A Routing Framework for Quantum Entanglements with Heterogeneous Duration

Yuhang Gan, Xiaoxue Zhang, Ruilin Zhou, Yi Liu, and Chen Qian
University of California, Santa Cruz

***Abstract*—Entanglement routing is a fundamental problem in the network layer of quantum networks, which has become an increasingly popular research topic. This paper addresses a common problem of existing entanglement routing algorithms: they assume the Synchronized Single-time-slot model (SynSts) that requires all entanglement generation and swapping to be completed in a single time slot. This model does not align with future large-scale quantum networks that will incrementally deploy heterogeneous devices: old ones generate short-duration entanglements while advanced devices can generate long-duration entanglements. Forcing long-duration entanglements to finish in one time slot will result in sub-optimal routing performance. This paper presents a new routing framework for quantum entanglements with heterogeneous duration, including a Synchronous Multi-time-slots (SynMts) routing model and two algorithms to manage routing requests and link states. The framework is designed in such a generalized way that all existing algorithms designed for the SynSts model can still run with the SynMts model while achieving higher performance. We conduct experiments using three recently-proposed routing algorithms and find they all achieve evident throughput improvement in the new framework.**

I. Introduction

Recently, quantum networks have been experiencing rapid development as a new type of network architecture that uses special hardware (quantum repeater equipped with quantum memory) to enable the transmission of quantum bits (qubits) [1] [2]. A quantum network enables unconditional security [3] [4] of the transmitted information by a physical statement called the no-cloning theorem [5]. Besides, it enables several important applications such as distributed quantum computing [6], [7] and quantum key distribution (QKD) [8] [9]. Quantum networks are not designed to replace the classic Internet. Instead, it complements the Internet to deliver certain crucial information that is difficult to deliver on the Internet, such as shared quantum states for distributed quantum computing and secret keys that need to be shared.

There are two main types of quantum networks. The first type transmits qubits in a hop-by-hop manner [10], similar to the "store-and-forward" packet switching networks. However, it requires all repeaters in the network to be trusted, which is only applicable for private networks and is an impractical assumption in future large-scale quantum Internet [11]. The second type relies on entanglement routing [12] [11] based on the DLCZ protocol [13], which attempts to first establish entanglements (called external links) between every pair of consecutive repeaters along the way from the source to destination, and uses these external links to establish an end-to-end source-destination (S-D) connection through *entanglement swapping* [14]. Fig. 1 shows an example of entanglement swapping. To deliver a qubit from S to D, S creates a shared entanglement pair (p, q) with a neighboring repeater R, using two qubits: p at S and q at R. Similarly, D creates a shared entanglement pair (m, n) with R. Then R performs an entanglement swapping (also called Bell state measurement) on q and m. After that, an S-D connection is established with p and n. S can then deliver a qubit to D. From a networking perspective [11], we can consider there are two *external links* (p, q) and (m, n) and one internal link (q, m) that connect the whole end-to-end path from S to D. This paper focuses on the entanglement routing problem [12] [11], which is a method to find end-to-end paths consisting of external and internal links.

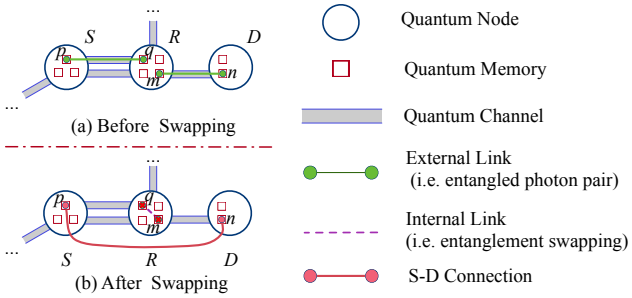


Fig. 1: Entanglement Swapping

Most solutions of entanglement routing use the ***Syn**chronized **S**ingle-**t**ime-**s**lot model* (SynSts model) [11] [15] [16] [17]. This model requires that every qubit transmission is completed in one time slot. It assumes all repeaters generate the entanglements with their neighbors and complete swapping within a time slot. The model strongly implies that all repeaters in the network carry similar hardware; hence their entanglements have similar lifetimes within a time-slot duration.

We argue that this model does not align with the evolution of future quantum networks. Similar to the history of the Internet, building a large-scale quantum network for public users is a long-term task and consists of numerous *incremental deployments* of repeaters with heterogeneous hardware. Hence, those heterogeneous devices will co-exist because they are deployed at different times and by different network providers – there is no way to shut down a shared quantum network entirely and replace all old hardware with upgraded ones. We examine the recent designs using SynSts model [11] [15] [16] and find that the typical length of each time slot is 1-

2 seconds. Moreover, the recent development of advanced quantum network hardware enables quantum memory to store photons with high fidelity for several minutes or even longer [18] [19] [20] [21], which is much longer than one time-slot. Therefore, the restriction that "the network must clear all external links at the end of each time slot" of the existing SynSts routing model will definitely result in sub-optimal results because this model forces advanced devices to use the same short entanglement time as the old devices.

In this paper, we develop a quantum network routing framework with a new ***Syn**chronous **M**ulti-**t**ime-**s**lots (**SynMts**)* model to keep external links with heterogeneous time duration. This model is a perfect balance to achieve two objectives that are necessary steps for future quantum networks: 1) support heterogeneous devices with different entanglement duration, and 2) allow adaptation of the existing algorithms of the SynSts model to the new model. The model sets the time duration of an entanglement of weak devices to be within one time slot and the stronger devices that are deployed incrementally generate entanglements that last for multiple time slots. We put forward a new parameter, "Slots To Live" (STL), to quantify the capability of quantum memory to maintain external links, which corresponds to the "Time To Live" (TTL) of traditional networks.

The new routing framework introduces several new research issues that need to be addressed. For example, which requests should be served and which links should be used in the current time slot? Intuitively short-duration links are preferred to use first and long-duration links can be saved to use in future slots. However, when combined with network traffic patterns and resource contentions, these problems can be complex. We introduce two algorithms, a *request management algorithm* and a *predictable link scheduling algorithm*. A network typically encompasses multiple routing requests, each of which demands the transmission of one or more qubits. Consequently, every transmission request persists within the network for a duration, representing diverse network flows. Given the finite resources available in the network (e.g., limited quantum memory), efficient management among flows is imperative for attaining enhanced network throughput. In our framework, we propose `ReqUp`, a request management algorithm responsible for managing the state of various transmission requests. By sorting requests prior to executing the routing algorithm, `ReqUp` ensures higher resource utilization. Although preserving external links across multiple time slots can augment network efficiency, link retention may increase network link state complexity and lead to potential resource occupancy issues, ultimately diminishing network throughput. To mitigate these concerns, we introduce a predictable link scheduling algorithm to optimize link retention and disposal, maximizing resource utilization and circumventing resource occupancy problems.

To our knowledge, **this is the first work that considers a quantum network with heterogeneous entanglement duration on mixed types of devices, which is an inevitable step of developing future quantum networks.** The proposed framework resolves the limitation of most existing algorithms that requires all operations of qubit transmission to be completed in one time slot. We summarize the main contributions of our work as follows:

- We analyze the literature on quantum network hardware, find out the limitations of the existing SynSts routing model and propose the SynMts routing model that aligns with the development of quantum hardware, because supporting heterogeneous devices is an inevitable step in building future quantum networks.
- In the new framework, we design a request management algorithm, `ReqUp`, which improves the network resources utilization in the network and a predictable links scheduling algorithm to manage the links in the network. With these two management algorithms, we show that many existing routing algorithms proposed for the SynSts model [11], [15] can be easily extended to run in the SynMts model.
- We conduct simulation experiments and analysis. The results show that many existing entanglement routing algorithms such as Q-PASS [11], Q-CAST [11], and REPS [15] receive clear performance improvements in the proposed framework compared to using the SynSts routing model.

