EE/ CprE/ SE 492 - sddec23-17

Simulated Design of Quantum Networks

Biweekly Status Report

November 9 - November 22

Client: Dr. Durga Paudyal

Faculty Advisor: Dr. Durga Paudyal

Team Members:

Benjamin Amick - Network security engineer

Derrick Wright - System integration engineer

Ohik Kwon- System component designer

Steven Tompary- Network engineer

Past Week Accomplishments

-Integration progress.
Integration is wholly and fully done at this point. At the time of writing, no work remains to be done. Initially we were using the quantum function written by Ohilk in its entirety, and sending the data to the router at the end of the function but at the behest of Ohik, who wanted the simulation to more closely resemble a realistic running of the algorithm, the function was split into five separate functions that interact with each other and send data to the router where appropriate.

Before - Qwire_final

```
def quantum_function():
    #Step 0. We set simulator as "statevector_simulator"
    simulator = Aer.get_backend('statevector_simulator')

    #Step 1. initialize random qubit (it is our information)
    psi = random_state(1)
    init_gate = Initialize(psi)
    inverse_init_gate = init_gate.gates_to_uncompute()
    init_statevector = information_initialize_to_statevector(init_gate, simulator, False)
    compound_statevector = compound_information_zero_states(init_statevector, False)

    #Step 2. Proceed Alice's quantum operation (not measuring)
    Bell_info_statevector = Alice_quantum_operation(compound_statevector,simulator,False)

    #Step 3. Alice Meausre her qubit.
    #Measured outcome also inverted.(little endian) 1st : information, 2nd : Alice's Bell states
    Alice_measurement_result , Alice_statevector = Alice_measure(Bell_info_statevector, simulator, False)

    #Step 4. Bob's quantum operation (not measureing)
    Bob_statevector = Bob_quantum_opertaion(Alice_measurement_result, Alice_statevector, simulator, False)

    #Step 5. Bob Meausre his qubit.
    Bob_measurement_result = Bob_measure(Bob_statevector, inverse_init_gate, simulator, False)

    return(Alice_measurement_result,Bob_measurement_result)
```

Before - integration_node

```
for n in nodes:
    n.connect()
    print("Node", n.ip, "connected")
    data = n.doWork()
    n.sendMessage(data)
    print("Node", n.ip, "on standby")
    n.sendMessage("quit")
    print("Node", n.ip, "disconnected")

time.sleep(1)
```

Before - node.py

```
def doWork(self):
    return quantum_function()
```

After - Qwire_final

```
def quantum_function_step_one():
    #Step 0. We set simulator as "statevector_simulator"
    simulator = Aer.get_backend('statevector_simulator')

    #Step 1. initialize random qubit (it is our information)
    psi = random_state(1)
    init_gate = Initialize(psi)
    inverse_init_gate = init_gate.gates_to_uncompute()
    init_statevector = information_initialize_to_statevector(init_gate, simulator, False)
    compound_statevector = compound_information_zero_states(init_statevector, False)
    return simulator,inverse_init_gate, compound_statevector

def quantum_function_step_two(simulator,compound_statevector):
    #Step 2. Proceed Alice's quantum operation (not measuring)
    Bell_info_statevector = Alice_quantum_operation(compound_statevector,simulator,False)
    return Bell_info_statevector

def quantum_function_step_three(simulator, Bell_info_statevector):
    #Step 3. Alice Meausre her qubit.
    #Measured outcome also inverted.(little endian) 1st : information, 2nd : Alice's Bell states
    Alice_measurement_result , Alice_statevector = Alice_measure(Bell_info_statevector, simulator, False)
    return Alice_measurement_result, Alice_statevector

def quantum_function_step_four(simulator,Alice_measurement_result, Alice_statevector):
    #Step 4. Bob's quantum operation (not measureing)
    Bob_statevector = Bob_quantum_opertaion(Alice_measurement_result, Alice_statevector, simulator, False)
    return Bob_statevector

def quantum_function_step_five(simulator,Bob_statevector, inverse_init_gate):
    #Step 5. Bob Meausre his qubit.
    Bob_measurement_result = Bob_measure(Bob_statevector, inverse_init_gate, simulator, False)

    return(Bob_measurement_result)

    # data1[0] = simulator
    # data1[1] = inverse_init_gate
    # data1[2] = compound_statevector
    # data2[0] = Bell_info_statevector
    # data3[0] = Alice_measurement_result
    # data3[1] = Alice_statevector
    # data4[0] = Bob_statevector
    # data5[0] = Bob_measurement_result
```

