Quantum Networks

A Mini-Project Summary

Pranav K Nayak - ES20BTECH11035

What are Quantum Networks

- Classical information is encoded in bits.
- Classical networks allow for computers to communicate by sending bits between each other
- Quantum information is encoded in the quantum analog of a bit, the *qubit*.
- Quantum Networks perform the same role of their classical counterparts.

Challenges

- Classical networking comes with its own collection of challenges:
  - Performance of the network
  - Delays
  - Security
  - Fidelity
  - and others
- The challenges of a quantum network are two-fold:
  - Those that are the result of an underlying classical architecture
  - Those that are the result of the probabilistic nature of quantum information

Challenges

Some concrete examples of quantum networking challenges are:

- Establishing connections between nodes
- Path selection
- Resource allocation
- Request scheduling

Why Quantum Networks

- The nature of quantum information can change the intractability of certain classical problems
- For example, the factoring of numbers on a quantum computer has been shown by Peter Shor (1994) to be possible in time polynomial in the number of bits.
- The best classical analog to this algorithm is superpolynomial in the number of bits.
- The delicate nature of quantum information can also be used to detect eavesdropping.
- Quantum networks open up new cryptographical possibilities.

Request Scheduling in Quantum Networks (Cicconetti et al.)

- “Request Scheduling in Quantum Networks” is a paper written by Claudio Cicconetti, Marco Conti, and Andrea Passarella
- The paper was published in IEEE Transactions on Quantum Engineering.
- The paper puts forward a framework for understanding the problem of establishing end-to-end entanglement of qubits in the form of two sub-problems:
    - Path Selection: The problem of determining the best set of repeaters to use for an end-to-end establishment of entanglement, assuming all local entanglements are successful.
    - Request Scheduling: The problem of determining which pair of end nodes are serviced in the current time slot, given a collection of nodes that are requesting to be serviced.

Preliminaries:

Entanglement:

- Qualitatively, entanglement can be understood as the probabilities of the entangled qubits not being independent.
- Measurement will lead to a set of states whose probabilities are not the products of the individual states of the qubits.
- Importantly, the qubits can be (and in application, often are) separated in space.

Preliminaries:

The Bell Pair:

- Also called the EPR Pair, the most common example of entanglement.
- There are four forms for the Bell Pair.
- All four share the property that measuring one qubit will allow us to predict, with certainty, the result of measuring the other.
- Only two possible states can exist for each Bell Pair.
- Applying local operations on one qubit of a pair will not affect the entanglement, it will simply change the form of the Bell Pair.

Preliminaries:

The no-cloning theorem:

- Quantum data cannot be backed up or copied.
- A qubit whose wavefunction has not collapsed due to measurement, cannot be replicated by any means.
- Copying a qubit that has been measured is not really copying quantum information, but rather the classical result of a measurement.

Preliminaries:

Gates:

- The NOT gate swaps the roles of the $|0\rangle$ and the $|1\rangle$states.
- The Z gate leaves the $|0\rangle$ unchanged and changes the sign for $|1\rangle$.
- The Hadamard gate changes the state to halfway between it and its complement.
- The CNOT gate acts on two qubits, NOT-ing the second in response to a signal from the first.

Preliminaries:

Teleportation: A Quick Outline

- Alice wishes to send the state of her qubit to Bob, with whom she shares one half of an EPR pair.
- Alice passes her two qubits through a CNOT gate, using the target qubit as the control qubit.
- She then passes the target qubit through a Hadamard gate, following which she performs a measurement on her two qubits.
- Because her initial two qubits are unentangled, the measurement results in one of four possible states.

Preliminaries:

Teleportation: A Quick Summary

- Alice sends the results of her measurement to Bob through a classical channel, since the result can be encoded as two bits.
- Depending on the result, Bob can perform a sequence of zero or more X- and Z-gates on his qubit (the one belonging to the entangled pair).
- These operations will result in Bob’s qubit ending up in the same state that Alice’s was in at the start.

Preliminaries:

Entanglement Swapping:

- In order to establish teleportation between remote nodes, an entangled pair needs to be shared between them.
- The establishment of this end-to-end entanglement can be brought about through the use of *repeaters*, intermediate nodes that teleport one qubit of the entangled pair from one end node to the other.
- Each time the entangled qubit is teleported from one repeater to the next, the initial EPR pair shared between the repeaters is used up, and must be replenished.

