QUANTUM NETWORK **PARAMETERS**

June 2023

Table of Contents

1 Contents

Abstract 2
1. Introduction 2
   1.1 Defining parameters 2
2. Standards bodies 2
   2.1. Internet Engineering Task Force 2
   2.2. European Telecommunications Standards Institute 3
   2.3. International Telecommunication Union 3
   2.4. Institute of Electrical and Electronic Engineers 4
   2.5. Global System for Mobile Communications 5
   2.6. Quantum Internet Alliance 5
3. Quantum network simulation packages 6
   3.1. SimulaQron 6
   3.2. Quantum Network Simulator 8
   3.3. Simulator for Quantum Networks and Channels 8
   3.4. Quantum Internet Simulator Package 10
   3.5. Simulator of Quantum Network Communication 10
   3.6. The NETwork Simulator for QUantum Information with Discrete Events 11
   3.7. Quantum Key Distribution Network Simulation Module 12
   3.8. Quantum/Classical 12
   3.9. QuNET 12
   3.10. NetQASM 13
4. Conclusion 13

About the Quantum Economic Development Consortium

The Quantum Economic Development Consortium (QED-C®) is an industry-driven consortium managed by SRI International. With a diverse membership representing industry, academia, government, and other stakeholders, the consortium seeks to enable and grow the quantum industry and associated supply chain. For more about QED-C, visit our website at quantumconsortium.org.

Acknowledgments

QED-C thanks the QED-C Standards and Performance Metrics Committee and, particularly, GE Research and Stephen F. Bush for material contributions to this report.

Abstract

This report provides findings on quantum network parameters developed by defining high-level components, input/output, and controls based upon examination of proposed abstractions from standards bodies and quantum network simulation packages. This is the second in a series of two QED-C reports on quantum networks; the prior report presented findings on quantum network abstraction.[1]

1. Introduction

This report details findings derived from the abstractions defined in a companion report[2] and seeks a minimal set of control parameters and observables to facilitate quantum communication and networking protocol development. These parameters are abstracted with the goal of being universally applicable regardless of how quantum network technology evolves. There are currently a massive number of parameters from which to choose when designing a quantum network. The art in standards development is to cull the list to those that are most essential.

1.1 Defining parameters

It is relatively easy to expose all control and observable parameters of a system, although this is generally unwise as it leads to overhead, complexity, and security issues. Selecting the minimal set of parameters to expose is extremely difficult. Enough control and observability must be exposed for the system architecture to operate given uncertainties in applications and technical evolution. This report explores the abstractions derived from a previous report[1] to examine their selected sets of exposed parameters.

2. Standards bodies

This section examines standards development organizations and their attempts to abstract a quantum network.

2.1. Internet Research Task Force

The Internet Research Task Force (IRTF) Quantum Internet Research Group (QIRG) has been working on identifying use cases, frameworks, and principles for creating a quantum internet.[3] There are few explicitly defined parameters, which include the notion of a quantum virtual circuit where an application can indicate quality of service (QoS) parameters such as the required capacity in end-to-end Bell pairs per second (BPPS) and the required fidelity of the Bell pairs. As an analogy, Multiprotocol Label Switching (MPLS) routes traffic using the shortest path based on “labels,” rather than network addresses, to handle forwarding over private, wide area networks (WANs). Network applications specify the required bandwidth in bits per second (BPS) and other constraints when they create a new label switched path, LSP, that is a path through an MPLS network, set up by a signaling protocol.

[1] QED-C, *Quantum Network Abstraction* (April 2023).
[2] *Quantum Network Abstraction*.
[3] Wojciech Kozlowski, Stephanie Whener, Rodney Van Meter, Bruno Rijsman, Angela Sara Cacciapuoti, Marcello Caleffi, and Shota Nagayama, “Architectural Principles for a Quantum Internet,” (2022), IEFT Datatracker, https://datatracker.ietf.org/doc/draft-irtf-qirg-principles/.

2.2. European Telecommunications Standards Institute

The Industry Specification Group (ISG) on Quantum Key Distribution (QKD) has performed groundbreaking standard development work in laying a foundation for specifying quantum channels.[4] In particular, their terminology and characteristics have been adopted by the National Institute of Standards and Technology (NIST) for its dictionary of single-photon terms. However, the abstractions developed within the European Telecommunications Standards Institute (ETSI) QKD specifications are primarily oriented toward quantum key generation and key management, as they should be. The specifications are comprised of the following and mapped to the QED-C standards visualization diagram in Figure 1.

