Socially-Aware Opportunistic Routing With Path Segment Selection in Quantum Networks

Shao-Min Huang[§], Ming-Huang Chien[§], Cheng-Yang Cheng[§],
Ting-Yuan Wen[‖], Qian-Jing Wang[‖], and Jian-Jhih Kuo[*]
Dept. of Computer Science and Information Engineering, National Chung Cheng University, Chiayi, Taiwan

***Abstract*—The conventional quantum teleportation schemes enable high-security network communications by establishing end-to-end entangled paths. However, those schemes focus on time synchronization and thus cause lots of idle time. Recent research suggests adopting an opportunistic scheme to forward data qubits as far as it can. However, this scheme lacks security since data qubits may be stored at malicious repeaters, which may peek at, destroy, or fake the data qubits. To this end, we design a new scheme called SOAR that considers trusted repeaters via social networks. Moreover, SOAR promotes the parallelism of swapping operations and thus leads to a less idle time of network resources than the other existing schemes. Furthermore, we design an algorithm called SAGE that can best fit SOAR by linking multiple subpaths via appropriate trusted repeaters to get an ideal path and augmenting least-hop paths to utilize the resources in quantum networks better. Simulation results manifest that SOAR outperforms the other schemes by 54%-89%; SAGE outperforms the other routing algorithms by 50% on average on SOAR.**

***Index Terms*—quantum network, entanglement routing, opportunistic routing, trust repeaters**

I. Introduction

Quantum networks (QNs) have been validated and realized to transmit information to facilitate technical frontier development [1]. A QN consists of quantum nodes, and any two *adjacent* nodes are interconnected with a bunch of optical fibers (i.e., *quantum channels*) [2]. Each node can be a source, repeater, or destination on a routing path. The smallest transmission unit is a quantum bit (i.e., *qubit*), and each node has quantum memory to store qubits for a limited period [3]. A data qubit (i.e., blue dots in Fig. 1) can be teleported via an entangled path which may be constructed with some entangled links and swapping links. The technology is called *quantum teleportation*, and an entangled path can teleport only one qubit and will destroy after teleportation [4]. An entangled link (i.e., blue solid lines in Fig. 1) can be constructed by adjacent nodes that are connected by quantum channels (i.e., blue dash lines in Fig. 1) and will occupy one quantum memory on both nodes and a quantum channel. Swapping links (i.e., red solid lines in Fig. 1) constructed at a node can glue two entangled paths (or entangled links) into a longer entangled path.[1] However, constructing entangle links and swapping links may fail and thus it is not easy to generate a long entangled path [5]–[9].

The other method different from creating swapping links is *store-and-forward*. By this method, the data qubit can be

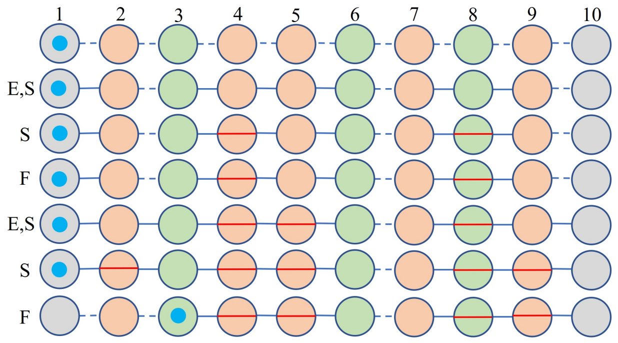


Fig. 1. A routing example for SOAR (E: Entangle, S: Swap, F: Forward).

teleported by entangled path, temporarily stored at repeater, and then transmitted to another repeater or destination [1]. For example, in Fig. 1, there is a data qubit at node 1 needed to be sent to node 10. At first, we try to construct the whole end-to-end path from node 1 to node 10. Unfortunately, some entanglement and swapping operations failed. At least the good thing is that the path from node 1 to node 3 is constructed successfully, so we can teleport the data qubit to node 3 and temporarily store it at node 3. Also, we can use the entangled path from node 3 to node 6 to send the qubit further in later time slots. However, storing data qubits at nodes will occupy nodes' quantum memory, possibly blocking the preferred paths of other requests. Hence, it is better not to transmit data one hop by one hop and must construct some swapping links at some nodes. Besides, an entangled link may decohere after its lifetime (e.g., 1.46 sec [10]), so the swapping process needs to be finished successfully before the entangled link is outdated.

Up to now, many quantum forwarding schemes have been developed to provide efficient quantum routing [5], [10], [11]. Although they may have a little difference, a typical conventional quantum forwarding scheme (CON) has the following three properties. 1) Time is divided into discrete *periods*, and a period is usually set to the lifetime of an entangled link. Thus, the entangled links and entangled paths will be flushed as a period ends for time synchronization. 2) CON attempts to create entangled links on paths for requests once in parallel for each period and swap sequentially in each period. 3) A data qubit is forwarded only after the complete entangled path is created (i.e., end-to-end teleportation) in each period. In summary, most existing routing algorithms are designed based on CON. However, CON transmits data qubits conservatively, causing inefficient network resource utilization [12].

Pioneering research [12] proposed an *opportunistic* quan-

This work was supported by the National Science and Technology Council, Taiwan, under Grant NSTC 111-2628-E-194-001-MY3.

[§][‖]: equal contributions; [*]: corresponding author (lajacky@cs.ccu.edu.tw)

[1]To avoid ambiguity, a long connection created by gluing multiple entangled links together via swapping is called an "entangled path."

tum forwarding scheme, OPP, to exploit a store-and-forward method to utilize network resources better. Different from CON, OPP has the following revolutions. 1) Time is divided into *shorter* discrete periods. For ease of reading, we term them as *time slots* in the rest of the paper. Each time slot is set to the total time of once entangling plus once swapping. Since the lifetime of an entangled link can span several time slots, OPP does not flush entangled links or paths. Thus, they can avoid unnecessary resource waste owing to OPP's shorter period design (i.e., time slot). 2) OPP initiates entangling and then swapping at a time slot. Swapping no longer waits for all the entangled links on a path to be created. 3) OPP forwards every data qubit as soon as possible even if only an entangled link on the path is created. Overall, compared to CON, OPP reduces the average waiting time effectively [12].