Preliminaries:

Entanglement Swapping:

- The precise source and direction of the movement of the final EPR pair used for establishing end-to-end entanglement is not necessarily fixed.
- For example, the EPR pair can be generated at some intermediate repeater and each qubit of the pair could be teleported across repeaters until they end up at a single end node each.
- The entanglement swap primitive using repeaters is the predominant one that research is being conducted on, and is the one assumed in the paper.

Assumptions

- The paper assumes that, given a network of nodes, not all local entanglements succeed, but that some network controller element can aggregate the information of successful entanglements across the network.
- This controller then accepts requests from an upper layer in the protocol and satisfies them, within a certain time interval.
- After such a time interval, another round of local link entanglements is attempted, and the queue of pending requests is updated and serviced.

The State of the Art

- Research is currently being pursued to address multiple sub-problems in the design of quantum networks:
  - The definition of protocol stacks for a quantum internet
  - Studies on tools for evaluating the performance of quantum networks
  - The identification of capacity regions (?)
  - Multipartite entanglement distribution
  - The design of quantum networking applications
- The problem of path selection is important to the solutions presented later in the paper.

Path Selection

- The problem of deciding along which repeaters to establish end-to-end entanglement between nodes, when there are multiple alternatives for such a set.
- A central facet of this problem is that of choosing a *routing metric*.
- The routing metric problem cannot immediately borrow its solution from classical networks, since classical solutions fail to account for the nature of quantum information.
- Analyses show that Dijkstra's SPF algorithm achieves reasonable performance goals, and that a suitable physical metric is the throughput of a single link, in EPR pairs generated per unit time.

Multipath Selection:

- Most prior research assumes that at any given point in time, the network can only accommodate a single end-to-end entangled pair, and for another node's request to be serviced, the current connection must end.
- The model proposed in the paper supports the establishment of multiple end-to-end entanglements concurrently.
- Research that has explored the problem of multipath selection has not explored the problem of request scheduling, making this paper unique.

The System Model

The Network Architecture:

The network consists broadly of the following two components

- Quantum Computers: The systems that send the requests for end-to-end establishment of entanglement.
- Quantum Repeaters: Devices that enable the entanglement to occur beyond the maximum feasible distance possible between two immediate nodes by “passing on” the entanglement to other repeaters in the network.

Prerequisites for Entanglement

For end-to-end entanglement to be established, there must be some path σ that satisfies the following:

- Local entanglement between all pairs of adjacent repeaters along the path is successful
- The intermediate nodes successfully teleport the required half of the EPR pair along the path upto the end node

The Quantum Network Controller

The controller is responsible for the following:

- Maintaining a record of the requests from the end nodes for end-to-end entanglement.
- Maintaining a record of the results of local link entanglements across the network.
- Choosing which requests to serve (*scheduling*), and selecting the best path for each such request.
- Notifying the repeaters of the direction in which they are supposed to send the results of their measurement.

