A Heuristic Remote Entanglement Distribution Algorithm on Memory-Limited Quantum Paths

Lutong Chen, *Graduate Student Member, IEEE*, Kaiping Xue, *Senior Member, IEEE*, Jian Li, *Member, IEEE*, Nenghai Yu, Ruidong Li, *Senior Member, IEEE*, Jianqing Liu, *Member, IEEE*, Qibin Sun, *Fellow, IEEE*, and Jun Lu

***Abstract*— Remote entanglement distribution plays a crucial role in large-scale quantum networks, and the key enabler for entanglement distribution is quantum routers (or repeaters) that can extend the entanglement transmission distance. However, the performance of quantum routers is far from perfect yet. Amongst the causes, the limited quantum memories in quantum routers largely affect the rate and efficiency of entanglement distribution. To overcome this challenge, this paper presents a new modeling for the maximization of entanglement distribution rate (EDR) on a memory-limited path, which is then transformed into entanglement generation and swapping sub-problems. We propose a greedy algorithm for short-distance entanglement generation so that the quantum memories can be efficiently used. As for the entanglement swapping sub-problem, we model it using an Entanglement Graph (EG), whose solution is yet found to be at least NP-complete. In light of it, we propose a heuristic algorithm by dividing the original EG into several sub-problems, each of which can be solved using dynamic programming (DP) in polynomial time. By conducting simulations, the results show that our proposed scheme can achieve a high EDR, and the developed algorithm has a polynomial-time upper bound and reasonable average runtime complexity.**

***Index Terms*— Quantum network, entanglement swapping, entanglement distribution, dynamic programming.**

I. Introduction

QUANTUM network is a promising networking technology to transmit quantum bits (qubits), enabling new applications such as quantum key distribution (QKD) and quantum clock synchronization. With the fast prototyping of quantum devices, the quantum network will soon become the reality [1]. Compared with the classic Internet, a quantum network enables unprecedented network services for end users. Specifically, quantum networks make it possible to generate and distribute entangled pairs (a.k.a., ebits) between arbitrary endpoints even with a long spatial separation [2], [3]. Those entangled pairs are fundamental for many upper-layer applications such as secure quantum state transmission [4], quantum key distribution [5], [6] and distributed quantum computation [7], [8].

In quantum networks, entangled pairs can be produced by an entanglement generator and distributed to two adjacent nodes through optical fiber [2], or in free space [9], [10]. We refer to this process as link entanglement generation, or entanglement generation for short. However, one of the major difficulties is distributing entangled pairs to two remote nodes. It is largely due to the path loss incurred by scattering and absorption in the transmission medium, leading to quick decoherence of entanglement. Here, we coin the research task of distributing entanglement over a long distance as *remote entanglement distribution*. To offer a solution to this task, quantum routers (or repeaters) [11], [12] with the capability of Bell State Measurement (BSM) are introduced to perform a quantum joint measurement. As a result, two original entangled pairs are consumed, but a new entangled pair is created across two links, thereby extending the entanglement distribution distance. By repetitively performing BSM in intermediate quantum routers, one can create entanglement over a much longer distance. This operation is called *entanglement swapping* (or swapping for short). In brief, several quantum routers are placed in the middle of endpoints and not separated too far. Any adjacent nodes — endpoint and router — first generate link entanglement. Then, by performing entanglement swapping at quantum routers, an end-to-end entangled pair over a longer distance is formed.

The primary problem in quantum networks is the optimal routing problem [13], [14], which can be further divided into two challenges. The first challenge is how to select a path. When an entanglement distribution request is issued, a specific routing algorithm will be called to select a specific path in the network and to pre-allocate resources for such entanglement distribution [15], [16], [17], [18], [19]. The second challenge is

Manuscript received 18 December 2021; revised 7 May 2022 and 1 September 2022; accepted 2 September 2022. Date of publication 12 September 2022; date of current version 18 November 2022. This work by L. Chen, K. Xue, J. Li, N. Yu is supported in part by Anhui Initiative in Quantum Information Technologies under grant No. AHY150300 and Youth Innovation Promotion Association Chinese Academy of Sciences (CAS) under Grant No. Y202093. The work by J. Liu was supported in part by the National Science Foundation under grant OMA-2231357. The associate editor coordinating the review of this article and approving it for publication was A. S. Cacciapuoti. *(Corresponding authors: Kaiping Xue; Jian Li.)*

Lutong Chen, Kaiping Xue, Jian Li, Nenghai Yu, and Qibin Sun are with the School of Cyber Science and Technology, University of Science and Technology of China, Hefei, Anhui 230027, China (e-mail: kpxue@ustc.edu.cn; lijian9@ustc.edu.cn).

Ruidong Li is with the College of Science and Engineering, Kanazawa University, Kakuma, Kanazawa 920-1192, Japan.

Jianqing Liu is with the Department of Computer Science, North Carolina State University, Raleigh, NC 27606 USA.

Jun Lu is with the School of Cyber Science and Technology and the Department of Electronic Engineering and Information Science, University of Science and Technology of China, Hefei, Anhui 230027, China.

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCOMM.2022.3205683.

Digital Object Identifier 10.1109/TCOMM.2022.3205683

how to distribute entanglement efficiently on the selected path, considering that quantum operations are imperfect, e.g., these operations may fail and/or cause entanglement to decohere, and the size of a quantum memory is limited. The solution to the second challenge is of vital importance to the establishment of a large-scale quantum network, so it is the focus of this paper.

To this end, we first model the problem of maximizing the entanglement distribution rate (EDR) on a resource-limited, noisy, and unreliable path. We then decompose the problem into the entanglement generation problem and the entanglement swapping problem. Next, we prove that the best strategy is to perform entanglement generation operations greedily. That is to say, entangled pairs need to be generated significantly to the extent that they can make full use of any available quantum memory in the quantum routers.

