Adaptive Entanglement Routing for Quantum Networks with Cutoff

Jiaheng Xiong*, Qiaolun Zhang*‡, Alberto Gatto*, Francesco Musumeci*, Raouf Boutaba†, Massimo Tornatore*
* Politecnico di Milano, Italy † University of Waterloo, Canada
‡ Corresponding author: qiaolun.zhang@mail.polimi.it

***Abstract*—Quantum networks, with applications like Quantum Key Distribution (QKD), are gaining significant attention. However, their implementation faces challenges due to low entanglement generation success rates and quantum decoherence. Recent quantum technology advancements have extended entanglement memory lifetimes to one minute, termed *cutoff*, opening new opportunities for entanglement routing. We propose the Adaptive Entanglement Routing (AER) algorithm, which optimizes resource utilization to improve the success probability of serving entanglement and ultimately reduce the time needed for entanglement establishment. AER includes two phases: 1) determine redundant paths based on load and 2) utilize shared entanglements for entanglement swapping. Moreover, we design the highest-success-path (HSP) algorithm to maximize the success probability of entanglement routing with limited quantum memory. These innovative routing algorithms significantly reduce entanglement request failures, resulting in up to 70% reduction in average waiting times.**

***Index Terms*—Entanglement routing, quantum network, success probability**

I. Introduction

Quantum networks have garnered significant attention in recent years due to their promising applications in quantum teleportation [1], quantum key distribution (QKD) [2], [3], improved sensing [4], etc. In a quantum network, nodes (sources, quantum repeaters, and destination nodes) [5] are interconnected via entangled links. When two nodes need to exchange information, they establish an entanglement connection, occupying one unit of quantum memory in both end nodes of the entanglement [5]. Multiple entanglements can be generated simultaneously between adjacent nodes using wavelength-division multiplexing (WDM) technology. These entanglements can be used also for non-adjacent node connections through *entanglement swapping* [6].

Unlike traditional networks, quantum networks are subject to constraints imposed by quantum mechanics (decoherence effects) [7], enforcing maximum storage time (defined as *cutoff time* [8]) that entanglement can support to ensure the desired quality. Entanglements established earlier are shared and can be used by multiple requests [9]. Our work employs a discrete-time model where time is divided into discrete *time-slots* [8] during which entanglement generation and swapping operations can be completed. Here, the *age* of an entanglement indicates the number of time-slots since its creation. Entanglements exceeding the cutoff time are removed. During swapping, the newly generated entanglement's age is the maximum of the ages of the two original entanglements involved [6].

Another critical issue of quantum networks is that both entanglement generation and swapping may fail due to various factors like device imperfections, noise, and decoherence [6]. These challenges make it hard to create an entanglement with a single attempt, potentially leading to a lengthy waiting time for successful entanglement generation. While some recent works have introduced opportunistic approaches to reduce waiting times for entanglement creation [5], addressing the challenge of extended waiting times, particularly on longer paths due to failure probabilities and inefficient resource utilization, remains unexplored. To address these issues, we aim to efficiently utilize network resources, including free entanglements (entanglements available for any request), while minimizing latency and enhancing overall network performance.

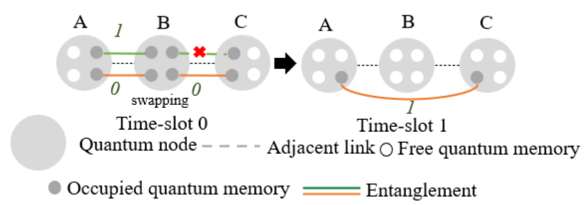


Figure 1. Illustration of entanglement swapping.

Fig. 1 illustrates swapping between nodes A and C using node B as quantum repeater, with a cutoff of 2. We use two paths (green and orange), where green is primary and orange is backup. In time-slot 0, we create a green entanglement between nodes *(A, B)* with an age of 1. In the same slot, we try to establish orange entanglements and a green one between nodes *(B, C)*. Orange entanglements succeed with an age of 0. At the end of time-slot 0, we perform entanglement swapping on node B using the newly created orange entanglements. In time-slot 1, swapping succeeds, resulting in a new orange entanglement between nodes *(A, C)* with an age of 1. Simultaneously, the green entanglement between nodes *(A, B)* is removed as its age reaches the cutoff of 2.