Even so, storing data qubits at arbitrary nodes is insecure since malicious nodes may peek at, destroy, or fake the data qubits [13], [14], especially in future large-scale QNs [10], [15]. To avoid this issue, this paper makes the first attempt to choose suitable repeaters based on trust relationships between quantum node owners in social networks (SNs). In SNs, any two nodes interconnected with an edge trust each other and will not peek at each other's data qubit. For data security, sources select only their trusted repeaters to store data qubits temporarily during data transmission. For example, if the owner of source node s is connected with the owner of a node c in SNs, then node s is willing to temporarily store its data qubit at node c when sending its data qubit toward node d in QNs since s's owner trusts c's owner.

However, *socially-aware opportunistic quantum forwarding* has not been proposed to solve the problem. Therefore, we make the first attempt to design a socially-aware opportunistic quantum forwarding scheme as well as a socially-aware quantum routing algorithm for QNs, which raises three main research challenges as follows. 1) *More efficient mechanism.* Although OPP has more entanglement chances than CON, it cuts the chance of swapping in half. The lack of swapping may waste constructed entangled links. Thus, OPP encourages data qubits to be stored and forwarded by more repeaters. Nevertheless, due to the security issue, OPP could not use every node as an intermediate repeater. Therefore, entangled links will expire before being used, and we need a more efficient mechanism to improve the utilization of entangled links. 2) *Better routing metric*. CON flushes the entangled links and entangled paths created in the previous period, and thus the available probability of an entangled link is easy to acquire. However, entangled links created can remain until time out in the opportunistic scheme. The properties make it challenging to predict the availability of an entangled link precisely for path selection. 3) *Socially-aware path selection.* The existing quantum routing algorithms do not consider trust relationships between nodes. They may select routing paths with a few trusted repeaters for temporarily storing data qubits and cannot fully benefit from a socially-aware opportunistic quantum forwarding scheme. In contrast, painstakingly detouring to include more trusted repeaters may increase the waiting time conversely (e.g., zigzag paths). Thus, socially-aware path selection is challenging since it has to decide the numbers and the positions of trusted repeaters on the paths.

To address the above challenges, we present a novel quantum forwarding scheme termed **S**ocially-aware **O**pportunistic **A**ggressive **R**outing (SOAR). SOAR considers trust relationships between nodes in the SN to avoid storing data qubits at untrusted repeaters. Meanwhile, SOAR innovates *aggressive quantum forwarding* (AGG) (detailed later) to impose an aggressive swapping process during the entangling process. To optimize SOAR's performance, we design a new quantum routing algorithm called **S**ocially-aware P**a**th Se**g**ment S**e**lection (SAGE). SAGE can precisely calculate each entangled link's available probability in the long term to estimate each path's quality cleverly. Then, it iteratively selects suitable subpaths to form a routing path with an appropriate number and positions of trusted repeaters for possibly storing data qubits.

Our contributions are summarized as follow.

1) We consider trust relationships between node owners in QNs to propose a new socially-aware opportunistic quantum forwarding scheme, SOAR, to make sure the safety of data qubits, while improving the routing efficiency.
2) We design a novel socially-aware path selection algorithm, SAGE, to trade off the waiting time and the number and positions of trusted repeaters to fit SOAR perfectly. To this end, we derive a formula for estimating the available probability of an entangled link and also convert it into a more useful additive weight to predict path quality and benefit path selection for SAGE.

II. Background & System Model

In this section, we introduce the network components, review CON and OPP, and then propose our scheme SOAR.[2]

A. Network Components

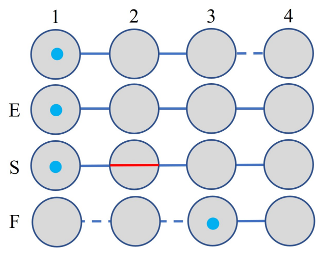
A QN consists of several quantum nodes, every node has limited quantum memory and connects to its neighboring nodes with a limited number of quantum channels [2]. Quantum memory can temporarily store a data qubit for R time slots, where R ranges from seconds to minutes [3].

An entangled link between two adjacent nodes is created by a pair of entangled qubits on a quantum channel, as shown by the blue lines in Fig. 1. The success probability of creating an entangled link will decrease exponentially with the distance of the channel [5], [8], [17], i.e., $\mathcal{P}(u,v) = e^{-\alpha \cdot l(u,v)}$, where $l(u,v)$ is the quantum channel's length between nodes u, v, and α is a given constant based on the optical fiber's material [3]. An entangled link has a certain lifetime (i.e., link decoherence time), and we let L denote the number of time slots an entangled link can remain after being created.

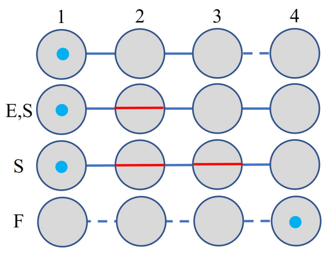
Furthermore, a swapping link that is generated by entanglement swapping can glue two entangled links (or paths) into an entangled path. Fig. 1 shows a red swapping link glues two entangled links into an entangled path. Swapping at node u is also associated with a success probability, denoted by $\mathcal{Q}(u)$.

Moreover, each node belongs to a specific owner. An owner may use its node s to send a data qubit to another node d

[2]Due to the page limit, the related work is placed in Appendix A of [16].