The rest of the paper is organized as follows. In Section II, we discuss the quantum network model, background, and our observations that motivate this work. Section III presents the proposed framework with the SynMts model. In Section IV, we analyze the benefit brought by the proposed framework and discuss the adaptability of the framework for rapidly evolving quantum network hardware. Section V presents the simulation results to demonstrate the advantages of the proposed framework. The related work is discussed in Section VI. We conclude this work in Section VII.

II. Network Model, Background, and Motivation

A. Quantum network model

We briefly present the quantum network model that will be used in this work.

Quantum node. A quantum node in a quantum network is a device that carries quantum memory and quantum repeater functions. It can be the source, destination, or an intermediate node for a multi-hop path. Quantum memory is responsible for storing photons. Source and destination nodes use their quantum memory to store information qubits. And repeater nodes use their memory to store entangled photon pairs to establish external links between two adjacent repeaters. One quantum memory can store only one qubit. One node usually has about tens of quantum memory.

Classic Internet. Each node in a quantum network also carries the functions of the classic Internet. All nodes can communicate directly using the Internet.

Routing information base (RIB). Similar to existing quantum network architecture [15], [22]–[24], there is a central network controller called the routing information base (RIB) to

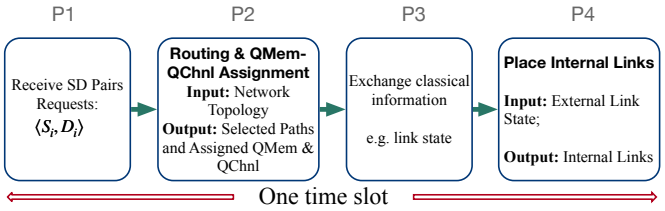


Fig. 2: SynSts model with a 4-Phases time slot [11] [15] [16]

compute routing paths and schedule available quantum links, which runs in a central cloud and connects all quantum nodes via the Internet. It adds extra 100ms latency to each time slot.

Quantum channels and links. A quantum channel is a physical medium (e.g., optical fiber) that transmits entangled photon pairs to establish external links. The process of generating external links between two neighboring nodes has two steps: 1) Generate a pair of entangled photons by an entanglement generator located inside one of the nodes. 2) The generator distributes the pair of entangled photons to the two nodes using a quantum channel. Each node uses a quantum memory to store its photon. When both nodes successfully store their photons, we say that the two nodes share an entanglement or a logical *external link* between them. Therefore, to establish an external link between adjacent nodes, we need to assign an exclusive quantum channel between the two nodes and one quantum memory for each of them.

Entanglement routing. The successfully established external link can directly transmit qubits between adjacent nodes. For nodes without direct quantum channels, we need to find a path from the source S to the destination D using a routing algorithm, and then establish external links between the intermediate repeater nodes one by one. After that, each intermediate repeater node performs an *entanglement swapping* operation on two photons belonging to two different external links. If this process is successfully performed inside every intermediate repeater node, we end up with a source-destination (SD) connection and can use this connection to transmit a qubit from S to D, as shown in Fig. 1. A sequence of qubits for the same SD pair is called a *flow* and applications request flows to the network. The above process is also known as DLCZ protocol [13]. Most current entanglement routing algorithms use the Synchronous Single-time-slot (SynSts) routing model [11] [15] [17]. A network-level synchronized time interval is called a time slot. A time slot is divided into 4 phases as shown in Fig. 2. In each phase, the network sequentially completes "receive SD pairs requests", "routing and establish external links", "communicate external links states", and "establish internal links and transmit qubits". And based on the assumption in the SynSts model that quantum memory can only store entangled photons for one time slot, all established external links and internal links need to be cleared after a transmission attempt is completed. Therefore, the routing of each time slot can be considered independent. Successfully established external links cannot be used in the subsequent time slots.

B. Characteristics of Quantum Networks

1) Quantum links are unreliable: In practice, photons can easily get lost in a quantum channel (e.g., optical fiber). In addition, quantum memory may also fail to preserve photons. Therefore, the above process of establishing external links does not always succeed and is *probabilistic*. Even if the external links are successfully established, they have to meet the quality requirements of the task. Fidelity is a parameter that measures the quality of the external link [25] [11] [22]. If the fidelity of the external linkis lower than the threshold required by a task, this link cannot be used. In fact, the transmission success rate of photons in the optical fiber decays *exponentially* with distance.

2) Network traffic has temporal and spatial locality: The traffic flow pattern of quantum networks is similar in consecutive slots. This is due to the bandwidth limitation of the quantum networks, where a successfully established SD connection can only transmit one qubit. Therefore, even if an SD request needs to transmit only 1000 qubits (much smaller compared to traditional networks, where the maximum transmission unit of an Ethernet frame is 1500 bytes), 1000 successfully established SD connections are required. SD pairs will be relatively stable over a period of time. We say the traffic state of quantum networks has temporal and spatial locality.

3) Existing SynSts model has unnecessary waste: The current SynSts routing model is based on the assumption that "quantum memory is not powerful enough to store entangled photon with a high fidelity for more than one time slot". This will reset all allocated hardware resources (mainly quantum memory and quantum channel) and clear all external links at the end of each time slot, regardless of whether they are used or not. The current SynSts model default to a time slot of about 1-2 seconds. However, according to recent quantum hardware development advanced quantum memory is able to store qubit state for several minutes [18] [19] [26] [20]. Recent and future quantum memories are able to maintain external links with high fidelity for multiple time slots. Hence forcing all quantum memory to finish in one time slot results in sub-optimal performance.

C. Key Observations and Motivation

By summarizing the above characteristics of quantum networks, we present **some observations that have not been fully considered in existing entanglement routing design**:

- Link establishment is probabilistic and the link state of a quantum network is highly dynamic;
- The network traffic pattern for consecutive slots is similar.
- Some quantum memories are now good enough to keep external links for much longer time than one time slot;
- Maintaining an external link is more likely to succeed than re-establishing it.
- The unused external links in each time slot are wasted in the existing SynSts model.

So, an intuitive idea is that *if we keep some of the established but unused external links and reuse them in subsequent time*

slots, we can improve the success rate of establishing SD connections and improve the network's performance.

Therefore, we propose our Synchronous Multi-time-slots (SynMts) quantum network routing model. Our model follows the overall 4-Phases process in SynSts model to ensure generality. Compared to the SynSts model, the link state in our model is more complicated. In a time slot, it is likely that newly established links and old links exist simultaneously. This makes the system design more difficult. To ensure the generality and flexibility of our model, we provide appropriate abstractions. **Many existing routing algorithms using the SynSts routing model can also be adapted to run with the SynMts model with little modification.** Our task is then to design algorithms that manage links and flow requests to maximize routing performance *across* different time slots.