Main technical contributions of this work are as follows. (i) We investigated, for the first time to the best of our knowledge, how to adaptively utilize redundant entanglements in different time-slots to reduce the latency of establishing entanglements in quantum networks with cutoff. (ii) We devise an Adaptive Entanglement Routing (AER) algorithm and a highest-success-path (HSP) algorithm to adaptively utilize redundant resources for entanglement generation and swapping.

The rest of the paper is organized as follows. Section II discusses related work. Section III formally presents the problem statement, and Section IV describes our proposed routing algorithm to solve the problem. Section V presents numerical results. Finally, we conclude the paper in Section VI.

II. Related Work

In the field of quantum networks, various entanglement routing algorithms have been proposed. Due to the short time duration of one entanglement, lots of work consider to solve the entanglement routing problems with only one time-slot [10], [11]. With recent advancements in quantum technologies, entanglements can last for 1 min [12], [13]. Consequently, researchers have turned their attention towards investigating the discrete-time quantum network model as a means of capitalizing on pre-established, longer-lasting entanglement connections [14], [15]. This shift in focus holds the potential to enhance the success rate of establishing entanglements and concurrently reduce latency in their establishment [15]. For instance, two algorithms Q-PASS and Q-CAST, are proposed in [15] to identify optimal routes in a quantum network in order to reduce latency. Moreover, the opportunistic routing strategy is investigated to reduce the latency by adapting the waiting time to perform entanglement swapping in [5]. It is worth noting that there has been no investigation into adaptively serving requests based on the current network resource occupation. This work is the first to investigate adaptively utilizing changing redundant entanglements in different time-slots to reduce latency in quantum networks.

III. Adaptive Entanglement Routing Problem

A. System Model and Problem Statement

The quantum network, represented as graph $G = (V, E)$ with nodes in V and links in E, allocates finite memory units for entanglement creation. Requests are processed within time-slots, with entanglements having a limited lifespan (cutoff) before they expire. The age of an entanglement represents the time-slots elapsed since its creation. Entanglement routing involves two phases, namely, generation and swapping, with success probabilities p_e and p_s, respectively. Considering that node memory is much smaller than channel capacity (e.g., up to 30 channels per link), no channel number constraints are imposed.

The Adaptive Entanglement Routing problem is defined as follows: **Given** network topology, node memory, cutoff, and generation/swapping success probabilities, **determine** redundant entanglement generation and routing (with entanglement swapping on all paths for requests), **subject to** (1) *quantum memory capacity*, (2) *generation constraint* (quantum link attempts entanglement once per time-slot), and (3) *cutoff time* (age of new entanglement equals to the maximum age of the swapped entanglements). The **objective** is to minimize the average *waiting time*, defined as the time-slots from request generation to fulfillment.

B. Illustrative Example of Adaptive Entanglement Routing

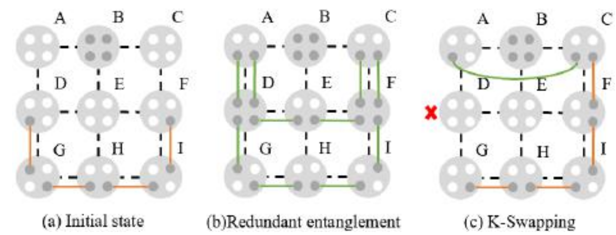


Figure 2. Procedure of Adaptive Entanglement Routing with 2 paths (k=2).

To provide redundant resources, we introduce an adaptive redundancy factor k for entanglement routing. Specifically, entanglement generation and swapping aim to find k redundant paths per request, adapting in function of network conditions.

Fig. 2 illustrates an example of redundant routing in a quantum network (k=2). Assume that we have a request to establish entanglement between node pair *(A, C)*. The used and unused quantum memories are denoted with filled and hollow circles, respectively. Before serving the request, the network has several free entanglements (marked with orange links), and node B is unavailable since all memories are occupied as in Fig. 2 (a). To serve the request, two paths are identified (marked with green links) as in Fig. 2(b). The first path (*A, D, E, F, C*) succeeds in establishing entanglements between node pair (A, C) while the second path (*A, D, G, H, I, F, C*) fails to establish entanglement due to failure of entanglement swapping in node D as shown in Fig. 2 (c). In addition, the remaining entanglements in the second path are marked with an orange line. Even if the second path fails, the entanglement between A and C is established through the other path. Our AER algorithm enhances entanglement success by assigning multiple paths and efficiently using free entanglements to overcome generation failure.