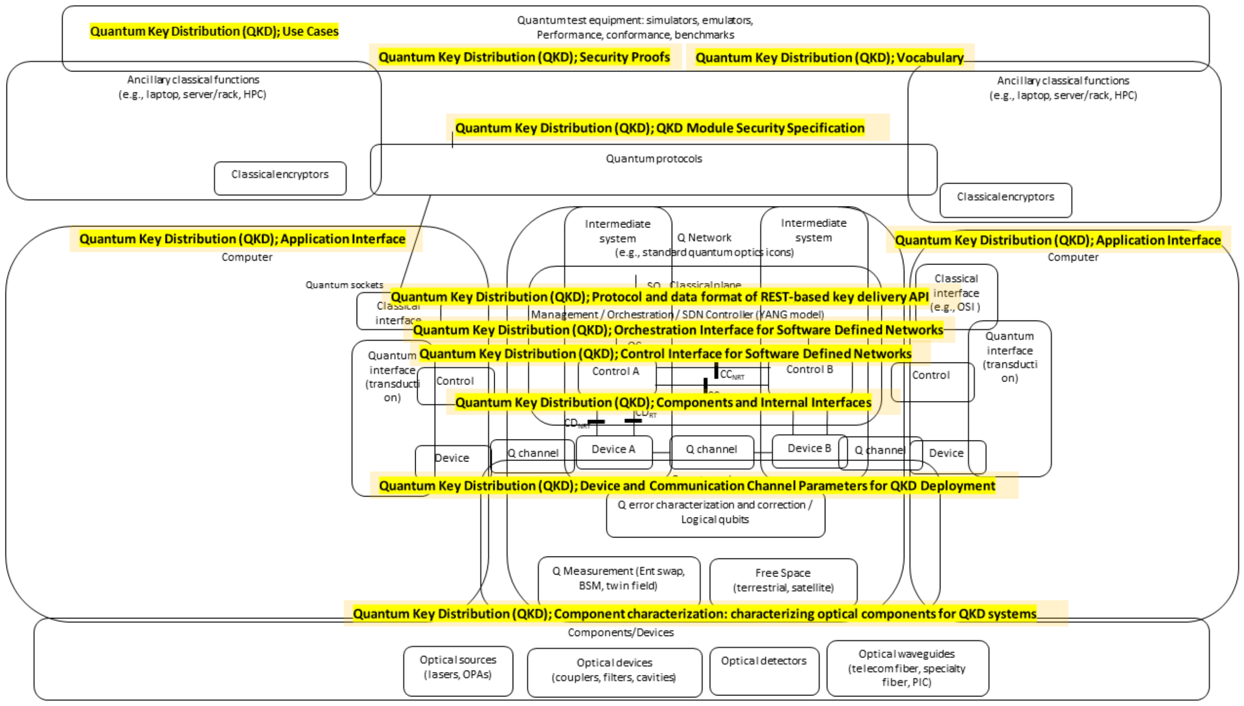

Quantum Key Distribution (QKD); Use Cases
Quantum test equipment: simulators, emulators, Performance, conformance, benchmarks
Quantum Key Distribution (QKD); Security Proofs
Quantum Key Distribution (QKD); Vocabulary
Ancillary classical functions (e.g., laptop, server/rack, HPC)
Ancillary classical functions (e.g., laptop, server/rack, HPC)
Quantum Key Distribution (QKD); QKD Module Security Specification
Quantum protocols
Classical encryptors
Classical encryptors
Quantum Key Distribution (QKD); Application Interface
Computer
Intermediate system
Q Network
(e.g., standard quantum optics icons)
Intermediate system
Quantum Key Distribution (QKD); Application Interface
Computer
Quantum sockets
Classical interface
SQ Classical plane
Quantum Key Distribution (QKD); Protocol and data format of REST-based key delivery API
Classical interface (e.g., OSI)
Management / Orchestration / SDN Controller (YANG model)
Quantum Key Distribution (QKD); Orchestration Interface for Software Defined Networks
Quantum interface (transduction)
Quantum Key Distribution (QKD); Control Interface for Software Defined Networks
Quantum interface (transduction)
Control
Control A
CC_NRT
Control B
Control
Quantum Key Distribution (QKD); Components and Internal Interfaces
CD_NRT
CD_RT
Device
Q channel
Device A
Q channel
Device B
Q channel
Device
Quantum Key Distribution (QKD); Device and Communication Channel Parameters for QKD Deployment
Q error characterization and correction / Logical qubits
Q Measurement (Ent swap, BSM, twin field)
Free Space (terrestrial, satellite)
Quantum Key Distribution (QKD); Component characterization: characterizing optical components for QKD systems
Components/Devices
Optical sources (lasers, OPAs)
Optical devices (couplers, filters, cavities)
Optical detectors
Optical waveguides (telecom fiber, specialty fiber, PIC)

Figure 1: ETSI ISG QKD specifications mapped to the QED-C standard visualization diagram. (Source: QED-C Standards and Performance Metrics TAC)

2.3. International Telecommunication Union

The International Telecommunication Union (ITU) has several sets of QKD specifications including:

- ITU Telecommunication Standardization Sector (ITU-T) Focus Group on Quantum Information Technology for Networks (FG QIT4N)[5]
- Y.3800-Y.3999 Quantum key distribution networks[6]

[4] ETSI, "Industry Specification Group (ISG) on Quantum Key Distribution (QKD)" (2023), https://www.etsi.org/committee/1430-qkd.
[5] ITU, "ITU-T Focus Group on Quantum Information Technology for Networks (FG-QIT4N)" (2023), https://www.itu.int/en/ITU-T/focusgroups/qit4n/Pages/default.aspx.
[6]ITU-T Publications, eY.3800 : Overview on Networks Supporting Quantum Key Distribution," (8 October 2020), https://www.itu.int/rec/T-REC-Y.3800/en.

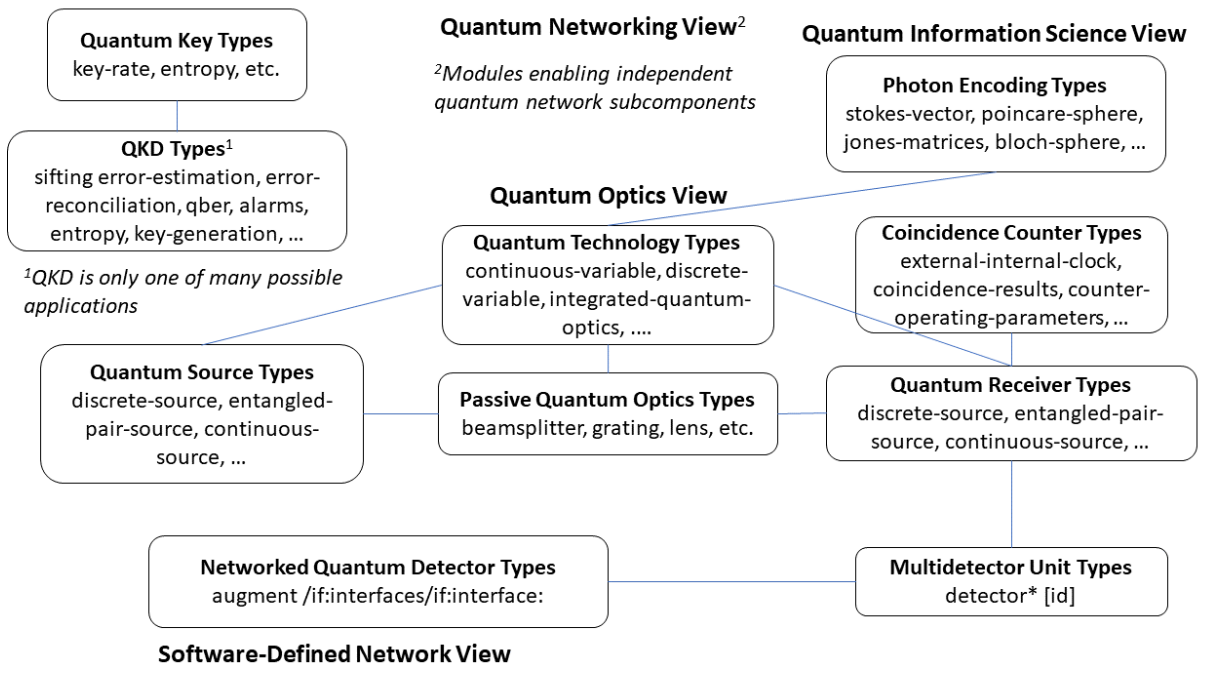
The ITU-T Focus Group on Quantum Information Technology for Networks (FG QIT4N) references the following parameters:

- Quantum channel status
    - QBER
    - channel loss
    - estimated secret key rate
- QKD module status
    - decoy state setting
    - output raw key rate
    - output secure secret key rate
- Quantum channel status
    - QBER
- QKD module status
    - output raw key rate
    - output secure secret key rate
- Quantum channel status
    - excess noise
    - channel transmission
    - estimated secret key rate
- QKD module status
    - shot noise variance
    - sender modulation variance
    - output raw key rate
    - output secret key rate

2.4. Institute of Electrical and Electronic Engineers

The Institute of Electrical and Electronics Engineers (IEEE) P1913 YANG Model for Software-Defined Quantum Communication project has been operating under the COM/NetSoft-SC/QuantumComm sponsorship of IEEE Standards.[7] This standard defines a YANG model that enables configuration of quantum endpoints in a communication network to dynamically create, modify, or remove quantum protocols or applications. The protocol design facilitates future integration with Software-Defined Networking. The IEEE P1913 YANG Model for Software-Defined Quantum Communication project variables considered are illustrated in Figure 2.