(a) OPP



(b) AGG

Fig. 2. Illustrations of OPP and AGG.

in QNs as a request. Additionally, each owner trusts a subset of owners. Storing data qubits at trusted owners' nodes is *secure* in QNs since the trusted repeater will not peek at, destroy, or fake data qubits [13], [14]. In order to coordinate all nodes and execute phases, the information of SNs and QNs including entangling and swapping results is periodically collected by a central controller. It is reasonable since the one-way propagation delay takes at most tens of milliseconds [18].

B. Prior Work, Motivation, and New Design

We review the opportunistic quantum forwarding scheme OPP proposed in [12] from various perspectives, point out its shortcomings, and then discuss the rationale behind SOAR.

1) Parallelism: Subsequently, we examine the *parallelism* of OPP which consists of two phases, Path Selection Phase and Opportunistic Forwarding Phase. The former receives requests, gets paths by any routing algorithm, and gives the paths to the latter. The latter contains two steps in a time slot: 1) entangle and 2) swap and forward, and both steps take one *time step*.[3] For example, assume that we have a path with two entangled links created and one not yet created as shown in Fig. 2(a). OPP initiates the entangling of the rightmost links, and then all entangled links are available after the first time step (E). Then, OPP initiates the swapping at node 2 at the second time step (S), and the swapping succeeds. However, we can observe that node 2's entangled links were already available at the first time step, but OPP wasted the chance to execute node 2's swapping. To this end, we modify the Opportunistic Forwarding Phase as the Aggressive Forwarding Phase to derive a new forwarding scheme termed the aggressive quantum forwarding (AGG) AGG imposes *aggressive swapping* during the entangle step, if possible. Take Fig. 2(b) for example, node 2 can initiate swapping at the first time step, originally for entangling only in OPP, and thus the data qubit can move further when the forwarding is initiated. It is obvious that the modified phase tends to perform better than the original one.

Next, we study the time granularity of CON, OPP, and AGG. For simplicity, assume the routing algorithm's running time is negligible. Then, the lengths of a period, a time slot, and a time step are shown in Fig. 3. For CON, the entangle step (i.e., the blue block in the bar) can initiate once entangling process at the beginning of a period, and the remaining time is for the swapping process (i.e., the yellow part in the bar). As the period ends, CON flushes all the resources. For time synchronization,

[3]In this paper, we use the three time terms, period, time slot, and time step. Note that they are not equal to each other and will be detailed in Fig. 3.

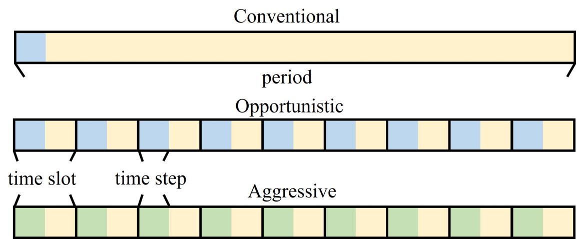


Fig. 3. Parallelism comparison of CON, OPP, and AGG (adopted by SOAR).

CON sets a period to a long time (e.g., the lifetime of entangled links) to ensure that nodes can complete swapping. Unlike CON, OPP divides time into small time slots, each of which contains the entangle step and swapping step, as shown in Fig. 3. Clearly, OPP has about half time for swapping but gains much more chances for entangling than CON. In contrast, AGG perfectly doubles the swapping chances by imposing aggressive swapping during entangle step and making them run in parallel (i.e., a green block is a combination of a blue block and a yellow block in Fig. 3).

From the theoretical perspective, it can be proved that AGG outperforms CON on average in the following theorem. The proof is not difficult because AGG has more chances to initiate entangling and swapping processes than CON. Due to the page limit, the detailed proof is provided in Appendix B in [16].

Theorem 1. Assume that the routing algorithm's running time is negligible. Then, with AGG, the available probability of any entangled path p is greater than that with CON, during a period (i.e., L time slots).

2) Security vigilance: Finally, we observe the *security vigilance* of OPP to check how seriously OPP may forward data qubits to malicious repeaters. Unfortunately, OPP may forward a data qubit to any repeater by chance. However, if the data qubit is stored at any malicious repeater (i.e., orange circles in Fig. 1), it may be peeked at, destroyed, or faked by the malicious repeater [13]. To this end, we make the first attempt to consider trust relationships between node owners in SNs to ensure that data qubits are always stored temporarily by only trusted repeaters (i.e., green circles in Fig. 1).

In summary, we find OPP has the following shortcomings:

1) OPP has only half the time for swapping as CON.
2) OPP lacks parallelism for entangling and swapping processes and may idle network resources.
3) OPP puts data qubits at risk because it may store them at malicious repeaters by chance during teleportation.

C. SOAR in a Nutshell

To overcome OPP's shortcomings, we design a new quantum forwarding scheme termed **S**ocially-aware **O**pportunistic **A**ggressive **R**outing (SOAR), which 1) adopts AGG and 2) stores data qubits on trusted repeaters only, according to the observations and insights in Section II-B. We assume that SOAR uses a central controller to manage QN in a time-synchronous manner. Specifically, time is divided into time slots, and each time slot contains two time steps. The controller periodically performs the two phases, Path Selection Phase and Aggressive Forwarding Phase, at each time slot. Then, each phase consists of two steps (i.e., P1-1, P1-2, P2-1, and P2-2). For ease of presentation, we assume that the running time for P1-1 and P1-2 is negligible,[4] and each of steps P2-1 and P2-2 takes a time step. The four steps are briefly described as follows:

1) P1-1 – Receive requests: SOAR receives the generated routing requests from the users in the QN.

2) P1-2 – Get paths by any routing algorithm: SOAR calls a quantum routing algorithm to select the paths for each routing request based on the state of network resources. Then, SOAR reserves the network resources required by the paths.

3) P2-1 – Entangle with aggressive swap: SOAR attempts to create the entangled links on the paths for each routing request. Meanwhile, SOAR initiates the swapping operations at the nodes to glue two adjacent entangled links (created at the previous time slots) on the same path into a (longer) entangled path by swapping, if possible.

