Quantum Internet Research Group
Internet-Draft
Intended status: Informational
Expires: March 13, 2020

R. Van Meter
T. Matsuo
Keio University

September 10, 2019

Connection Setup in a Quantum Network
draft-van-meter-qirg-quantum-connection-setup-01

Abstract

Near-term quantum networks will grow to form a Noisy, Intermediate-Scale Quantum Internet (NISQI). Connection setup will require adapting behavior along the path to the noise levels of individual elements. In this proposal, path creation is triggered by an application at the Initiator, information is accumulated node-by-node on an outbound pass in a series of QCap (quantum capability) blocks, then the RuleSets are created at the Responder. RuleSets are installed at the individual nodes on the return pass. This document describes the architecture of connection setup in a network. Details of the RuleSets and QCaps, addressing architecture, link protocols, routing, resource allocation (multiplexing), extension of this setup procedure to an internetwork, and extension to multiparty communications are beyond the scope of this document.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on March 13, 2020.

Copyright Notice

Table of Contents

1. Introduction

Building a connection across a quantum network [theqi] is a classical task. Because of the low success probability of quantum

communication due to photon loss and the extremely high error rates due to the fragile nature of quantum information, quantum communication between two nodes more closely resembles a coordinated computation distributed among the set of nodes forming the path

between the two nodes than a store-and-forward network session [qnetworking].

Use of the quantum network is driven by applications running at two (or more) classical nodes. Overall behavior is similar to client-server computing. The connection is initiated from a node similar to client and responded to by a node similar to a server. The details of the sending and receiving of the classical messages are not specified in this document, but can be modeled as if being sent over a TCP socket. Messages are assumed to be reliable and delivered in order. These messages have no hard real time requirement, though the subsequent data phase of the operation may.

This connection setup process must collect information about the hardware (channels and buffer memories) to be used, because of the heterogeneity of the underlying hardware. Loss in optical channels naturally varies with channel length and other factors, and has a large impact on quantum communication performance. Individual quantum buffers holding quantum bits (qubits) will vary in quality, as well.

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

2. Concepts and Glossary

The following terms will be used:

Bell pair a common form of entangled quantum state useful in communications.

End node a quantum network node with a single interface. End nodes may have stationary quantum memories, or may be capable only

of measuring photons; this distinction is beyond the scope of this document.

Entanglement the condition of a group of qubits (typically two qubits in this document) in a shared state that cannot be described using only real, non-negative, classical probabilities.

Entanglement Swapping executed at node B splices an entangled state shared with node A to an entangled state shared with node C, creating A-C entanglement and disentangling B from both nodes.

Fidelity a measure of the quality of a quantum state; roughly, the probability that the system holds the desired state.

Initiator the initiator of the classical process of establishing the connection by sending a message toward the Responder.

Purification an error detection mechanism on quantum states. Typically, one quantum state is used to test the condition of a second state; the first state is destroyed in the process. If the purification fails, it is unknown whether the first or second state was in error, and the second state is discarded as well. If purification succeeds, our confidence in the state is improved.

QCap an information block describing the quantum capabilities of a particular node and link.

Qubit a quantum system with two states that can be stored in memory or transmitted through a channel, manipulated in a constrained set of operations, entangled with other qubits, and measured.

Repeater a quantum network node with two interfaces, typically sitting in the middle of a chain. Repeaters do not require routing functionality, but otherwise have the same capabilities as routers. As spacing between nodes may be required to be as short as ten kilometers, depending on technology, what would be single fiber hops in a classical network will be a long chain of repeaters.

Responder is the classical endpoint of the connection setup process, where the message sent by the Initiator terminates. The Responder creates the RuleSets for all nodes in the path, and commonly will be the smarter node.

Router a quantum network node with a more than two interfaces, requiring routing capability.

RuleSet describes the actions that a nodes should take when certain conditions occur. The contents of RuleSets are beyond the scope of this document.

The terms "source" and "destination" are not appropriate at the connection level in a quantum network, because distributed quantum states are not necessarily used for the unidirectional transfer of information. Therefore, we use Initiator and Responder to designate roles in the connection setup process, but those roles do not not

necessarily correspond to any asymmetry during the connection lifetime. Source and destination are not appropriate because:

1. There may not even be data transferred between nodes; the entanglement might be used for some shared operation that doesn't involve qubits moving back and forth via teleportation. Quantum key distribution (QKD) is an obvious example, where both ends measure the entangled state and destroy it in order to get a classical bit.

2. Temporally, operations may not even happen left-to-right along the chain of repeaters, again violating the notion that data is moving.

"Source" and "destination" may be used to describe the movement of an individual classical message.

Links are assumed to be point-to-point. Multidrop physical layers are possible, but quantum broadcast or multicast are not directly possible at the physical level, and would have to be emulated.

3. Connection Setup Phases

3.1. Short Description of Phases

The single-network, two-node connection setup procedure consists of three basic phases:

1. The outbound request is routed from Initiator to Responder using a standard NextHop-based forwarding table, accumulating information about the path along the way in a stack of QCaps.