[7] See "IEEE SA - The IEEE Standards Association" (2023), https://standards.ieee.org/ieee/1913/.



Quantum Key Types
key-rate, entropy, etc.
QKD Types¹
sifting error-estimation, error-reconciliation, qber, alarms, entropy, key-generation, …
¹QKD is only one of many possible applications
Quantum Networking View²
²Modules enabling independent quantum network subcomponents
Quantum Information Science View
Photon Encoding Types
stokes-vector, poincare-sphere, jones-matrices, bloch-sphere, …
Quantum Optics View
Quantum Technology Types
continuous-variable, discrete-variable, integrated-quantum-optics, ….
Coincidence Counter Types
external-internal-clock, coincidence-results, counter-operating-parameters, …
Quantum Source Types
discrete-source, entangled-pair-source, continuous-source, …
Passive Quantum Optics Types
beamsplitter, grating, lens, etc.
Quantum Receiver Types
discrete-source, entangled-pair-source, continuous-source, …
Networked Quantum Detector Types
augment /if:interfaces/if:interface:
Multidetector Unit Types
detector* [id]
Software-Defined Network View

Figure 2: Abstraction into different categories of quantum network YANG modules (Source: Stephen F Bush).

2.5. Global System for Mobile Communications

Global System for Mobile Communications (GSMA) Internet Group (IG)[8] does not appear to explicitly suggest specific parameters, although they do note some parameters missing from standards in gaps:

- the level of amplified spontaneous emission (ASE) noise from optical amplifiers, or in band crosstalk from classical channels, that can be tolerated by a quantum channel receiver
- the maximum transmission power that should be allowed to a classical channel not to interfere with a QKD channel
- attenuation/reflection characteristics of all components and sub-systems in a quantum node/system
- interfaces of all parts of all components and sub-systems in a quantum node/system
- interfaces for interoperability and interworking of quantum nodes/systems within current networks
- interfaces for management, control, and orchestration
- assurance (policies and processes to ensure that services offered over networks meet a pre-defined service quality level for an optimal subscriber experience)

2.6. Quantum Internet Alliance

The Quantum Internet Alliance (QIA) is a consortium of partners from quantum research groups and high-tech companies in Europe coordinated by QuTech from Delft University in the Netherlands.[9] The Quantum Internet Alliance (QIA) was a European project launched in October 2018, running until

[8] GSMA, *Quantum Networking and Service Version 1.0* (2021), https://www.gsma.com/newsroom/wp-content/uploads/IG-12-Quantum-Networking-and-Service.pdf.
[9] See "Quantum Internet Alliance" (2023), https://quantum-internet.team.

September 2021. The QIA targeted a blueprint for a pan-European Quantum Internet through ground-breaking technological advances, culminating in the first experimental demonstration of a fully integrated network stack running on a multi-node quantum network. Most of the QIA documents appear to be a re-packaging of U Delft papers and are covered elsewhere.

3. Quantum network simulation packages

This section examines quantum network simulation packages and explores their attempts to abstract the quantum network.

3.1. SimulaQron

SimulaQron is not intended to simulate quantum networks, but rather to simulate/emulate quantum application development in a platform-independent manner that would utilize a quantum internet.[10] Thus, its abstractions are suited toward that end. It is implemented in Python.

Arguably the most interesting abstraction is the universal interface, the Classical Quantum Combiner (CQC). CQC can be understood as an extended instruction set that—next to supporting "standard" quantum instructions such as performing gates or measurements—contains special features tailored to a quantum network. This includes, for example, commands to produce entanglement or transmit qubits. Applications can be realized in the platform-independent part of the quantum internet system by sending the appropriate instructions using CQC to the underlying quantum processing system.

Messages in the CQC interface are abstracted into different types. One of these types is COMMAND, which instructs SimulaQron (or the physical hardware) to perform a certain command. An example would be a gate operation or sending a qubit to another node. In addition to the type of message and what command to perform, CQC messages also contain information regarding, for example, the CQC version, an application identifier, the length of the message (including additional commands if any) and, if applicable, the qubit identifiers a command concerns, the IP and port-number for the node to which a qubit should be sent. Other options that can be specified is whether notification should be returned when the command is finished and if the node should be blocked during the execution of the command.

Options that can be specified when sending a COMMAND message are listed by option name and effect:

- OPT_NOTIFY : Send a notification when command is done
- OPT_ACTION : Execute further commands when done
- OPT_IFTHEN : Execute further commands based on result
- OPT_BLOCK : Block until command is done

A HELLO message can be used to check that the connection to the hardware/SimulaQron is up and to get some specifications about the hardware. The commands that can be specified using the type COMMAND are discussed in the following section. FACTORY is a type similar to COMMAND, but here a certain command should be executed a specified number of times. An example of the use of FACTORY is to instruct the hardware to continuously produce a specified number of entangled EPR-pairs. An EPR-pair could then be ready to be used whenever it is needed by the protocol running on the network.

[10] A Dahlberg and S Wehner, "SimulaQron—A Simulator for Developing Quantum Internet Software," *Quantum Science and Technology* 4:1, p. 015001 (2019), doi:10.1088/2058-9565/aad56e.