III. Designing Routing Framework with the SynMts Model

A. Overview

In the proposed SynMts model, we define a new parameter to quantify the lifetime of each external link: Slots To Live (STL), which indicates how many time slots the external link can last in the network. Each external link has two related parameters: maximal STL (*maxSTL*) and remaining STL (*rmnSTL*). We define *strong links* to be those whose $maxSTL \geq 2$, while *weak links* are those whose $maxSTL$ are 1. The existing SynSts model can be treated as a special case in the SynMts model that all the external links are weak links. We define three operations on each existing strong link in a time slot: *Use* to be used for existing requests, *Reserve* for the next time slot, and *Release* to free the quantum memory.

In a given time slot, newly established external links and alive strong links reserved from previous time slots with $rmnSTL \geq 1$ may exist at the same time. This makes the overall link states of the network more complicated. Therefore, we propose to split the entanglement routing design space into the *network layer* and *link layer* and keep the routing algorithm only running at the network layer while the link states management takes place at the link layer by an independent link scheduling algorithm. Previously in the SynSts model, all routing designs are in the network layer and there is no need to manage the link state. The new framework with the SynMts model uses the link layer to decide whether to use, reserve, or release each link based on the flow requests and provide a proper abstraction for the network layer. The routing algorithm can only consider the case where $maxSTL = 1$ for all external links, in line with the original SynSts model. Therefore all existing entanglement routing algorithms can still be used in the SynMts model.

The new algorithms that are needed in the link layer include:

Request management. In the SynMts model, the routing information base (RIB) is responsible for request management, routing, resource allocation, and link scheduling. With multiple simultaneous requests coming, first-come first-serve (FCFS) may cause unfairness or starvation among requests. Thus, a rational request management algorithm is needed

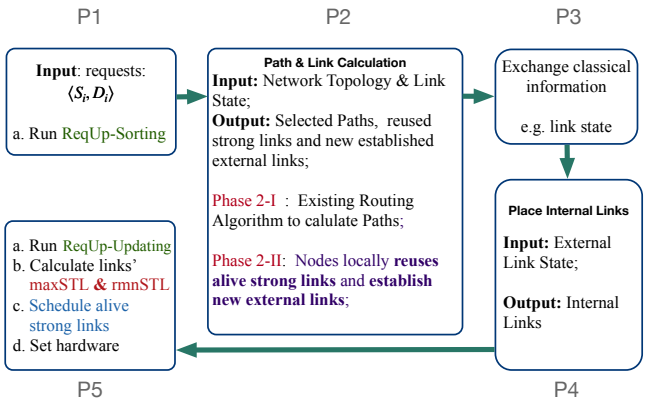


Fig. 3: Overview of SynMts model

to ensure both efficiency and fairness. Also, some requests may suffer from extremely slow transmission speeds due to link failures or intense resource contention in quantum networks and cannot be satisfied before the request timeouts. Hence all future resources spent on them will be wasted. We call them *straggler requests* and it is necessary to promptly detect straggler requests and remove them. We propose the request management algorithm, called `ReqUp`, including two processes: Sorting and Updating. `ReqUp-Sorting` sorts the requests at the beginning of each time slot according to their priority to gain higher network efficiency and better fairness. And `ReqUp-Updating` is responsible for updating request states, detecting and removing the straggler requests in the network at the end of each time slot.

Link scheduling. The SynMts model allows strong links to be kept for multiple time slots, but maintaining them consumes hardware resources. Thus, a link scheduling algorithm is needed to reserve the links that will likely to be reused and release other links. We propose a predictable link scheduling algorithm, which retains strong links for unfinished requests and uses a prediction algorithm to determine which links to retain or release for completed requests. The predictor comprises a key-value pairs table and uses Bloom filters to predict the usage of links. At the end of each time slot, alive strong links are categorized as *Use*, *Reserve*, or *Release*.

We then present the process of the SynMts model in one time slot as shown in Fig. 3.

- In Phase 1, the RIB runs the request management algorithm, `ReqUp-Sorting`, to sort all the requests according to their priority.
- In Phase 2, there are two sequential parts:
  - Phase 2-I: Network layer routing algorithm calculates the paths as well as allocates corresponding hardware resources.
  - Phase 2-II: Based on the routing results, each node locally first uses its existing strong links to satisfy some of the links assignments and then tries to build other external links using the remaining hardware resources at the link layer.
- Phase 3 and 4 are the same as the SynSts model [11], to

communicate external links states, and establish internal links to transmit qubits respectively.

- In Phase 5 at the end of each time slot,
  - The *maxSTL* and *rmnSTL* parameters are calculated and updated for each existing strong link according to the task fidelity requirement in this time slot.
  - The RIB executes the requests management algorithm, `ReqUp-Updating`, to update the requests' status.
  - The RIB then runs the **Predictable Link Scheduling algorithm**, determining the actions for all remaining strong links - to *Use*, *Reserve*, or *Release* - within the network and broadcasting the decision to each node.
  - Each node removes expired strong links, links of the "Release" state, and weak links, and finally resets the hardware.

B. Phase 1

Request management: `ReqUp-Sorting`**.** At the beginning of Phase 1, the network receives multiple S-D requests. The RIB is responsible for maintaining request status, including the size of the data to be transmitted and the timeout for each request. Based on this status information, the RIB can better allocate resources and schedule requests. To achieve higher network efficiency and fairness, `ReqUp-Sorting` is designed to sort the requests based on priority before routing and resource allocation. Requests with higher priority will be served first, and others can only allocate the remaining resources. We use a 5-tuple to represent the metadata of each request: $\langle\langle src, dst\rangle, transBits, sentBits, elpasedSlots, totalSlots\rangle$. The first tuple denotes the source and destination of the request, followed by the size of the data to be transmitted, the size of the data already transmitted, the number of time slots elapsed, and the maximum timeout of the request. The priority of each request is defined as follows:

$$\alpha_1 \times \frac{1}{totalSlots - elapsedSlots} + \alpha_2 \times \frac{sentBits}{transBits} \quad (1)$$

This formula consists of two terms. The first term mainly indicates how soon the current request will time out, while the second term shows the progress of the current transmission request. When sorting requests, we consider both time and progress factors: requests closer to the time-out have a higher priority to minimize the probability of request time-out failures; requests with faster progress have higher priority, so they can be completed sooner and release resources for new requests. The framework can also adjust the weight of these two terms by assigning α_1 and α_2 different values.

C. Phase 2

Network layer routing. In Phase 2-I, the routing algorithm computes paths without the knowledge of the actual link states. We let the routing algorithm run on a "virtual network topology with complete hardware resources". Just like in traditional networks, the routing algorithm runs only at the network layer and does not care about the link states. The routing algorithm only needs to consider single-time-slot scenarios, in line with the SynSts model. Therefore, existing routing algorithms under the SynSts model can also be used directly in the new framework with little modification.

Link layer hardware resource allocation. After computing paths for all the requests in Phase 2-I, the RIB distributes the path information to each node. At the link layer, each node manages its resource independently and assigns resources to build external links. After receiving the computed paths, each node allocates hardware resources to establish external links accordingly. Nodes will not start building external links at this time. They first check available strong links remaining from the last time slot locally. If the strong links could meet the fidelity requirement of the requests, the nodes directly use them for the paths. If the paths cannot be fulfilled by the existing strong links, nodes will then establish new external links according to the path requirements.