IV. Adaptive Entanglement Routing Algorithm

In this section, we present the details of the proposed *AER* algorithm and the *HSP* algorithm, which can enhance the probability of entanglement establishment and reduce the average waiting time.

A. Auxiliary Graph Model

We create an auxiliary graph for entanglement routing in our quantum network, including free entanglements and adjacent links formed by establishing entanglements between neighboring nodes at the current time-slot. In Fig. 2(a), the green links represent entanglements between nodes A and C, dedicated to serving a particular request and inaccessible for others. Free entanglements are in orange, and adjacent links are grey. This auxiliary graph retains only idle entanglements available during routing, including free entanglements and adjacent links, while excluding entanglements assigned to specific requests (non-free entanglements). In Fig.2, we utilize existing free entanglements (orange links) from Fig.2(a), marked as green links in Fig. 2(b). Additionally, we use adjacent links (also marked as green in Fig. 2(b)) between node pairs (A, D), (D, E), (E, F), and (C, F) for routing.

The algorithm utilizes the availability status of each node represented in the auxiliary graph to perform the HSP algorithm. The different nodes can be distinguished based on the current state and assigned to one of the following classes:

- Available: The node is available for use in the path search process, regardless of the direction the path takes.
- Conditionally available: The node is available in certain directions during the path search process. The algorithm will record the unavailable combinations of this node and each neighbor as the availability conditions of the node.
- Unavailable: The node is unavailable for use in the path search process due to resource exhaustion.

For conditionally available quantum nodes, we initially determine their unavailability conditions and store this information in the attributes of the respective nodes.

Algorithm 1: AER Algorithm

```
Data: Request Set N, Network Topology, p_e, p_s
Result: Served request, Average waiting time
1  initialize Request Set N, Network Topology, p_e, p_s;
2  while any request is not served in N do
3      k = 1;
4      while any request has available path do
5          for N do
6              HSP(G, source, target) and build external links;
7          k = k + 1;
8      for N do
9          for k do
10             swapping;
11     time-slot = time-slot + 1;
12 calculate the average waiting time;
```

B. Weight Assignment Policies

The goal of our weight assignment strategy in this context is to maximize the success probability of each route. This is achieved by assigning weights that take into account the entanglement generation and swapping probabilities.

For a given path $s_i = n_1 \rightarrow n_2 \rightarrow n_3 \cdots \rightarrow n_k$ with i free entanglements $(i < k)$, we calculate the success probability as $p_s^{k-2} p_e^{k-1-i}$, where p_s^{k-2} is the success probability of entanglement swapping along the path and p_e^{k-1-i} is the success probability of creating $k-1-i$ entanglements along the path. For instance, in Fig. 2(a), we calculate the success probability for the paths (A, D, E, F, C) and (A, D, G, H, I, F, C) for the request (A, C) as (1). $P_1 = p_s^3 p_e^4$ and (2). $P_2 = p_s^5 p_e^2$, respectively.

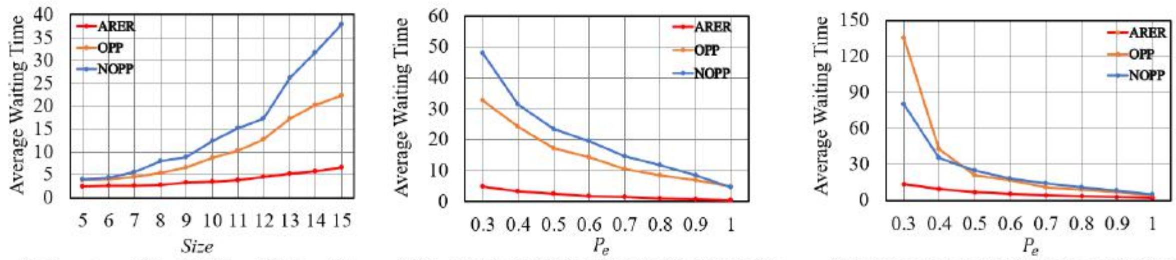
Specifically, we use the logarithmic operation to transform the multiplicative probability calculations into additive cost calculations, which is required by algorithms based on Dijkstra's algorithm. If a node performs swapping, it incurs a cost of $-\ln p_s$, which is incorporated into the overall cost, equivalent to $-\frac{1}{2}\ln p_s$, associated with the cost of the two links for swapping. If neither of the end nodes of a free entanglement is the same as the end nodes of a request, its weight is set to $-\ln p_s$ since both of the end nodes of the free entanglement are used for swapping; if one of the end nodes of a free entanglement is the same as the end nodes of a request, the weight is $-\frac{1}{2}\ln p_s$ since only one end node of the free entanglement is used for swapping. For adjacent links without end nodes, the weight is $-\ln(p_e) - \ln(p_s)$, while those with end nodes carry a weight of $-\ln(p_e) - \frac{1}{2}\ln p_s$. Thus, the weight of the paths in Fig. 2 (path *(A, D, E, F, C)* and *(A, D, G, H, I, F, C)*) can be calculated as $W_{P_1} = -\ln p_s^3 p_e^4$ and $W_{P_2} = -\ln p_s^5 p_e^2$. Even if the path (A, D, G, H, I, F, C) is longer, the success probability is equal. Since the weight of a path is negatively correlated with the success probability, the path with minimum cost is the path with the highest success probability. Finally, to alleviate local congestion, we incorporate load balancing by increasing adjacent link weights by 0.001 before the HSP stage, favoring lower load paths.

