cmd> follow 1
Command completed successfully
Cmd> follow 2
Command completed successfully
Cmd> list
Command completed successfully
All users: 1, 2, 3,
Followers: 1, 2,
Cmd> timeline
Command completed successfully
Now you are in the timeline
1 (Fri Apr 11 02:07:25 2025) >> p11
1 (Fri Apr 11 02:07:25 2025) >> p12
2 (Fri Apr 11 02:07:42 2025) >> p21
2 (Fri Apr 11 02:07:43 2025) >> p22
p31
p32
^C
csce438@880584a029fa:~/mp2_2$ ./tsc -h localhost -k
9000 -u 3

========= TINY SNS CLIENT =========
 Command Lists and Format:
 FOLLOW <username>
 UNFOLLOW <username>
 LIST
 TIMELINE
=====================================
Cmd> follow 5
Command completed successfully
Cmd> list
Command completed successfully
All users: 1, 2, 3, 5,
Followers: 1, 2, 5,
Cmd> timeline
Command completed successfully
Now you are in the timeline
1 (Fri Apr 11 02:07:25 2025) >> p11
1 (Fri Apr 11 02:07:25 2025) >> p12
2 (Fri Apr 11 02:07:42 2025) >> p21
2 (Fri Apr 11 02:07:43 2025) >> p22
5 (Fri Apr 11 02:10:30 2025) >> p51
5 (Fri Apr 11 02:10:31 2025) >> p52
p33
p34
2 (Fri Apr 11 02:11:15 2025) >> p23
2 (Fri Apr 11 02:11:16 2025) >> p24
1 (Fri Apr 11 02:12:31 2025) >> p13
1 (Fri Apr 11 02:12:32 2025) >> p14
```

```
csce438@880584a029fa:~/mp2_2$ pkill -f './tsd -c 2 -s 1
'
csce438@880584a029fa:~/mp2_2$ ./tsc -h localhost -k 900
0 -u 5

========= TINY SNS CLIENT =========
 Command Lists and Format:
 FOLLOW <username>
 UNFOLLOW <username>
 LIST
 TIMELINE
=====================================
Cmd> list
Command completed successfully
All users: 1, 2, 3, 5,
Followers:
Cmd> follow 1
Command completed successfully
Cmd> follow 2
Command completed successfully
Cmd> follow 3
Command completed successfully
Cmd> list
Command completed successfully
All users: 1, 2, 3, 5,
Followers: 1, 2, 3,
Cmd> timeline
Command completed successfully
Now you are in the timeline
p51
p52
3 (Fri Apr 11 02:10:48 2025) >> p33
3 (Fri Apr 11 02:10:49 2025) >> p34
2 (Fri Apr 11 02:11:15 2025) >> p23
2 (Fri Apr 11 02:11:16 2025) >> p24
1 (Fri Apr 11 02:12:31 2025) >> p13
1 (Fri Apr 11 02:12:32 2025) >> p14
```

Figure 3: testing resilience