Types of messages from the application to the hardware/SimulaQron are listed by type name and effect:

- TP_HELLO : Alive check (get hardware specification)
- TP_COMMAND : Execute (list of) command
- TP_FACTORY : Execute (list of) command repeatedly
- TP_GET_TIME : Get creation time of qubit

Types of messages from the hardware/SimulaQron to the application are listed by type and effect:

- TP_NEW_OK : Qubit was allocated
- TP_EXPIRE : Qubit has expired
- TP_DONE : Done with command
- TP_RECV : Received qubit
- TP_EPR_OK : Created EPR pair
- TP_MEASOUT : Measurement outcome
- TP_INF_TIME : Return timing information
- ERR_GENERAL : General-purpose error
- ERR_NOQUBIT : No more qubit
- ERR_UNKNOWN : Unknown qubit ID
- ERR_UNAVAILABLE : Cannot allocate qubit
- ERR_DENIED : No access to qubit
- ERR_VERSION : CQC version not supported
- ERR_UNSUPP : Sequence not supported
- ERR_TIMEOUT : Timeout

Commands that can be specified when using the type COMMAND and are listed by command name and effect:

- CMD_NEW : Ask to allocate a new qubit
- CMD_ALLOCATE : Ask to allocate multiple qubits
- CMD_RELEASE : Release qubit to be used by other app.
- CMD_RESET : Reset qubit to |0>
- CMD_MEASURE : Measure qubit (demolition)
- CMD_MEASURE_INPLACE : Measure qubit (non-demolition)
- CMD_SEND : Send qubit to another node
- CMD_RECV : Ask to receive qubit
- CMD_EPR : Create EPR pair with another node
- CMD_RECV_EPR : Ask to receive half of EPR pair
- CMD_SWAP : Entanglement swapping
- CMD_I : Identity
- CMD_X : Pauli X
- CMD_Y : Pauli Y
- CMD_Z : Pauli Z
- CMD_H : Hadamard
- CMD_K : K gate (Z to Y)

- CMD_T : T gate
- CMD_ROT_X : Rotation around X (multiple of 2 Pi/256)
- CMD_ROT_Y : Rotation around Y (multiple of 2 Pi/256)
- CMD_ROT_Z : Rotation around Z (multiple of 2 Pi/256)
- CMD_CNOT : CNOT (this qubit as control)
- CMD_CPHASE : CPHASE (this qubit as control)

3.2. Quantum Network Simulator

The Quantum NETwork SIMulator (QuNetSim) is implemented in a Python framework, and its abstraction is to focus solely on the network layer.[11] The goal of QuNetSim is to make it easier to investigate and test quantum networking protocols over various quantum network configurations and parameters. Network objects include:

- Message
- Qubit
- Quantum Storage
- Classical Storage
- Quantum Connection
- Classical Connection
- Packet
- Routing Packet

QuNetSim supports limiting the number of qubits stored at a host. The number of entangled qubits and data qubits can be limited separately or a limit for the combined number of qubits can be set. The host methods for setting the limits are as follows:

- set_epr_memory_limit
- set_data_qubit_memory_limit.

There are three parameters (other than the host running the protocol):

- the ID of the distributor
- the receiver that is the holder of the other half of the EPR pair
- an agreed upon ID for the EPR pair that will be generated.

It should be noted that qubits have IDs for easier synchronization between parties.

3.3. Simulator for Quantum Networks and Channels

Simulator for Quantum Networks and Channels (SQUANCH) is implemented as a Python library and appears to be rather highly abstracted.[12] Their definition for a quantum agent is essentially a networked quantum Turing machine.

[11] S DiAdamo, J Nötzel, B Zanger, and M Mert Beşe, "QuNetSim: A Software Framework for Quantum Networks" (2020), arXiv:2003.06397.
[12] B Barlett, "A Distributed Simulation Framework for Quantum Channels," (2018), arXiv:1808.07047.

A quantum agent is represented by a 5-tuple (C; MC; MQ; G; T):

- C is the configuration of channels connecting the agent to other agents
- MC is the classical memory
- MQ is the quantum memory
- G is an arbitrary but finite set of quantum operations
- T is a classical Turing machine

In the non-networked case that C = (0,0,0,0), the Agent has full access to the state space of their quantum system, and this model is seemingly reducible to the definition of a quantum Turing machine.

Class squanch.qubit.QSystem(num_qubits, index=None, state=None) represents a multi-body, maximally-entangleable quantum system, and contains references to constituent qubits as well as its parent QStream if applicable. Quantum state is represented as a density matrix in the computational basis.

Class squanch.qubit.Qubit(qsystem, index) is a wrapper class representing a single qubit in an existing quantum system.

Class squanch.qstream.QStream(system_size, num_systems, array=None, agent=None) efficiently represents many separable quantum subsystems in a contiguous block of shared memory. Qsystems and Qubits can be instantiated from the state of this class.

Class squanch.agent.Agent(qstream, out=None, name=None, data=None) represents an entity (e.g., Alice or Bob) that can send messages over classical and quantum communication channels. Agents have the following properties:

- Incoming and outgoing classical and quantum channels connecting them to other agents
- Classical memory, implemented simply as a Python dictionary
- Quantum memory, implemented as a Python dictionary of qubits stored in keys of agent names
- Runtime logic in the form of an Agent.run() method

Class squanch.channels.QChannel(from_agent, to_agent, length=0.0, errors=()) is the base class for a quantum channel connecting two agents. Class squanch.channels.CChannel(from_agent, to_agent, length=0.0) is the base class for a classical channel connecting two agents.

Class squanch.errors.QError(qchannel) is a generalized quantum error model apply(qubit) that applies the error to the transmitted qubit. This method can be overwritten in child classes.

3.4. Quantum Internet Simulator Package

The Quantum Internet Simulator Package (QuISP) is implemented in OMNeT++ and introduces at least two abstractions: error handling and ruleset operation.[13] Currently, there are more than 50 types of parameters that can be given through the INI file. The following parameters are most likely to be configured by users:

- Network: The name of the network written in the NED language, specifying the number and types of nodes and the network topology
- Channel errors: The errors on a quantum channel. This specifies how much and what types of errors occur over a particular distance. Currently, all Pauli errors and photon loss errors are supported, and users can specify the ratio of each error.
- Memory errors: In addition to Pauli errors, it supports relaxation and excitation errors. As the simulation time passes, this error accumulates on each qubit.
- Gate errors: In a realistic situation where we assume that devices are imperfect, there are always small operational errors in quantum gates. We can handle errors on basic gate sets (Pauli gates, Hadamard, Controlled-X).
- Measurement errors: Quantum memories must be measured to get internal information that affects the quantum state itself. Measurement errors are composed of Pauli errors.
- BSA errors: It is possible to have different errors for internal and external BSAs. Tunable parameters are photon loss, photon detection rate, and dark count probability.
- Photon emission success probability: The probability that a quantum memory successfully emits a photon that is then captured by the fiber.
- Traffic pattern: A configurable parameter for traffic generation pattern. Currently, this simulator supports single traffic generation from one node to the other random node and multiple traffic generation from all end nodes to randomly selected nodes.
- Measurement count: The number of measurements in tomography. When the required precision of tomography is higher, the number of measurements must be larger.

The simulator returns the performance statistics at the end of simulation when the target configuration contains tomography. The output files contain the following information:

- Fidelity: Calculated by the software tomography process. This indicates the quality of Bell pairs between two end nodes.
- Bell-pair per sec: The number of Bell pairs generated in one second (in simulation time).
- Tomography time: The time until the entire tomography process finishes.
- Tomography measurements: The number of measurements of tomography.

