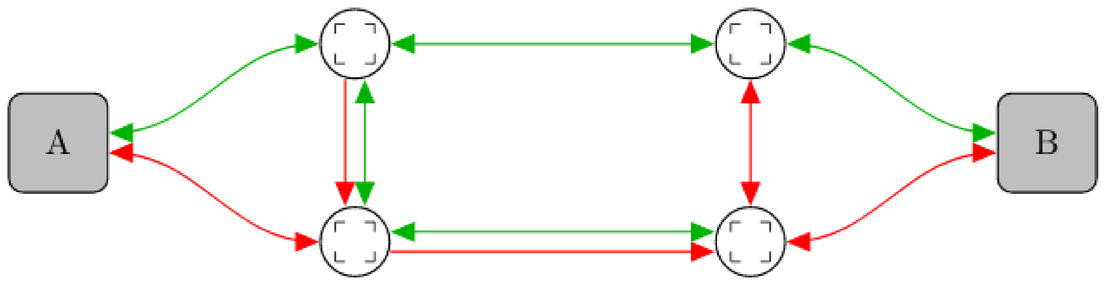
QuNetSim: A Software Framework for Quantum Networks

Stephen DiAdamo, Janis Nötzel, Benjamin Zanger, Mehmet Mert Beşe

TU München: Lehrstuhl für Theoretische Informationstechnik

Technische Universität München



What is QuNetSim?

QuNetSim is:

- A Python simulation framework for developing quantum network protocols
- Used to develop and test protocols at a high level
- Used to develop and test routing algorithms for quantum networks
- A tool for teaching about quantum networks
- Open source (feel free to contribute to the project on Github!)

Who should use QuNetSim?

Students:

- Can use QuNetSim to learn about quantum network protocols by implementing them in software
- Can design master's and research projects by taking a well known protocol and implementing it under various simulated network configurations

Instructors:

- Can demonstrate to their students using software (rather than research papers) quantum networking protocols

Researchers:

- Can use QuNetSim to test application and routing protocols under various simulated network conditions and security threats

How Does QuNetSim Work?

- QuNetSim simulates a layered architecture similar to the (classical) Internet (e.g. OSI layering)
- Adds a layer of thread synchronization for asynchronous simulated communication
- Uses qubit simulators like ProjectQ [1], CQC [2], and EQSN [3] as qubit engines which can be easily swapped depending on the simulation requirements

QuNetSim Links

- **Github Repo:** www.github.com/tqsd/QuNetSim
- **Online Documentation:** tqsd.github.io/QuNetSim/
- **arXiv:** arXiv:2003.06397

References

[1] Steiger, Damian S., Thomas Häner, and Matthias Troyer. "ProjectQ: an open source software framework for quantum computing." Quantum 2 (2018): 49.

[2] Dahlberg, Axel, and Stephanie Wehner. "SimulaQron-a simulator for developing quantum internet software." Quantum Science and Technology 4.1 (2018): 015001.

[3] Zanger, Benjamin, Stephen DiAdamo, and Janis Nötzel, EQSN, Github repository, `https://github.com/tqsd/EQSN_python`

stephen.diadamo@tum.de, www.lti.ei.tum.de

Example Simulation: CHSH Game

Referee:

```
def ref_(ref, alice, bob):
    x = random.choice([0, 1])
    ref.send_classical(alice, x)
    y = random.choice([0, 1])
    ref.send_classical(bob, y)

    alice_response = ref.get_classical(alice, seq_num=0, wait=5)
    bob_response = ref.get_classical(bob, seq_num=0, wait=5)

    a = alice_response.content
    b = bob_response.content

    if x & y == a ^ b:
        print('Winners!')
    else:
        print('Losers!')
```

Alice:

```
def alice_(alice, ref, bob)
    referee_message = alice.get_classical(ref, seq_num=0, wait=5)
    x = referee_message.content
    epr = alice.get_epr(bob)

    if x == 0:
        res = epr.measure()
        alice.send_classical(ref, res)
    else:
        epr.H()
        res = epr.measure()
        alice.send_classical(ref, res)
```

Bob:

```
def bob_(bob, ref, alice):
    referee_message = bob.get_classical(ref, seq_num=0, wait=5)
    y = int(referee_message.content)
    epr = bob_host.get_epr(alice)
    if y == 0:
        epr.ry(-2.0 * math.pi / 8.0)
        res = epr.measure()
        bob.send_classical(ref, res)
    else:
        epr.ry(2.0 * math.pi / 8.0)
        res = epr.measure()
        bob.send_classical(ref, res)
```

Network:

```
network = Network.get_instance()
network.start()
# Build network
alice = Host('alice')
alice.add_connection('ref')
alice.start()
bob = Host('bob')
bob.add_connection('ref')
bob.start()
ref = Host('ref')
ref.add_connections(['alice', 'bob'])
ref.start()
# Logic for EPR generation skipped (see online docs)
network.add_hosts([alice, bob, ref])

alice.run_protocol(alice_, (ref.host_id, bob.host_id))
bob.run_protocol(bob_, (ref.host_id, alice.host_id))
ref.run_protocol(ref_, (alice.host_id, bob.host_id), blocking=True)
```