Implementing the 3 Phase Commit

Group: 10
Shariar Kabir
ID: 0419052047

Problem Statement

The goal of this project is to implement a consistent distributed music “playlist” using **three phase commit(3PC)**.

This is an un-ordered list of pairs of the form ⟨song_name, URL⟩.

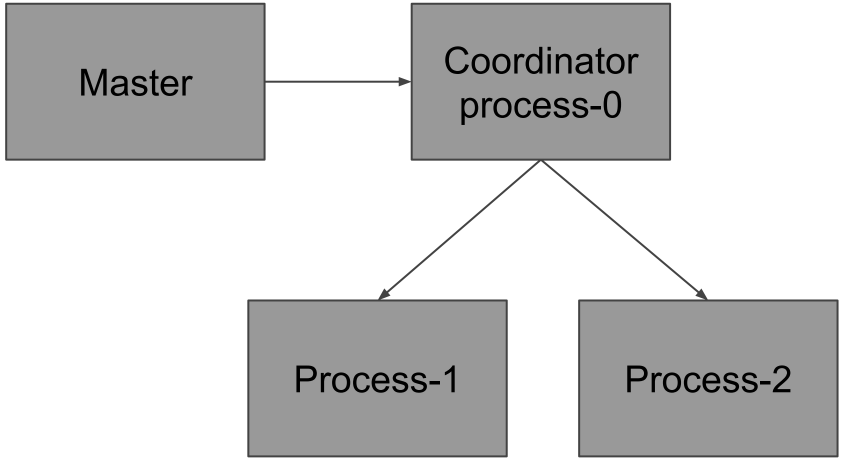
Commands to modify the playlist should obey the four interfaces:

- Add
- Get
- Delete and
- Edit

Basic Structure

The system consists of a set of **servers**, one of which serves as the **coordinator**. A client is is used for sending commands to the coordinator and receives responses from the coordinator. We refer to it as **the master**

The coordinator has pid: -1



Master Client

This client is used for

- sending commands to the coordinator and receives responses from the coordinator.
- The master may also send commands to any server (including a participant) to ask that server to crash.
- A list of these commands is shown in Table 1.

Master Commands

command	purpose	usage
halt	stops the client. Additionally, sends a halt message to each of the servers. Use this to terminate the entire simulation.	halt
start	start a client process	<pid> start <total> <port>
add	Used to add a song to the playlist. If the song already exists then it will have no effect on the playlist.	-1 add song1 URL1
edit	Used to edit a song in the playlist. If the song does not exist then it will have no effect on the playlist.	-1 edit song1 URL1
delete	Used to delete a song from the playlist. If the song does not exist then it will have no effect on the playlist.	-1 delete song1
get	Used to get a song's url from the playlist. If the song does not exist then it will return NONE.	-1 get song1
crash	Immediately stops the client	<pid> crash