3.5. Simulator of Quantum Network Communication

The Simulator of Quantum Network Communication (SeQUeNCe) is implemented in Python.[14] To create entanglement between two nodes, the application makes a reservation request consisting of the following six elements:

[13] R Satoh, "QuISP: A Quantum Internet Simulation Package" (2021), arXiv:2112.07093.

[14] Xiaoliang Wu, et al., "SeQUeNCe: A Customizable Discrete-Event Simulator of Quantum Networks," *Quantum Sci. Technol*. 6 045027 (2021), doi:10.1088/2058-9565/ac22f6.

- Initiator: The initiator of the entanglement connection that is sending a reservation request (classical message) toward the responder.
- Responder: The other end of the connection setup process (and future entanglement connection), where the message sent by the initiator terminates.
- Fidelity: The target fidelity of distributed entanglement.
- Memory size: The memory provided by the initiator to distribute entanglement and that is requested of the responder.
- Start time: The time from when the resources need to be available for use by the application.
- End time: The time when the resources can be released.

Parameters and values used in the simulator include:

- Memory efficiency
- Memory frequency
- Memory coherence time
- Atom-cavity cooperativity
- Memory array size
- Detector efficiency
- Detector count rate
- Detector dark count
- Detector resolution
- Attenuation
- Speed of light
- Channel Time Division Multiplexing (TDM) time frame
- Gate fidelity
- Swap success probability

3.6. The NETwork Simulator for QUantum Information with Discrete Events

The NETwork Simulator for QUantum Information with Discrete events (NetSquid) is a Python discrete event simulator.[15] The parameters and design overview of NetSquid are available online.[16]

Parameters include:

- Source total number of modes (spectral x temporal x spatial)
- Source emission probabilities
- Fiber coupling loss probability
- Fiber attenuation
- Visibility
- Detector dark count probability
- Detector detection efficiency
- Detectors number-resolving (yes/no)

[15] T Coopmans, R Knegjens, A Dahlberg, et al. "NetSquid, a NETwork Simulator for QUantum Information Using Discrete Events. Commun Phys 4, 164 (2021), doi:10.1038/s42005-021-00647-8.
[16] See Table II and Supplementary Figure 4 in Supplementary Information to NetSquid, a Network Simulator for Quantum Information using Discrete Events.

Parameters for the two different atomic-ensemble memory technologies include:

- spectral modes
- temporal modes
- spatial modes
- maximum efficiency (t = 0)
- memory lifetime
- time dependence statistical distributions

3.7. Quantum Key Distribution Network Simulation Module

QKD is a commercialized quantum network technology that has reached a relatively high level of maturity. Quantum networking can learn much from QKD successes and, perhaps more importantly, its failures. There are many QKD simulators, both free and commercial, but in this report, we focus on a free version: Quantum Key Distribution Network Simulation Module (QKDNetSim).[17] This simulator was implemented as an NS-3 simulator module that assumed users were familiar with the classical NS-3 simulation package.

QKDNetSim focuses on the classical traffic generated by a QKD system. It relies heavily on the AIT R10 QKD software.[18] The simulator appears to follow the ETSI QKD standards and is a good example of a practical, useful tool.

3.8. Quantum/Classical

In this section, the quantum/classical abstraction[19] is not a simulator per se, but rather a proposed abstraction simulated with QuNetSim, discussed earlier. Here, for example, a node can make use of the optional parameters of the proposed quantum Ethernet frame that can state a maximum storage time for the quantum payload before it should be dropped. If the payload requires a higher fidelity and the maximum duration for storage has elapsed, the payload can be dropped. Parameters are available in the literature.[20]

3.9. QuNET

QuNET is a highly abstracted quantum network simulator written in Julia (see VI B 6. Network abstraction).[21] It is open-source software for simulating entanglement networks and benchmarking entanglement routing algorithms. QuNet uses a cost vector methodology. Rather than track quantum states themselves, it tracks their associated costs as they traverse the network. Costs are arbitrary properties that accumulate additively as qubits traverse networks. QuNet can express physical degradation such as loss, dephasing, or depolarizing processes in this form, and also non-physical costs such as monetary ones. Tracking the accumulation of costs acting on Bell pairs is claimed to be equivalent to directly tracking the states themselves.

---

[17] M Mehic, O Maurhart, S Rass, et al. “Implementation of Quantum Key Distribution Network Simulation Module in the Network Simulator NS-3.” Quantum Inf Process 16, 253 (2017), doi:10.1007/s11128-017-1702-z.
[18] Austrian Institute of Technology, “AIT Optical Quantum Technologies Projects,” (2018), https://sqt.ait.ac.at/software.
[19] S DiAdamo, “Integrating Quantum Simulation for Quantum-Enhanced Classical Network Emulation” (2021), arXiv:2110.01437.
[20] See Table 1 of DiAdamo, S., Qi, B., Miller, G., Kompella, R., and Shabani, A., “Packet switching in quantum networks: A path to the quantum Internet”, Physical Review Research, vol. 4, no. 4, 2022.
[21] H Leone, NR Miller, D Singh, NK Langford, and PP Rohde, “QuNet: Cost Vector Analysis & Multi-Path Entanglement Routing in Quantum Networks” (2021), arXiv:2105.00418.

3.10. NetQASM

QASM originated as a language for formally defining a quantum circuit to render images for visualization purposes. As quantum computation evolved, the language was adopted to specify quantum circuits as input to a quantum computer. NetQASM extends this to quantum networks[22] and has similar goals to SimulaQron in terms of exploring the intersection of quantum processing and communication. It is another Python tool. Specifically, NetQASM, is a low-level instruction set architecture for quantum internet applications. It is a universal, platform independent, and extendable instruction set with support for local quantum gates, classical logic, and quantum networking operations for remote entanglement generation. It enables exploration of close integration of classical logic and communication at the application layer with quantum operations at the physical layer. Thus, its abstractions are geared toward this goal.

Parameters and explanation:

- electron_T1: T1 of communication qubit (electron)
- electron_T2: T2 of communication qubit (electron)
- electron_init: Fidelity to initialize communication qubit (electron)
- electron_rot: Fidelity for Z-rotation on communication qubit (electron)
- carbon_T1: T1 of storage qubit (carbon)
- carbon_T2: T2 of storage qubit (carbon)
- carbon_init: Fidelity to initialize storage qubit (carbon)
- carbon_z_rot: Fidelity for Z-rotation on storage qubit (carbon)
- carbon_xy_rot: Fidelity for X/Y-rotation on storage qubit (carbon)
- ec_controlled_dir_xy: Fidelity for native two-qubit gate
- prob_error_meas_0: Probability of flipped measurement outcome for |0>
- prob_error_meas_1: Probability of flipped measurement outcome for |1>
- link_fidelity: Fidelity of generated entangled pair

4. Conclusion

Table 1 summarizes the parameters derived from the abstractions discussed in this report. Source identifies the venue describing the abstraction typically either a publication or standard; abstraction is an attempt at a very concise description of the "novel" abstraction(s); the high-level components column attempts to provide more detail about significant parts of the abstraction; and the parameters are the control and observable parameters exposed by the abstraction, which is the focus.