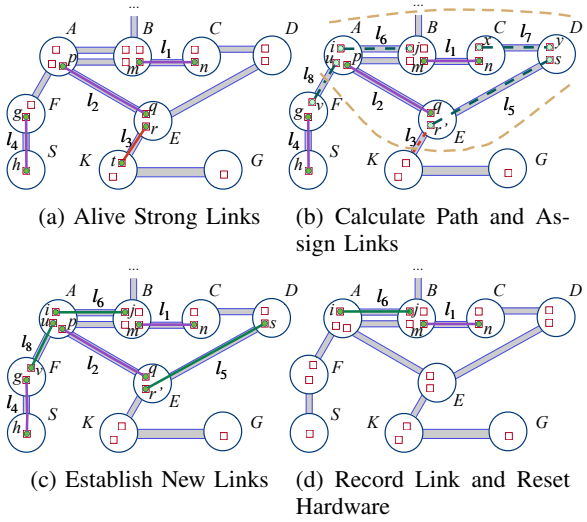
D. Phase 5

Links' STL calculation. For each external link, nodes locally obtain the corresponding STL parameters based on the reliability of their quantum memory and the task requirements of link fidelity. For instance, if a task demands a minimum fidelity of 85% for links, and one time slot lasts for 2 seconds, a link can be maintained for approximately 20 seconds before its fidelity decays to 85%. The maximum STL for this link would be 10 in this case. The *rmnSTL* is initialized to *maxSTL* upon the successful establishment and decreases as time flows. The link is considered expired when the *rmnSTL* becomes 0. At the beginning of Phase 5, each node calculates the STL parameters of its remaining unused strong links. The RIB collects the updated STL parameters of all the remaining strong links from nodes for future use in link scheduling.

Request management: `ReqUp-Updating`**.** In Phase 5, the RIB needs to update the state of the requests according to the transmission results of the current time slot. To achieve this, we introduce the `ReqUp-Updating` algorithm. The pseudocode of `ReqUp-Updating` running in the central controller is shown in Algorithm 1. First, the RIB counts the amount of data transmitted for each request during the current time slot and updates the $sentBits$ (Lines 2-3). Then, the controller increases the values of $elapsedSlots$ by 1 (Line 4). For completed requests ($sentBits == transBits$) and time-out requests ($elpasedSlots == totalSlots$), the RIB removes them from the request pool (Lines 6-10). The RIB also needs to remove those straggler requests that cannot be satisfied before it expires (Lines 11-13). We define a flow is a straggler if:

$$\frac{elapsedSlots}{totalSlots} \geq f_1 \;\&\&\; \frac{sentBits}{transBits} \leq f_2 \quad (2)$$

where f_1 and f_2 are two thresholds for the remaining time and transmission progress respectively. For example, with $f_1 = 0.75$ and $f_2 = 0.25$, a request is considered a straggler request



(a) Alive Strong Links (b) Calculate Path and Assign Links

(c) Establish New Links (d) Record Link and Reset Hardware

Fig. 4: Example of SynMts Routing Model

if 75% of the time has elapsed but less than 25% of the data has been transmitted.

Algorithm 1 ReqUp - Updating

Input: $R = \{r_i\}, S = \{s_i\}$
// R: Set of all existing requests r_i in network
// S: Set of all requests r_i's 5-tuple states
Output: Updated R, S

1: **for** each five-tuple state $s_i \in S$ **do**
2: $succ_i \leftarrow$ Succeeded transmission number of request r_i in this time slots;
3: $sentBits_i \leftarrow sentBits_i + succ_i$;
4: $elapsedSlots_i \leftarrow elapsedSlots_i + 1$;
5: **end for**
6: // F : Finished requests set;
7: // T : Time-out requests set;
8: $F \leftarrow$ Subset of all requests that $sentBits == transBits$ in R;
9: $T \leftarrow$ Subset of all requests that $elapsedSlots == totalSlots$;
10: $R \leftarrow R - F - T$;
11: // S : Straggler requests set;
12: $S \leftarrow$ Subset of all requests that $\frac{elpasedSlots}{totalSlots} \geq f_1$ && $\frac{sentBits}{transBits} \leq f_2$ in R;
13: $R \leftarrow R - S$;
14: // N : New coming requests set;
15: $R \leftarrow R + N$;
16: Update S according to R
17: **return** R, S

Predicatable link scheduling. In the SynMts model, unused strong links in the current slot can be reserved for future use in the subsequent time slot, because generating a new link in the next slot is probabilistic and does not always succeed. On the other hand, not all strong links should be reserved, because

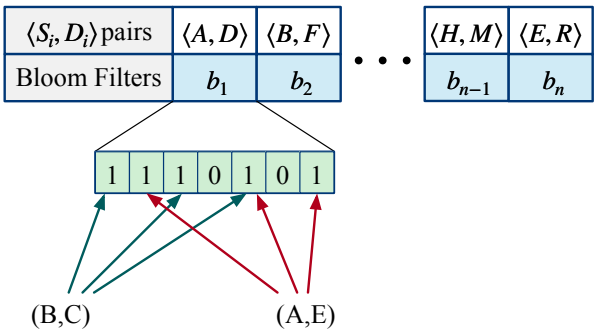


Fig. 5: Link Usage Predictor Table Example

reserving external links still consumes quantum memory. If a link will not be used in the future, reserving it is a waste. For instance, in Fig. 4a, a strong link $l_3 : (r, t)$ is reserved from the last time slot. In the current time slot, a new request $\langle A, D\rangle$ comes in, and the RIB computes the path $A \rightarrow E \rightarrow D$ for this request. This request cannot succeed because node E does not have enough quantum memory to build an external link with D. Its quantum memory is locked by the reserved strong link l_3.

We propose a predictable link scheduling algorithm to manage strong links to maximize efficiency and save costs. By predicting link usage, we can maintain necessary links to improve S-D connection success rates and release unnecessary links. Considering the spatial locality of the network flows, we assume that the network situation in two consecutive time slots are similar, and the routing paths are similar as well. Thus, the algorithm makes all alive links related to the paths of unfinished requests continue to be reserved in the following time slot. For other links, we need to determine the benefits of retaining them. We use a prediction table to estimate whether new requests will utilize them, reserving those anticipated to be used and releasing those not expected to be used.

The prediction table uses Bloom Filters [27], which are probabilistic data structures for membership queries, to record the links of paths that are used in the current slot. Fig. 5 shows the structure of this predictor. The main body of the predictor is a key-value pair table, where the keys are the $\langle src, dst\rangle$ tuples of the most recent x requests in the network, and the value is a Bloom filter. The Bloom filter contains all the edges the routing algorithm has chosen for the current $\langle src, dst\rangle$ since the predictor table entry was created. We update the corresponding Bloom filters based on routing results at the end of each time slot. For example, the edge (B, C) and (A, E) are selected for request $\langle A, D\rangle$ in the current time slot. The prediction table will update the corresponding entries in the Bloom filter of this request to be 1.

For each unused strong link, if the Bloom Filter returns positive, the link will be reserved. Otherwise, it will be released. After getting all these results, the RIB sends the reserve/release information to all nodes.