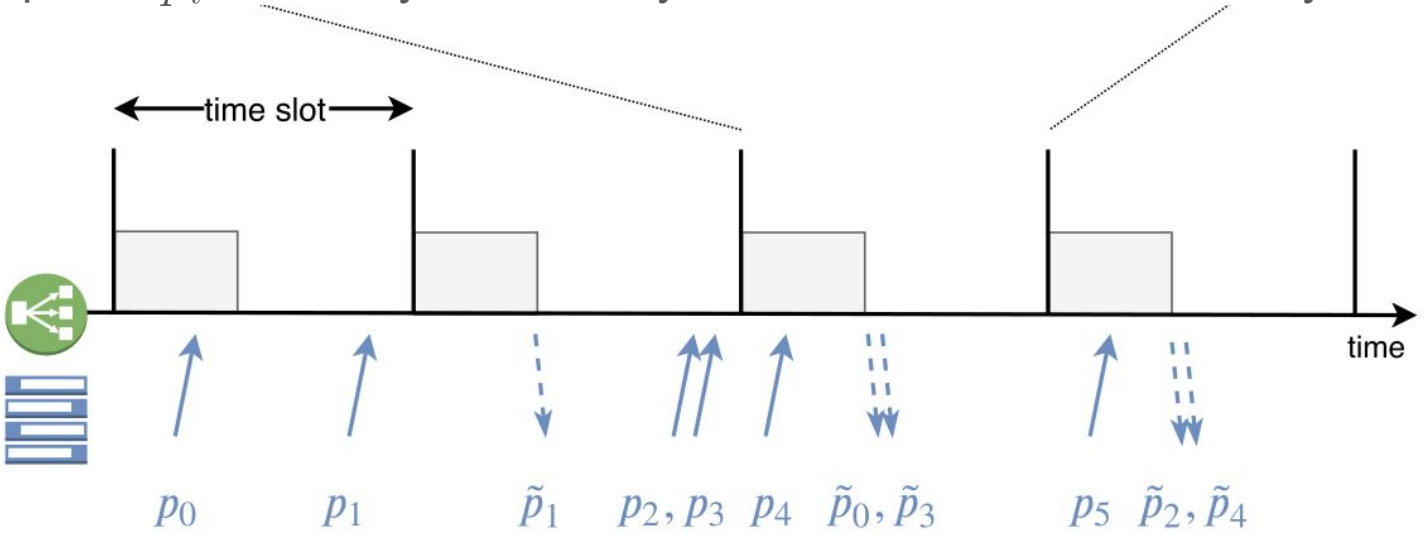
Framing Structure

The time-slotted nature of the problem can be described as follows:

- Time is divided into slots, with each slot consisting of two broad phases.
- The first phase consists of establishing entanglement between end nodes.
- The second phase consists of the operations of those end nodes using the entanglement established
- The paper focuses its attention entirely on the first phase, but notes that the deadline for establishing entanglement is technically until the end of the slot, and can thus spill over into what would otherwise be the second phase.

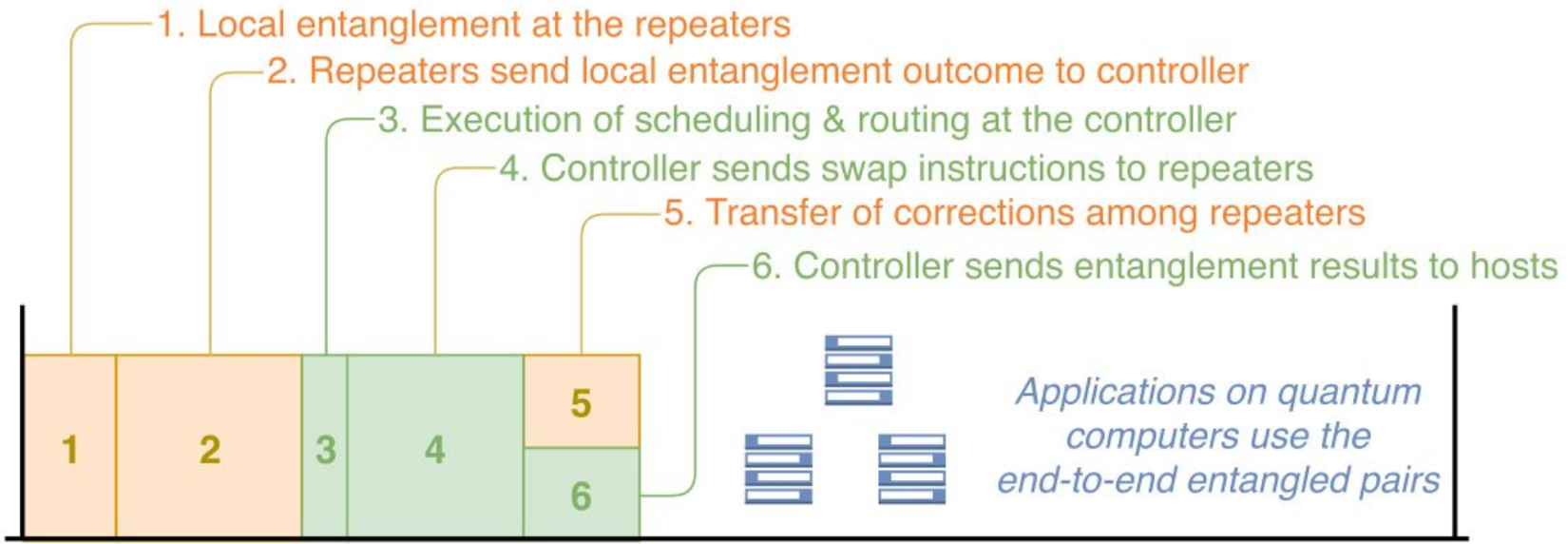
Framing Structure

The requests p_i arrive asynchronously and must be accounted for by the controller



