Simulation of Quantum Information Systems

1st Benjamin Amick
Iowa State University
bcamick@iastate.edu

2nd Derrick Wright
Iowa State University
dwright@iastate.edu

3rd Ohik Kwon
Iowa State University
ohik@iastate.edu

4th Steven Tompary
Iowa State University
stompary@iastate.edu

I. Introduction

Quantum information systems have emerged as a promising avenue for revolutionizing information processing, offering unparalleled capabilities in terms of data transmission and computation. This paper presents a design framework for a Quantum Information System (QIS)simulation which can be used to collect quantum information among multiple quantum computers.We demonstrate how this quantum router can facilitate this distribution, thereby enabling the seamless integration of a large number of quantum devices into a quantum network.

II. Design Constraints

Our network must be considered as a physical aspect of the quantum. When we run our simulation, although we don't use actual quantum bits unless we're purchasing computation resources from IBM, we should consider the nature of quantum such as quantum entanglement restriction such as distance, since our design must have a simulated quantum network for cluster computing which reflects limitations as much as possible. Also, our network must be available to work as cluster computing, not just for simple communication between different nodes. Although the degree of decentralization can be changed due to the property of our project, it must be working as a network for cluster computing. As well as functioning, our network must be cost-effective, as using too many Q bits means higher costs for our clients to run our network on IBM quantum computers. Our design goal is to save their research funding as much as possible related quantum networks to fabricate actual quantum nodes.

As our design is originally created for research, our network must be easy to implement when our researchers have all requirements to run our network such as quantum computers for running simulations to make them available to keep focus on Quantum information research.

III. Importance of Modularity in Quantum Information Science

IV. Network Design

Our network is built around two main networking systems, classcial and quantum network. Because our network will have a underlying classical network, we are able to use it for things which would be difficult to implement such as error correction as well as starting and stopping the nodes. This allows us to reduce the complexity of these nodes and improve our ability to maximize the number of nodes we can create. The interaction between classical and quantum network and our entire network's working protocol is depicted as Figure 1.

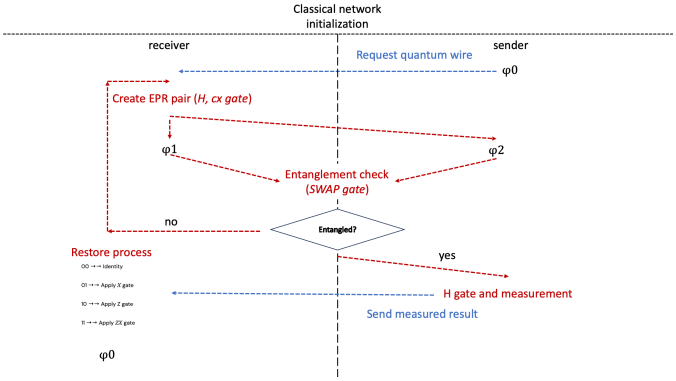

Fig. 1. Network protocol
Entire network protocol. It shows how quantum devices from the receiver and sender works after the initiation of the classical network.

V. Classical Network

The classical portion of our network is critical to allow the Quantum Nodes and Router to communicate with ease which is not found in Quantum computing. This information being sent is the command and control information needed to start, stop, and send instructions to the nodes. This classical network is also created to handle error correction as well. For design, we originally took inspiration from classical routers and task schedulers found in classical CPU's and routing protocols. For our nodes to communicate, we use TCP IP socket ports to send this information. Using these, we will be able to easily implement communication as well as error correction into our nodes.

VI. Qauntum Network

For simulating a quantum device, we are using python's Quiskit to simulate quantum circuits. Our quantum nodes were created to use the minimal amount of computation needed for the quantum operations. This was done so that the number of nodes and qubits could be easily increased as our algorithm required.

A. Quantum Teleportation

As a basis of all quantum communication, our network relies on quantum teleportation[1] to transmit the information

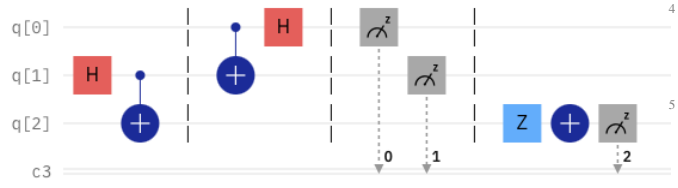

Fig. 2. Quantum teleporation circuit

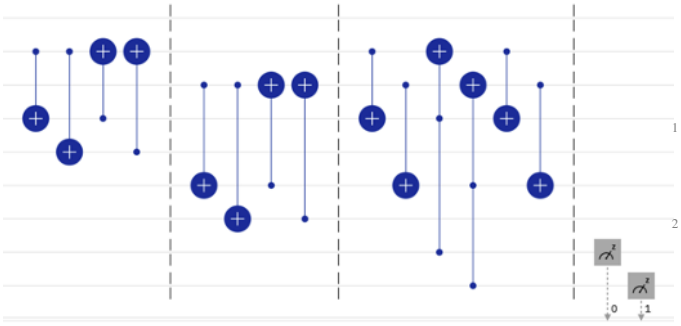

Fig. 3. Entanglement verification quantum circuit

between nodes and the router. Quantum teleportation circuit we used is same as below figure 2.

B. Quantum Entanglement verification

As quantum teleportation requires entangled qubits, it is essential to verify that our quantum wire can hold entangled quantum states. To verify this, we apply recent works on quantum entanglement verification protocols using two controlled SWAP gates[2]. Entire quantum circuits for entanglement verification protocol is same as below Figure 3.