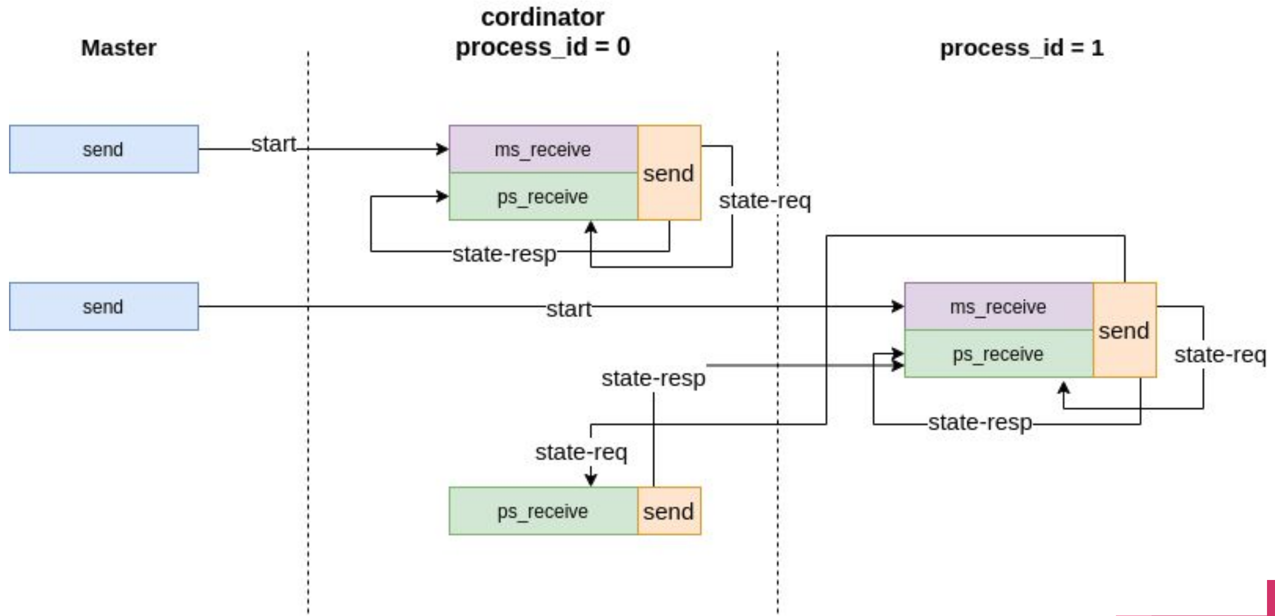
Client Commands

command	purpose	usage
<ctrl> c	use this to instantly stop a node. (We don't catch this signal, the node can stop at any point in execution.)	<ctrl> c
crash	a more kind way of stopping a process. Type this into the terminal and the process will finish doing whatever it was doing and will terminate.	crash
vetonext	Cause the node to vote NO on the next votereq.	vetonext
playlist	Cause the node to print its current playlist.	playlist
actions	Cause the node to print all transactions it has performed on the playlist.	actions