After - integration_node

```
for n in nodes:
    n.connect()
    print("Node", n.ip, "connected")
    n.doWork()
    print("Node", n.ip, "on standby")
    n.sendMessage("quit")
    print("Node", n.ip, "disconnected")
```

After - node.py

```
def doWork(self):
    data1 = quantum_function_step_one ()
    data2 = quantum_function_step_two(data1[0], data1[2])
    data3 = quantum_function_step_three(data1[0],data2)
    self.sendMessage(data3[0])
    data4 = quantum_function_step_four(data1[0],data3[0], data3[1])
    data5 = quantum_function_step_five(data1[0],data4,data1[1])
    self.sendMessage(data5)
```

End array:

current message array: ['11', '0', '01', '0', '11', '0', '01', '0', '10', '0', '11', '0', '11', '0', '10', '0', '11', '0', '10', '0', '01', '0', '10', '0', '10',
'0', '00', '0', '01', '0', '11', '0', '10', '0', '11', '0', '11', '0', '11', '0', '11', '0', '11', '0', '11', '0', '11', '0', '01', '0', '01', '0', '10', '0', '00', '0',
'10', '0', '11', '0', '00', '0', '01', '0', '00', '0', '11', '0', '01', '0', '01', '0', '01', '0', '11', '0', '01', '0', '01', '0', '10', '0', '01', '0', '01', '0', '01',
'0', '10', '0', '10', '0', '01', '0', '01', '0', '01', '0', '10', '0', '11', '0', '10', '0', '01', '0', '00', '0', '00', '0', '01', '0', '11', '0', '01', '0', '01', '0',
'00', '0', '00', '0', '10', '0', '11', '0', '10', '0', '01', '0', '00', '0', '10', '0', '00', '0', '00', '0', '01', '0', '10', '0', '01', '0', '11', '0', '11', '0', '01',
'0', '10', '0', '10', '0', '01', '0', '11', '0', '00', '0', '01', '0', '00', '0', '01', '0', '00', '0', '00', '0', '00', '0', '10', '0', '11', '0', '10', '0', '10', '0',
'01', '0', '11', '0', '00', '0', '11', '0', '01', '0', '10', '0', '11', '0', '11', '0', '11', '0', '00', '0', '00', '0', '11', '0', '00', '0', '11', '0', '01', '0', '01',
'0', '10', '0', '00', '0', '00', '0', '10', '0', '10', '0', '00', '0', '01', '0', '10', '0', '00', '0', '01', '0', '01', '0', '10', '0', '01', '0', '00', '0', '00', '0',
'10', '0', '11', '0', '10', '0', '00', '0', '00', '0', '01', '0', '10', '0', '11', '0', '10', '0', '11', '0', '01', '0', '10', '0', '11', '0', '01', '0', '10', '0', '00',
'0', '11', '0', '10', '0', '11', '0', '10', '0', '10', '0', '01', '0', '11', '0', '01', '0', '11', '0', '11', '0', '11', '0', '11', '0', '00', '0', '10', '0', '01', '0',
'10', '0', '10', '0', '01', '0', '10', '0', '11', '0', '10', '0', '01', '0', '10', '0', '11', '0', '00', '0', '00', '0', '01', '0', '10', '0', '11', '0', '11', '0', '10',
'0', '11', '0', '01', '0', '00', '0', '11', '0', '10', '0', '00', '0', '01', '0', '00', '0', '10', '0', '10', '0', '11', '0', '00', '0', '01', '0', '11', '0', '01', '0', '11',
'11', '0', '10', '0', '11', '0', '01', '0', '11', '0', '11', '0', '10', '0', '01', '0', '11', '0', '01', '0', '01', '0', '01', '0', '11', '0', '01', '0', '00', '0', '00',
'0', '00', '0', '11', '0', '01', '0', '01', '0', '00', '0', '00', '0', '10', '0', '10', '0', '11', '0', '01', '0', '00', '0', '00', '0', '00', '0', '10', '0', '01', '0',
'10', '0', '00', '0', '01', '0', '11', '0', '00', '0', '10', '0', '00', '0', '01', '0', '10', '0', '00', '0', '10', '0', '11', '0', '10', '0', '11', '0', '00', '0', '00',
'0', '01', '0', '10', '0', '01', '0', '01', '0', '11', '0', '00', '0', '11', '0', '00', '0', '10', '0', '10', '0', '11', '0', '10', '0', '10', '0', '10', '0', '01', '0',
'00', '0', '00', '0', '01', '0', '00', '0', '10', '0', '01', '0', '00', '0', '01', '0', '11', '0', '01', '0', '10', '0', '01', '0', '00', '0', '11', '0', '10', '0', '10',
'0', '10', '0', '00', '0', '01', '0', '01', '0', '11', '0', '00', '0', '01', '0', '01', '0', '11', '0', '01', '0', '00', '0', '01', '0', '10', '0', '11', '0', '01', '0',
'01', '0', '01', '0', '01', '0', '01', '0', '11', '0', '11', '0', '10', '0', '01', '0', '10', '0', '01', '0', '11', '0', '01', '0', '01', '0', '01', '0', '01', '0', '10',
'0', '11', '0', '00', '0', '00', '0', '10', '0', '00', '0', '01', '0', '11', '0', '01', '0', '01', '0', '00', '0', '10', '0', '11', '0', '00', '0', '10', '0', '01', '0',
'11', '0', '01', '0', '00', '0', '00', '0', '11', '0', '01', '0', '11', '0', '01', '0', '11', '0', '10', '0', '11', '0', '01', '0', '11', '0', '10', '0', '11', '0', '00',
'0', '00', '0', '01', '0', '00', '0', '00', '0', '10', '0', '10', '0', '10', '0', '11', '0', '00', '0', '00', '0', '01', '0', '10', '0', '01', '0', '00', '0', '10', '0',
'11', '0', '10', '0', '10', '0', '01', '0', '01', '0', '01', '0', '01', '0', '01', '0', '01', '0', '11', '0', '11', '0', '00', '0', '00', '0', '01', '0', '00', '0', '11',
'0', '11', '0', '11', '0', '00', '0', '00', '0', '11', '0', '00', '0', '00', '0', '11', '0', '00', '0', '11', '0', '00', '0', '01', '0', '11', '0', '00', '0', '01', '0',
'11', '0', '00', '0', '00', '0', '11', '0', '11', '0', '10', '0', '10', '0', '11', '0', '01', '0', '10', '0', '01', '0', '00', '0', '11', '0', '01', '0', '10', '0', '11',