There are a lot of exposed parameters, some of which are fundamentally quantum technology dependent (e.g., spin versus photonic). If the classical internet exposed all technology dependent parameters, its development would be stunted as well. It is noted that the parameter "quality of entanglement," which applies to any quantum device or network, may highlight fundamental physics challenges and limitations for the feasibility of second quantum revolution goals.[23]

[22] A Dahlberg, "NetQASM-a Low-Level Instruction Set Architecture for Hybrid Quantum-Classical Programs in a Quantum Internet" 7:3 (2022), doi:10.1088/2058-9565/ac753f.
[23] P Liu, Z Liu, S Chen, and X Ma, "Fundamental Limitation on the Detectability of Entanglement," (2022), arXiv:2208.02518.

Table 1: Summary of Parameters

Source	**Abstraction**	**High-level Components**	**Parameters**
Source of the abstraction	Brief description of the "novel" abstraction(s)	Information exposed by the abstraction	Control and observable parameters based upon the abstraction
"Tools for quantum network design"	• Entangled links • Noise modeling • Time-dependencies	• Entangled pairs as "links" • Entanglement swapping • Noise modeling via depolarizing, dephasing, or amplitude damping • Time-dependent memory decoherence	• Time to distribute entanglement • Quality of entanglement • Timing profile of entanglement swapping, distillation, and cutoffs • Sensitivity analysis of performance with respect to hardware parameters • Pauli error tracking parameters • RuleSet-based communication protocol parameters
"Experimental demonstration of entanglement delivery using a quantum network stack"	• Platform-independent applications • Link layer abstraction of physical-layer entanglement • Remote state preparation and a hardware abstraction layer (HAL)	• Abstraction into independent tasks and services • Network controller implements the platform-independent stack • Link layer schedules entanglement requests • More stringent timing requirements are positioned lower in the software stack.	• Link layer request - Physical layer outcome - Description • INI - SUCCESS - Qubit initialization is always successful. • MSR - SUCCESS_0 - Measurement outcome is |0>. • SUCCESS_1 - Measurement outcome is |1>. • SQG - SUCCESS - Single qubit gates are always successful. • PMG - SUCCESS - Updating the pre-measurement gate information is always successful. • ENT - SUCCESS_PSI_PLUS - Entanglement generation was successful, the state generated was $\|\Psi^+ >$. • SUCCESS_PSI_MINUS - Entanglement generation was successful, the state generated was $\|\Psi^- >$. • ENM - SUCCESS_PSI_PLUS_0 - Entanglement generation was successful, the state generated was $\|\Psi^+ >$, and the measurement outcome was $\|0 >$. • SUCCESS_PSI_PLUS_1 - Entanglement generation was successful, the state generated was $\|\Psi^+ >$, and the measurement outcome was $\|1 >$. • SUCCESS_PSI_MINUS_0 - Entanglement generation was successful, the state generated was $\|\Psi^+ >$, and the measurement outcome was $\|0 >$. • SUCCESS_PSI_MINUS_1 - Entanglement generation was successful, the state generated was $\|\Psi^+ >$, and the measurement outcome was $\|1 >$. • ENT, ENM - ENT_FAILURE - Entanglement generation was attempted and failed. • ENT_SYNC_FAILURE - Entanglement generation was not attempted because the synchronization step failed (the other node is busy).

			• Any request - HARDWARE_FAILURE - The node has experienced a hardware problem and cannot fulfill requests.
"The Virtual Quantum Optics Laboratory"	• N/A	• N/A	• N/A
IEEE P1913 YANG Model for Software-Defined Quantum Communication	• Software-Defined Networking	• Separation of classical control from quantum operation (YANG model) • Quantum Case & UML diagrams	• Quantum Key Types • Key-rate, entropy, etc. • QKD Types • Sifting error-estimation, error-reconciliation, qber, alarms, entropy, key-generation, etc. • Photon Encoding Types • Stokes-vector, poincare-sphere, jones-matrices, bloch-sphere, etc. • Quantum Technology Types • Continuous-variable, discrete-variable, integrated-quantum-optics, etc. • Coincidence Counter Types • External-internal-clock, coincidence-results, counter-operating-parameters, etc. • Quantum Source Types • Discrete-source, entangled-pair-source, continuous-source, etc. • Passive Quantum Optics Types • Beamsplitter, grating, lens, etc. • Quantum Receiver Types • Discrete-source, entangled-pair-source, continuous-source, etc. • Networked Quantum Detector Types • Augment /if:interfaces/if:interface: • Multidetector Unit Types • Detector* [id]
Quantum Internet Research Group (qirg)	• Heralded entanglement abstraction • Software-defined networking • Quantum resource discovery in the network • Lower (faster) vs higher level (slower) signaling based upon timing requirements	• Three basic schemes for heralded entanglement generation (there is no upstream or downstream end of the pair) • Control information (distinct from control plane messages) messages that operate at the granularity of individual entangled pairs as part of quantum data plane • Types of quantum network nodes • Mechanism to discover and locate quantum resources in the network	• Quantum virtual circuit quality of service (QoS) • Quantum virtual circuit capacity of end-to-end Bell pairs per second (BPPS) • Quantum virtual circuit required fidelity of the Bell pairs
Industry Specification Group (ISG) on Quantum Key Distribution (QKD)	• Secure key management abstractions	• Trusted vs quantum connections	• Name – Description • qkdi_id - Interface id. Locally unique number, which is globally unique when combined with the SD-QKD node ID • qkdi_status - Status of a QKD interface of the SD-QKD node • qkdi_capabilities - Capabilities of the QKD system (interface)

