Sameer K.C. (84967w) & Fariha Nazmul (84999P)

Report on “Tuple Space”

LocalTupleSpace puts and gets the tuples into tuplespace. The tuplespace is synchronized to avoid concurrency related problems. To remain tuples unaffected after it has been put to tuplespace, cloning of passed tuple is done and then the clone is inserted into tuplespace.

While retrieving the tuples, the length and contents are checked and if it matches with any tuples in the tuplespace then those tuples are returned otherwise there is a wait condition applied.

ChatServer implements writing messages to channel and opening connection to new listeners. Tuples to track number of rows, message numbers in each channels and channel lists are created. **writeMessage** writes the new message to the specified channel. The old message number of the channel is fetched first and then incremented and then put into tuplespace with the new message. This way we can track how many messages are there in each channel and later on we can read the message by their message number in **ChatListener. openConnection** opens connection for a new listener. Whenever the message number for the channel exceeds the number of rows and a new listener enters, it subtracts the number of rows with message number and assigns the **readMsg** with the result of subtract. **readMsg** is used which message to read. By doing this, it is assured that whenever a new listener enters it will start reading the messages equal to number of rows not the whole messages sent to other listeners.

ChatListener reads the next message and closes the connection. In reading next message a variable **readMsg** is set which tracks which number of message to read. As we’ve inserted the message number with the message itself in the tuplespace, so this **readMsg** will match with message number and if any match is found then it returns that value. Close connection will put the variables used to its initial states.

Gets and puts are used in different places. To acquire some information we did get and to put the information back to tuple consecutively puts were done. Synchronized statements are used in localtuplespace to avoid deadlocks and concurrency related problems.

Problems

Initially the whole messages were coming to new listeners we tried different ways then finally we came to track the messages with a message number and implemented functionality so that a new listener, in case of messages greater than rows, will only read the last messages equal to number of rows. It was weird but sometimes some tests like consumer tests were coming fail sometimes and pass other times.

Testing

In GUI it was working fine. Messages could be sent to new listeners and messages equal to number of rows, incase of over amount of messages, to a listener were successfully sent. Closing connections to listeners didn’t have any bad effect on the execution. We didn’t pass the havoc yielding and chat distributed, other 17 tests were ok.