As for the entanglement swapping problem, it is found that the utility function, i.e., EDR, cannot be used directly due to the high evaluation cost. Therefore, we propose an alternative utility function M_τ, which is the sum of a contribution function of each entanglement swapping. In this way, solving the entanglement swapping problem is equivalent to finding the weighted Maximal Independent Sets (MIS), which is widely known to be hard in a polynomial-time [20]. To overcome this obstacle, we propose a heuristic algorithm, HSA, in which the solution turns out to be near-optimal in most scenarios. Specifically, we use a heuristic rule to decompose the entanglement swapping problem into several independent sub-problems. The heuristic rule is that a router can only perform at most one swapping in each sub-problem solution instance. In each sub-problem, a Dynamic Programming (DP) algorithm [21], [22] is used to calculate the solution in polynomial time. When aggregated, the overall swapping algorithm is proved to be a deterministic polynomial-time algorithm with the worst upper bound $O(m \cdot N^4)$, where N is the number of nodes on the path and m is the memory size.

We conduct simulations and performance analyses to compare HSA with other baseline algorithms. The results show that the proposed scheme has a performance improvement of up to 87.76% compared to baselines, and the average complexity is about $O(N^{2.109})$ in most scenarios which are far better than its worst case.

Our contributions in this paper are as following:

- We model the problem of maximizing the entanglement distribution rate on a memory-limited quantum path and propose a centralized and iterative end-to-end entanglement distribution framework to handle entanglement generation and swapping where only the swapping decision is made in a centralized way.
- We transform the entanglement swapping problem into a weighted Maximal Independent Set searching problem by introducing an Entanglement Graph (EG). We then propose a heuristic polynomial-time approximate algorithm to decompose the original problem into multiple sub-problems that can be solved easily by a DP algorithm.
- We develop a numerical evaluation platform, and the results show the performance advantages of our design in terms of EDR and memory use. Also, We evaluate the algorithm's time complexity numerically and theoretically. We deem that the designed algorithm is applicable to future large-scale quantum networks for entanglement distribution.

The organization of this work is as follows. In Section II, we introduce the background of the enabling technologies and related works. In Section III, we describe the system model and the modeling of the maximum EDR design problem. Then, in Section III-B, we propose the entanglement distribution framework in which short-distanced entanglement generation and entanglement swapping are performed alternatively in each decision-making time slot. Next in Section IV, a greedy entanglement generation algorithm is presented. In Section V, we provide our heuristic algorithm for entanglement swapping. We present the performance evaluation and analysis in Section VI. Finally, we discuss and conclude our work in Section VII.

II. Related Works

The quantum network is unique for its transmission of quantum bits (qubits) through teleportation [3], [23], which is based on the established entanglement between the source and the destination. Taking a 2-qubit entanglement as an example, a Bell entangled pair — the maximally entangled 2-qubit state — is pre-distributed between the source and the destination. Then, a target qubit can be teleported with the assistance of the established entangled pair [24] through several standard quantum operations.

Quantum network is the product of many disciplines spanning from physics to telecommunication that can hardly be surveyed in this work. Instead, we mainly summarize the works that are relevant to entanglement distribution. In essence, the critical quantum operations pertaining to distributing entanglement include entanglement generation and entanglement swapping [25]. In this procedure, quantum memory is needed to store the qubits' quantum state.

A. Enabling Quantum Operations

Link entanglement generation is a quantum operation that can produce two Bell-state entangled qubits at an EPR generator. The generator can be located at either a node or a third party. Then, qubits are sent to these adjacent nodes to form a link-layer entangled pair through a quantum link (e.g., an optical fiber). Nevertheless, due to the attenuation of the quantum link and the imperfection of the EPR generator, entanglement generation may fail, and we denote p_{gen} as the overall probability of a successful entanglement generation.

Entanglement swapping [26] is a quantum operation performed on quantum routers [26]. The router will consume two entangled pairs to create a new entangled pair with longer distances. For example, if the source and a router share an entangled pair, the destination and the router share another pair, the router can perform Bell State Measurement (BSM) — a key operation for entanglement swapping — to create a new entangled pair between the source and the destination. However, the consumed two entangled pairs are not available anymore. In practice, BSM cannot be guaranteed

to be successful due to the noise and imperfect operations [15], and we denote p_{swap} as the success probability.

The limitation of quantum memory is another contributing factor to EDR. Quantum memories [27], [28] can store qubits within a short time because the entanglement will decohere as time goes on. However, it is very complicated and expensive in the noisy intermediate-scale quantum (NISQ) ego [7]. As quantum memories in a quantum router are shared among multiple entanglement distribution requests, the memory allocated for a specific request becomes even more precious.

B. Challenges in Entanglement Distribution

The primary problem in quantum networks is to efficiently distribute entangled pairs between two remote nodes, which is named the optimal routing problem [13], [14]. To this end, it can be divided into two steps. First, a routing algorithm must be presented to select a path from the source to the destination for each communication request. After that, the entanglement distribution schema should be proposed to distribute end-to-end entangled pairs on that selected path. The major difficulty is that the network resources are limited, and the entangled pairs decohere quickly, resulting in a limited network capacity [17], [29]. Unfortunately, The overall optimal routing problem in arbitrary network topology is proved to be an NP-complete problem [16], [19], and we further prove that even finding an optimal entanglement distribution schema is also an NP-complete problem in Section V-B.

For the routing path selection between a source and a destination, several existing mechanisms have been proposed [15], [18], [19], [30]. To improve the capacity of a single path [17], [29], Pant *et. al* [15] proposed a multi-demand multi-path routing protocol. The entangled pair distribution scheme proposed in this paper can be directly adopted into a multi-path model and executed independently on every single path. Other routing algorithms focus on specific features. For example, Chakraborty *et al.* [19] proposed a path selection approach under different fidelity requests. Shi and Qian [14] proposed Q-Cast algorithm for both routing and allocating entangled pairs, and Zhang *et al.* [31] improved it by reusing entanglement fragments. Besides, Li *et al.* [16] proposed a multiple-path routing and entanglement allocation algorithm.