2. When the request arrives at the Responder, the Responder uses that information to create a complete RuleSet for every node. The RuleSets are assembled into a stack with the nearest node at the top.

3. The RuleSets are sent back along the original path, with each node removing its RuleSet from the message (popping the stack), then forwarding the remaining QCaps on until it returns to the Initiator.

3.2. Rationale for this Architecture

The outbound pass collects information about the nodes and links, to be used by the Responder to formulate the RuleSets. Why is the information collected in this fashion rather than shared more broadly across the network, e.g. as part of a modified routing protocol such

as OSPF [RFC2328]? Why does a single node create the RuleSets for all nodes, rather than allowing individual nodes to create their own RuleSets when they see the PathSetupRequest message?

1. Because Repeaters may be spaced as closely as every 10km, a full topology for a network listing every Repeater may be excessively large for routing purposes, but such information is needed for building RuleSets.

2. The information collected may be substantially larger in volume than simple link costs.

3. The information collected and used may be too dynamic for a routing protocol.

4. Sharing of this information can be unnecessary when routing is driven by policy decisions rather than technical capabilities.

5. Centralization of the RuleSet creation is necessary because all RuleSets must cooperate toward a single goal, and the correct breakdown of responsibility cannot be determined from partial information.

6. Centralization of RuleSet creation allows a Responder to upgrade its policies independently and to improve the process if its developers have found better tuning mechanisms. A distributed mechanism would require that all nodes in the path upgrade at the same time to avoid the creation of inconsistent policies, and limit the ability of Responders (often service providers of some sort) to innovate.

4. Message Contents and Elements

This section outlines the principal information to be carried in the messages. Detailed packet formats are beyond the scope of this document, and may vary from network to network.

4.1. PathSetupRequest

At minimum, the PathSetupRequest message must contain:

1. node addresses for the Initiator and Responder

2. the class of service requested [qiroadmap]

3. minimum performance parameters (fidelity and throughput)

4.2. Quantum Capabilities (QCap)

A QCap (quantum capabilities) block to be added to the stack in the PathSetupRequest message describes the functions, performance and quality of the node and link. This may include:

1. the fidelity of Bell pairs created by the quantum channel

2. the fidelity of local operations performed by the node for purification or entanglement swapping

3. the rate at which entanglement can be created (Bell pairs per second)

The details of the required information may differ between networks. A standardized form of this information for sharing between networks will be used for internetworking operation.

4.3. RuleSets

A RuleSet block in the stack in the PathSetupResponse message describes the rules to be executed at each node. A rule consists of a Condition clause and an Action clause. A Condition clause lists the existence of particular entangled states, or the reception of particular messages. The Action clause describes the actions of purification, entanglement swapping, or even discarding an entangled state, as appropriate. The details are beyond the scope of this document.

In order to implement multiplexing schemes (e.g. buffer-space multiplexing, time-division multiplexing, or statistical multiplexing) based on the RuleSet-based network architecture, a RuleSet may include descriptors that define the usable resources for each link involved in that specific connection.

If a link carries only a single connection, all resources available may be fully assigned to that single connection to maximize the throughput. However, a link may receive a second RuleSet generated for a new connection. In that case, the nodes must be able to correctly update and reassign the available resources. Further details of the resource reservation and reclamation process are beyond the scope of this document.

5. Processing the SetupRequest

5.1. Initiating a Connection Setup Request

An Initiator, driven by an application request for quantum network services between itself and the Responder, builds the PathSetupRequest, populates the first QCap block, selects the next hop, and sends the request. Note that there is no need for either the Initiator or the Responder to know the entire network topology, only be able to select a next hop appropriately. The details of the routing are beyond the scope of this document.

5.2. Outbound Processing

Creation of the RuleSets requires knowledge of the number of nodes involved. A quantum node adds its own address when receiving the request packet, before sending to the next node. The stack size indicates how many nodes are involved. Additionally, the RuleSet creator may require information regarding links between nodes along the path - e.g. to be used when optimizing the order of entanglement swapping.

The pseudocode below outlines the processing on receipt of the PathSetupRequest message.

```
   procedure ProcessFlatPathSetupRequest(Msg)
       Msg.HopStack.Push(MyHopInfo)
       if (MyAddr != Msg.ConnSpec.Responder)
           // Process and forward
           NextQuantumHop = GetNextQuantumHop(Msg.ConnSpec.Responder)
           LinkInfo = GetLinkInfo(NextQuantumHop)
           Msg.HopStack.Push(LinkInfo)
           Forward(NextQuantumHop,Msg)
       else
           // have reached the far end, need to build RuleSets
           // for everybody, then return
           ReturnMsg = ProcessFlatPath(Msg)
           MyRuleSet = ReturnMsg.RuleSetStack.Pop()
           InstallRuleSet(MyRuleSet)
           NextQuantumHop = ReturnMsg.RuleSetStack.Top.Addr
           Forward(NextQuantumHop,Msg)
       endif
   endprocedure
```

Note that although we use the term "NextQuantumHop" here, that refers to a neighboring quantum node, and does not imply that the classical node's neighbor is necessarily the same; it could, in theory, pass through multiple nodes to get there.