Subphases

The first phase can be divided into six subphases, which help break down the operation of the heuristic proposed:



Subphases

- At the beginning of the time slot, local entanglement is attempted between *every* pair of nodes sharing a link in the network.
- The local entanglement outcomes are communicated to the controller through classical means.
- Based on its knowledge of the current requests and the state of local entanglements, the controller performs *routing*:
  - Routing involves deciding, for each node on a path, which of its qubits it must perform the teleportation on, i.e. in which direction it is to teleport the required half of the EPR pair.
- It then communicates this decision to the nodes of the network, telling them which of their neighbors they must perform the entanglement swap with so that end-to-end entanglement is achieved.

Subphases

- Classical information is then sent across the nodes of the network to the end node(s) so that the corrections can be performed to solidify the entanglement.
- In the final subphase, the end node performs the corrections on its half of the Bell Pair with its nearest neighbor to transform it into the half of the Bell Pair that is entangled with the other end node.
- Finally, the results of the entanglement are sent to the hosts, signalling that the pair is ready to be used by them for the remainder of the time slot.

More Assumptions

Before moving on to the problem of routing (which is really a combination of scheduling and path selection), the paper makes the following assumptions:

- If a link level Bell Pair is not used within a time slot, it is not carried forward into the next time slot. At the start of every slot, fresh entanglement is attempted at *every* link in the network, regardless of if it previously contained an unused Bell Pair or not.
- The requests have no relative priority accompanying them. They are treated as equal, and all considerations of their scheduling are done assuming they have equal priority.

The Routing Problem

The input for the problem can be formally defined as:

- The logical topology of the network as a graph *(V, E)*
- The set of local entanglements that have succeeded *S*, from which a *reduced subgraph (V', E')* is constructed to contain only those edges whose corresponding links have successful entanglement.
- The list of pairs of nodes for which entanglement is to be established. This contains both requests that arrived in previous slots but were not serviced, as well as new requests that arrived in the current slot

Note that the problem is local to each time slot, and all analysis must be done again once a new time slot begins.

The Routing Problem

The output of the routing problem is a set of end-to-end entanglement paths. Each path consists of

- The list of nodes along the path, and the positions of the memory slots on each node on which measurement is to be performed.
- The positions of memory slots on the two end nodes.
- A path identifier to distinguish between paths that are proposed within a slot.
- An identifier of which end node to send the classical information to.

The Routing Problem

The paper assumes that a protocol to encode the messages sent between the controller and the intermediate nodes, as well as between the end nodes and the upper layer, exists already.

Such a protocol would have to handle *all* communication of instructions between the network components, and it would have to send these instructions through classical channels.

The Routing Problem

The controller's job is to offer the best possible solution given a set of requests. However, what is best within a single time slot need not be what is best across all time slots. An ideal controller would account for what happened in previous slots as well as what could happen in future slots when formulating its solution.

In response to this notion, the paper proposes an *idealized routing problem,* one whose solution accounts for the arrival of requests across *all* time slots. The problem is formulated mathematically, as a *multicommodity flow problem* spanning a time horizon from $t = 0$ to $t = T$.

The (Idealized) Routing Problem

Notation:

- $p_i = (s_i, d_i, a_i)$ represents a request, with s_i being the start node, d_i being the end node, and a_i being the time slot in which the request arrives.
- $f_{i,t}(u, v)$ is a binary variable, whose value is 1 if and only if the request i is routed through the edge (u, v) in the reduced subgraph at time t.
- τ Is the set of all time slots $\{0, \ldots, T\}$ and $\mathcal{P}$ is the set of all requests $\{0, \ldots, P\}$

The Objective

$$\max \sum_{i=1}^{P} \sum_{t=1}^{T} \sum_{w \in V'(t)} f_{i,t}(s_i, w)$$

Breaking down the objective from inside to out:

- The inner sum will be 1 if the request i has been serviced in time slot t.

