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:

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:

- Description createHeader()
- String[][] createVisualizationData()
- String[] createColumnHeader()
- List<String> getFlowsAndNodes(List<String> flows)
- String[][] reliabilityTableTo2dArray(ReliabilityTable table)
- ReliabiltityTable getReliabilities()
- ReliabilityTable getFakeDataTable()
- String getTitle()
- String getScheduler()
- String getM()
- String getE2E()
- String getnChannels()

Task 3:

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

- ReliabilityVisualization(WarpInteface warp)

- Description createHeader() - returns a description of all the data describing the file and how it will be analyzed
- String[][] createVisualizationData() - returns a 2d array of Strings containing all of the reliability probabilities
- String[] createColumnHeader() - returns an array in the form of <flowname>:<node in flow> that includes every node in every flow in priority order
- String getScheduler() - gets how the program is being scheduled formatted
- String getTitle() - creates the title for the reliability analysis description with the program's name
- String getM() - min packet reception rate formatted
- String getE2E() - end to end reliability formatted
- String getnChannels() - number of channels formatted
- List<String> getFlowsAndNodes(List<String> flows) - takes in a list of flow names and attaches each flow name to each of the nodes in the flow
- ReliabilityTable getReliabilties() - returns the reliability table from ReliabilityAnalysis but currently returns a fake data table for testing since we have not implemented the ReliabilityAnalysis class yet
- ReliabilityTable getFakeData() - returns a ReliabilityTable the same size as it should be for the given file but with randomly generated data for testing purposes
- String[][] reliabilityTableTo2dArray() - converts a ReliabilityTable object to a 2d string array

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:

1. Update the Readme
2. Write Tests
3. Complete ReliabilityAnalysis
4. Document the tests and Reliability Analysis
5. Sequence Diagram for ReliabilityAnalysis

6. Update UML and create new JavaDocs

Task 1:

Zach updated the readme to include a list of all the tasks, who completed the tasks, and when they were completed. As well as a list of all the documents added and when we met.

Task 2:

Evan will write the test cases. This entails creating the following tests:

- testGetReliabilities() - tests the getReliabilities method to check if the value returned is not null
- testGetColumnIndicesOfFlow() - tests the getColumnIndicesOfFlow method to get the flow nodes to represent the column indices for the reliability table in the correct order
- testGetOldRow() - tests the getOldRow method to return the previous row of reliability values
- testGetFlowNamesToResend() - tests the getFlowNamesToResend method to determine the flows that need to re-send the message based on the flow's period
- testGetFlowNodeToColumnAssociation() - tests the flowNodeToColumnAssociation method to check that check that each flow name and node name association paired to the column index of the reliability table matches the column header created in the ReliabilityVizualization class
- testGetNumRows() - tests the getNumRows method to get the number of rows from the getReliabilities table
- testGetNumColumns() - tests the getNumColumns method to get the number of rows from the getReliabilities table
- testVerifyReliabilities() - tests the verifyReliabilities method to return True if all the flows have met their minimum end-to-end reliabilities
- testGetFinalReliabilityRow() - tests the getFinalReliabilityRow method to return the correct reliabilities from the final row of the reliability table
- testGetColumnToFlowNodeAssociation() - tests the getColumnToFlowNodeAssociation method to check that the column index paired to a flow name and node name association for the reliability table matches the column header created in the ReliabilityVizualization class.
- testGetProgram() - tests the getProgram method to check that the program's schedule choice is not null
- testGetM() - tests the getM method to get the correct minimum packet reception rate
- testGetE2E() - tests the getE2E method to get the correct end-to-end reliability value of the program
- testNumTxPerLinkAndTotalTxCost() - tests the NumTxPerLinkAndTotalTxAttemps method to return the number of transmissions per link and total transmission cost based on whether its number of fault tolerance is fixed

- testGetNumTxAttemptsPerLinkAndTotalTxAttempts() - tests the getNumTxAttemptsPerLinkAndTotalTxAttempts method to return the number of transmission attempts per link and the total transmission attempts of a flow when the end-to-end reliability and minimum packet reception rate is specified
- testGetFixedTxPerLinkAndTotalTxCost() - tests the getFixedTxPerLinkAndTotalTxCost to return the number of transmissions per link and the total cost of the flow when the number of faults is specified
- testGetReliabilitiesEmptyWorkLoad() - tests the getReliabilities method with an emptyWorkLoad
- testGetColumnIndicesOfFlowEmptyFlowName() - tests the getColumnIndicesOfFlow with an empty flow name
- testGetOldRowInvalidRowNumber() - tests the getOldRow method with an invalid row number
- testGetFlowNamesToResendNegativeCycle() - tests the getFlowNamesToResend when the user specifies a negative value for the cycle
- testGetNumRowsEmptyWorkLoad() - tests the getNumRows method with an empty WorkLoad
- testGetNumColumnsEmptyWorkLoad() - tests the getNumColumns method with an empty WorkLoad
- testVerifyReliabilitiesEmptyWorkLoad() - tests the verifyReliabilities method with an empty WorkLoad
- testGetFinalReliabilityRowEmptyWorkLoad() - tests the getFinalReliabilityRow method with an empty WorkLoad
- testGetColumnToFlowNodeAssociationEmptyWorkLoad() - tests the getColumnToFlowNodeAssociation method with an empty WorkLoad
- testGetColumnIndicesOfFlowNonExistentFlow() - tests the getColumnIndicesOfFlow with a non-existent flow name
- TestGetOldRowRowOutOfBounds() – tests the getOldRow method with a row number that is out of bounds
- testGetFlowNamesToResendInvalidTimeStep() - tests the getFlowNamesToResend with an invalid or negative time step

Task 3:

Zach will implement getReliabilities by creating the following methods.

- ReliabilityTable getReliabilities() - returns the reliabilities instance variables if it exists. Otherwises it uses buildReliabilities to create the reliabilities.
- Void buildReliabilities() - builds a ReliabilityTable based off the program that was passed in the constructor

- List<Integer> getColumnIndicesOfFlow(String flow, Map<String, Integer> map) - This method takes in a flow name and a map of <flowname>:<nodename> to the index of the column. Then returns a list of the column indices of the nodes in the flow in order
- ReliabilityRow getOldRow(int row, Map<String, Integer> map, ReliabilityTable reliabilities) - uses information about the period to determine if a flow's reliabilities need to be reset and returns the next row used for calculations
- List<String> getFlowNamesToResend(int row) - given a time slot returns the names of the flows who are starting a new period
- Map<String, Integer> getFlowNodeToColumnAssociation() - returns a Map associating <flowname>:<nodename> to the column in the reliabiliites tables that it represents
- Int getNumRows() - returns the number of rows in the reliabilities tables
- Int getNumColumns() - returns the number of columns in the reliabilities table

and Abby will implement verifyReliabilities by creating the following methods

- Boolean verifyReliabilities() - returns whether all flows meet the e2e reliability requirement
- ArrayList<Double> getFinalReliabilityRow() - returns the last row of the ReliabilityTable
- Map<Integer, String> getColumnToFlowNodeAssociation – returns a hashmap mapping the column indices of ReliabilityTable to the corresponding flow and node names

Task 4:

Wei Ching will do all the documentation for the tests and ReliabilityAnalysis class.

Task 5:

Zach will create the sequence diagram to model the getReliabilities method

Abby will create the sequence diagram for verifyReliabilities

Task 6:

Wei Ching will update the UML Diagrams and create new JavaDocs

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.