Calculating the Reliabilities of Warp Workloads

Zach Buchholz, Evan Frankmann, Wei Ching Lim, Abby Hawken

Flamingo Team

4/3/2023

Overview:

The Warp program is built to schedule workloads. Workloads consist of flows that each represent a path from a source node to a sink node. The current program schedules the transmissions for these flows and nodes. In this project we will add onto the Warp codebase to develop a way to compute the probability that each node has received the message at each timeslot of the schedule. Next, we will use this information to determine if the E2E defined in the workload has been met for each flow. Finally, the probability information will be created and put into a *.ra file.

Sprint 1:

Tasks:

1. Create Sequence Diagram
2. Understand the Problem
3. Create Project Plan
4. Update ReadMe with high level plans
5. Regenerate UML Diagrams

Task 1:

Zach will create the Sequence Diagram; Wei Ching will double check the sequence diagram and convert it into a pdf to be placed in the Warp folder.

Task 2:

The team will meet on Monday April 3rd to understand the formula for calculating the transmission probabilities, and the ReliabilityVisualization class.

Task 3:

During the meeting on April 3rd, all of us will help create the project plan (this document).

Task 4:

Abby will update ReadMe with the higher-level project plans.

Task 5:

Evan will regenerate the UML diagrams and make sure that everything is correct with those.

Sprint 2:

Tasks: C:\Users\abbyh\git\cs2820\final-project

1. Update Readme with plans for Sprint 2
2. JUnit tests for ReliabilityVisualization Tests
3. Implement the ReliabilityVisualization class.
4. Document the ReliabilityVisualization class.
5. Sequence Diagram for ReliabilityVisualization
6. Update the UML Diagrams to reflect all the latest changes
7. Plan for Sprint 3

Task 1:

Zach will update the Readme with the plans for Sprint 2

Task 2:

Evan will create JUnit tests for the following ReliabilityVisualization methods:

- ReliabilityVisualization(WarpInteface warp)
- public Description visualization()
- public String getHeader()
- public String getScheduler()
- public String getM()
- public String getE2E()
- public String getnChannels()
- public String getFlows()
- public Description reliabilityTableToDescription(ReliabilityTable r)
- public ReliabilityTable getReliabilties()

Task 3:

Zach and Wei Ching will implement the ReliabilityVisualization class which will entail creating the following methods.

- public Description visualization() - This method will return a description object of all the rows that will be in the .ra file.
- ReliabilityVisualization(WarpInteface warp)
- public Description visualization()
- public String getHeader()
- public String getScheduler()
- public String getM()
- public String getE2E()
- public String getnChannels()
- public String getFlows()
- public Description reliabilityTableToDescription(ReliabilityTable r)
- public ReliabilityTable getReliabilties()

Task 4:

Abby will document the Tests and ReliabilityVisualization and make sure the code is clean.

Task 5:

Abby will create a Sequence Diagram for the ReliabilityVisualization class.

Task 6:

Evan will update all the UML diagrams.

Task 7:

Together the entire team will plan for the next sprint.

Sprint 3:

Terminology:

Reliability: The probability that data sent along a flow is received; the reliability of a node at a given time slot is the probability that the node has received the data that it is scheduled to receive.

Flow: The paths of nodes and edges in the transmission.

Node: The point of message transmission.

Source: The node that transmits the information to another node.

Sink: The node that receive information from another node

M: Minimum Packet Reception Rate on an edge in a flow.

E2E: The end-to-end reliability for each flow, flow: src->snk. The flow:src node has an initial probability of 1.0 and all other initial probabilities are 0.0. Each src->sink pair probability is computed as NewSinkNodeState = (1 – M) * PrevSnkNodeState + M*PrevSrcNodeState. This value represents the probability that the message has been received by the node SinkNode.