4) P2-2 – Swap and forward: SOAR initiates the swapping process again to create (longer) entangled paths. After that, SOAR forwards each request's data qubit if the current entangled path can connect to a trusted repeater (with sufficient storage) or the destination. Then, it releases useless resources.

Example. Fig. 1 illustrates an example of SOAR. Assume after P1-1 and P1-2, SOAR gets a routing path with nine links (not created yet) and three trusted repeaters marked as green circles. At the first time slot, SOAR executes P2-1 (i.e., E, S) to initiate the entangling process and obtains six entangled links $(1,2)$, $(3,4)$, $(4,5)$, $(5,6)$, $(7,8)$, and $(8,9)$. Note that P2-1 cannot initiate the swapping process since there is no created entangled link initially. Then, SOAR executes P2-2 (i.e., S) to initiate the swapping process to get two entangled paths from node 3 to 5 and from 7 to 9. When the first time slot ends, SOAR does not forward the data qubit since node 2 is not a trusted repeater.

At the second time slot, SOAR executes P2-1 to acquire entangled links $(2,3)$ and $(9,10)$. Meanwhile, SOAR lengthens the entangled path from node 3 to 6. SOAR goes to P2-2 to get one entangled path from 1 to 3 and the other from node 7 to 10. When the second time slot ends, SOAR forwards the data qubit to node 3 if node 3 has sufficient quantum memory for temporary storage. It is secure since node 3 is a trusted repeater for node 1. It is worth noting that if node 3 has no sufficient memory for storing the data qubit, it will initiate swapping at the next time slot to help the data qubit bypass it. ■

Remark that SOAR can also handle the cases with multiple paths for each routing pair. For example, assume there are two paths from the source to the destination. SOAR forwards the data qubit along the entangled path which is ready to use. If there are two or more created entangled subpaths (or paths), then SOAR will select the one closer to the destination and release the other paths' resources.

[4]In practice, the time slot length should include the adopted routing algorithm's running time for P1-1 and P1-2. Thus, the performance may vary based on the adopted routing algorithm. Later, we will show that our algorithm SAGE can run much faster than the existing ones in Section IV.

III. Algorithm Design — SAGE

In this section, we first derive the new *additive* metric to measure the quality of a link and a path in Section III-A. Then, we propose the algorithm SAGE to fit SOAR in Section III-B.

A. The Design of a Better Routing Metric for SOAR

Before selecting paths for requests, SAGE requires a good routing metric to measure the quality of a link and a path. Thus, we first derive and introduce the available probability of an entangled link at a time slot in the long term. Then, SAGE assigns every link a weight, representing the link quality, by converting the link's available probability. Later, SAGE identifies efficient paths by comparing the total weight of links on each path. For ease of reading, all the detailed proofs are presented in Appendix C to E in [16].

Suppose an entangled link can remain L slots. Let $l(u,v)$ be the channel length between nodes u, v. Then, link (u,v) has the success probability of entangling, $\mathcal{P}(u,v) = e^{-\alpha \cdot l(u,v)}$. Then, we derive a formula to predict the availability of a link.

Theorem 2. The available probability of a link (u,v) at an arbitrary time slot in the long term is $\mathcal{P}_g(u,v) = \frac{L\cdot\mathcal{P}(u,v)}{L\cdot\mathcal{P}(u,v)-\mathcal{P}(u,v)+1}$. (1)

After deriving the available probability of a single link, we can further estimate the probability of a path where all links are available simultaneously at a time slot, if the failure probability of swapping is negligible and all nodes have sufficient memory.

Lemma 1. Suppose $\mathcal{Q}(u) = 1$ for each node u and all nodes have sufficient quantum memory. Then, an entangled path p has the available probability $\mathcal{P}(p) = \prod_{(u,v)\in p} \mathcal{P}_g(u,v)$. (2)

To make Eq. (2) more useful, we convert the probability into the *additive* weight to make the existing shortest path algorithm can acquire the path with the highest available probability. Thus, we derive the following Eq. (3).

Lemma 2. Following Lemma 1, we give every link a weight $w(u,v) = -\ln \mathcal{P}_g(u,v)$. (3)

Then, a path with a smaller sum of link weights has a higher available probability. That is, for any two paths p_1, p_2, $\sum_{(u,v)\in p_1} w(u,v) > \sum_{(u,v)\in p_2} w(u,v)$ *iff* $\mathcal{P}(p_1) < \mathcal{P}(p_2)$.

Moreover, with Eq. (2), we can estimate the expected waiting time of a path by Eq. (4). The proof is trivial and omitted.

Lemma 3. Following Lemma 1, the expected waiting time for creating an entangled path p is $E[p] = \frac{1}{\mathcal{P}(p)}$. (4)

In addition, we can link any two paths with a common end node, which will store the data qubit temporarily at a time slot, to acquire a longer path. However, the expected waiting time of such a path is *non-trivial* (i.e., not just the addition of the two paths' expected time). The expected waiting time of a path can be calculated as follows.

Theorem 3 ([19])**.** Let p_1 and p_2 denote two paths with a common end node k, and R is the number of time slots in a node's storage time. Assume the data qubit takes a time slot being temporarily stored at k, so the waiting time of the path linking $p1, p2$ is as follows:

$$E[p_1, p_2] = \sum_{t=2}^{\infty} t \sum_{s_2=1}^{t - \lceil \frac{t}{R+1} \rceil} \binom{t-s_2-1}{\lceil (\frac{s_2}{R}) - 1 \rceil} \mathcal{P}(p_1)^{\lceil \frac{s_2}{R} \rceil} \times \mathcal{P}(p_2)(1-\mathcal{P}(p_1))^{t-s_2-\lceil \frac{s_2}{R} \rceil}(1-\mathcal{P}(p_2))^{s_2-1}, \quad (5)$$

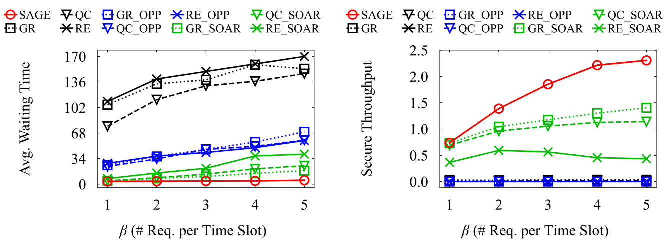
B. Algorithm Description

The traditional quantum routing algorithms, GREEDY [5], Q-CAST [10], and REPS [11], can also be employed to obtain the routing paths by SOAR. However, all of them are designed based on CON. Moreover, none of them considers trust relationships between nodes to select paths and balancing the waiting time for temporary storage. Therefore, their selected paths may have no or a few trusted repeaters such that they cannot utilize SOAR's advantages fully.