The Objective

$$\max \sum_{i=1}^{P} \sum_{t=1}^{T} \sum_{w \in V'(t)} f_{i,t}(s_i, w)$$

- The second sum will be 1 if the request i has been serviced throughout the time interval from $t = 1$ to $t = T$.
- The outer sum gives us the total number of requests serviced across all time slots.
- The objective is thus to simply maximize the number of requests across the time horizon.

The Constraints

$$\forall t \in \mathcal{T}\, \forall u, v \in E'(t) : \sum_{i=1}^{P} f_{i,t}(u, v) \leq 1$$

This constraint guarantees that no edge is used to service more than one request in any given time slot.

The Constraints

$$\forall i \in \mathcal{P} : \sum_{t=0}^{a_i} f_{u,t}(u, v) = 0$$

This constraint makes sure that no request can be serviced before it arrives.

The Constraints

$$\forall i \in \mathcal{P} : \sum_{t=a_i+1}^{T} f_{i,t}(u, v) \leq 1$$

This constraint guarantees that every pair, once it arrives, can only be serviced up to once.

The Constraints

$$\forall i \in \mathcal{P}\, \forall t \in \mathcal{T}: \quad \sum_{w \in V'(t)} f_{i,t}(u, w) - \sum_{w \in V'(t)} f_{i,t}(w, u) = 0, u \neq s_i, d_i$$

This constraint guarantees that for every edge that does not connect to a start or destination node, the number of flows remains balanced.

The Constraints

$$\forall i \in \mathcal{P}\, \forall t \in \mathcal{T}: \quad \sum_{w \in V'(t)} f_{i,t}(s_i, w) - \sum_{w \in V'(t)} f_{i,t}(w, s_i) \leq 1$$

$$\sum_{w \in V'(t)} f_{i,t}(w, d_i) - \sum_{w \in V'(t)} f_{i,t}(d_i, w) \leq 1$$

The above are basic network flow constraints that make sure that every source node has up to 1 flow outwards and every destination node as up to 1 flow inwards.

The (Idealized) Routing Problem

The problem as described is *ideal* for the following reasons:

- For real applications, the time horizon is not necessarily finite.
- Solving the Integer Programming problem necessitates knowledge of the arrival of requests ahead of time, specifically at the *start* of the time horizon, as well as knowledge about the state of the subgraphs at the start of each time slot.
- Even if we account for how the problem would be solved in practice, i.e. the IP is solved only for a single time slot, at the start of every time slot, the IP is NP-complete, and is difficult to solve exactly for even small instances.

An Alternative Route

- Solving the problem as soon as possible is extremely important, since in the time the controller takes to solve the problem, the qubits experience decoherence in the quantum memories.
- This leads to a reduction in the fidelity of the entanglement.
- The computation time of the above IP coupled with the need for a fast solution leads the paper’s authors to propose an alternative formulation for the problem.

Two Parts

The authors propose breaking up the routing problem into two subproblems:

- *Scheduling*, the problem of selecting which of the requests are serviced in the current time slot.
- *Path Selection*, the problem of finding the best path for a given request, in the subgraph of the current time slot.

The paper asserts that decoupling the two problems allows for a clean design and efficient implementation, letting each problem be solved with its most suitable tool.

Two Parts

The authors point out how the decoupling is really artificial, since scheduling and path selection can affect each other, with the following two examples:

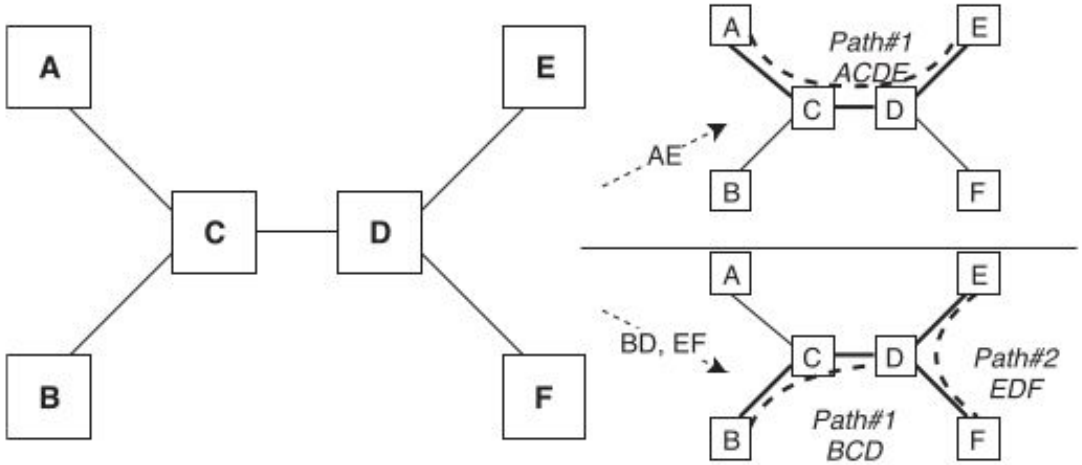


FIG. 5. Example: how scheduling affects path selection.

Two Parts

The authors point out how the decoupling is really artificial, since scheduling and path selection can affect each other, with the following two examples:

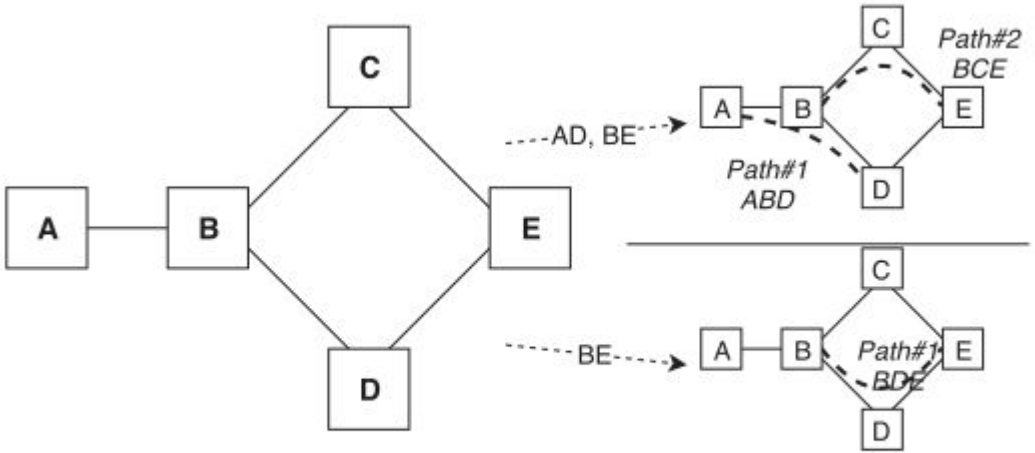


FIG. 6. Example: how path selection affects scheduling.

The Scheduling Problem

Algorithm 1: Iterative Scheduling Algorithm.

(V, E') is the reduced subgraph after the links where local entanglement has failed have been removed
$\mathcal{P}$ is the set of pending pairs sorted on chronological order (older first); ties are broken arbitrarily
$\mathcal{S} \leftarrow \emptyset, \mathcal{C} \leftarrow \emptyset$
for $p \in \mathcal{P}$ **do**
  $\sigma = path\_selection(V, E')$
  if $\sigma = \emptyset \vee skip(\sigma)$ **then**
    $\mathcal{S} \leftarrow \mathcal{S} \cup p$
  else
    $E' \leftarrow E' \setminus \{e \in \sigma\}$
    $\mathcal{C} \leftarrow \mathcal{C} \cup \sigma$
    $\mathcal{P} \leftarrow \mathcal{P} \setminus p$
  end
end
for $p \in \mathcal{S}$ **do**
  $\sigma = path\_selection(V, E')$
  if $\sigma \neq \emptyset$ **then**
    $E' \leftarrow E' \setminus \{e \in \sigma\}$
    $\mathcal{C} \leftarrow \mathcal{C} \cup \sigma$
    $\mathcal{P} \leftarrow \mathcal{P} \setminus p$
  end
end
return $\mathcal{C}$

The algorithm proposed performs two passes on the list of pending requests, sorted in FIFO order. In the first pass, for every pair *p*, path selection is performed. If a valid path σ is found, one of two decisions is taken.