For remote entangled pair distribution schema, there are also several mechanisms. Among them, Evgeny Shchukin *et al.* [32] and Liang Jiang *et al.* [33] focused on the distribution of the first qubit and reducing the delay time. However, more entangled pairs are needed and distributed continuously in most quantum applications. Bernardes *et al.* [34] proposed a remote distribution algorithm using multiplexing and calculating the rate of remote distribution. However, it uses a strict entanglement scheme (we call it BTSA later), which greatly restricts EDR. Dai *et al.* [35] proposed an optimal algorithm for long-distance distribution with noisy intermediate-scale quantum technologies, where the imperfect quantum routers have been considered. However, they have not considered that the limited size of a quantum memory also greatly impacts the entanglement distribution rate, as we will show later. In [19], Kaushik Chakraborty *et al.* presented a simple but efficient distribution algorithm called **prepare and swap** protocol, but it did not consider the quantum memory either.

III. System Model and Problem Formulation

A. System Model

A quantum network is composed of several quantum nodes, including endpoints and quantum routers. These quantum nodes are equipped with quantum devices to perform quantum measurements and quantum memories to store qubits. These quantum nodes are also equipped with one or more interfaces bonded to quantum links such as optical fiber or free space. Those quantum links are capable of transmitting qubits. The primary function of the quantum network is to transmit qubits between arbitrary source-destination pairs by distributing a pair of entangled qubits between them. This operation is known as quantum teleportation [24]. We assume that an integrated classic network is presented along with quantum networks to deliver classic control messages reliably between any quantum nodes.

In this paper, we assume that a routing protocol exists to select a path and allocate quantum memories for each request (i.e., a source-destination pair). To be more specific, when a request starts, a routing protocol will calculate one or more paths in the quantum network and also allocate a certain number of quantum memories on each node of the selected path. The advantage of the pre-selected path model is that it can guarantee a better performance for each request. Consider a path with N quantum nodes including 2 end points and $N-2$ quantum routers. u_i ($\in \mathbb{U}$) denotes the i-th node in $\mathbb{U}$ where $\mathbb{U}$ is the set of nodes on that path. The cardinality of $\mathbb{U}$, i.e., $|\mathbb{U}|$, is equal to N. We assume that there is an m-qubit quantum memory allocated for one interface in each quantum node. That is to say, the source and the destination allocate an m-qubit memory, while quantum routers should have at least a size of $2m$ for their two interfaces (one for incoming and the other for outgoing). With m-qubit quantum memory, at most m entangled pairs can be generated between any adjacent nodes. Fig. 1 shows the model of a quantum path with 4 nodes. It also shows the internal details within a quantum router. Quantum memories (with the size of 4) is available on both interfaces of a quantum router, so at most 4 entangled pairs can be restored on each link. Provided that only two interfaces are involved in each quantum router u_i for any request, we use inbound and outbound to distinguish the two interfaces, where inbound is closer to the source (i.e., u_1) and outbound is closer to the destination (i.e., u_N).

We use $e_{i,j}$ to represent an entangled pair between u_i and u_j. In case of u_i and u_j sharing multiple entangled pairs, we use the notation $e_{i,j}^k$ to represent the k-th entangled pair. Let $\mathbb{E}$ be the set of all existing entangled pairs and $E(i,j)$ be the the number of entangled pairs between u_i and u_j.

The quantum network is assumed to follow a discrete time slot model, where the network control decision is made at the beginning of a time slot. Specifically, let $\tau \in \mathbb{N}^+$ denote a time slot, and $\mathbb{E}_\tau$ be the set of existing entangled pairs at τ. A centralized controller will instruct a selection of quantum

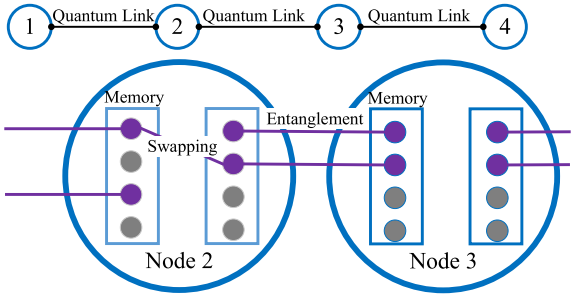


Fig. 1. Example of a quantum path and an abstracted model within quantum routes ($N = 4, m = 4$).

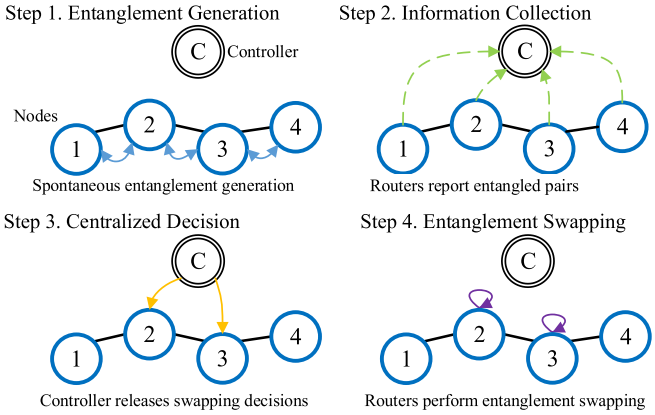


Fig. 2. Procedure of remote entanglement distribution.

nodes in each time slot to perform entanglement generation or entanglement swapping. Without loss of generality, we assume a decision set to represent the control decisions for entanglement generation and swapping in a time slot.

Specifically, for the link entanglement generation, let the decision set $\mathbb{S}_\tau^g$ be a set of entangled pairs. For each element $e_{i,i+1}$ in $\mathbb{S}_\tau^g$, it means that u_i and u_{i+1} will perform entanglement generation and have a probability of p_{gen} to generate $e_{i,i+1}$ in time slot $\tau + 1$.