'0', '10', '0', '11', '0', '01', '0', '10', '0', '00', '0', '10', '0', '10', '0', '11', '0', '10', '0', '00', '0', '10', '0', '11', '0', '10', '0', '10', '0', '10', '0',
'10', '0', '10', '0', '11', '0', '00', '0', '10', '0', '01', '0', '11', '0', '00', '0', '10', '0', '01', '0', '11', '0', '01', '0', '10', '0', '00', '0', '11', '0', '10',
'0', '00', '0', '10', '0', '01', '0', '01', '0', '01', '0', '10', '0', '00', '0', '00', '0', '10', '0', '01', '0', '11', '0', '00', '0', '00', '0', '01', '0', '01', '0',
'10', '0', '01', '0', '11', '0', '01', '0', '10', '0', '10', '0', '00', '0', '01', '0', '00', '0', '11', '0', '01', '0', '01', '0', '01', '0', '11', '0', '11', '0', '00',
'0', '11', '0', '00', '0', '10', '0', '01', '0', '11', '0', '00', '0', '11', '0', '01', '0', '00', '0', '10', '0', '10', '0', '11', '0', '00', '0', '11', '0', '10', '0',
'11', '0', '10', '0', '01', '0', '01', '0', '10', '0', '11', '0', '11', '0', '11', '0', '10', '0', '11', '0', '11', '0', '10', '0', '11', '0', '00', '0', '11', '0', '11',
'0', '11', '0', '01', '0', '10', '0', '10', '0', '01', '0', '00', '0', '00', '0', '10', '0', '11', '0', '00', '0', '10', '0', '01', '0', '00', '0', '10', '0', '01', '0',
'11', '0', '01', '0', '11', '0', '10', '0', '10', '0', '11', '0', '00', '0', '00', '0', '10', '0', '11', '0', '11', '0', '11', '0', '01', '0', '00', '0', '10',
'0', '00', '0', '11', '0', '11', '0', '11', '0', '11', '0', '10', '0', '10', '0', '01', '0', '01', '0', '01', '0', '01', '0', '10', '0', '00', '0', '11', '0', '10', '0',
'10', '0', '01', '0', '00', '0', '11', '0', '01', '0', '11', '0', '10', '0', '00', '0', '01', '0', '00', '0', '10', '0', '10', '0', '10', '0', '01', '0', '10', '0', '11',
'0', '01', '0', '10', '0', '00', '0', '11', '0', '11', '0', '01', '0', '10', '0', '01', '0', '00', '0', '11', '0', '01', '0', '01', '0', '01', '0', '01', '0', '01', '0',
'11', '0', '00', '0', '00', '0', '10', '0', '11', '0', '01', '0', '10', '0', '01', '0', '00', '0', '11', '0', '11', '0', '10', '0', '00', '0', '01', '0', '11', '0', '00',
'0', '10', '0', '00', '0', '11', '0', '01', '0', '01', '0', '00', '0', '10', '0', '11', '0', '01', '0', '11', '0', '00', '0', '11', '0', '01', '0', '01', '0', '01', '0',
'00', '0', '11', '0', '00', '0', '11', '0', '00', '0', '11', '0', '10', '0', '10', '0', '00', '0', '00', '0', '10', '0', '11', '0', '10', '0', '11', '0', '11', '0', '10',
'0', '11', '0', '11', '0', '10', '0', '01', '0', '00', '0', '00', '0', '00', '0', '01', '0', '10', '0', '01', '0', '00', '0', '00', '0', '01', '0', '00', '0', '11', '0',
'11', '0', '00', '0', '00', '0', '10', '0', '11', '0', '01', '0', '11', '0', '11', '0', '10', '0', '10', '0', '01', '0', '00', '0', '01', '0', '11', '0', '00', '0', '01', '0',
'10', '0', '01', '0', '00', '0', '11', '0', '01', '0', '01', '0', '11', '0', '10', '0', '01', '0', '01', '0', '01', '0', '11', '0', '00', '0', '01', '0', '11', '0', '00',
'0', '10', '0', '00', '0', '11', '0', '01', '0', '01', '0', '00', '0', '10', '0', '11', '0', '01', '0', '11', '0', '00', '0', '11', '0', '01', '0', '01', '0', '01', '0',
'00', '0', '11', '0', '00', '0', '11', '0', '00', '0', '11', '0', '10', '0', '10', '0', '00', '0', '00', '0', '10', '0', '11', '0', '10', '0', '11', '0', '11', '0', '10',
'0', '11', '0', '11', '0', '10', '0', '01', '0', '00', '0', '00', '0', '00', '0', '01', '0', '10', '0', '01', '0', '00', '0', '00', '0', '01', '0', '00', '0', '11', '0',
'11', '0', '00', '0', '00', '0', '10', '0', '11', '0', '01', '0', '11', '0', '11', '0', '10', '0', '11', '0', '01', '0', '01', '0', '11', '0', '10', '0', '01', '0', '11',
'0', '00', '0', '11', '0', '01', '0', '01', '0', '01', '0', '10', '0', '00', '0', '10', '0', '01', '0', '11', '0', '00', '0', '11', '0', '01', '0', '01', '0', '01', '0',
'10', '0', '10', '0', '11', '0', '01', '0', '00', '0', '11', '0', '11', '0', '11', '0', '10', '0', '10', '0', '01', '0', '00', '0', '01', '0', '10', '0', '10', '0', '00',
'0', '10', '0', '11', '0', '00', '0', '00', '0', '01', '0', '11', '0', '11', '0', '10', '0', '01', '0', '10', '0', '10', '0', '10', '0', '10', '0', '01', '0', '10', '0',
'10', '0', '10', '0', '01', '0', '01', '0', '01', '0', '01', '0', '11', '0', '01', '0', '01', '0', '01', '0', '00', '0', '11', '0', '00', '0', '11', '0', '01', '0', '10',
'0', '00', '0', '11', '0', '00', '0', '10', '0', '11', '0', '10', '0', '11', '0', '01', '0', '10', '0', '01', '0', '00', '0', '10', '0', '00', '0', '01', '0', '01', '0',
'01', '0', '01', '0', '01', '0', '00', '0', '00', '0', '00', '0', '00', '0', '01', '0', '01', '0', '11', '0', '00', '0', '01', '0', '11', '0', '00', '0', '00', '0', '10',
'0', '01', '0', '00', '0', '01', '0', '01', '0', '10', '0', '11', '0', '01', '0', '11', '0', '00', '0', '10', '0', '10', '0', '10', '0', '11', '0', '11', '0', '01', '0',