Algorithm 2: HSP Algorithm

```
Input: Graph G, source node source, target node target
Output: Optimal path
Function HSP(G, source, target)
    Initialize open set with source, visited set(VS), visited
      with conditions set(VCS);
    while open set not empty do
        Pop current node and path with lowest cost from
          open set;
        if current node not in VS then
            Add current node to VS;
            if current node is not unavailable then
                if current node is neither source nor target
                  then
                    for each neighbor do
                        if neighbor is conditional available
                          then
                            prev_node = current_path[-1];
                            if prev_node not in VCS then
                                Add current node,
                                  predecessor, successor to
                                  VCS;
                                Remove current node from
                                  VS;
                        Calculate minimum edge cost;
                        Update nodes, cost, path;
                        Push updated info to open set;
                else if current node is source then
                    for each neighbor do
                        if availability is False then
                            Continue loop;
                        Calculate minimum edge cost;
                        Update cost, steps, path;
                        Push updated info to open set;
                else if current node is target then
                    if availability is True then
                        return path including target;
    return None;
```

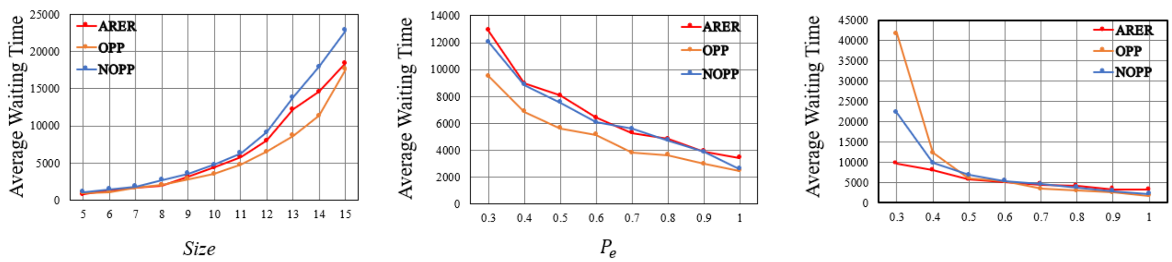
C. Overall Procedures of AER Algorithm

Algorithm 1 outlines the overall process of our AER algorithm. In each time-slot, we start with k=1 (line 1) and find the available min-cost path for each request. We increment



(a) $p_e = p_s = 0.8, cutoff = 30, N = 20$ (b) $p_s = 0.8, cutoff = 30, N = 20, size = 10$ (c) (b) $p_s = 0.8, cutoff = 6, N = 20, size = 10$

Figure 3. Average waiting time for different scenarios



(a) $p_e = p_s = 0.8, cutoff = 30, N = 20$ (b) $p_s = 0.8, cutoff = 30, N = 20, size = 10$ (c) (b) $p_s = 0.8, cutoff = 6, N = 20, size = 10$

Figure 4. Total memory usage for different scenarios

k in each iteration until a request can't find an available path, obtaining the initial redundancy factor (lines 4-7). During swapping, we identify k min-cost paths, comprising only free entanglements for each request, and execute simultaneous swapping operations (lines 8-10). If multiple paths successfully establish entanglements for a request, one is chosen to fulfill the request, while the others become free entanglements for the next time-slot.

By adaptively adjusting the redundancy factor and based on current network resource availability to determine the number of redundant paths, we aim to maximize network resource utilization. If all requests cannot find an available path, we consider the network to be overloaded and thus stop the iteration process.