Let $\mathbb{S}_\tau^s$ be a swapping decision set which contains several swappable candidates $(e_{i,j}, e_{j,k})$, where $e_{i,j}$ and $e_{j,k}$ are two existing entangled pairs in the current time slot τ. u_j will consume these two entangled pairs and perform entanglement swapping. As a result, this joint measurement operation has a probability of p_{swap} to distribute a new entangled pair $e_{i,k}$ in time slot $\tau + 1$. Regardless of whether the operation is successful, both $e_{i,j}$ and $e_{j,k}$ will be consumed.

Consider that the size of the quantum memory on each quantum node is limited. For each quantum node u_i, it cannot hold more than m entangled pairs for each interface, and we get the following constraints on the limited size of the quantum memory:

$$C_{out}(i) = \sum_{j=i+1}^{N} E(i,j) \leq m, \tag{1}$$

$$C_{in}(i) = \sum_{j=1}^{i-1} E(j,i) \leq m, \tag{2}$$

where $C_{out}(i)$ is the number of qubits existing in the outbound interface of node u_i and $C_{in}(i)$ is the number of qubits in the inbound memory. Here $i \in [1, N]$.

We do not consider the entanglement distillation [36], [37], but instead, we require a reliably link-layer protocol [38] to produce high-fidelity entangled pairs. From the network architecture perspective, entanglement distillation can be performed at both the link layer and the application layer. In the link layer, distillation can obtain high-fidelity entangled pairs between any adjacent nodes [38]. In the application layer, the upper layer applications can use multiple distributed remote entangled pairs for purification to obtain high-fidelity end-to-end entangled pairs to address the fidelity downgrade during entanglement swapping. For those qubits that have a low fidelity and cannot be used, quantum memory will cut off these entangled pairs, Vardoyan *et. al* provides a cutoff policy in quantum memories [39].

B. Remote Entanglement Distribution Framework

In this subsection, we propose a remote entanglement distribution framework to make an optimal decision for remote entanglement distribution, and we decompose it into the entanglement generation problem and the swapping problem.

Quantum nodes have limited information about the whole network, and distributed decision-making processes cannot efficiently utilize the network resources. Therefore, it is better to introduce a centralized controller to manage the resources of any end-to-end sessions [16], [30]. Specifically, in our problem setting, the role of a centralized controller includes entanglement information collection, decisions making for entanglement generation and swapping, requesting quantum routers to perform entanglement swapping or generation, and obtaining the response from them. We assume a synchronous time-slotted system where the controller runs the algorithm at each time slot and instructs the selected nodes to execute entanglement generation or entanglement swapping.

Fig. 2 shows the remote entanglement distribution framework. The procedure in each time slot can be further divided into four steps:

Step 1. The controller collects the entanglement information over all candidate paths, such as the available size of quantum memories. Then it instructs the quantum routers to generate entangled pairs. In Section IV, we find that the best entanglement generation strategy is to generate link-layer entangled pairs at the maximized rate to fill quantum memories. Therefore, this step can be implemented by a static rule that all quantum nodes spontaneously complete the short-distanced entanglement generation by themselves.

Step 2. The controller keeps track of the status information of the entangled pairs. This is because not all short-distanced entanglement generations will succeed, and the existing entangled pairs have the possibility of decoherence.

Step 3. The controller performs the distribution algorithm to produce and release $\mathbb{S}_\tau^s$ to a selection of routers.

Step 4. Finally, the selected quantum routers will perform entanglement swapping under the controller's instruction.

In Section IV, we illustrate the algorithm for entanglement generation. We find the strategy to be greedy. In Section V, we model the problem of entanglement swapping and propose a heuristic algorithm, HSA, for its solution.

C. Problem Formulation

The upper-layer applications usually require long-distanced quantum entanglement distribution at a higher rate. For example, in the scenario of QKD, two endpoints with a higher entanglement distribution rate will achieve a higher security level and performance.

We model the problem of maximizing the remote entanglement distribution rate $\mathcal{P}_{\text{MEDR}}$ as follow: Let $\mathcal{A}_{L(N,m)} : \mathbb{E}_\tau \rightarrow \{\mathbb{S}^g_\tau, \mathbb{S}^s_\tau\}$ denote the algorithm used for remote entanglement distribution at time slot τ, where $L(N,m)$ is a N-nodes quantum path and each interface has a m-qubit quantum memory. Also, let $\mathcal{A}^*_L$ be the optimal algorithm on $L(N,m)$.

The optimization goal is to increase the number of distributed remote entangled pairs. Let $\text{EDR}_{\mathcal{A}_L}$ be the entanglement distribution rate, and therefore, we obtain the utility function of $\mathcal{P}_{\text{MEDR}}$ as follows,

$$\text{EDR}_{\mathcal{A}_L} = \lim_{T\rightarrow\infty} \frac{1}{T} \sum_{\tau=1}^{T} E_\tau^{A_L}(u_1, u_N), \tag{3}$$

where $E_\tau^{\mathcal{A}_L}(u_1, u_N)$ represents the entangled pairs distributed between u_1 and u_N at time slot τ. $\mathcal{P}_{\text{MEDR}}$ can be formally described as:

$$\mathcal{A}^*_L = \arg\max_{\mathcal{A}_L} \text{EDR}_{\mathcal{A}_L},$$

Briefly, we aim to design an algorithm that can better use the limited quantum memory and further distribute remote entanglements with a higher EDR rate in the long term.

How to distribute the first several entangled pairs is not the focus of this paper based on the following two reasons: First, most quantum applications and upper-layer protocols will not just use one entangled pair. Long-distanced entangled pairs need to be produced and transmitted continuously. Second, the establishment of quantum entanglement connections and long-distance distribution tunnels is costly. It is dramatically wasteful if it is to distribute only one entangled pair.