'11', '0', '10', '0', '10', '0', '10', '0', '00', '0', '10', '0', '00', '0', '11', '0', '11', '0', '10', '0', '01', '0', '00', '0', '01', '0', '11', '0', '01', '0', '01', '0', '11', '0', '11', '0', '01', '0', '00', '0', '10', '0', '01', '0', '00', '0', '11', '0', '00', '0', '10', '0', '10', '0', '01', '0', '11', '0', '00', '0', '01', '0', '00', '0', '00', '0', '01', '0', '11', '0', '00', '0', '11', '0', '10', '0', '00', '0', '00', '0', '10', '0', '10', '0', '00', '0', '01', '0', '00', '0', '11', '0', '00', '0', '11', '0', '10', '0', '00', '0', '01', '0', '01', '0', '00', '0', '00', '0', '00', '0', '00', '0', '00', '0', '11', '0', '10', '0', '10', '0', '10', '0', '11', '0', '10', '0', '10', '0', '11', '0', '11', '0', '10', '0', '10', '0', '00', '0', '11', '0', '00', '0', '01', '0', '00', '0', '10', '0', '11', '0', '01', '0', '00', '0', '00', '0', '01', '0', '00', '0', '00', '0', '10', '0', '00', '0', '01', '0', '11', '0', '01', '0', '10', '0', '11', '0', '10', '0', '01', '0', '10', '0', '00', '0', '11', '0', '10', '0', '01', '0', '00', '0', '00', '0', '01', '0', '11', '0', '11', '0', '01', '0', '10', '0', '00', '0', '01', '0', '10', '0']