VII. Process

To create our process for networking between the node and router, we implemented a modular quantum wire and split our code into easy-to-understand steps.

In the beginning of operations, our router first enables the classical network for communication by connecting the router socket to the node socket, where it sends a packet confirming the work the node is tasked with doing.

Step 1: Creating a random Qubit

Our first step is to generate a random qubit in the router. This qubit is created to simulate information generated by the router to be sent across our quantum network. Next, we generate an initialization gate with a random state and that gate's inverse. With those gates we just created, we create a quantum state vector using that initialization gate we just generated. Finally, combining the initialized state vector with a zero state creates a compound state vector.

```
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: Alice's Quantum Operations

Next, Alice performs her quantum operation on the compound state vector. This gives us the state vector, which combines information with Alice's Bell states.

```
Bell_info_statevector =
    Alice_quantum_operation(
    compound_statevector,simulator,False)
on_zero_states(init_statevector, False)
```

Step 3: Alice Measures Her Qubit

Alice's next step is to perform a measurement on her qubit in the Bell information state vector and return the result to send to the router using the classical network.

```
Alice_measurement_result ,
    Alice_statevector = Alice_measure(
    Bell_info_statevector, simulator,
    False)
    return(Alice_measurement_result)
```

Step 4: Bob's Quantum Operation

Bob's quantum operation is based on Alice's measurement result and the state vector obtained after Alice's operation.

```
Bob_statevector = Bob_quantum_opertaion(
    Alice_measurement_result,
    Alice_statevector, simulator, False)
```

Step 5: Bob's Quantum Measurement

Next, Bob measures his qubit in the state vector obtained after his quantum operation and returns the result to send to the router through the classical network.

```
Bob_measurement_result = Bob_measure(
    Bob_statevector, inverse_init_gate,
    simulator, False)
    return(Bob_measurement_result)
```

Step 6: Finalization and Output

Finally, these three bits, one received from Alice, one from Bob, and one used as a baseline, are received by the router and made to form a histogram to show the distribution of these values. We should see an even distribution between the four values if everything worked correctly.

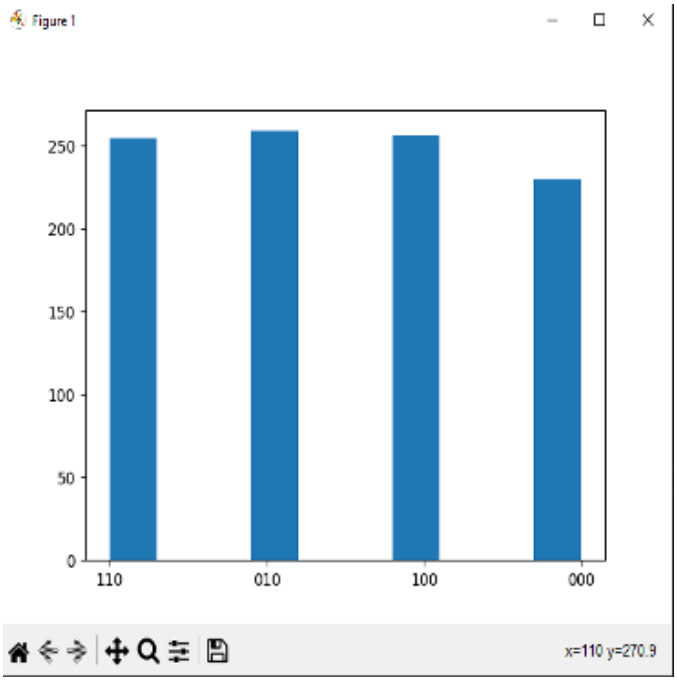

Fig. 4. Simulated result
Equally distributed measured outcome with zero measured information quantum bits shows that designed modular quantum network works well.

Extra Step: Entanglement Verification

We we briefly discussed before, checking feasibility of holding entangled states within quantum wire is essential to quantum network. We implemented two controlled SWAP gates to check the feasibility of holding of quantum entanglement verification[2]. It takes bell state and return wire's feasibility of keeping entangled states through 1024 times repeated measurement.

```
Entanglement verification =
    entanglement_verification(simulator,
    Bell_state, debug = False)
```

VIII. Result

This networking design allows for reliable and fast communication between as many quantum nodes as the operation requires. The result of testing running shows that our modular quantum networks works well. For the test, we randomly generate qubits from the state zero and teleported using our quantum network. After the quantum router received each quantum information, then it applies the inverse quantum operation to convert random qubit to zero state. Expected result is equally distributed measured information with zero for teleported measured quantum information. Figure 4 shows that the modular quantum teleportation networks works properly.

IX. Conclusion

In this work, we developed quantum modular network that using designed TCP-IP based classcial network with quantum entangled verification protocol.

References

[1] A. S. Cacciapuoti, M. Caleffi, R. Van Meter and L. Hanzo, "When Entanglement Meets Classical Communications: Quantum Teleportation for the Quantum Internet," in IEEE Transactions on Communications, vol. 68, no. 6, pp. 3808-3833, June 2020, doi: 10.1109/TCOMM.2020.2978071.

[2] Steph Foulds, Viv Kendon, and Tim Spiller, "The controlled SWAP test for determining quantum entanglement," in Quantum Science and Technology, 6 035002, April 2021, doi: 10.1088/2058-9565/abe458