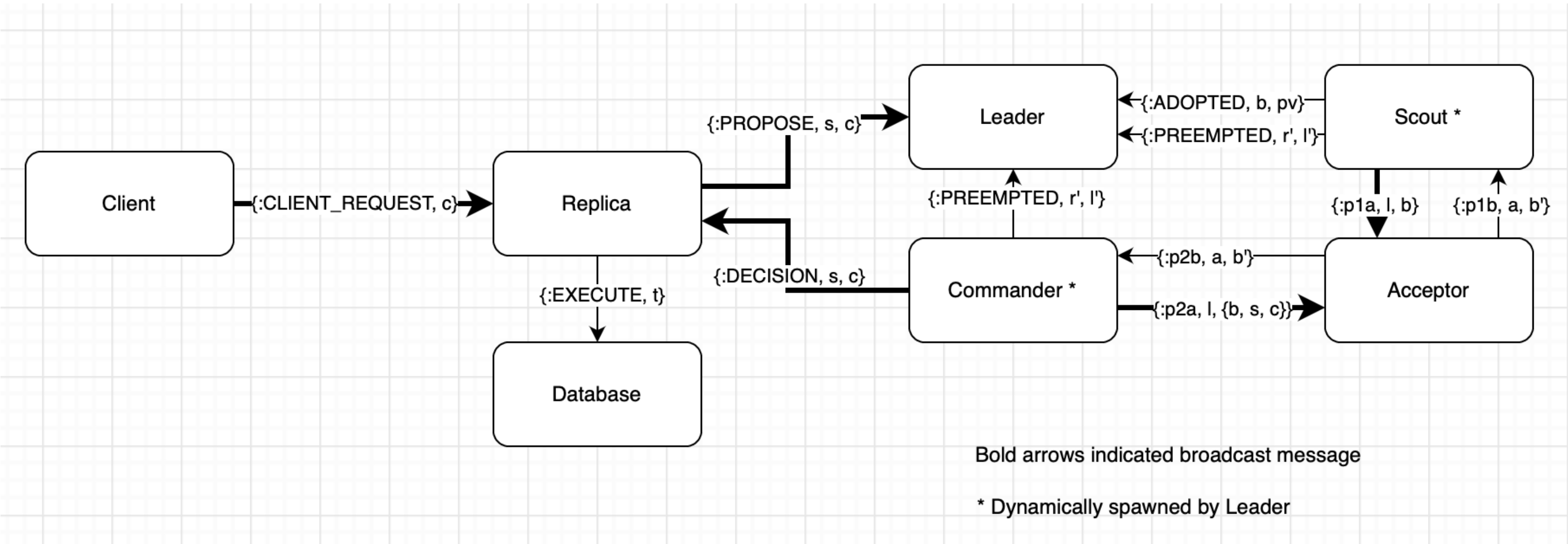
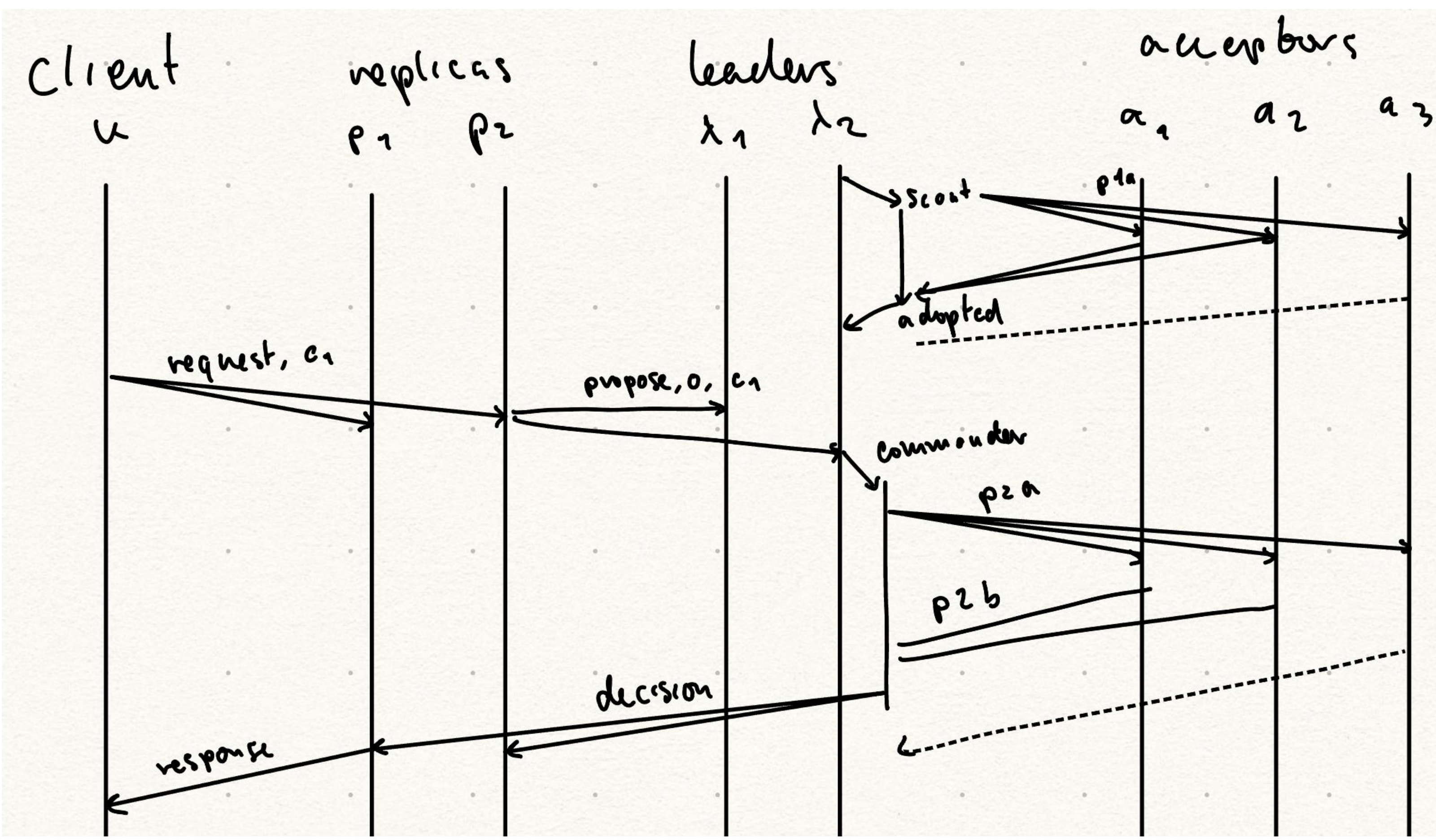
Luca Mehl (lsm20)

