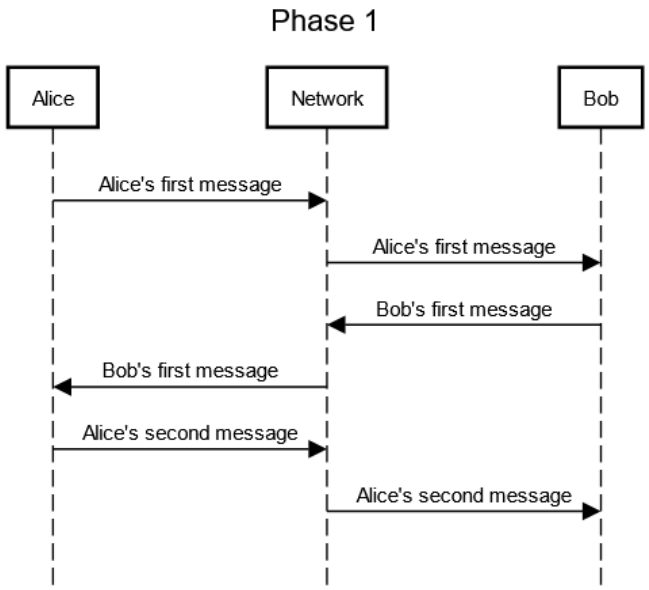
Jack Snowden - Justin Milliman
CS4710: Model-Driven Software Development, Fall 2022
SPIN Project 2 - Needham-Schroeder Protocol - Report

Phase 1

When the program starts, we initialize two instances of clients talking to one another denoted as Alice and Bob. Alice initiates the conversation by generating and sending a message into the network, where the network is just a simple sync channel. Both Alice and Bob idle waiting for responses, so once the message is posted, Bob pulls the initial message, and responds by posting another message into the network. Once Alice sees that message, she responds by posting another message and terminates. Bob then receives the second message and proceeds to terminate.



Verify LTL properties hold for both Alice and Bob.

To test, we specified several properties:

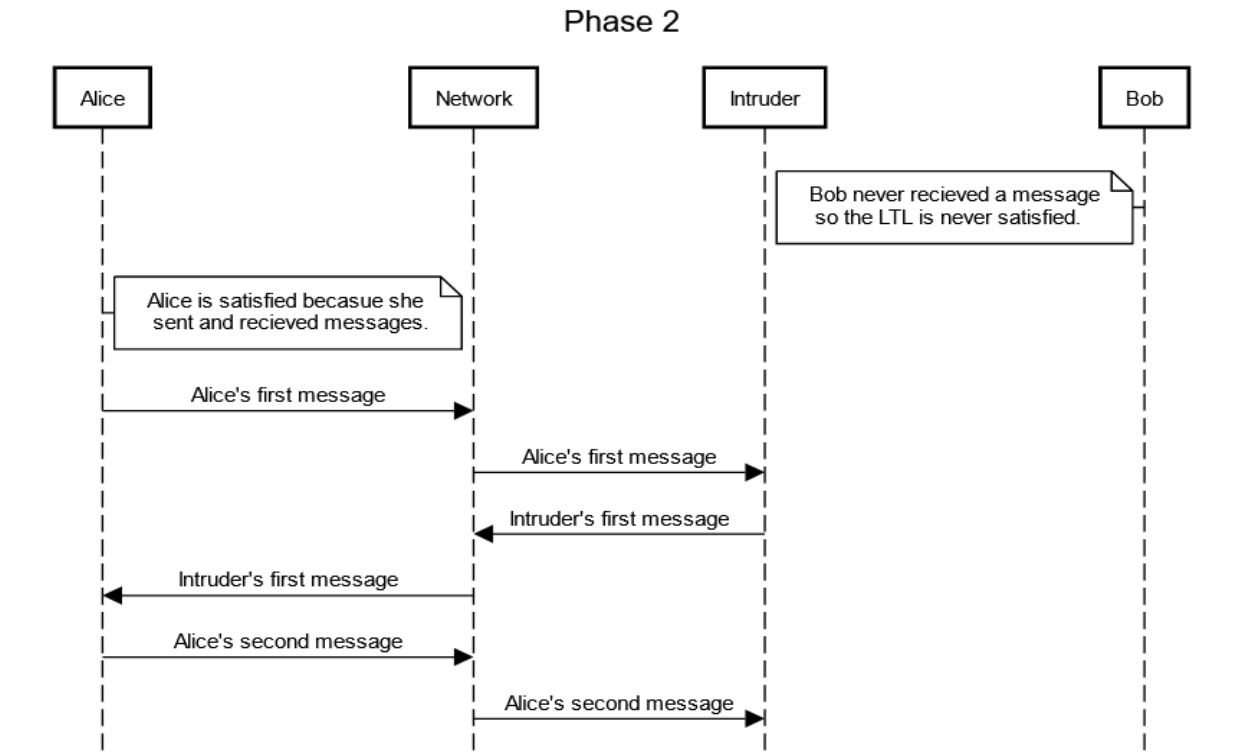
- Alice received a message (from anyone)
- Bob received a message (from anyone)
- Alice received one message from Bob
- Bob received two messages from Alice
- Both parties received the correct number of messages from the intended client and terminated

After running each LTL separately, we did not encounter any counterexamples. This means that the program ran correctly, and both clients were able to send and receive messages to and from the intended recipient.

Does weak fairness impact the result of verification?

No. When accessing the network, Alice and Bob spawn sub processes that are housed in atomic blocks. These blocks contain if blocks, and only ever run on a predetermined condition. Once the condition is satisfied, and a process has run, it simply never runs again. Since these conditions have been set manually, they will only ever run in a predefined order. Meaning, even if a weak fairness policy was attempting to starve a process, it would eventually run out of processes to starve it with.

Phase 2



Sometimes when we simulate our model everything works as intended and other times the intruder makes it so either Alice or Bob's processes never close because the intruder changes the expected outcome. When one party doesn't receive any messages this will cause the LTL for that user to be unsatisfied.

Verify LTL properties hold for both Alice and Bob. Explain any counterexample.

To test, we specified several properties:

- Alice received a message (from anyone)
- Bob received a message (from anyone)
- Alice received one message from Bob
- Bob received two messages from Alice
- Both parties received the correct number of messages from the intended client and terminated

They did not hold, in the scenario that the intruder takes Alice's messages to Bob and returns new one's to Alice, eventually Alice's process will end and Bob will still be waiting for a message. This is the case in our message sequence diagram, Bob never having received or sent a message will cause the LTL to not pass.

Does weak fairness impact the result of verification?

Yes, because both Alice and Bob need to complete their processes. This means that the model will fail verification because the intruder inhibits them from doing so. Weak fairness can cause the intruder to hog the network, essentially entering a cycle where he just talks to himself endlessly until the end of time. So even with the processes being handled in atomic blocks, the intruder will still cause the verification of the model to fail. If strong fairness was used there could be a case that eventually Alice would end but Bob would never have the opportunity to end because Alice will never respond as shown in the above counterexample.