E. Example of Routing with the SynMts model

We show an example of the proposed framework with nine nodes in the network in Fig. 4. Suppose in time slot t_{i-1}, there

are three requests in the network: $\langle B, D\rangle$, $\langle A, S\rangle$ and $\langle A, G\rangle$. The routing algorithm assigns path $p_1 : (B \to C \to D)$ for request $\langle B, D\rangle$, but only successfully established l_1:(m, n). The path assigned for request $\langle A, G\rangle$ is $p_2 : (A \to E \to K \to G)$, with l_2:(p, q) and l_3:(r, t) successfully established, while the edge (K, G) failed to establish an external link. And for request $\langle A, S\rangle$, routing algorithm calculates the $p_3 : (A \to F \to S)$, with only l_4:(g, h) established. Thus SD connections cannot be built for these three requests. And these unused links are left in the network. Fig. 4a shows the network at the end of time slot t_{i-1} before running the Predictable Link Scheduling Algorithm. Links l_1, l_2, l_3, and l_4 are all alive strong links. The $maxSTL$ and $rmnSTL$ of these links are locally computed and updated by their end nodes, and then sent to the central controller, as shown in Table I.

TABLE I: STL Table Example

	t_{i-1}		t_i	
Strong Link	maxSTL	rmnSTL	maxSTL	rmnSTL
l_1	3	2	3	1
l_2	4	3	4	2
l_3	4	3	-	-
l_4	4	3	4	2
l_6			5	4

We assume that in time slot t_{i-1}, request $\langle A, G\rangle$ and $\langle B, D\rangle$ are both timeout or fulfilled via other paths. So in Phase 5, both requests $\langle A, G\rangle$ and $\langle B, D\rangle$ are marked as committed and removed from the requests queue. $\langle A, S\rangle$ has not finished yet and will be executed again in the next time slot t_i. Then, the RIB runs the Predictable Link Scheduling Algorithm to manage the remaining links, determining whether to reserve or release them. First, for unfinished $\langle A, S\rangle$, the action of link l_4 can be directly updated to *Use*. Suppose we have a new request $\langle A, D\rangle$ to be executed in the next time slot t_i. For this request, we assume it has shown up in the network before, and the predictor table has a corresponding entry and a Bloom filter b_1 for it already, as shown in Fig. 5. According to b_1, the routing path for request $\langle A, D\rangle$ may pass through edges (B, C) and (A, E) corresponding to l_1 and l_2, so these two links are set to be *Reserve*, while l_3 is *Release*.

The routing algorithm calculates two paths for $\langle A, D\rangle$: $p_4 : (A \to B \to C \to D)$ and $p_5 : (A \to E \to D)$ (as shown in Fig. 4b). p_4 can use l_1 directly. We only need to establish l_6:(i, j) and l_7:(x, y), and thus increase the success rate of connection establishment. For p_5, l_2 retained from time slot t_{i-1} can be used directly. We still need to build external links between (A, E) and (E, D). The released l_3 freed up the quantum memory in node E, which can be used to establish l_5:(r', s) for path p_5, avoiding resource contention; otherwise, l_5 could not be established. Finally for $\langle A, S\rangle$, the routing algorithm still assigns p_3. We reuse l_4 and establish new $l_8 : (u, v)$. At the end of Phase 2, as shown in Fig. 4c, p_4 fails to establish an external link on edges (C, D), while p_5 successfully established l_5. Next, in Phases 3 and 4, nodes perform entanglement swapping and try to build SD connections. In Phase 5, we update link states, schedule the alive strong links (l_1 and l_6 in this example) and reset other hardware.

IV. Performance Analysis

We quantify the performance improvement of the SynMts routing model. For a quantum network $G = \langle V, E\rangle$, we set d SD pair requests in a time slot on average, and the routing algorithm computes b paths for each SD pair with an average width of c for each path, meaning there are c parallel links at each edge of the path. Ideally, the throughput (# of SD connections) of the network is dbc. However, the establishment of both external links and internal links is probabilistic. We first define the success rate of each external link as $p_i, i \in E$ and the success rate of each internal entanglement swapping in each repeater node as $q_j, j \in V$, and we assume that the length of one path is l. Then the probability that an SD connection is successfully established is

$$Q = \prod_{i=1}^{l} p_i \prod_{j=1}^{l-1} q_j \tag{3}$$

We set the expectation of p_i to be $\bar{p}$, the expectation of q_i to be $\bar{q}$ and the average path length is $\bar{l}$ hops, then the expectation of an SD connection establishment success rate in the network is $\bar{Q} = \bar{p}^{\bar{l}}\bar{q}^{\bar{l}-1}$. The expected throughput of the overall network is $dbc\bar{Q}$. So, there are $M = dbc(1-\bar{Q})$ finally failed SD connections.

In Phase 2, there will be $M\bar{p}^{\bar{(l)}}$ SD connections that succeed in establishing every external link (but may still fail in Phase 4), we call them P2 succeeded connections. Thus, there will be $M(1-\bar{p}^{\bar{(l)}})$ Phase 2 failed connections, and their average length is $\bar{l}\bar{p}$. So we have a total of $L_1 = M(1-\bar{p}^{\bar{l}})\bar{l}\bar{p}$ established external links. Since we do not assume that the routing algorithm will try to reuse them, these links will be directly reserved for the next time slot.

In Phase 4, $M\bar{p}^{\bar{(l)}}$ SD connections will try to establish internal links. We assume that when entanglement swapping is performed, it is tried sequentially along the path. So it may fail at any repeater node. The expectation of the number of final remaining links in an SD connection is $S = (l-2)q^0(1-q)^1 + (l-3)q^1(1-q) + ... + (l-(i+1))q^{i-1}(1-q) + ... + (l-(l-1))q^{l-2}(1-q))$, i.e.

$$S = (1-q)\sum_{1}^{l-2}(l-(i+1))q^i$$

Therefore, in phase 4, there will eventually be $L_2 = M\bar{p}^{\bar{(l)}}\bar{l}S$ unused links. At the end of a time slot, we will have $L = L_1 + L_2$ links reserved for the next time slot t_{i+1}.

We then calculate the performance gain of using these alive strong links at the next time slot t_{i+1}. In time slot t_{i+1}, we assume that these links are evenly distributed among each SD connection that will be attempted, with a usage rate of β, $(0 \leq \beta \leq 1)$. So each SD connection has an average of $\bar{m} = \frac{\beta L}{dbc}$ alive strong links. The probability of successful establishment of each SD connection is Eq. 3, and if we use alive strong links

instead of assigned links, then the expectation of successful establishment of each SD connection will be

$$\bar{Q}' = \prod_{i=1}^{\bar{l}-\bar{m}} p_i \prod_{j=1}^{l-1} q_j$$

i.e., the performance of the network will be improved by

$$\frac{\bar{Q}' - \bar{Q}}{\bar{Q}}$$

.

V. Performance Evaluation

In this section, we evaluate the performance of our SynMts model through simulations by extending an existing quantum network simulator that is open-sourced [11]. One simulation generates random network topologies with the required network parameters. The performance of the main hardware resources and the SD pair requests are set according to the given parameters.

A. Methodology

1) Network Topology: We randomly generate the network topology and set the entire quantum network to be distributed in a 100 $\times$ 100 unit square, each unit can be treated as 1km. To generate a specific quantum network, we need to set node number n, average edge degree E_d, the average external links establishment success rate E_p, the average internal links success E_q and the average number of quantum memory each node has. The length of each edge is at least $\leq \frac{50}{\sqrt{n}}$. Besides, we generate edges between nodes following the Waxman model [28].

2) Requests Distribution: Taking into account the possibility that the traffic patterns of quantum networks exhibit similar features to the Internet. such as following the Zipfan distribution [29]–[31], we generate the routing requests with a Zipfan distribution characterized by a parameter $\alpha = 1.05$.