IV. The Entanglement Generation Algorithm

The entanglement generation algorithm runs in **Step 1** in each time slot. The strategy for generating entangled pairs between two adjacent nodes is discussed in this section. The conclusion is that entangled pairs should be generated greedily, i.e., generates link-layer entangled pairs at a maximized rate. Here is a brief proof.

In an optimal strategy for maximizing EDR, new entangled pairs should be generated whenever the quantum memories have the available capacity to store them. Let A^* be the algorithm that entangled pairs are distributed in the most recent time slot, and A' is another algorithm that has the same behavior before t as A^* except that one entangled pair is distributed later than A^*. Assume that the entangled pair is distributed in t^* time slot in A^* while it is distributed in $t > t^*$ in A'. Then we have the following proposition.

TABLE I
NOTATIONS TABLE

Notation	Meaning
τ	Time slot
T	# of total time slots
N	# of nodes
m	Size of the memory allocated for each interface
$L(N,m)$	A path with N nodes and m-qubit memories
$\mathbb{U}$	Quantum node set
p_{gen}	Success probability of entanglement generation
p_{swap}	Success probability of entanglement swapping
u_i	The i-th node
$e^k_{i,j}$	The k-th entangled pair between u_i and u_j
$\mathbb{E}_\tau$	Existing entangled pairs in time slot τ
$E(i,j)$	# of entangled pairs between u_i and u_j
$C_{out}(i)$	# of occupied qubits on the outbound interface of u_i
$C_{in}(i)$	# of occupied qubits on the inbound interface of u_i
$\mathbb{S}^g_\tau$	Generation decision set in time slot τ
$\mathbb{S}^g_\tau$	Swapping decision set in time slot τ
M_τ	Instantaneous swapping contribution goal in time slot τ
f	Contribution function for each entangled pair
$\mathcal{A}_L$	The solution for path L
$\mathcal{A}^*_L$	The optimal solution for path L
$\text{EDR}_{\mathcal{A}_L}$	EDR in solution $\mathcal{A}_L$
$\mathcal{P}_{\text{MEDR}}$	Problem of maximizing EDR
$\mathcal{P}_{swap}$	Problem of entanglement swapping
$S[N]$	Best swapping decision for subproblem from node u_1 to u_N
$M[N]$	Maximized M_τ for subproblem from node u_1 to u_N

Proposition 1: For two algorithm A^* and A' and for any time slot τ, the following inequality holds: $E_\tau^{A^*_L}(u_1, u_N) \geq E_\tau^{A'_L}(u_1, u_N)$.

Proof: For $\tau < t^*$, A^* and A' has the same behavior and we can obtain $E_\tau^{A^*_L}(u_1, u_N) = E_\tau^{A'_L}(u_1, u_N)$. As for $t^* \leq \tau < t$, the entanglement distribution is not carried out in A' and $E_\tau^{A^*_L}(u_1, u_N) \geq E_\tau^{A'_L}(u_1, u_N)$. For $\tau \geq t$, we have that the entanglement distribution is carried out in both A^* and A', but $E_\tau^{A^*_L}(u_1, u_N) \geq E_\tau^{A'_L}(u_1, u_N)$, as memories to distribute the entanglement is faster to be reused to distribute other entangled pairs then A'_L. In conclusion, we have the proposition holds. Thus, we can conclude that A^* is not worse than A'. □

More specifically, the best entanglement generation strategy is to carry out every entanglement generation once quantum memories have free capability. Assume there are two strategies, A_g and A'_g, where A_g prefers to carry out entanglement generation greedily, and it performs generation fully till there is no free quantum memory available; while A'_g keeps the memory not fully used. The strategies of A_g and A'_g are almost the same before t, except that one new entanglement $e_{i,i+1}$ can be generated in A_g between u_i and u_{i+1} in t while A'_g chooses to postpone the execution of $\Delta t(> 0)$ time slots. We have the following proposition.

Proposition 2: For those two algorithms A_g and A'_g, we have $E_\tau^{A_{g_L}}(u_1, u_N) \geq E_\tau^{A'_{g_L}}(u_1, u_N)$ in any time slot τ.

The proof can be articulated as follows. The entanglement $e_{i,i+1}$ is either going to be a component which successfully help the distribution of the remote entangled pair $e_{1,N}$, or it decays due to the failure of quantum generation or swapping. In the success case, A_g generates $e_{i,i+1}$ faster than A'_g and

we can obtain: $E_\tau^{A_{g_L}}(u_1, u_N) \geq E_\tau^{A'_{g_L}}(u_1, u_N)$. While in the failure case, there is no additional entanglement distributed in both A_g and A'_g. However, the quantum memory used to generate $e_{i,i+1}$ can be reused faster in A_g than that in A'_g, where the quantum memory is occupied.

In conclusion, in order to maximize $\text{EDR}_{\mathcal{A}^*{}_L}$, entanglement generation should be performed whenever the quantum memory is free for generating an entangled pair between two adjacent nodes. Thus, we propose the entanglement generation in **Algorithm 1**. Specifically, it first counts the usage of the quantum memory on each node, which is represented as $C_{out}(i)$ and $C_{in}(i)$, to calculate the maximum number of entangled pairs that can be established between any two adjacent nodes. Once it is possible to generate a new entangled pair, the algorithm updates the generation decision set $\mathbb{S}_\tau^g$ to allow this generation operation. Since **Algorithm 1** is a greedy strategy and there is no trade-off between different generation options, it can be delegated to each quantum node. As a result, these quantum nodes can spontaneously and simultaneously execute **Algorithm 1** at the beginning of the time slot.