D. HSP Algorithm for Resilient Entanglement Routing

Algorithm 2, the HSP algorithm, is modified from Dijkstra's algorithm by considering the neighbor availability (defined in Sec. IV. A). The HSP algorithm can be summarized as follows: It begins by initializing three sets, namely, open set (record the nodes to visit and current path), visited set (VS, record the visited nodes), and visited nodes with conditions (VCS, record the visited nodes under specific predecessor and successor nodes) (line 1). Note that we maintain a VCS for every node because the availability conditions of different nodes are different. Then, it iteratively processes nodes based on whether they are the source, target, or intermediate node (line 8, 18, 25). For intermediate nodes, it checks neighbor availability and validates conditions. If conditions are met, it moves the node and the predecessor to VCS (line 13). If the predecessor has not been recorded in VCS, the current node will be removed from visited set (line 14) because it can not be used under current predecessor, successor combination. For the source node, it checks neighbor availability and updates related data if the node is available (line 18-24). If the neighbor is in the availability conditions, HSP will continue to find the next neighbor since the combination of the current node and this neighbor is unavailable (line 21). For the target node, it checks if the node is available and returns the path if conditions are met or if the node is available (line 25-27). The algorithm stops the iteration until the open set is empty or the target node is reached.

V. Illustrative Numerical Results

We perform our numerical evaluations on a set of MxM manhattan topology network with different values of M as in Ref. [5]. All the requests are randomly generated between node pairs in the first time-slot, and each simulation episode ends when all the requests are served. We compare our *AER* algorithm with *OPP* (Opportunistic Entanglement Routing) [5] and *NOPP* (Non-Opportunistic Entanglement Routing) [16]. *OPP* attempts to swap along the selected path when k consecutive edges have available entanglements, even if not all entanglements along the path are available. Since [5] demonstrated that *OPP* achieves the lowest waiting time when k equals 1, we assume $k = 1$ in our work to have a more compelling baseline than adopting other values for k. In contrast, *NOPP* waits until all entanglements along the path are available for swapping. We evaluate these algorithms in three scenarios: network size, probability of entanglement generation, and cutoff time. Node memory size is fixed at 6, and the number of requests N is set to 20 in all scenarios. For network size evaluation, we set the success probability of entanglement generation (p_e) and entanglement swapping (p_s) to 0.8. When examining the impact of generation probability and cutoff time, we fix p_s at 1 and the network size at 5. When assessing generation probability, the cutoff is set to 30, while for a low cutoff time evaluation, the cutoff is set to 6. All results are averaged over 100 problem instances.

Impact of network size. Fig. 3(a) shows the average waiting time of our proposed *AER* algorithm for manhattan topology networks when increasing the network size. Results show that *AER* significantly outperforms all the baseline algorithms. Specifically, we reduced the average waiting time by 15%-70%, and, for larger network size, the performance of the proposed *AER* algorithm compared to the previous methods becomes even more effective, as the number of available redundant paths for a single request also increases.

Impact of success probability of entanglement generation. As shown in Fig. 3(b), when varying the success probability of entanglement generation, *AER* still significantly outperforms OPP, showing that redundant paths mitigate the impact of failures of entanglement generation. This validates that the *AER* algorithm, modulated by an adaptive redundancy factor, capitalizes on the existing free entanglements within the network and circumvents the failure probability of external links.

Impact of cutoff time. Finally, we compare the performance of the two methods in a more dynamic network in Fig. 3(c), where the cutoff time is set to 6 rather than 30 as in Fig. 3(b). With a smaller cutoff, entanglement lifetimes decrease, requiring more entanglements to be established in each time-slot for low waiting times. In Fig. 3(c), as p_e decreases, *AER* maintains a significant advantage over OPP, even with the reduced cutoff time. At lower success probabilities, OPP performs worse than NOPP because it initiates swapping earlier. Due to the age constraint in swapping, newly created entanglements after swapping match the age of the oldest participating entanglement, leading to more entanglements reaching the cutoff and being removed. This effect is more pronounced at lower success probabilities.