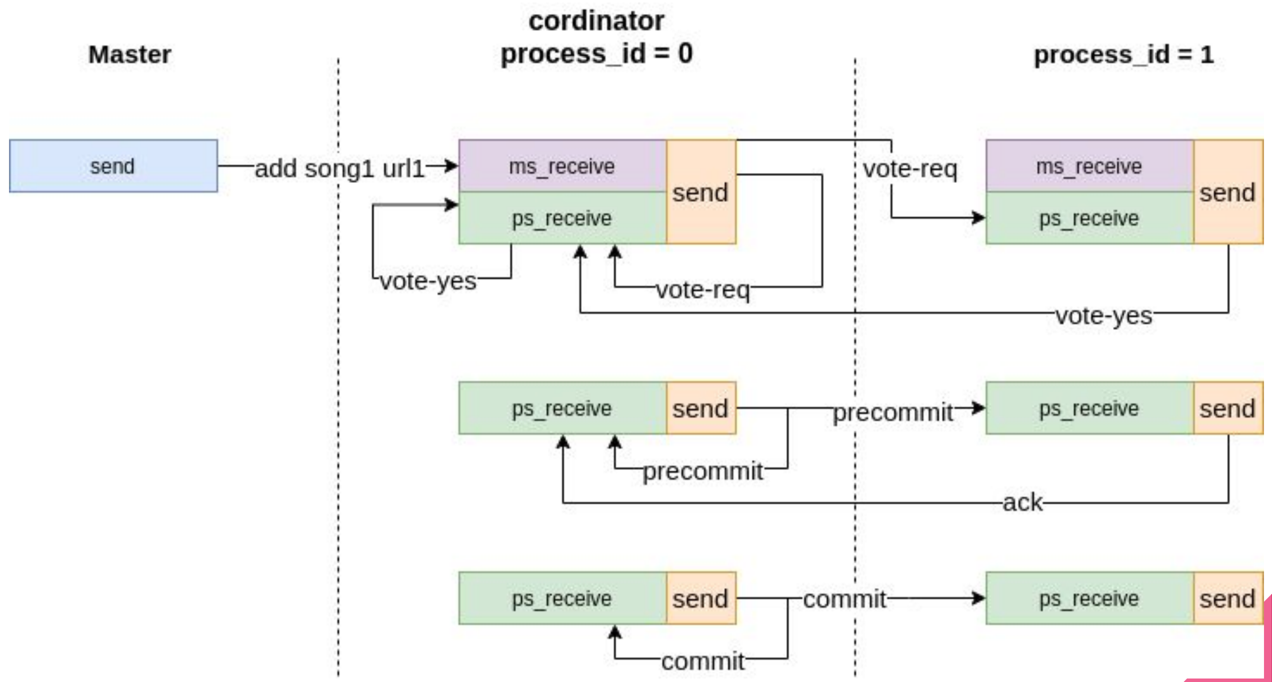
Client Commands

command	purpose	usage
crashBeforeVote	Process will crash before the next vote.	crashBeforeVote
crashAfterVote	Process will crash after the next vote.	crashAfterVote
crashAfterAck	Process will crash after the next ack.	crashAfterAck
crashVoteReq	Cordinator will crash before the next vote-req.	crashVoteReq
crashPartialPreCommit	Cordinator will crash before the next precommit.	crashPartialPreCommit
crashPartialCommit	Cordinator will crash before the next commit	crashPartialCommit

Start

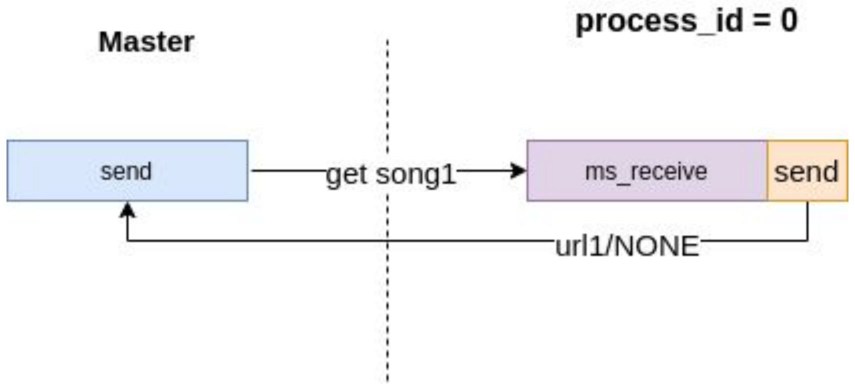

Master
cordinator
process_id = 0
process_id = 1
send
start
ms_receive
ps_receive
send
state-req
state-resp
send
start
ms_receive
ps_receive
send
state-req
state-resp
state-resp
state-req
ps_receive
send

Add



Master
cordinator
process_id = 0
process_id = 1
send
add song1 url1
ms_receive
ps_receive
send
vote-req
vote-yes
vote-req
ms_receive
ps_receive
send
vote-yes
ps_receive
send
precommit
precommit
ps_receive
send
ack
ps_receive
send
commit
commit
ps_receive
send

Get



Crashes/Timeout

- Process
  - **crashBeforeVote**: process_timeout_vote will occur action will be aborted
  - **crashAfterVote**: process_timeout_ack will occur action will be committed
  - **crashAfterAck**: no timeout will occur action will be committed
- Co-ordinator:
  - **crashVoteReq** coordinator_timeout_vote_req will occur re-election and termination_protocol will decide abort
  - **crashPartialPreCommit** coordinator_timeout_precommit will occur re-election and termination_protocol will decide abort as processes are uncertain (TR3)
  - **crashPartialCommit** coordinator_timeout_commit will occur re-election and termination_protocol will decide commit as processes are committable (TR4)

Recovery Scenarios

- Node fails while no transaction is running and gets back online
  - Participant will have the latest playlist
  - Actions/Transactions performed by that client will be reset
- A participant P fails during a transaction X and gets back online
  - If P fails before casting a vote, X will be aborted
  - If P fails before giving ack, X will be committed
  - If P fails after giving ack, X will be committed

Test Cases

1. No failure, but a participant votes NO
    a. vetonext
2. Participant failure
    a. crashBeforeVote
    b. crashAfterVote
    c. crashAfterAck
3. Coordinator failure:
    a. crashVoteReq
    b. crashPartialPreCommit
    c. crashPartialCommit