Algorithm 1 Algorithm for Entanglement Generation

Input: The current entanglement state, $\mathbb{E}_\tau$;
Output: $\mathbb{S}_\tau^g$
$\mathbb{S}_\tau^g \leftarrow \{\}$;
calculate the usage of the quantum memory on each node,
$C_{out}(i) = \sum_{j=i+1}^{N} E_\tau(i,j), C_{in}(i) = \sum_{j=1}^{i-1} E_\tau(j,i)$;
for $i = 1$ *to* $N-1$ **do**
  while $m - \max\{C_{out}(i), C_{in}(i+1)\} > 0$ **do**
    $\mathbb{S}_\tau^g = \mathbb{S}_\tau^g \cup \{e_{i,i+1}\}$;
    $C_{out}(i) = C_{out}(i) - 1$;
    $C_{in}(i+1) = C_{in}(i+1) - 1$;
  end
end
return $\mathbb{S}_\tau^g$;

V. Algorithm for Entanglement Swapping

In order to distribute remote entangled pairs, entanglement swapping must be conducted on every quantum router. However, the optimal strategy for swapping is not trivial, mainly due to two challenges.

The first challenge is that the entanglement swapping strategy should maximize EDR in Eq. 3. However, EDR is the time-averaged distributed entangled pairs over infinite time slots. Since the result is delayed after multiple time slots, it is difficult to guide the swapping algorithm instantly. Even the evaluation of EDR is also infeasible. Therefore, we propose an instantaneous swapping contribution goal M_τ and the corresponding algorithm to replace EDR in Section V-A, which can be evaluated instantly with low computational complexity.

Second, we model the entanglement swapping strategy as an optimization problem $\mathcal{P}_{swap}$. However, we find it is at least an NP-complete problem as it can be reduced to a weighted Maximum Independent Set problem (weighted MIS in Section V-B) [40]. The major difficulty comes from the fact that entanglement swapping needs two entangled pairs as the material. We call those two entangled pairs a swappable candidate (i.e., a swappable candidate contains two entangled pairs that share a node). However, one entangled pair can be used to group multiple swappable candidates, which leads to the conflict that only one of these candidates can be selected to perform an entanglement swapping. Inspired by the graph theory, we use a heuristic to decompose the original problem into multiple sub-problems in Section V-C. Then, dynamic programming algorithms can be used to solve sub-problems in polynomial time. As a result, we propose a heuristic algorithm to obtain an effective and efficient solution for the entanglement swapping problem in polynomial time.

A. M_τ: An Instantaneous Swapping Contribution Goal

The original optimization goal is to maximize EDR in Eq. (3). Nevertheless, this value is hard to evaluate in real time because it averages the distributed entangled pairs over infinite time slots. Only after some time slots can we observe the effectiveness of the decision at this time slot, but it is difficult to guide the swapping algorithm instantly. As a result, EDR is more likely to be an evaluation goal rather than an instructive goal.

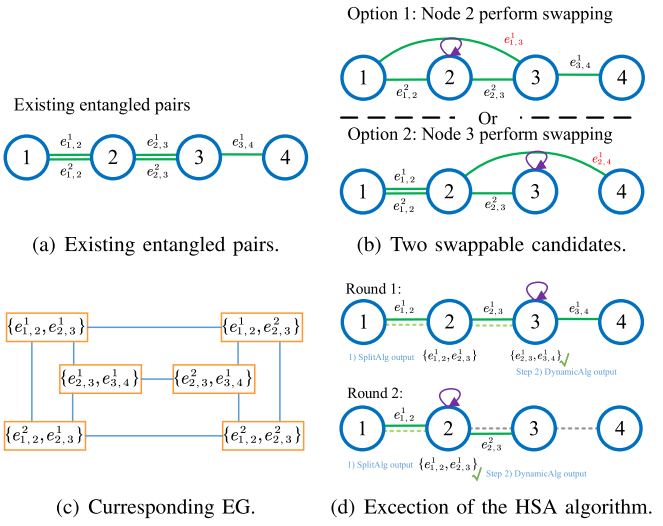
Real-time estimation of EDR is also prohibitive due to the computational complexity. Brand *et al.* [41] proposed an efficient method to estimate the waiting time for entanglement distribution in polynomial time, but the order of entanglement swapping is strictly fixed. As a result, it can only be used to evaluate the entanglement swapping in a specific mode, which we call Binary Tree Swapping Algorithm (BTSA). Apart from this, the estimation takes the time complexity $O(e^N)$ in general cases. Even worse, this estimation needs to be done several times for each optional swapping decision in each time slot. Consequentially, the overhead is not-negligible and unacceptable, for we expect the running time to be restricted to avoid entanglement decoherence even in a large-scale network.

To address this issue, we propose an instantaneous swapping contribution goal M_τ as the optimization goal, which can be calculated at a low cost:

$$M_\tau = \sum_{(e_{i,j}, e_{j,k}) \in \mathbb{S}_\tau^s} f(e_{i,j}, e_{j,k}), \tag{4}$$

where $f(e_{i,j}, e_{j,k})$ is a function that represent the contribution of a swappable candidate $(e_{i,j}, e_{j,k})$. Because the input of $f(\cdot)$ is only related to the existing entangled pairs, and its value is non-negative, representing the contribution, M_τ can be calculated with a negligible computational time overhead.

To determine the close form of the optimal objective M_τ, we leverage the fact that each entanglement swapping can make a contribution to the final EDR, and we use the function $f(e_{i,j}, e_{j,k})$ to quantify this contribution. As a result, we assume that the optimization goal M_τ is the sum of all entanglement swapping contributions in the current time slot. Another advantage of this form of M_τ is that it is neat to be an optimization goal and makes it easier to perform further



(a) Existing entangled pairs. (b) Two swappable candidates.

(c) Corresponding EG. (d) Excection of the HSA algorithm.

Fig. 3. Example of entangled pairs swapping conflict. Fig. 3(a) shows the existing entangled pairs on a 4-node path. Fig. 3(b) shows two available swappable candidates, but they conflict with each other. 3(c) shows the corresponding EG and 3(d) illustrates the two rounds of HSA.

optimization. However, it is difficult to find an optimal contribution function $f(\cdot)$, and we use a heuristic one as a replacement, which can be obtained through theoretical derivation or experimental simulation. For example, let $f(e_{i,j}, e_{j,k}) = 1$ be a constant function, M_τ represents the number of executions of entanglement swapping in the current time slot. In this case, the strategy for entanglement swapping is greedy, i.e., finding a swapping decision containing the most swappable candidates and allowing them to perform entanglement swapping.