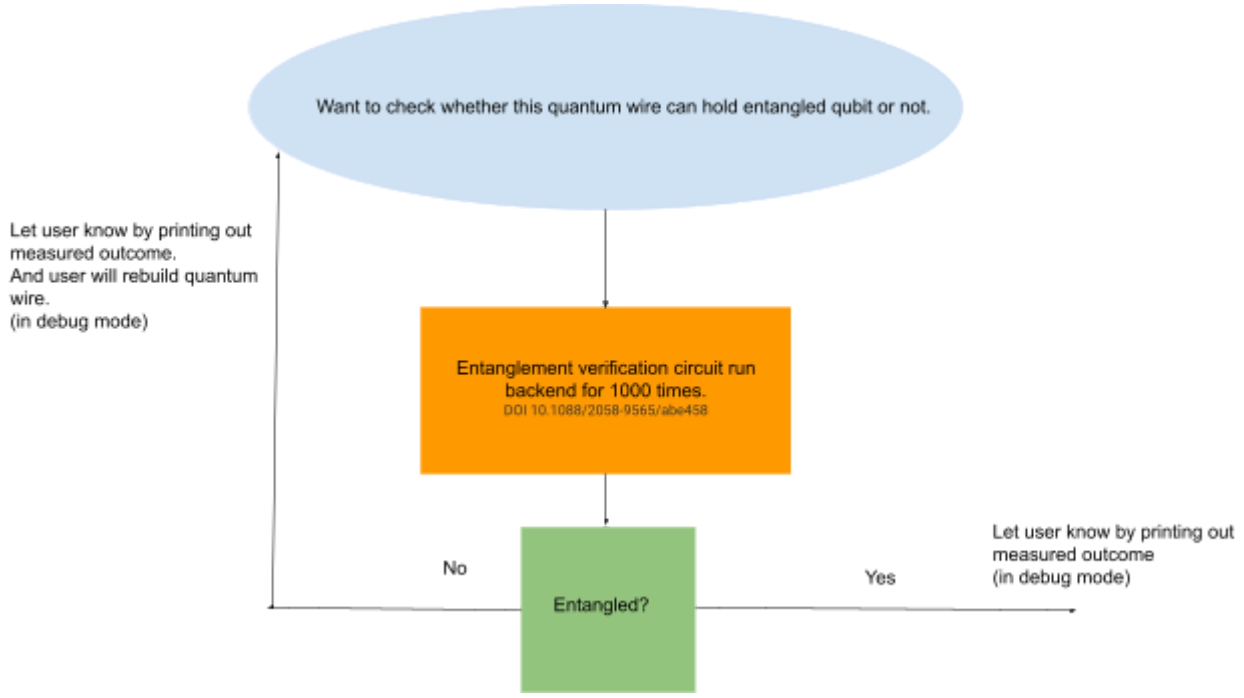
As you can see, a number of things have changed. The quantum function was split up, and the method of calling and utilizing these functions had to change in integration_node and node.py respectively. Results are now returned when they are calculated for each party as well. Additionally, the manner in which the nodes are connected, and disconnected changed slightly as well. This change allows for more graceful termination and also brings with it error handling for accidental disconnections, though that should not ever be an issue with our use case.