3) Metrics and Parameter Selection: We use several routing algorithms to run in the proposed framework with the SynMts model and demonstrate their throughput improvement compared to their performance with the SynSts model. We use the absolute throughput improvement and relative improvement ratio as the main performance metrics. In order to assess the effectiveness of the `ReqUp-Sorting` requests management algorithm, we initially let each request's time-out limitation ($\#\ of\ totalSlots$) be equal to the number of transmitted bits ($\#\ of\ transBits$). To examine the performance of the Link Scheduling algorithm, we adopt the "Link Hit Ratio" as the principal metric, which represents the proportion of the alive strong links reserved in the previous time slots that are selected by the routing algorithm in the current time slot.

We vary the number of nodes n within the set $\{50, 100, 200, 400\}$, the average successful establishment rate of external links E_p within $\{0.1, 0.2, 0.3, 0.4, 0.5\}$, and the success rate of internal links q within $\{0.1, 0.2, 0.3, 0.4, 0.5\}$. We select the number of simultaneous requests running in the network from $\{5, 10, 15, 20, 25\}$. The maximum $maxSTL$ for strong links is 5, and the proportion of strong links in the network, r, is chosen from $\{0.0, 0.2, 0.4, 0.6, 0.8, 1.0\}$. When r is 0.0, the link scheduling algorithm is disabled; only the requests management algorithm is executed.

4) Default Parameters: To control variables, we set the default settings as follows: the number of nodes, $n = 100$; the average success rate of external link establishment, $E_p = 0.3$; the success rate of internal links, $q = 0.3$; and the average number of quantum channels between adjacent nodes is 6. The number of quantum memories per node is uniformly chosen from 20 to 25. We assume the efficiency of quantum memory is 1 (qubits can be stored and read out successfully every time). We set the default to have 10 requests running simultaneously in each time slot. The transmission size of a single request is chosen with equal probability from 16 to 32 bits. The default proportion of strong links r in the network's external links is 0.6, with the average $maxSTL$ of each strong external link being 5. We set the maximal size of the predictor table to 100, the size of each Bloom filter to 500, and rebuild a new predictor table every 50 time slots.

For a given set of parameters, we simulate 10,000 time slots on ten different network topologies and provide the average results.

B. Evaluation results

We chose three different types of recent entanglement routing algorithms to implement in the original SynSts model and the proposed framework and compared the results. These algorithms are Q-PASS, Q-CAST [11], and Redundant Entanglement Provisioning and Selection (REPS) [15]. Q-PASS is an offline algorithm that computes all paths in the initialization step. Q-CAST is an online algorithm that calculates the paths online based on the link state and remaining resources in the network. And REPS is a network flow optimization based routing algorithm. The code of Q-PASS and Q-CAST is publicly available [11]. As the REPS code is not publicly available, we endeavored to implement it as faithfully as possible based on the descriptions provided within the research paper.

1) Throughput improvement: Figures 6 to 9 display the throughput for the three distinct types of routing algorithms executed on the SynMts model in comparison to the original SynSts model. And Figures 10 to 13 show the relative improvement ratios of the throughput. Under default parameters, the throughput improvement for all three algorithms, Q-PASS, Q-CAST, and REPS, is notable, with average increases of 23%, 32%, and 9%, respectively. This outcome shows the effectiveness of the SynMts transmission model. In particular, since the most significant difference between the SynMts model and the SynSts model is that the SynMts model attempts to preserve alive strong links across time slots, Fig. 10 shows the variation of throughput improvement with the average success rate of external links. The improvement ratio gradually decreases as E_p increases. This trend occurs because, as E_p increases, the success rate of re-establishing an external link rises, subsequently reducing the benefit of preserving an

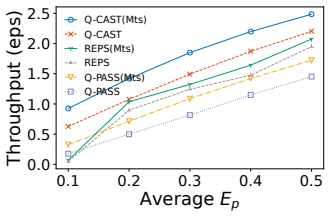


Fig. 6: Throughput vs. external links success rate

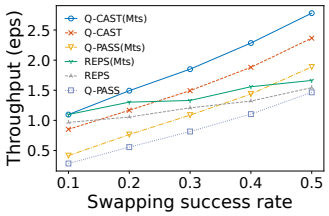


Fig. 7: Throughput vs. internal links success rate

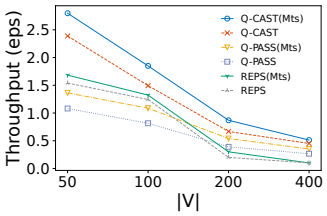


Fig. 8: Throughput vs. network size

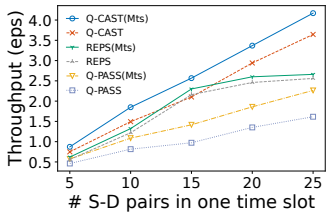


Fig. 9: Throughput vs. # of S-D pairs

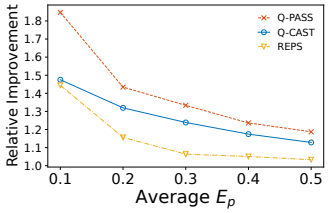


Fig. 10: Relative improvement vs. external links success rate

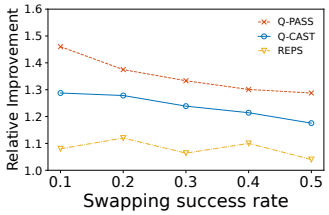


Fig. 11: Relative improvement vs. internal links success rate

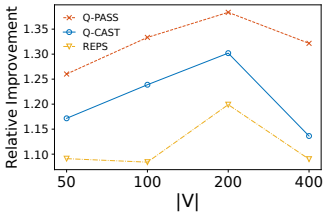


Fig. 12: Relative improvement vs. network size

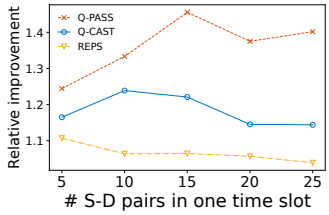
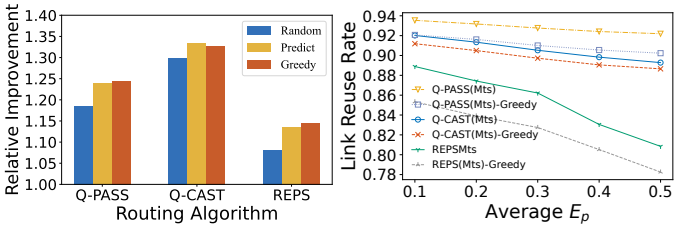


Fig. 13: Relative improvement vs. # of S-D pairs

external link. Nevertheless, given that the real-world maximum success rate of establishing an external link at the quantum network link layer does not surpass 0.5, the benefits yielded by the SynMts model remain substantial.

2) Link reuse rate: We define the link reuse rate as the proportion of preserved alive strong links in the current time slot that are used in subsequent time slots. To validate the efficiency of our link scheduling algorithm, we conducted a comparative analysis with Random and Greedy algorithms. The Greedy algorithm preserves all alive strong links at the conclusion of each time slot, whereas the Random algorithm decides whether to retain a link with a probability of 0.5. Figure 14a demonstrates the throughput improvement ratios for the three routing algorithms under various link scheduling algorithms. The Random algorithm exhibits the lowest improvement ratio, as it indiscriminately releases 50% of the alive strong links in the network. Without resource competition, this will reduce the available alive strong links in the next time slot by 50%, causing resource waste and reduced network efficiency. Greedy and our predictable link scheduling algorithm show quite similar performance improvement ratios. To better illustrate the difference in performance between the Greedy algorithm and the proposed link scheduling algorithm, we show the link reuse rate of the two algorithms (Predictable Link Scheduling and Greedy) in Fig.14b, which is the proportion of retained alive strong links reassigned by the routing algorithm in subsequent time slots. This metric represents the proportion of preserved alive strong links employed in subsequent time slots. Our predictable link scheduling algorithm possesses a higher link reassigning rate, signifying reduced resource waste and a decreased likelihood of resource competition.