B. Problem of Entanglement Swapping

The problem $\mathcal{P}_{swap}$ is now changed to select a set of the swappable candidate and perform swapping so that M_τ can be maximized.

Before presenting $\mathcal{P}_{swap}$ formally, we first introduce a concept called Entanglement Graph (EG) to describe the conflict of entangled pair selection. Each entangled pair can be potentially used by multiple entanglement swapping candidates and those candidates are conflicted with each other. An example of entanglement conflict is shown in Fig. 3(a) and (b). Consider a path with four nodes, and there are 5 entangled pairs $e^1_{1,2}$, $e^2_{1,2}$, $e^1_{2,3}$, $e^2_{2,3}$ and $e^1_{3,4}$ in Fig. 3(a). $e^1_{2,3}$ can be used to perform swapping with either $e^1_{1,2}$ or $e^2_{1,2}$ to distribute $e^1_{1,3}$. Alternatively, it can also be used to distribute $e^1_{2,4}$ with $e^1_{3,4}$. However, these three candidates are conflict and only one of them can be permitted. Fig. 3(b) shows two feasible swappable candidates that are conflicted with each other.

We now describe the problem $\mathcal{P}_{swap}$ formally and draw the following conclusions:

Proposition 3: $\mathcal{P}_{swap}$ can be reduced and formally expressed as a weighted Maximal Independent Set (MIS) problem.

Proof: We propose EG $G(\mathcal{V}, \mathcal{E})$ to express entanglement conflict. Herein $\mathcal{V}$ is the set of nodes representing every swappable candidate. A swappable candidate contains two entangled pairs that share the same node and can be used to perform entanglement swapping. Also, $\mathcal{E}$ is the set of edges that connect two swappable candidates if they use the same entangled pair. Thus, two adjacent swappable candidates can not perform entanglement swapping concurrently as they use the same entangled pair.

Edges in $\mathcal{E}$ represent two feasible but conflict entanglement swappable candidates. As shown in Fig. 3(c), G has six nodes that represent all swappable candidates. In Fig. 3(b), there are edges between $(e^1_{1,2}, e^1_{2,3})$ and $(e^1_{2,3}, e^1_{3,4})$ because they all contain a specific entanglement $e^1_{2,3}$.

The goal of $\mathcal{P}_{swap}$ is to maximize M_τ by selecting an independent set in G, as two connected swappable candidates are conflict in G.. Notice that the input parameter of f is the two entangled pairs in a swappable candidate, and f can also be considered as the weight of nodes in G. Therefore, the entanglement swapping problem, i.e., $\mathcal{P}_{swap}$, can be equivalently described as the following integer programming problem:

$$\begin{aligned} \max\ & M_\tau = \sum_{v \in \mathcal{V}_{sel}} f(v), \\ \text{s.t. } & \mathcal{V}_{sel} = \{v | x_v = 1, \quad \forall v \in \mathcal{V}\}, \\ & N(v) \cup \mathcal{V}_{sel} = \varnothing, \quad \forall v \in \mathcal{V}_{sel}, \\ & x_v \in \{0, 1\}, \quad \forall v \in \mathcal{V}, \end{aligned} \tag{5}$$

where x_v is the boolean for whether select $v \in \mathcal{V}$ to execute swapping and $\mathcal{V}_{sel}$ is the selected subset of swapping candidates. $N(v)$ denotes the neighbor set of v.

Finally, $\mathcal{P}_{swap}$ can be reduced into weighted-MIS, and the common *MIS* is a special case of $\mathcal{P}_{swap}$ when the contribution function is $f(v) = 1$. □

$\mathcal{P}_{swap}$ is at least an NP-complete problem as *MIS* is proved to be an NP-complete problem. It means that there is no optimal polynomial-time algorithm to solve $\mathcal{P}_{swap}$. Therefore, we propose a heuristic algorithm in the following section.

C. HSA: A heuristic algorithm for Entanglement Swapping

There is no efficient algorithm for searching all *MISs* in polynomial time [42], [43], so $\mathcal{P}_{swap}$ is hard to solve as well. Nonetheless, we observe that if the degree of G is less than 2, the *MIS* problem has a polynomial-time solution [44]. It is equivalent to the case of the single-tunnel model (i.e., $m = 1$), as there are at most two nodes that will use a common entangled pair to perform entanglement swapping, and there is no loop in G. Therefore, we propose a heuristic algorithm based on the basic idea of dividing $\mathcal{P}_{swap}$ into several subproblems $\mathcal{P}^{m=1}_{swap}$ where $\mathcal{P}^{m=1}_{swap}$ is the strict version of $\mathcal{P}_{swap}$ in the single-tunnel model.

Then, we propose a Dynamic Programming (DP) algorithm to address $\mathcal{P}^{m=1}_{swap}$ as illustrated in **Algorithm 2**. The idea of the algorithm is to first solve the problem on a partial path from node u_1 to u_i. The maximized M_τ on this partial path is recorded in notation $M[i]$, and the best strategy is recorded in $S[i]$. Then, the algorithm will solve the problem from node u_1 to u_{i+1}, and finally, the algorithm gives the best entanglement swapping strategy on the whole path. The entanglement swapping decision is in $S[N]$, and $M[N]$ is

Algorithm 2 DynamicAlg: Algorithm for Entanglement Swapping in Single-Tunnel Model ($m = 1$)

Input: The current entanglement state, $\mathbb{E}_\tau$; Nodes on the Path, $\mathbb{U}$; The total number of nodes, N;
Output: The strategy $\mathbb{S}^s_\tau$; And the maxmized M_τ;

