Language - C

Assumptions -

* Arbitrary number of users

protocol -

* base protocol - udp
* requirement - fully ordered multicast

operations -

* Start -

$$  dchat USER

Where, USER is the name or nickname of the user. After starting, dchat should print out the IP address and UDP port on which it is listening. For example, if "Bob" wants to start a chat:

$$ dchat Bob   Bob  started  a  new  chat,  listening  on  192.168.5.2:7432   Succeeded,  current  users:   Bob  192.168.5.2.7432  (Leader)

  Waiting  for  others  to  join...   This would indicate that the chat is now active, with Bob's client listening for messages on IP address 192.168.5.2, UDP port 7432. Note that you can run all clients on a single machine for this project, using the localhost (127.0.0.1) IP address.

* Join

To join an existing chat, the user invokes “dchat” as:

$$  dchat  USER  ADDR:PORT

Where (as before), USER is their own nickname and ADDR and PORT are the IP address and UDP port number of a currently chatting user (which can be any user in a chat, not necessarily the user who started the chat). When invoked this way, dchat should print out its own IP address and port and attempt to join the chat specified. If successful, it should print a list of the current users in the chat, with their IP addresses and port numbers. For example: $$  dchat  Alice  192.168.5.2:7432 Alice  joining  a  new  chat  on  192.168.5.2:7432,  listening  on   192.168.5.81:1923 Succeeded,  current  users:

Bob  192.168.5.2:7432  (Leader)

Eve  192.168.8.200:8333

Alice  192.168.5.81:1923

If there is no active chat user listening on the IP address/port specified, dchat should simply

fail and exit:   $$  dchat  Alice  192.168.5.2:7432

  Alice  joining  a  new  chat  on  192.168.5.2:7432,  listening  on   192.168.5.81:1923   Sorry,  no  chat  is  active  on  192.168.5.2:7432,  try  again  later.   Bye.

* Leave (control-D) - need to update all users

Modes of operation -

* participant - send messages; receive messages; send heart beats;
* sequencer - receive messages; sequence them; send them to all participants;

TODO -

* basic communication channels
* client - start, join and leave methods; message recovery and ordered delivery; detect failures; heartbeat; time out;
* sequencer - sequencing and total ordering of messages
* crash recovery
* leader election