3) Link lifetime: Fig. 15 shows the CDF of the average link lifetime when setting the lifetime of the links to infinity, i.e., $maxSTL = \infty$ in three different routing algorithms. One link will not expire before it is used in this setting. It can be



(a) Link Scheduling Algorithm vs. Speed Up Ratio (b) Predictable and Greedy Comparison

Fig. 14: Link Scheduling Algorithm Comparison

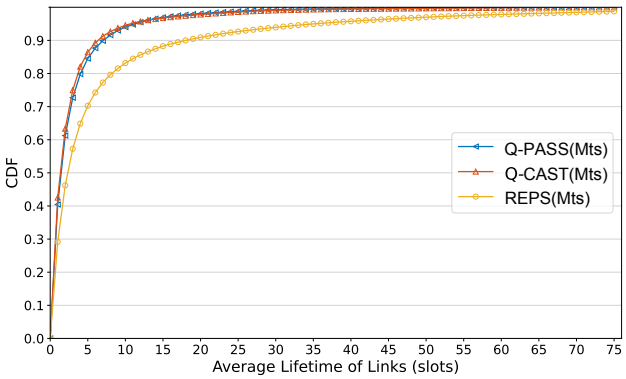


Fig. 15: Link Lifetime CDF

seen that even if the $maxSTL = \infty$, most of the links of the three algorithms will be used and consumed soon after they are established. For the three different types of algorithms (REPS, Q-CAST, and Q-PASS), the ratio of the average lifetime ≤ 5 links in the network to $64.86\%, 86.42\%, 84.50\%$.

The reason for this phenomenon is the locality of network traffic. Usually, quantum network traffic has a stronger locality compared to traditional networks, hence the traffic conditions within several adjacent time slots will be similar. The SD requests of the first and next several time slots will partially

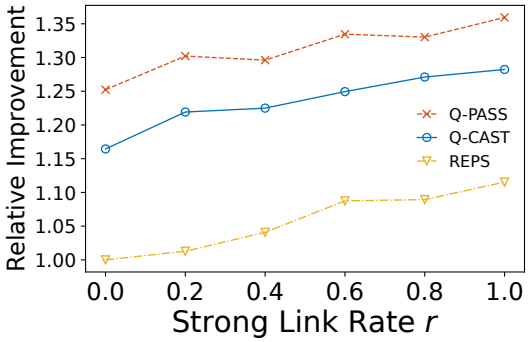


Fig. 16: Speed Up vs. Strong Link Ratio

overlap, and therefore the paths computed by the routing algorithm in these time slots will also quite overlap. The links established in the first time slot but not used are reserved for the several next time slots and will be chosen with a high probability. The remaining new links are established in the second time slot, and the old links and the new links are combined to form a complete SD connection. Therefore, most of the strong links created in the first time slot but not successfully used will be selected and consumed in the next several time slot. This is also consistent with the traffic locality of the quantum network and verifies the effectiveness of the SynMts model.

4) Incremental deployment: As the hardware of quantum networks, especially quantum memory, is evolving rapidly, quantum networks may have hardware with different performances within a network. Strong and weak links may exist in a network simultaneously, and the proportion of these two will gradually change with the evolution of the quantum networks. The SynMts model is naturally suitable for fast-growing hardware because we consider the lifetime of links to maximize the utilization of strong links. Fig. 16 shows how the performance of different algorithms varies with the proportion of strong links to the total number of links under the default parameters settings. We let the ratio of strong links of all links r vary in the set $\{0.0, 0.2, 0.4, 0.6, 0.8, 1.0\}$. It can be seen that the performance of the network increases with the increase of the strong link ratio. Notice that, even when the r is 0.0, the improvement for Q-PASS and Q-CAST is not 1.0. This is because the `ReqUp` algorithm provides a more efficient order of requests by sorting and detecting straggler requests. REPS is a global network flow algorithm, so the order of the requests will not be considered during routing. The results show that all three different types of routing algorithms can benefit more from the strong link ratio increasing by using the framework with the SynMts model in continuously evolving quantum networks.

5) Scalability: Finally, we evaluate the scalability of the SynMts model. We still use default parameters. We let the number of nodes n vary within the set $\{50, 100, 200, 400\}$ and observe the performance of the algorithm. Fig. 12 shows that the algorithms running on the SynMts model have an improvement ratio on throughput greater than 1.0.

VI. Related Work

The quantum networks routing problem has received special attention. The studies of this field can be divided into two parts: routing frameworks and routing algorithms. The routing framework design aims to design a reasonable network architecture and technology stack, while the routing algorithm aims to design a particular routing algorithm to improve the transmission performance of the network. In recent years, several transmission protocols and network architectures have been proposed. [32] proposed a high-level quantum network stack, but did not consider some critical parts, such as establishing links is probabilistic. [25] proposed a link layer protocol that provides a robust entanglement generation service and proposed a quantum network stack inspired by the TCP/IP stack. [33] designed a single time slot network layer quantum data plane protocol based on [25]. Some complementary functional allocations for entanglement distillation have also been proposed to complement the existing network stack [34] [35], but they do not take into account the difference in reliability of different quantum network hardware and are limited to a single time slot routing model.

In the field of routing algorithm design, many algorithms have been proposed [36] [12]. Shi and Qian [11] define the SynSts model and propose Q-PASS and Q-CAST that can run on arbitrary network topology. Zhao *et al.* [15] propose a network flow based optimization to provide additional backups to cope with possible failures when creating external links. Chakraborty *et al.* [37] propose to guarantee the fidelity of a connection by limiting the length of routing paths. A centralized entanglement distillation and fidelity guarantee route schedule is also proposed [22]. All these algorithms are based on the SynSts model. Link scheduling algorithms [23] [24] [38] have also been proposed for quantum networks. However, none of these methods consider heterogeneous devices in quantum networks and resource reuse and scheduling across time slots.

VII. Conclusion

This paper identifies a common problem of all existing entanglement routing algorithms: they all assume the SynSts routing model that requires all entanglements to be used in the current time slot. We propose the SynMts model that is aligned with the future quantum networks that include heterogeneous devices. In addition to the SynMts model, we present the new routing framework that includes request and link scheduling algorithms across multiple time slots. Analysis and simulation results show that the new routing framework can effectively exploit the advantages of long-duration entanglements and achieve higher throughput for the network even when using the same routing algorithms.

Acknowledgment

The authors were partially supported by NSF Grants 1750704, 1932447, and 2114113, and DoE Grant DE-SC0022069. We thank the anonymous reviewers for their comments.

REFERENCES

[1] Z.-S. Yuan, Y.-A. Chen, B. Zhao, S. Chen, J. Schmiedmayer, and J.-W. Pan, "Experimental demonstration of a bdcz quantum repeater node," *Nature*, 2008.

