Experimental demonstration of entanglement delivery using a quantum network stack

QIRG | IETF 116

Carlo Delle Donne (he/him)
Ph.D. candidate
QuTech | TU Delft (The Netherlands)
c.delledonne@tudelft.nl

27 March 2023
Outline

- Background and goals
- Programmable quantum network nodes
- Implementation of a quantum networking stack
- Experimental results
- Future work
This presentation is based on:

*Experimental demonstration of entanglement delivery using a quantum network stack*

M. Pompili, C. Delle Donne, I. te Raa, B. van der Vecht, M. Skrzypczyk, G. M. Ferreira, L. de Kluijver, A. J. Stolk, S. L. N. Hermans, P. Pawełczak, W. Kozlowski, R. Hanson, S. Wehner

npj Quantum Information 8.1 (2022)

DOI: 10.1038/s41534-022-00631-2
Background and goals
Assumptions:
- Entanglement-based quantum networks
- Heralded entanglement generation

Existing tools:
- Multi-node quantum networks (e.g. [1])
- Protocol designs for quantum data plane (e.g. [2, 3])

[1] “Realization of a Multinode Quantum Network of Remote Solid-State Qubits”. Pompili et al., Science 372.6539 (2021)
[2] “A Link Layer Protocol for Quantum Networks”. Dahlberg et al., SIGCOMM (2019)
[3] “Designing a Quantum Network Protocol”. Kozlowski et al., CoNEXT (2020)
Goals

- **Heterogeneous quantum networks**
  Control end nodes and switches that are based on various quantum platforms

- **Programmable network nodes**
  Control of nodes must be platform-independent and application-agnostic (not ad-hoc)

- **Configurable resource allocation**
  Node resources are allocated according to local demand and global needs
Programmable quantum network nodes
Programmable quantum network
Programmable quantum network

Heterogeneous quantum networks
Programmable quantum network

Programmable network nodes

Heterogeneous quantum networks
Programmable quantum network

Centralized network controller

Node controller
- Network schedule
- Network stack
- QoS policies
- Operating system
- Quantum device driver
- Quantum device

Node controller
- Network schedule
- Network stack
- QoS policies
- Operating system
- Quantum device driver
- Quantum device

Node controller
- Network schedule
- Network stack
- QoS policies
- Operating system
- Quantum device driver
- Quantum device

Configurable resource allocation

Programmable network nodes

Heterogeneous quantum networks
Programmable quantum network

Configurable resource allocation

Programmable network nodes

Heterogeneous quantum networks
Implementing a quantum networking stack
## A quantum network stack

| Transport | Qubit transmission          |
|-----------|----------------------------|
| Network   | Long-distance entanglement |
| Link      | Robust entanglement generation |
| Physical  | Attempt entanglement generation |

[1] “Realization of a Multinode Quantum Network of Remote Solid-State Qubits”. Pompili et al., Science 372.6539 (2021)
[2] “A Link Layer Protocol for Quantum Networks”. Dahlberg et al., SIGCOMM (2019)
[3] “Designing a Quantum Network Protocol”. Kozlowski et al., CoNEXT (2020)

Implementing a quantum networking stack | 14
Network schedule
- timeslots with millisecond resolution
- generated on control plane

Entanglement attempts
- triggered at nanosecond resolution
- synchronized across two nodes
Deployment

Node A  
NV center  

2 meters  

Node B  
NV center  

Host

Runtime (Py)

PC  Eth

Network stack

Stack (C++)

FreeRTOS (C)

MicroZed (Zynq-7000)

Eth  SPI

QDevice

QDevice stack

ADwin-Pro II

SPI  I/O

Qubit control
Experimental results
Applications and metrics

Applications:
› Entanglement requests at various fidelities
› Full tomography of entangled state
› Remote state preparation

Metrics:
› Quantum performance $\rightarrow$ fidelity of entangled state
› Classical performance $\rightarrow$ entanglement request latency
Applications and metrics

Simplified application code (uses NetQASM SDK [4])

```python
req_fidelities = [0.50, 0.55, 0.60, 0.65, 0.70, 0.75, 0.80]
meas_bases = [(-X, -X), (-X, +X), (-Y, -Y), (-Y, +Y), ...]

for rep in range(125):
    for fid in req_fidelities:
        for meas in meas_bases:
            client_basis = meas[0]
            client_qubit = create_ent(min_fidelity=fid)
            client_qubit.rotate_basis(client_basis)
            outcomes[rep, fid, meas] = client_qubit.measure()
```

[4] “NetQASM”. Dahlberg et al., Quantum Sci. Technol. 7 (2022) — github.com/QuTech-Delft/netqasm
Results: Measured fidelity

Targeted: physical layer tries to deliver fidelity higher than requested

Measured: almost always larger than requested 🎉 🎄
Results: Latency breakdown

Physical layer: overhead increases with requested fidelity
Results: Latency breakdown

- Link layer: roughly constant overhead

![Graph showing latency breakdown with different requested fidelities and categories: Link layer protocol, Interface, Physical layer (CR check), Physical layer (entanglement).]
Future work
Future work

- Operating system for quantum network nodes (ONGOING)
- More complex applications / concurrent applications (ONGOING)
- Larger-scale experiments (> 2 nodes at least)
- Node- and network-level scheduling
- Design and implementation of control plane
Experimental demonstration of entanglement delivery using a quantum network stack

QIRG | IETF 116

Carlo Delle Donne (he/him)
Ph.D. candidate
QuTech | TU Delft (The Netherlands)
c.delledonne@tudelft.nl