To perfectly fit SOAR, for each request, SAGE iteratively anchors a trusted repeater to the routing path until no repeater can be added to reduce the expected waiting time. The rationale behind the design of SAGE is that a longer entangled path has more opportunities to be decomposed into two entangled paths to reduce the expected waiting time. Overall, SAGE includes the following stages: 1) Path Sampling Stage (PSS), 2) Path Linking Stage (PLS), and 3) Path Augmenting Stage (PAS). Three stages are explained as follows:

1) Path Sampling Stage (PSS): In this stage, for each link (u, v), SAGE converts the success probability of entangling (i.e., $\mathcal{P}(u, v)$) to the weight of the link (i.e., $w(u, v)$) by Eqs. (1) and (3). With the link weights, SAGE calculates all-pair shortest paths and stores those paths with their expected waiting time by Eq. (4) in the path table T. Then, for every three different nodes s, k, d, SAGE acquires the shortest paths, p_1 from s to k and p_2 from k to d via the path table T, where k must be a trusted repeater of s. Afterward, SAGE calculates the expected waiting time of the path linking p_1 and p_2 by Eq. (5) and then stores the path with the expected time in the path table T. Note that PSS is a pre-processing, and by doing so, the quality of any shortest path can be known immediately without recomputing Eqs. (4) and (5) in PLS at every time slot. PLS makes SAGE meet SOAR's requirement of short time slot.

2) Path Linking Stage (PLS): In PLS, SAGE iteratively adds trusted repeaters to the path to form the routing paths for each request. To avoid the starvation issue, PLS iteratively examines the requests in increasing order of the request's arrival time. For each request (e.g., from s to d), SAGE initially sets the path as the shortest path from s to d by querying the table T built in PSS. Next, SAGE examines s's every trusted repeater (denoted by k) to generate a detour path by linking the shortest path from s to k and the shortest path from k to d. With the table T, SAGE can acquire the detour path's expected waiting time immediately. If there exists any detour path with a *shorter expected waiting time*, then SAGE will replace the current path with the detour path via the corresponding trusted repeater, denoted by k'. Subsequently, SAGE checks the subpath from s to the trusted repeater k' selected in the previous iteration and continues the above actions until no detour path can reduce the



(a) β vs Avg. Waiting Time (b) β vs Secure Throughput

Fig. 4. Effect of number of requests per time slot on different metrics.

expected waiting time. Last, SAGE will reserve the resources required by the request's path if the resources are sufficient, and then obtain a new residual graph. SAGE will repeat the stage until no more path can be found via table T.

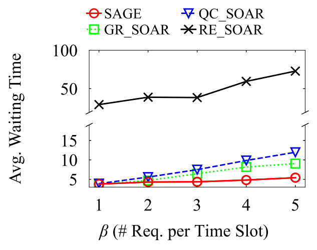
3) Path Augmenting Stage (PAS): Last, SAGE attempts to extract more paths for requests and exhaust the network resources. Similarly, SAGE iteratively examines the requests in increasing order of the request's arrival time. At first, for each request with no routing path allocated in PLS, SAGE uses the Breadth-First Search algorithm to find a least-hop path in the residual graph, and then reserves the path's required resources. After that, if the network still has remaining resources, SAGE will continue finding a least-hop path for each request iteratively until the network cannot support more paths.

IV. Performance Evaluation

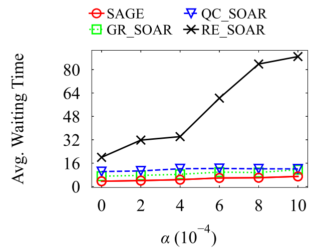
A. Simulation Settings

We follow [10], [11] to generate QNs by Waxman model [20] and use MEDICI [21] to generate SNs for the simulation. We randomly scatter 100 quantum nodes on the 2000 km by 4000 km rectangle area and assign them to 20 owners in the SN. We set $\delta = 0.85$ and $\epsilon = 0.02$ for Waxman model to build an edge with the probability $\delta e^{-l(u,v)/\epsilon \mathcal{L}}$, where $\mathcal{L}$ is the largest distance between any two nodes and $l(u, v)$ is the distance between u and v to ensure the average distance between two adjacent nodes is about 300 km. The social network density is set to 25% in MEDICI to build social links. The number of quantum channels of each edge is randomly set from 3 to 7 and the number of qubits of each node is randomly set from 10 to 14 [10]. By default, the success probability of entangling is about 0.94 and 0.83 by setting α to 2×10^{-4} and 6×10^{-4} in $\mathcal{P}(u, v) = e^{-\alpha \cdot l(u,v)}$ for Figs. 4 and 5, respectively.[5] The success probability of swapping at a node (i.e., $\mathcal{Q}(u)$) is set to 0.9. We select $\beta = 5$ random source-destination pairs to generate requests at each of the first 10 time slots and then run the simulation until no request is left. The link lifetime L and the storage time R are both set to 5 time slots. We change one parameter in each simulation to observe its effect on the different metrics. We compare SAGE, with GREEDY (GR) [5], Q-CAST (QC) [10], and REPS (RE) [11] on the different forwarding schemes, CON, OPP, and SOAR. Each result is averaged over 30 trials.