In conclusion, all of the changes have allowed for the successful integration of classical and quantum components.

-Quantum Entanglement Verification work

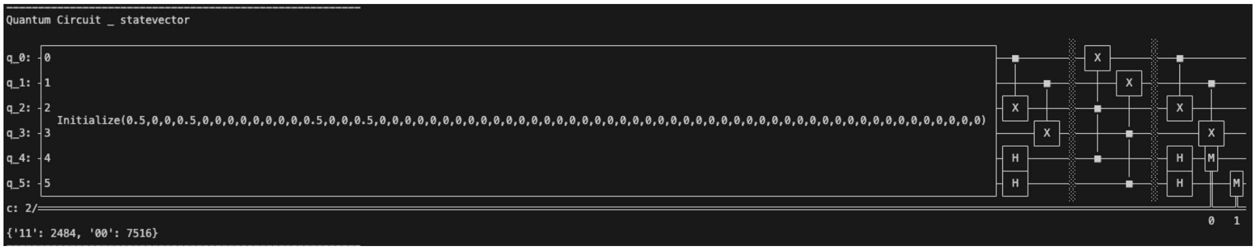
Quantum Entanglement verification protocol was also implemented successfully. Since there is no real-time quantum entanglement verification with qubits that are supposed to be measured, we supposed that this entanglement verification protocol is for testing that the quantum wire can hold any entanglement qubits before we send quantum information. So it is basically a capability check. The reason why we can't use quantum information which will be sent for entanglement verification is that we have to measure some qubits, and that will destroy the entangled qubit.

The diagram for how quantum entanglement circuit will use in our network is same as shown below.



[Figure: system diagram of our quantum entanglement verification diagram.]

We only use this function as debug mode since our simulation accounts only the ideal case, and we already verified that our network is always entangled. Thus, this function is more like our effort to embrace "realistic" situations as much as we can. The circuit diagram in our simulation and measured result is the same as down below.



[Figure: Quantum entanglement verification circuit and measured outcome]

According to the paper, there is at most ¼ probability to measure ‘11’ in 100 or 1000 times of measurement cycles when two qubits are entangled. If not, theoretically we can’t get any of ‘11’ from measurement. Our measurement outcome said that we measured 2484 times of ‘11’ and 7516 times of ‘00’ for 10000 times of repetition. That is ~24.84% chance to measure ‘11’ which is very close to the theoretical max value. That means our quantum wire from simulation verified that it can hold entangled qubits. We implemented this functionality in our network only available in debug mode.

-Quantum teleportation progress.

- **Ben** - Worked with Steven to build classical network protocol and finished to make our first primitive classical network. Assisting Derrick with integration.
- **Ohik** - Worked to build Quantum Entanglement verification circuit.
- **Steven** - Worked with Steven to build classical network protocol and finished to make our first primitive classical network.
- **Derrick** - Done integration part and preparing the visualization work.

Resources

Our git repository
https://github.com/Kcops11/SeniorDesignQuantum17

Books we are reading

- Quantum Computation and Quantum Information, Michael A. Nielson

Articles we found this week and reading

- Github Qiskit Community Tutorials
- When Entanglement meets Classical Communications: Quantum Teleportation for the Quantum Internet, IEEE Transactions on Communication, 2020, 10.1109/TCOMM.2020.2978071
- The controlled SWAP test for determining quantum entanglement, Quantum Science and Technology, 2021, *https://doi.org/10.1088/2058-9565/abe458*

Pending Issues

- There are no pending issues for this week since we all agreed on detailed functionalities of our first iteration network.
- We should prepare the final paper and presentation.

Individual Contributions

Team Member	Contribution	Weekly Hours	Total Hours
Benjamin Amick	Worked on classical network coding	10	86
Derrick Wright	Made visualization tool for presentation	10	86
Ohik Kwon	Worked on quantum teleportation simulation coding	10	86
Steven Tompary	Worked on classical network coding	10	86

Plans for Coming Week

- We will work on the final paper and presentation.