The Scheduling Problem

Algorithm 1: Iterative Scheduling Algorithm.

(V, E') is the reduced subgraph after the links where local entanglement has failed have been removed
$\mathcal{P}$ is the set of pending pairs sorted on chronological order (older first); ties are broken arbitrarily
$\mathcal{S} \leftarrow \emptyset, \mathcal{C} \leftarrow \emptyset$
for $p \in \mathcal{P}$ **do**
  $\sigma = path\_selection(V, E')$
  if $\sigma = \emptyset \vee skip(\sigma)$ **then**
    $\mathcal{S} \leftarrow \mathcal{S} \cup p$
  else
    $E' \leftarrow E' \setminus \{e \in \sigma\}$
    $\mathcal{C} \leftarrow \mathcal{C} \cup \sigma$
    $\mathcal{P} \leftarrow \mathcal{P} \setminus p$
  end
end
for $p \in \mathcal{S}$ **do**
  $\sigma = path\_selection(V, E')$
  if $\sigma \neq \emptyset$ **then**
    $E' \leftarrow E' \setminus \{e \in \sigma\}$
    $\mathcal{C} \leftarrow \mathcal{C} \cup \sigma$
    $\mathcal{P} \leftarrow \mathcal{P} \setminus p$
  end
end
return $\mathcal{C}$

The path is evaluated for whether it should be “skipped” or not.

- If it is skipped, then the request is placed into a list of skipped pairs, sorted FIFO.
- If it is not skipped, the controller communicates with all the nodes on the path to establish the entanglement. The pair *p* is removed from the list, and the subgraph is pruned of all edges belonging to σ.

The Scheduling Problem

Algorithm 1: Iterative Scheduling Algorithm.

(V, E') is the reduced subgraph after the links where local entanglement has failed have been removed
$\mathcal{P}$ is the set of pending pairs sorted on chronological order (older first); ties are broken arbitrarily
$\mathcal{S} \leftarrow \emptyset, \mathcal{C} \leftarrow \emptyset$
for $p \in \mathcal{P}$ **do**
  $\sigma = path\_selection(V, E')$
  if $\sigma = \emptyset \vee skip(\sigma)$ **then**
    $\mathcal{S} \leftarrow \mathcal{S} \cup p$
  else
    $E' \leftarrow E' \setminus \{e \in \sigma\}$
    $\mathcal{C} \leftarrow \mathcal{C} \cup \sigma$
    $\mathcal{P} \leftarrow \mathcal{P} \setminus p$
  end
end
for $p \in \mathcal{S}$ **do**
  $\sigma = path\_selection(V, E')$
  if $\sigma \neq \emptyset$ **then**
    $E' \leftarrow E' \setminus \{e \in \sigma\}$
    $\mathcal{C} \leftarrow \mathcal{C} \cup \sigma$
    $\mathcal{P} \leftarrow \mathcal{P} \setminus p$
  end
end
return $\mathcal{C}$

After one pass has been made through the list, a second pass is made on the list of pairs that have been skipped, and if a valid path is found for them, then the pairs are serviced.

In this pass, no skipping is done.

The Scheduling Problem

Properties of this approach are:

- It favors older pairs, since there is no risk of starvation. The chances of a pair being served do not decrease over time.
- It is efficient, since each pair is evaluated at most twice.
- It allows for deferred path selection, since paths that are not deemed efficient can be skipped and the pair can be serviced later.
- If, however, an efficient / valid path does open up, the pairs skipped will be serviced until there are no more resources available to service requests.

Skipping Policies

The authors provide three skipping policies as examples:

- Strict FIFO: Never skip paths
- Best FIFO: Skip a path if its cost, in terms of the same metric used by path selection, is greater than the long-term average cost.
- Random FIFO: Skip a path with a probability P_{skip}, where

$$P_{\text{skip}} = \max\left\{0, 1 - \frac{E[c]}{c(\sigma)}\right\}$$

Skipping Policies

Strict FIFO: Never skip paths

- This policy ensures fairness.
- All requests are treated the same, and are served on a first-come first-serve basis.
- Only a single pass is required.
- Fairness comes at the cost of efficiency, since an extremely inefficient path, which would otherwise have been skipped by the other two policies, will consume resources being serviced before a potentially more efficient request that arrived later.