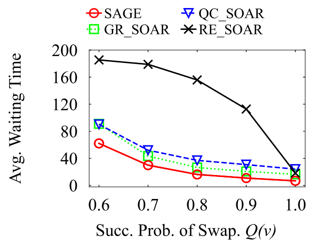
[5]We set a lower α (i.e., a higher success probability of entangling) for Fig. 4 to ensure that CON and OPP can finish all pairs' transmission in a reasonable time. Otherwise, they may have zero throughput and cannot stop.



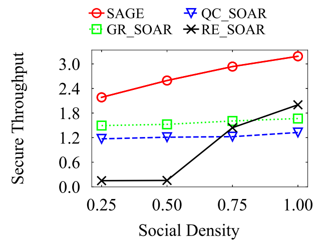
(a) β vs Avg. Waiting Time



(b) α vs Avg. Waiting Time



(c) $\mathcal{Q}(v)$ vs Avg. Waiting Time



(d) Soc. Den. vs Secure Throughput

Fig. 5. Effect of different parameters on different metrics.

B. Numerical Results

Overall, SOAR significantly improves the average waiting time and secure throughput (i.e., the average number of requests finished securely at a time slot) compared with other forwarding schemes. Also, SAGE outperforms other algorithms on SOAR, as shown in Figs. 4 and 5. More simulation results such as the running time, effect of life time, and the effect of storage time can be found in Appendix F in [16].

1) Effect of Forwarding Scheme: Fig. 4(a) shows that when β increases, the waiting time grows because more requests share the limited resources. Overall, SOAR outperforms CON and OPP by $84-89\%$ and $54-77\%$, respectively, since SOAR adopts AGG to improve the waiting time. Fig. 4(b) shows CON and OPP has almost zero secure throughput since CON has a long average waiting time while OPP may choose untrusted repeaters for temporary storage. In contrast, SOAR identifies the trusted repeaters and initiates the swapping process efficiently, leading to high secure throughput.

2) Effect of Routing Algorithm: In Fig. 5(a), SAGE outperforms GR, QC, and RE on SOAR by up to 44%, 57%, and 93%. Figs. 5(b)–5(c) show the average waiting time with different α and $\mathcal{Q}(u)$. The waiting time increases as the success probabilities of entangling and swapping decrease (i.e., a higher α and a lower $\mathcal{Q}(u)$) for all the algorithms on SOAR. Figs. 5(a) and 5(b) also show SAGE has the best performance since SAGE selects the path that can balance the waiting time and the number of trusted repeaters to perfectly fit SOAR. It is worth noting that RE performs worse than the other algorithms in most cases since RE's expected throughput error will accumulate and increase drastically as the environment is challenging (i.e., high α and low $\mathcal{Q}(v)$).

3) Effect of Repeater Selection: Fig. 5(d) shows that a higher social density gives more chances for requests to find trusted repeaters, so it also leads to higher security throughput. Moreover, SAGE outperforms the others under different social density because SAGE takes trusted repeaters into account for constructing the routing path in PLS. The more effects of social density are shown in Figs. 8(a) and 8(b) in [16].

V. Conclusion

This paper proposes a novel socially-aware opportunistic scheme SOAR to speed up the quantum routing while not putting user data qubits at risk. To overcome the shortcomings of the existing opportunistic quantum forwarding scheme OPP, we add one more swapping process aggressively in SOAR to promote swapping efficiency. Moreover, to further optimize SOAR's performance, we derive a promising metric to measure path quality for our proposed socially-aware routing algorithm SAGE to select a path with appropriate trusted repeaters for each request. Finally, simulation results show that, with any quantum routing algorithm, SOAR can outperform OPP by up to 77%. Also, SAGE can perfectly fit SOAR and reduce the average waiting time by around 50% compared to the others.

References

[1] C. Elliott, "Building the quantum network," *New J. Phys.*, vol. 4, p. 46, 2002.
[2] H. J. Kimble, "The quantum internet," *Nature*, vol. 453, pp. 1023–1030, 2008.
[3] N. Sangouard *et al.*, "Quantum repeaters based on atomic ensembles and linear optics," *Rev. Mod. Phys.*, vol. 83, p. 33, 2011.
[4] R. Ursin *et al.*, "Entanglement-based quantum communication over 144 km," *Nature physics*, vol. 3, no. 7, pp. 481–486, 2007.
[5] M. Pant *et al.*, "Routing entanglement in the quantum internet," *NPJ Quantum Inf.*, vol. 5, pp. 1–9, 2019.
[6] Pan *et al.*, "Experimental entanglement swapping: entangling photons that never interacted," *Physical review letters*, vol. 80, no. 18, p. 3891, 1998.
[7] Guccione *et al.*, "Connecting heterogeneous quantum networks by hybrid entanglement swapping," *Sci. Adv.*, vol. 6, no. 22, p. eaba4508, 2020.
[8] S. Pirandola *et al.*, "Fundamental limits of repeaterless quantum communications," *Nat. Commun.*, vol. 8, 2017.
[9] M. Caleffi, "Optimal routing for quantum networks," *IEEE Access*, vol. 5, pp. 22 299–22 312, 2017.
[10] S. Shi and C. Qian, "Concurrent entanglement routing for quantum networks: Model and designs," in *ACM SIGCOMM*, 2020.
[11] Y. Zhao and C. Qiao, "Redundant entanglement provisioning and selection for throughput maximization in quantum networks," in *IEEE INFOCOM*, 2021.
[12] A. Farahbakhsh and C. Feng, "Opportunistic routing in quantum networks," in *IEEE INFOCOM*, 2022.
[13] Y.-Y. Fei, X.-D. Meng, M. Gao, H. Wang, and Z. Ma, "Quantum man-in-the-middle attack on the calibration process of quantum key distribution," *Sci. Rep.*, vol. 8, no. 1, pp. 1–10, 2018.
[14] S. Song and M. Hayashi, "Secure quantum network code without classical communication," *IEEE Transactions on Information Theory*, vol. 66, no. 2, pp. 1178–1192, 2019.
[15] R. V. Meter and J. Touch, "Designing quantum repeater networks," *IEEE Commun. Mag.*, vol. 51, pp. 64–71, 2013.
[16] S.-M. Huang *et al.*, "Socially-aware opportunistic routing with path segment selection in quantum networks (technical report)," May 2023. [Online]. Available: https://github.com/forward0606/SAGE
[17] S. Wehner, D. Elkouss, and R. Hanson, "Quantum internet: A vision for the road ahead," *Science*, vol. 362, no. 6412, p. eaam9288, 2018.
[18] I. Parvez *et al.*, "A survey on low latency towards 5G: RAN, core network and caching solutions," *IEEE Commun. Surv. Tutor.*, vol. 20, pp. 3098–3130, 2018.
[19] S.-M. Huang *et al.*, "Socially-aware concurrent entanglement routing with path decomposition in quantum networks," in *IEEE GLOBECOM*, 2022.
[20] B. Waxman, "Routing of multipoint connections," *IEEE J. Sel. Areas Commun.*, vol. 6, pp. 1617–1622, 1988.
[21] D. F. Nettleton, S. Nettleton, and M. C. i. Farriol, "MEDICI: A simple to use synthetic social network data generator," in *MDAI*, 2021.