We compare memory usage among algorithms in Fig. 4. AER consumes at most 40% additional memory compared to OPP due to its redundancy mechanism but usually stays within 15% for most network sizes. With a larger cutoff time (Fig.4(b)), AER's memory consumption becomes comparable to NOPP and up to 40% higher than OPP. In Fig.4(c) with a smaller cutoff time, AER uses about the same resources as OPP. However, with low p_e, AER's memory usage can be up to 77% and 57% lower than OPP and NOPP, respectively. In summary, when P_e is small, AER outperforms OPP and NOPP in both waiting times and memory efficiency.

VI. Conclusion and Future Work

We proposed an adaptive redundant routing algorithm for quantum networks, which adaptively uses redundant entanglements to improve the success probability of creating entanglements and, ultimately the latency. To efficiently exploit existing idle resources, we developed an adaptive entanglement routing (AER) algorithm. Moreover, the *AER* algorithm utilizes our devised novel highest-success-path (HSP) algorithm based on Dijkstra's algorithm, which can find the path with the highest success probability of creating entanglements considering the constraint of limited quantum memories. Numerical results show that our proposed *AER* algorithm reduces the average waiting time by up to 70%. In future work, we plan to develop an Integer Linear Programming (ILP) model for the adaptive entanglement routing problem and evaluate the optimality gap of the proposed *AER* algorithm. Moreover, we will extend the existing quantum computers to evaluate the proposed approach.

Acknowledgment

This work was supported by the Italian Ministry of University and Research (MUR) and the European Union (EU) under the PON/REACT project and also partly funded by MUR PON2014_2020 "QUANCOM" Project (MIUR ARS01_00734).

References

[1] S. Shi, Y. Wang, L. Tian, W. Li, Y. Wu, Q. Wang, Y. Zheng, and K. Peng, "Continuous variable quantum teleportation network," *Laser & Photonics Reviews*, vol. 17, no. 2, p. 2200508, 2023.

[2] P. K. Tysowski, X. Ling, N. Lütkenhaus, and M. Mosca, "The engineering of a scalable multi-site communications system utilizing quantum key distribution (qkd)," *Quantum Science and Technology*, vol. 3, no. 2, p. 024001, 2018.

[3] Q. Zhang, O. Ayoub, A. Gatto, J. Wu, F. Musumeci, and M. Tornatore, "Routing, channel, key-rate and time-slot assignment for qkd in optical networks," *IEEE Transactions on Network and Service Management*, 2023, early access.

[4] C. L. Degen, F. Reinhard, and P. Cappellaro, "Quantum sensing," *Reviews of modern physics*, vol. 89, no. 3, p. 035002, 2017.

[5] A. Farahbakhsh and C. Feng, "Opportunistic routing in quantum networks," in *IEEE Conference on Computer Communications*, 2022.

[6] Á. G. Iñesta, G. Vardoyan, L. Scavuzzo, and S. Wehner, "Optimal entanglement distribution policies in homogeneous repeater chains with cutoffs," *arXiv preprint arXiv:2207.06533*, 2022.

[7] M. Mosca, "Quantum algorithms," *arXiv preprint arXiv:0808.0369*, 2008.

[8] Á. Iñesta, G. Vardoyan, L. Scavuzzo, and S. Wehner, "Optimal entanglement distribution policies in homogeneous repeater chains with cutoffs," *Bulletin of the American Physical Society*, 2023.

[9] 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*, vol. 5, no. 1, p. 25, 2019.

[10] N. K. Panigrahy, T. Vasantam, D. Towsley, and L. Tassiulas, "On the capacity region of a quantum switch with entanglement purification," *arXiv preprint arXiv:2212.01463*, 2022.

[11] Y. Zhao and C. Qiao, "Redundant entanglement provisioning and selection for throughput maximization in quantum networks," in *IEEE Conference on Computer Communications*, 2021.

[12] G. Heinze, C. Hubrich, and T. Halfmann, "Stopped light and image storage by electromagnetically induced transparency up to the regime of one minute," *Physical review letters*, vol. 111, no. 3, p. 033601, 2013.

[13] 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*, vol. 12, no. 1, p. 2381, 2021.

[14] K. Chakraborty, F. Rozpedek, A. Dahlberg, and S. Wehner, "Distributed routing in a quantum internet," *arXiv preprint arXiv:1907.11630*, 2019.

[15] S. Shi and C. Qian, "Concurrent entanglement routing for quantum networks," in *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*. ACM, jul 2020.

[16] E. Schoute, L. Mancinska, T. Islam, I. Kerenidis, and S. Wehner, "Shortcuts to quantum network routing," *arXiv preprint arXiv:1610.05238*, 2016.