Skipping Policies

Best FIFO:

- While this seems to account for efficiency, it heavily penalizes those requests whose paths will always be inherently inefficient. For example, a request for establishing entanglement between two nodes at opposite ends of the network will never be serviced on a first pass.
- A degree of randomness in the skipping policy should alleviate this problem.

Skipping Policies

Random FIFO:

$$P_{\text{skip}} = \max\left\{0, 1 - \frac{E[c]}{c(\sigma)}\right\}$$

- Here $c(\sigma)$ is the cost of the path σ, and $E[c]$ is the long-term average path cost.
- If the cost of a path is below average, P_{skip} evaluates to zero and the path is never skipped.
- If it is, however, above average, then it is *likely*, but not certain, that the path will be skipped

The Path Selection Problem

The paper defers to other research that has extensively covered the area of path selection in a quantum network. It focuses on the fact that its scheduling algorithm allows for a multitude of existing path selection frameworks to be used.

Significant attention has been devoted to the use of SPF, implemented via the Djikstra or Bellman-Ford algorithms. Assuming some form of SPF is used, the problem is really one of choosing the cost function.

The Path Selection Problem

Were the path selection to be performed before knowing the results of local entanglement, some form of expectation of the success of these entanglements would have to be factored into evaluating the cost of a path.

This tends to favour shorter paths, or paths where the physical hardware used to generate local entanglements is more reliable.

Since, in the framework given when discussing scheduling, path selection is performed *after* the state of the reduced subgraph at that instant is communicated to the controller, the cost metric is greatly simplified.

The Path Selection Problem

A commonly used metric is the hop count. The paper asserts that this, however, does not account for the decoherence of qubits in quantum memory.

The fidelity of the entangled pair of qubits is dependent on the time taken for the correction bits to be sent across the network. The path with minimum hops does not necessarily have to be the path along which the time taken to send the correction bits is minimum.

The Path Selection Problem

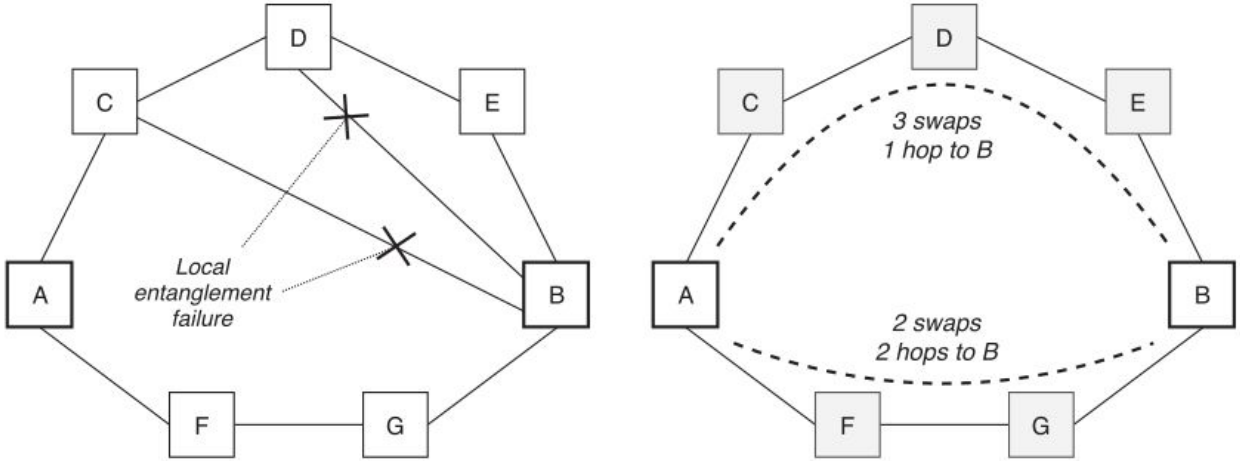


FIG. 8. Example of end-to-end entanglement between A and B with lower latency on longer path.

The Path Selection Problem

The above considerations lead the authors to propose MinMax Path Selection*:*

- For a given pair of nodes, the path that minimizes the maximum physical distance between any two nodes on the path is chosen.
- Ties are broken by choosing the path with fewer hops.