Appendix A
Related Work

For implementation, Elliott *et al.* proposed the first idea of QNs and implemented secure communications [1]. Meter *et al.* proposed a large QN architecture with layered recursive repeaters, where repeaters may not trust each other, and then designed new protocol layers to support quantum sessions to ensure robustness and interoperable communication [15]. For optimization, Pirandola *et al.* discussed the limits of repeater-less quantum communications and proposed general benchmarks for repeaters [8]. Caleffi *et al.* designed a routing protocol to assure the highest end-to-end entanglement rate between any two nodes [9]. Pant *et al.* provided a greedy algorithm to create entangled paths by gluing multiple established entangled links [5]. Shi *et al.* proposed a routing method Q-CAST with a Dijkstra-based algorithm to find paths. [10]. Q-CAST uses recovery paths to mitigate entanglement failures' effect. Zhao *et al.* presented an LP-based algorithm REPS with a central controller to maximize throughput [11]. All of the above methods adopt conventional forwarding schemes which adopt a long period for synchronization and cause lots of idle time. Farahbakhsh *et al.* presented an opportunistic forwarding scheme that forwards data qubits as soon as possible [12]. However, both the conventional and opportunistic ones still lack parallelism and miss trust relationships between node owners and thus cause a long waiting time compared to SOAR.

Appendix B
Proof of Theorem 1

Proof. We fist calculate the probability of creating an entangled path using CON. Such a probability is equal to the probability of all the entangling and swapping operations on the path succeeding for the first time in a period, i.e., the probability is

$$\mathcal{P}_{con} = \prod_{(u,v)\in p} \mathcal{P}(u,v) \cdot \prod_{u\in p} \mathcal{Q}(u)$$

Then, we consider AGG. Let $\mathcal{P}'(u,v)$ denote the available probability of a certain link between nodes u and v using AGG. By Lemmas 1 and 2 and Theorem 2, we know $\mathcal{P}'(u,v) \lesssim \mathcal{P}_g(u,v)$ in the long term while $\mathcal{P}'(u,v) \gtrsim \mathcal{P}(u,v)$ in the short term. Therefore, we have the following inequality.

$$\mathcal{P}(u,v) \leq \mathcal{P}'(u,v) \leq \mathcal{P}_g(u,v).$$

Afterward, if the swapping process fails, AGG will recreate the links that were decohered and initiate another swapping process after that. Note that AGG has more chances to initiate entangling and swapping processes as CON, as shown in Fig. 3. Thus, AGG must gain more probability than CON since CON does not recreate the links decohered in a period. Finally, let $\mathcal{P}_{agg}$ denote the available probability of an entangled path in a period using AGG. By the above reasons, we can derive

$$\begin{aligned}\mathcal{P}_{agg} &> \prod_{(u,v)\in p} \mathcal{P}'(u,v) \prod_{u\in p} \mathcal{Q}(u)\\ &\geq \prod_{(u,v)\in p} \mathcal{P}(u,v) \prod_{u\in p} \mathcal{Q}(u) = \mathcal{P}_{con}.\end{aligned}$$

Therefore, the theorem holds. □

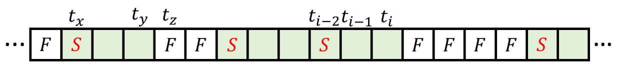


Fig. 6. Example of creating an entangled link (u,v) in sequential time slots.

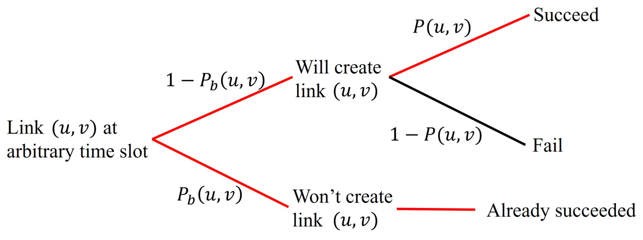


Fig. 7. Available probability of link (u,v) at an arbitrary time slot.