Architecture



This graph shows the architecture of the various modules in Multi-Paxos. The Server and Multipaxos modules have been omitted for brevity.



This time diagram shows a client, two replicas, two leaders (with dynamically generated scouts and commanders), and three acceptors, with time progressing downward.
The leader first runs a scout to become active. Later, when a replica proposes a command (in response to a client's request), the leader runs a commander, which notifies the replicas upon learning a decision.

Liveness

To implement the liveness property, I used the Additive Increase Multiplicative Decrease (AIMD) concept from the paper. When receiving a :PREEMPTED message, alerting the leader that there is another ballot in circulation with a higher ballot number, the leader waits for *timeout* before attempting to spawn a new scout, and multiplicatively increases the timeout period.

```
{:PREEMPTED, {r_new, leader_id_new}} ->
  if {r_new, leader_id_new} > ballot_num do

    # Discovers there is another ballot in circulation with higher number,
    # give the other leader time to complete
    Process.sleep(timeout)

    # Multiplicatively increase timeout for next attempt
    timeout = min(round(timeout * 1.2), 10_000)
```

Meanwhile, if a ballot number is successfully :ADOPTED by a majority of acceptors, then contention is low and the leader linearly reduces its timeout.

```
{:ADOPTED, ^ballot_num, pvalues} ->

  # Success, linearly decrease timeout for next attempt
  timeout = max(timeout - 50, 0)
```

Evaluation

This evaluation was conducted on an M1 Macbook Pro with 8 cores and 16GB of RAM (5Gb free).