			• qkdi_capabilities/role_support - QKD node support for key relay mode services • qkdi_capabilities/ • wavelength_range - Range of supported wavelengths (nm) (multiple if it contains a tunable laser) • qkdi_capabilities/max_absorption - Maximum absorption supported (in dB) • qkdi_model - Device model (vendor/device) • qkdi_type etsi-qkdn-types - Interface type (QKD technology). • qkdi_att_point - Interface attachment point to an optical switch • qkdi_att_point/ - Unique ID of the optical switch (or passive component) to which the interface is connected
ITU-T Focus Group on Quantum Information Technology for Networks (FG QIT4N)	• Layering • Software-defined network context	• Quantum, key management, QKDN control, QKDN management and service layers	• QBER • Channel loss • Estimated secret key rate • Decoy state setting • Output raw key rate • Output secure secret key rate • Excess noise • Channel transmission • Estimated secret key rate • Shot noise variance • Sender modulation variance
Y.3800-Y.3999 Quantum key distribution networks	• Modular functional architecture diagram • AI/ML abstractions for QKD networks	• Abstraction for integration of different quantum technologies	• No explicit set of parameters
GSMA Internet Group (IG)	• Quantum services network reference framework	• Quantum services network components	• Scalability • Point-to-point and multi-point key transport • Efficiency • Performance of key supply and relay node routing solutions • Key Performance Indicators (KPI) in real-time • Capacity to keep Service level agreements (SLA) • Robustness • Device and hop-by-hop link fault detection and recovery methods • Policy control • Interfaces parameters (classical and quantum channels) • Internal parameters • QoS planning requirements • QoS monitoring requirements • QoS optimization requirements • QoS provisioning requirements • QoS protection recovery requirements • Application-oriented • API compatibility

			• Standards Compliance Certification
Quantum Internet Alliance	• Most of the QIA documents appear to be a re-packaging of U Delft papers and are covered elsewhere in this report.	• Most of the QIA documents appear to be a re-packaging of U Delft papers and are covered elsewhere in this report.	• Most of the QIA documents appear to be a re-packaging of U Delft papers and are covered elsewhere in this report.
SimulaQron	• Quantum communication channels interconnect local quantum processors. • Platform-independent portion of the quantum internet for applications	• "Stand-in" for platform dependent quantum processing systems • Classical Quantum Combiner (CQC) is an extended instruction set with special features tailored to a quantum network.	• Type - Effect • TP_NEW_OK Qubit was allocated • TP_EXPIRE Qubit has expired • TP_DONE Done with command • TP_RECV Received qubit • TP_EPR_OK Created EPR pair • TP_MEASOUT Measurement outcome • TP_INF_TIME Return timing information • ERR_GENERAL General-purpose error • ERR_NOQUBIT No more qubit • ERR_UNKNOWN Unknown qubit ID • ERR_UNAVAILABLE Cannot allocate qubit • ERR_DENIED No access to qubit • ERR_VERSION CQC version not supported • ERR_UNSUPP Sequence not supported • ERR_TIMEOUT Timeout • Command Effect • CMD_NEW Ask to allocate a new qubit • CMD_ALLOCATE Ask to allocate multiple qubits • CMD_RELEASE Release qubit to be used by other app. • CMD_RESET Reset qubit to j0i • CMD_MEASURE Measure qubit (demolition) • CMD_MEASURE_INPLACE Measure qubit (non-demolition) • CMD_SEND Send qubit to another node • CMD_RECV Ask to receive qubit • CMD_EPR Create EPR pair with another node • CMD_RECV_EPR Ask to receive half of EPR pair • CMD_SWAP Entanglement swapping • CMD_I Identity • CMD_X Pauli X • CMD_Y Pauli Y • CMD_Z Pauli Z • CMD_H Hadamard • CMD_K K gate (Z to Y) • CMD_T T gate

			• CMD_ROT_X Rotation around X (multiple of 2 Pi / 256) • CMD_ROT_Y Rotation around Y (multiple of 2 Pi / 256) • CMD_ROT_Z Rotation around Z (multiple of 2 Pi/256) • CMD_CNOT CNOT (this qubit as control) • CMD_CPHASE CPHASE (this qubit as control)
Quantum NETwork SIMulator (QuNetSim)	• Focuses solely on the network layer (lower layers are separate "backend" software) • Emphasizes synchronization logic programmatically at a high level	• Message • Qubit • Quantum Storage • Classical Storage • Quantum Connection • Classical Connection • Packet • Routing Packet	• Limitation of number of qubits stored at a host • Maximum number of entangled qubits • Maximum number of data qubits • Host methods for setting the limits: 1) set_epr_memory_limit; 2) set_data_qubit_memory_limit. • ID of the entanglement distributor • Receiver that is the holder of the other half of the EPR pair • ID for the EPR pair that will be generated • IDs resulting in easier qubit synchronization between parties
Simulator for Quantum Networks and CHannels (SQUANCH)	• Parallelized simulation of distributed quantum information processing Agents. • Channels (quantum and classical) interconnect Agents. • Error models apply to the transmitted information. • Agents maintain internal clocks. • Timing considering photon pulse widths, signal travel speeds, length of channels, etc. into account. • QSystem represents a multi-particle quantum state as a density matrix. • Gates manipulate the state of a quantum system.	• Agents • Channels (quantum and classical) • Error models • QSystem • QStreams • Qubits • Gates	• A quantum agent is represented by a 5-tuple (C; MC; MQ; G; T): • C is the configuration of channels connecting the agent to other agents. • MC is the classical memory. • MQ is the quantum memory. • G is an arbitrary but finite set of quantum operations. • A classical Turing machine T
Quantum Internet Simulator Package (QuISP)	• Error abstraction • QuISP Ruleset operation has the feel of OpenFlow flow tables.	• Only deviations from the ideal quantum state are tracked and managed. • Only Pauli X, Y, and Z errors as well as relaxation and excitation errors • Photon loss • Quantum state at any instant in time is represented by an error probability vector. • Evolution of error probability vector is given by a transition rate matrix and modeled as a Markov process.	• 50+ types of parameters that can be given through the INI file • Network. The name of the network written in the NED language, specifying the number and types of nodes and the network topology • Channel errors. The errors on a quantum channel. This specifies how much and what types of errors in a particular distance. Currently, all Pauli errors and photon loss errors are supported, and users can specify the ratio of each error.