[2] H.-J. Briegel, W. Dür, J. I. Cirac, and P. Zoller, "Quantum repeaters: the role of imperfect local operations in quantum communication," *Physical Review Letters*, 1998.

[3] M. S. Sharbaf, "Quantum cryptography: An emerging technology in network security," in *Proceedings of IEEE International Conference on Technologies for Homeland Security (HST)*, IEEE, 2011.

[4] J. Yin, Y. Cao, Y.-H. Li, S.-K. Liao, L. Zhang, J.-G. Ren, W.-Q. Cai, W.-Y. Liu, B. Li, H. Dai, *et al.*, "Satellite-based entanglement distribution over 1200 kilometers," *Science*, 2017.

[5] J. L. Park, "The concept of transition in quantum mechanics," *Foundations of physics*, 1970.

[6] R. Van Meter and S. J. Devitt, "The path to scalable distributed quantum computing," *Computer*, 2016.

[7] A. S. Cacciapuoti, M. Caleffi, F. Tafuri, F. S. Cataliotti, S. Gherardini, and G. Bianchi, "Quantum internet: networking challenges in distributed quantum computing," *IEEE Network*, 2019.

[8] C. H. Bennett and G. Brassard, "Quantum cryptography: Public key distribution and coin tossing," *Proceedings of IEEE International Conference on Computers, Systems, and Signal Processing (CCSP)*, 1984.

[9] A. K. Ekert, "Quantum cryptography and bell's theorem," in *Quantum Measurements in Optics*, 1992.

[10] C. Elliot, "Building the quantum network," *New Journal of Physics*, 2002.

[11] S. Shi and C. Qian, "Concurrent entanglement routing for quantum networks: Model and designs," in *Proceedings of ACM SIGCOMM*, 2020.

[12] M. Pant, H. Krovi, D. Towsley, L. Tassiulas, L. Jiang, P. Basu, D. Englund, and S. Guha, "Routing entanglement in the quantum internet," *npj Quantum Information*, 2019.

[13] L.-M. Duan, M. D. Lukin, J. I. Cirac, and P. Zoller, "Long-distance quantum communication with atomic ensembles and linear optics," *Nature*, 2001.

[14] R. Van Meter, T. Satoh, T. D. Ladd, W. J. Munro, and K. Nemoto, "Path selection for quantum repeater networks," *Networking Science*, 2013.

[15] Y. Zhao and C. Qiao, "Redundant entanglement provisioning and selection for throughput maximization in quantum networks," in *Proceedings of IEEE INFOCOM*, IEEE, 2021.

[16] Y. Zeng, J. Zhang, J. Liu, Z. Liu, and Y. Yang, "Multi-entanglement routing design over quantum networks," in *Proceedings of IEEE INFOCOM*, IEEE, 2022.

[17] A. Farahbakhsh and C. Feng, "Opportunistic routing in quantum networks," in *Proceedings of IEEE INFOCOM*, IEEE, 2022.

[18] Y. Wang, M. Um, J. Zhang, S. An, M. Lyu, J.-N. Zhang, L.-M. Duan, D. Yum, and K. Kim, "Single-qubit quantum memory exceeding ten-minute coherence time," *Nature Photonics*, 2017.

[19] C. E. Bradley, J. Randall, M. H. Abobeih, R. Berrevoets, M. Degen, M. A. Bakker, M. Markham, D. Twitchen, and T. H. Taminiau, "A ten-qubit solid-state spin register with quantum memory up to one minute," *Physical Review X*, 2019.

[20] H. Bartling, M. Abobeih, B. Pingault, M. Degen, S. Loenen, C. Bradley, J. Randall, M. Markham, D. Twitchen, and T. Taminiau, "Entanglement of spin-pair qubits with intrinsic dephasing times exceeding a minute," *Physical Review X*, 2022.

[21] Y. Ma, Y.-Z. Ma, Z.-Q. Zhou, C.-F. Li, and G.-C. Guo, "One-hour coherent optical storage in an atomic frequency comb memory," *Nature communications*, 2021.

[22] Y. Zhao, G. Zhao, and C. Qiao, "E2e fidelity aware routing and purification for throughput maximization in quantum networks," in *Proceedings of IEEE INFOCOM*, 2022.

[23] M. Ghaderibaneh, H. Gupta, C. Ramakrishnan, and E. Luo, "Pre-distribution of entanglements in quantum networks," in *IEEE International Conference on Quantum Computing and Engineering (QCE)*, pp. 426–436, 2022.

[24] P. Fittipaldi, A. Giovanidis, and F. Grosshans, "A linear algebraic framework for quantum internet dynamic scheduling," in *IEEE International Conference on Quantum Computing and Engineering (QCE)*, pp. 447–453, 2022.

[25] A. Dahlberg, M. Skrzypczyk, T. Coopmans, L. Wubben, F. Rozpedek, M. Pompili, A. Stolk, P. Pawełczak, R. Knegjens, J. de Oliveira Filho, *et al.*, "A link layer protocol for quantum networks," in *Proceedings of ACM SIGCOMM*, 2019.

[26] C. P. Anderson, E. O. Glen, C. Zeledon, A. Bourassa, Y. Jin, Y. Zhu, C. Vorwerk, A. L. Crook, H. Abe, J. Ul-Hassan, *et al.*, "Five-second coherence of a single spin with single-shot readout in silicon carbide," *Science advances*, 2022.

[27] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, 1970.

[28] B. M. Waxman, "Routing of multipoint connections," *IEEE journal on selected areas in communications*, 1988.

[29] W. Fang and L. Peterson, "Inter-as traffic patterns and their implications," in *Seamless Interconnection for Universal Services. Global Telecommunications Conference. GLOBECOM'99.(Cat. No. 99CH37042)*, vol. 3, pp. 1859–1868, IEEE, 1999.

[30] Y. Zhang, L. Breslau, V. Paxson, and S. Shenker, "On the characteristics and origins of internet flow rates," in *Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications*, pp. 309–322, 2002.

[31] J. Wallerich, H. Dreger, A. Feldmann, B. Krishnamurthy, and W. Willinger, "A methodology for studying persistency aspects of internet flows," *ACM SIGCOMM Computer Communication Review*, vol. 35, no. 2, pp. 23–36, 2005.

[32] A. Pirker and W. Dür, "A quantum network stack and protocols for reliable entanglement-based networks," *New Journal of Physics*, 2019.

[33] W. Kozlowski, A. Dahlberg, and S. Wehner, "Designing a quantum network protocol," in *Proceedings of ACM CoNEXT*, 2020.

[34] R. Van Meter and J. Touch, "Designing quantum repeater networks," *IEEE Communications Magazine*, 2013.

[35] L. Aparicio, R. Van Meter, and H. Esaki, "Protocol design for quantum repeater networks," in *Proceedings of the 7th Asian Internet Engineering Conference*, 2011.

[36] S. Brand, T. Coopmans, and D. Elkouss, "Efficient computation of the waiting time and fidelity in quantum repeater chains," *IEEE Journal on Selected Areas in Communications*, 2020.

[37] K. Chakraborty, D. Elkouss, B. Rijsman, and S. Wehner, "Entanglement distribution in a quantum network: A multicommodity flow-based approach," *IEEE Transactions on Quantum Engineering*, 2020.

[38] A. Chandra, W. Dai, and D. Towsley, "Scheduling quantum teleportation with noisy memories," in *IEEE International Conference on Quantum Computing and Engineering (QCE)*, pp. 437–446, 2022.