Appendix C
Proof of Theorem 2

Proof. For an arbitrary slot in the long term, there are two disjoint cases as follows. 1) the two nodes u, v try to create an entangle link in this time slot. For example, in Fig. 6, they try and succeed in time slot t_x while trying but fail in time slot t_z, where $L = 3$. 2) The entangled link has been created while remaining in this time slot. For example, the u and v did not try at t_{i-1} and t_i since the entangling has succeeded at t_{i-2} and it can remain available for $L = 3$ time slots in Fig. 6. We first denote by $\mathcal{P}_b(u,v)$ the probability of the second case. Then, both cases are depicted in Fig. 7, where the red branches represent the conditions of the entangled link available in the time slot. Thus, we know the available probability

$$\begin{aligned}\mathcal{P}_g(u,v) &= \mathcal{P}_b(u,v) + (1-\mathcal{P}_b(u,v)) \cdot \mathcal{P}(u,v)\\ &= \mathcal{P}_b(u,v) \cdot (1-\mathcal{P}(u,v)) + P(u,v). \qquad (6)\end{aligned}$$

Clearly, u and v do not try to create an entangled link in this time slot (i.e., the second case) if and only if they tried to create an link and succeeded in a time slot among the previous $(L-1)$ time slots (i.e., $(L-1)$ events). In other words, the second case's probability $\mathcal{P}_b(u,v)$ only depends on the $(L-1)$ events. The probability of each of $(L-1)$ events in the long term is also $(1-\mathcal{P}_b(u,v)) \cdot \mathcal{P}(u,v)$ as shown in Fig. 7. Since the $(L-1)$ events are independent, we know

$$\mathcal{P}_b(u,v) = (L-1) \cdot (1-\mathcal{P}_b(u,v)) \cdot \mathcal{P}(u,v). \qquad (7)$$

By rearranging Eq. (7), we acquire

$$\mathcal{P}_b(u,v) = \frac{(L-1) \cdot \mathcal{P}(u,v)}{(L-1) \cdot \mathcal{P}(u,v) + 1}. \qquad (8)$$

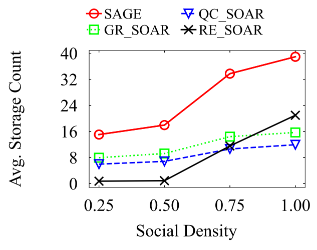
With Eqs. (6) and (8), we can get

$$\begin{aligned}\mathcal{P}_g(u,v) &= \frac{(L-1) \cdot \mathcal{P}(u,v)}{(L-1) \cdot \mathcal{P}(u,v) + 1} \cdot (1-\mathcal{P}(u,v)) + \mathcal{P}(u,v)\\ &= \frac{L \cdot \mathcal{P}(u,v)}{L \cdot \mathcal{P}(u,v) - \mathcal{P}(u,v) + 1}.\end{aligned}$$

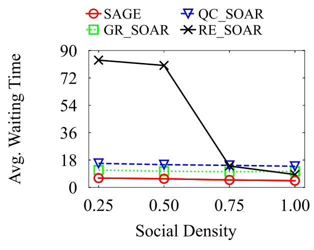
Therefore, the theorem holds. □

TABLE I
RUNNING TIME ON SERVER WITH INTEL XEON GOLD 6248R CPU (SEC)

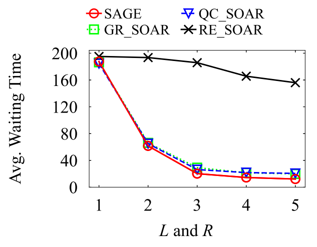
β	1	2	3	4	5
SAGE	0.009	0.009	0.010	0.012	0.011
GR	0.090	0.149	0.218	0.315	0.393
QC	0.078	0.257	0.577	0.931	1.237
RE	1.906	2.844	4.884	5.864	6.235



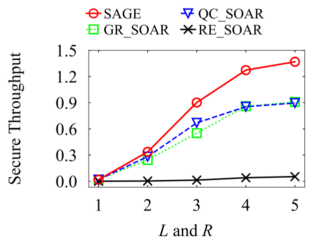
(a) Soc. Den. vs Avg. Storage Count



(b) Soc. Den. vs Avg. Waiting Time



(c) L and R vs Avg. Waiting Time



(d) L and R vs Secure Throughput

Fig. 8. Effect of different parameters on different metrics.

APPENDIX D
PROOF OF LEMMA 1

Proof. The creation of an entangled path succeeds if and only if all entangled link and swapping link are available simultaneously. Since all nodes have enough quantum memory to support all links, the creations of all links are independent. Thus, $\mathcal{P}(p) = \prod_{(u,v)\in p} \mathcal{P}_g(u,v)$. The lemma holds. $\square$

APPENDIX E
PROOF OF LEMMA 2

Proof. By Eq. (3), we can easily derive $\sum_{(u,v)\in p_1} w(u,v) > \sum_{(u,v)\in p_2} w(u,v)$ if and only if $-\ln\Big(\prod_{(u,v)\in p_1} \mathcal{P}_g(u,v)\Big) > -\ln\Big(\prod_{(u,v)\in p_2} \mathcal{P}_g(u,v)\Big)$ holds. By Eq. (2), the latter statement holds if and only if $\mathcal{P}(p_1) < \mathcal{P}(p_2)$. $\square$

APPENDIX F
MORE NUMERICAL RESULTS

1) Effect of Repeater Selection: Fig. 8(a) shows that the average temporary storage count per time slot rises when the social density increases. It is because a higher social density gives more opportunities for requests to find trusted repeaters. Furthermore, SAGE adaptively leverages more temporary storage of trusted repeaters to speed up the routing and thus reduce the average waiting time in Fig. 8(b).

2) Effect of Link Lifetime and Storage Time: Fig. 8(c) shows the average waiting time decreases and converges quickly as L (link lifetime) and R (storage time) grow, implying that a smaller L (or R) can also improve the performance significantly for SOAR. Fig. 8(d) manifests that SAGE can increase secure throughput and outperform the others as L and R grow since SAGE can anchor more trusted repeaters on paths and leverage them better in PLS.

3) Running time: Table I shows that SAGE runs much faster than other algorithms on SOAR since it benefits from the preprocessing in PSS. In contrast, the others may be impractical since their running time is greater than $1.46/5 = 0.292$ seconds, where the link lifetime of current technology is about 1.46 seconds [10] and $L = 5$.