			• Memory errors. In addition to Pauli errors, we support relaxation and excitation errors. As the simulation time passes, this error accumulates on each qubit. • Gate errors. In a realistic situation where we assume that devices are imperfect, there are always small operational errors in quantum gates. We can handle errors on basic gate sets (Pauli gates, Hadamard, Controlled-X). • Measurement errors. Quantum memories must be measured to get internal information which affects the quantum state itself. Measurement errors are composed of Pauli errors. • BSA errors. It is possible to have different errors for internal and external BSAs. Tunable parameters are photon loss, photon detection rate, and dark count probability. • Photon emission success probability. The probability that a quantum memory successfully emits a photon which is then captured by the fiber • Traffic pattern. A configurable parameter for traffic generation pattern. Currently, this simulator supports single traffic generation from one node to the other random node and multiple traffic generation from all end nodes to randomly selected nodes. • Measurement count. The number of measurements in tomography. When the required precision of tomography is higher, the number of measurements needs to be larger. • The simulator returns the performance statistics at the end of simulation when the target configuration contains tomography. The output files contain the following information: o Fidelity. Calculated by the software tomography process, this indicates the quality of Bell pairs between two end nodes. o Bell pair per sec. The number of Bell pairs generated in one second (in simulation time)

			○ Tomography time. The time until the entire tomography process finishes ○ Tomography measurements. The number of measurements of tomography
Simulator of Quantum Network Communication (SeQUeNCE)	• Focus on realism of quantum states • Quality of entanglement is a key quantum network performance metric, and loss and decoherence that affect it • Quantum networks are time-sensitive systems; the arrival times of photons determine their identity. • Bounds on certain operations • Modularized design that separates functionality into modules that contain protocols that can be reprogrammed • Arrival times of photons determine their identity.	• States can be encoded as time bins, in the polarization of light, or as states in quantum memories. • Bra–Ket and density matrix representations of quantum states are utilized. • Hardware models record entanglement fidelity. • Quantum networks are time-sensitive systems. • Lifetime of qubits in memories is limited. • Simulator operates with picosecond precision. • Modularized design that separates functionality into modules that contain protocols that can be reprogrammed	• Reservation request consisting of the following six elements: ○ Initiator: the initiator of the entanglement connection that is sending a reservation request (classical message) toward the responder ○ Responder: the other end of the connection setup process (and future entanglement connection), where the message sent by the initiator terminates ○ Fidelity: the target fidelity of distributed entanglement ○ Memory size: the memory provided by the initiator to distribute entanglement and requested of the responder ○ Start time: the time from when the resources need to be available for use by the application ○ End time: the time when the resources can be released • Memory efficiency • Memory frequency • Memory coherence time • Atom-cavity cooperativity • Memory array size • Detector efficiency • Detector count rate • Detector dark count • Detector resolution • Attenuation • Speed of light • Channel TDM time frame • Gate fidelity • Swap success probability
NETwork Simulator for QUantum Information with Discrete events (NetSquid)	• Quantum and classical component-level class abstraction	• QState, Operator • Entity • Event, EventHandler, EventExpression • Model, ErrorModel, DelayModel • Port, Component, QuantumMemory • Channel, Node, Network • Connection • ServiceProtocol, NodeProtocol, Protocol • Data-Collector	• Qubit, KetState, DMState (density matrix), StabState (stabilizer), GSLCState (graph states with local Cliffords) • Source total number of modes (spectral x temporal x spatial) • Source emission probabilities • Fiber coupling loss probability • Fiber attenuation • Visibility • Detector dark count probability • Detector detection efficiency

			• Detectors number-resolving • Memory maximum efficiency • Memory lifetime • Time dependence distribution
QKDNetSim	• Conforms to ETSI QKD standards • Focus on classical channel; minimizes details of quantum channel	• Relies heavily on AIT R10 QKD software for quantum impact on classical traffic • Quantum devices are simply treated as black-box classical key generators.	• Key identification (ID) • Key size • Key value • Key generation timestamp • Amount of pre-shared key material • The key material storage depth Mmax, used to denote the maximal number of keys that can be stored in QKD buffer • The current value Mcur(t), representing the amount of key material in QKD buffer at the time of measurement t, where it holds that Mcur(t)<=Mmax • The threshold value Mthr(t) is the state of QKD buffer where it holds that Mthr(t)<=Mmax
"Integrating Quantum Simulation for Quantum-Enhanced Classical Network Emulation"	• Abstraction strives toward tight integration of quantum and classical networking. • Hybrid classical-quantum data frames: classical header, quantum payload, classical trailer • Quantum Ethernet frame and type-length-value (TLV) structures	• Classical abstraction of a packet and packet switching can be maintained in a quantum network. • Concept of burst-switched quantum networks - sends control information ahead of the payload "just in time" • Requires precise classical/quantum transmission scheduling to avoid storing quantum states	• Quantum Data Unit comprised of • Frame Preamble • Frame Destination MAC Address • Frame Source MAC Address • Frame Ethertype • Frame Header/Trailer TLV Type = 1 • Frame Optional TLV(s) • End of QDU TLV Type = 0 • Frame Check Sequence • • TLV Type 0 End of QDU Optional • TLV Type 1 Header/Trailer Mandatory • TLV Type 2 Quantum payload duration Mandatory • TLV Type 3 Other payload information Optional • TLV Type 4 Time spent in memory Optional • TLV Type 5-126 Reserved - • TLV Type 127 Custom TLV Optional • Optional parameters of the proposed quantum Ethernet frame state a maximum storage time for the quantum payload before it should be dropped. • If the payload requires a higher fidelity and the maximum duration for storage has elapsed, the payload can be dropped.
QuNET	• Quantum cost-vector analysis to simulate and benchmark routing in	• Operates at a high level of abstraction as opposed to	• Parameters are closer to a quantum level of abstraction

	multi-user entanglement networks in a way that is highly scalable • Abstraction considers a subset of error channels that are expressible in terms of additive cost metrics.	representing details of a low-level link layer	(fidelities and operators) to track loss of fidelity in abstract manner.
NetQASM	• Universal, platform-independent and extendable instruction set with support for local quantum gates, classical logic, and quantum networking operations for remote entanglement generation • Quantum network applications to be programmed in high-level platform-independent software • Quantum network processing unit (QNPU) is assumed to reside within end-nodes in a quantum network. • Quantum network applications run on the end-nodes; communication via classical message passing and quantum entanglement is abstracted away by a network stack.	• Low-level instruction set architecture for quantum internet applications • Close integration of classical logic and communication at the application layer with quantum operations at the physical layer • End-nodes hold (1) communication qubits, which can be used to generate entanglement with remote nodes and (2) storage qubits, which can be used to store quantum states and apply operations.	• electron_T1 - T1 of communication qubit (electron) • electron_T2 - T2 of communication qubit (electron) • electron_init - Fidelity to initialize communication qubit (electron) • electron_rot - Fidelity for Z-rotation on communication qubit (electron) • carbon_T1 - T1 of storage qubit (carbon) • carbon_T2 - T2 of storage qubit (carbon) • carbon_init - Fidelity to initialize storage qubit (carbon) • carbon_z_rot - Fidelity for Z-rotation on storage qubit (carbon) • carbon_xy_rot - Fidelity for X/Y-rotation on storage qubit (carbon) • ec_controlled_dir_xy - Fidelity for native two-qubit gate • prob_error_meas_0 - Probability of flipped measurement outcome for |0> • prob_error_meas_1 - Probability of flipped measurement outcome for |1> • link_fidelity - Fidelity of generated entangled pair