5.3. Responder Processing

The Responder accepts the final PathSetupRequest message with the complete stack of information about node capabilities and links, and builds a corresponding stack of RuleSets, one per node in the path. The Responder's processing is outlined in the "then" clause of the pseudocode above. The details of this creation process are beyond the scope of this document, and may be kept secret from other nodes in the path.

5.4. Return Processing

The pseudocode below outlines the processing on receipt of the PathSetupReturn message.

```
    procedure ProcessFlatPathSetupReturn(Msg)
        MyRuleSet = ReturnMsg.RuleSetStack.Pop()
        InstallRuleSet(MyRuleSet)
        If (ReturnMsg.RuleSetStack.Size != 0)
            NextQuantumHop = ReturnMsg.RuleSetStack.Top.Addr
            Forward(NextQuantumHop,Msg)
        endif
    endprocedure
```

The RuleSetStack should only be empty after the Initiator node of the original request removes its RuleSet, so this should be followed by activating the connection.

6. Rejection and Robustness of the Setup Process

6.1. Rejection by a Repeater or Router

A repeater or router that receives a PathSetupRequest may reject the request if it has no quantum communication resources available. It should not reject the request simply because it believes the requirements of the request (fidelity or rate) to be difficult to fulfill; that responsibility lies with the Responder.

When a node rejects the PathSetupRequest, it shall inform the other nodes along the portion of the path that have already received the PathSetupRequest by creating a PathSetupResponse message with an error code that indicates failure and sending that message to the node on the top of the stack. As with a successful PathSetupResponse, the list of nodes to which the message must be sent is created as a stack. Other than the addresses and the error

code, the message may be empty; no RuleSets are required. The

message is then iteratively returned, with each node popping its own address and forwarding to the next.

6.2. Rejection by a Responder

A Responder may reject a PathSetupRequest for any reason:

1. As with any classical system, it may simply choose to reject the request for any service-related reason, such as security, licensing, etc.

2. It may determine that the request cannot be fulfilled with the resources offered by nodes in the path.

When a node rejects the PathSetupRequest, it shall inform the other nodes along the path by creating a PathSetupReturn message with an error code that indicates failure and sending that message to the node on the top of the stack. As with a successful PathSetupResponse, the list of nodes to which the message must be sent is created as a stack. Other than the addresses and the error code, the message may be empty; no RuleSets are required. The message is then iteratively returned, with each node popping its own address and forwarding to the next.

6.3. Robustness

As the rate of connection initiation increases, competition for resources will also increase. A soft reservation mechanism that temporarily allocates resources in the anticipation of reception of a RuleSet may be used, with the reservation timing out and resources being released if no RuleSet arrives within a certain period. Specification of this mechanism is beyond the scope of this document.

Deeper integration of routing with real-time availability of resources is beyond the scope of this document.

7. Contributors

Besides the authors, Luciano Aparicio, Clement Durand, Dominic

Horsman, Shota Nagayama, Takahiko Satoh, Shigeya Suzuki, Amin Taherkhani, and Joe Touch have made substantial contributions to the network architecture and the concepts described here.

We also thank Chia-Hung Chien, Kaori Ishizaki, Bill Munro, Kae Nemoto, Takafumi Oka, Shinnosuke Ozawa, and Thaddeus Ladd.

Comments by Wojciech Kozlowski, Gyananjay Rai and Patrick Gelard are reflected in this draft.

8. IANA Considerations

This memo includes no request to IANA.

9. Security Considerations

Security implications of this entire process are extensive.

To minimize the probability of tampering, each information block added to the request on the outbound leg should be signed by the node adding the block.

Each information block describes hardware configuration, and therefore inherently leaks information about the network topology and condition. This document addresses only connection setup within a single network. Internetwork connection setup will require mechanisms to limit the leaking of sensitive network information across organizational boundaries.

Likewise, each RuleSet should be signed to prevent tampering during the PathSetupResponse phase.

Both the Request and Response phase may be encrypted using appropriate public key mechanisms.

It is also known that quantum networks may be vulnerable to attacks not possible in classical networks. These concerns are beyond the scope of this document.

10. References

10.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

10.2. Informative References

[qiroadmap] Wehner, S., Elkouss, D., and R. Hanson, "Quantum internet: A vision for the road ahead", Science 362, 2018.

[qnetworking] Van Meter, R., "Quantum Networking", Wiley-iSTE , 2014.

[RFC2328] Moy, J., "OSPF Version 2", STD 54, RFC 2328, DOI 10.17487/RFC2328, April 1998, <https://www.rfc-editor.org/info/rfc2328>.

[theqi] Kimble, J., "The Quantum Internet", Nature 453, 1023-1030, 2008.

Authors' Addresses

Rodney Van Meter
Keio University
5322 Endo
Fujisawa, Kanagawa 252-0882
JP

Phone: +81-46-649-3529
Email: rdv@sfc.wide.ad.jp

Takaaki Matsuo
Keio University
5322 Endo
Fujisawa, Kanagawa 252-0882
JP

Phone: +81-46-649-3529
Email: kaaki@sfc.wide.ad.jp