My solution to part 2 unfortunately contained a bug, and therefore operations were not synchronized, and not all requests done. In general, when 50 requests were sent any number between 40 and 48 of them would be completed before progress stopped being made.

As such, an extensive evaluation could not be carried out, but some properties were still observed.
Once timeouts were implemented, the proportion of completed messages was significantly higher than before. As seen in Appendix 1, after only 1000ms 48/50 requests have already been completed, however due to the bug in part 2 no progress is made beyond that point.

APPENDIX 1:

```
--> Starting Multipaxos at multipaxos_03_lucamehl@127.0.0.1 (192.168.8.117)
  client_sleep 2
  client_stop 15000
  crash_servers %{}
  debug_level 1
  line_num 0
  max_amount 1000
  max_requests 10
  monitor #PID<0.145.0>
  n_accounts 100
  n_clients 5
  n_servers 5
  node_location "multipaxos_03_lucamehl@127.0.0.1 (192.168.8.117)"
  node_name "Multipaxos"
  node_num ""
  node_suffix "03_lucamehl@127.0.0.1"
  node_type "Multipaxos"
  param_setup :default
  print_after 1000
  send_policy :broadcast
  start_function :cluster_start
  timelimit 15000
  window_size 10
--> Starting Server1 at server1_03_lucamehl@127.0.0.1 (192.168.8.117)
--> Starting Server2 at server2_03_lucamehl@127.0.0.1 (192.168.8.117)
--> Starting Server3 at server3_03_lucamehl@127.0.0.1 (192.168.8.117)
--> Starting Server4 at server4_03_lucamehl@127.0.0.1 (192.168.8.117)
--> Starting Server5 at server5_03_lucamehl@127.0.0.1 (192.168.8.117)
--> Starting Client1 at client1_03_lucamehl@127.0.0.1 (192.168.8.117)
```

```
--> Starting Client2 at client2_03_lucamehl@127.0.0.1 (192.168.8.117)
--> Starting Client3 at client3_03_lucamehl@127.0.0.1 (192.168.8.117)
--> Starting Client4 at client4_03_lucamehl@127.0.0.1 (192.168.8.117)
--> Starting Client5 at client5_03_lucamehl@127.0.0.1 (192.168.8.117)
  Client 1 going to sleep, sent = 10
  Client 3 going to sleep, sent = 10
  Client 2 going to sleep, sent = 10
  Client 4 going to sleep, sent = 10
  Client 5 going to sleep, sent = 10
time = 1000 client requests seen = [{1, 50}, {2, 50}, {3, 50}, {4, 50}, {5, 50}]
time = 1000     db requests done = [{1, 48}, {2, 48}, {3, 48}, {4, 48}, {5, 48}]
time = 1000            scouts up = [{1, 4}, {2, 3}, {3, 3}, {4, 2}, {5, 1}]
time = 1000          scouts down = [{1, 4}, {2, 3}, {3, 3}, {4, 2}, {5, 1}]
time = 1000        commanders up = [{1, 140}, {2, 70}, {3, 144}, {4, 74}, {5, 70}]
time = 1000      commanders down = [{1, 140}, {2, 70}, {3, 144}, {4, 74}, {5, 70}]

time = 2000 client requests seen = [{1, 50}, {2, 50}, {3, 50}, {4, 50}, {5, 50}]
time = 2000     db requests done = [{1, 48}, {2, 48}, {3, 48}, {4, 48}, {5, 48}]
time = 2000            scouts up = [{1, 4}, {2, 3}, {3, 3}, {4, 2}, {5, 1}]
time = 2000          scouts down = [{1, 4}, {2, 3}, {3, 3}, {4, 2}, {5, 1}]
time = 2000        commanders up = [{1, 140}, {2, 70}, {3, 144}, {4, 74}, {5, 70}]
time = 2000      commanders down = [{1, 140}, {2, 70}, {3, 144}, {4, 74}, {5, 70}]
```