$\forall u_i, M[i] \leftarrow 0, S[i] \leftarrow \{\}$;
for $u_j \in \mathbb{U},\ s.t.\ e_{i,j}$ *and* $e_{j,k} \in \mathbb{E}_\tau$ **do**
  if $(*, e_{i,j}) \notin S[i]$ **then**
    // no conflict exists, allow u_j to perform swapping.
    $S[j] = S[j-1] \cup \{(e_{i,j}, e_{j,k})\}$;
    $M[j] = M[j-1] + f(e_{i,j}, e_{j,k})$;
  else
    $(e_{q,i}, e_{i,j}) \in S[i]$;
    // find u_i that uses $e_{i,j}$
    if $M[q] + f(e_{i,j}, e_{j,k}) > M[i]$ **then**
      // allow u_j to use $e_{i,j}$ and cancel u_i's swapping.
      $S[j] = S[j-1]/(e_{q,i}, e_{i,j}) \cup \{(e_{i,j}, e_{j,k})\}$;
      $M[j] = M[j-1] - f(e_{q,i}, e_{i,j}) + f(e_{i,j}, e_{j,k})$;
    else
      // reject u_j to perform swapping.
      $S[j] = S[i-1]$;
      $M[j] = M[i-1]$;
    end
  end
end
Return $\mathbb{S}^s_\tau \leftarrow S[N], M_\tau \leftarrow M[N]$;

Algorithm 3 SplitAlg: Algorithm for Entanglement Swapping in Multi-Tunnel Model ($m > 1$)

Input: The current entanglement state, $\mathbb{E}_\tau$; Nodes on the Path, $\mathbb{U}$; The total number of nodes, N; The size of a quantum memory, m;
Output: The entanglement swapping strategy $\mathbb{S}^s_\tau$;

while *True* **do**
  $C \leftarrow \{\}$;
  for $u_j \in \mathbb{U}, s.t.\ e_{i,j}$ *and* $e_{j,k} \in \mathbb{E}_\tau$ **do**
    $e_1 \leftarrow e_{i,j}\ s.t.\ \forall e_{i,j}, e_{p,j} \in \mathbb{E}_\tau, i \le p$;
    $e_2 \leftarrow e_{j,k}\ s.t.\ \forall e_{j,q}, e_{j,k} \in \mathbb{E}_\tau, k \ge q$;
    $C \leftarrow C \cup (e_1, e_2)$;
  end
  if $C = \varnothing$ **then**
    Break;
  end
  $\mathbb{S}^s_\tau \leftarrow \mathbb{S}^s_\tau \cup DynamicAlg(C, u, N, m)$;
  $\mathbb{E}_\tau \leftarrow \mathbb{E}_\tau / \{e_1, e_2\},\ \forall (e_1, e_2) \in \mathbb{S}^s_\tau$;
  // Update unallocated entanglement
end
Return $\mathbb{S}^s_\tau$;

the maximized goal. To be more specific, as mentioned in Section V-A, $f(\cdot)$ is non-negative and M_τ is monotonically increasing. Thus, if the current two entangled pairs can be used to perform swapping and this swapping does not conflict with other swapping decisions, the algorithm should allow it to be executed and put it into $\mathbb{S}^s_\tau$. When there is a conflict, the algorithm must choose the best option to maximize M_τ. Without loss of generality, we assume u_j can perform swapping with $(e_{i,j}, e_{j,k})$ while $u_i (i < j)$ is allowed to perform swapping with $(e_{q,i}, e_{i,j})$. Since both options use the same entangled pair $e_{i,j}$, they cannot be permitted simultaneously. The algorithm blocks u_j's execution when

$$M[i] \geq M[q] + f(e_{i,j}, e_{j,k}),$$

which means that $(e_{q,i}, e_{i,j})$ is a better option to maximize $M[i]$. Otherwise, the algorithm revokes u_i from executing and chooses u_j to execute swapping. In general, the recursive expression of the algorithm is

$$M[j] = \max\left(M[i], M[q] + f(e_{i,j}, e_{j,k})\right).$$

As a result, the problem $\mathcal{P}^{m=1}_{swap}$ can be solved in the time complexity of $O(N)$.

In case of multi-tunnel model (i.e., $m > 1$), the proposed algorithm should divide the problem $\mathcal{P}_{swap}$ into multiple $\mathcal{P}^{m=1}_{swap}$ sub-problems. We implement the algorithm through two heuristic ideas: First, priority is given to entangled pairs with longer distances. This is because their fidelity levels are usually low and it is easy for them to lose coherence after multiple swapping operations compared to the shorter distance entangled pairs. This will further cause blocking of the distribution of other entangled pairs. Second, each sub-problem expects to deal with the most number of swappable candidates if it does not violate the assumption of $m = 1$ to reduce the time cost of solving the whole problem to find the optimal solution and reduce the runtime overhead.

Algorithm 3 shows how divide the original problem into multiple $\mathcal{P}^{m=1}_{swap}$ sub-problems. It runs multiple rounds. In each round, it generates a sub-problem and solves it using **Algorithm 2**. The strategy used to generate sub-problems is that the controller only produces one swappable candidate on each node. This strategy guarantees that the degree of the EG in the generated sub-problem is less than 2. It is because a Bell state entangled pair has two qubits and is related to at most two nodes. If the two nodes use the same entangled pair to generate swappable candidates, this entangled pair will be used in no more than two swappable candidates.

In each round, **Algorithm 3** checks all nodes in turn and looks for two entangled pairs with the longest distance on each of interfaces to form a swappable candidate. It takes the selected two entangled pairs as a swappable candidate of this node and puts this swappable candidate in the candidates set (C). After calculating a swappable candidate on every node, multiple swappable candidates are selected, but they could be conflicted with each other. Thus, the algorithm uses the candidate set (C) as the input parameter and executes **Algorithm 2** to solve the sub-problem. After that, some swappable candidates from C are chosen to perform entanglement swapping while others are revoked. The algorithm then updates $\mathbb{E}_\tau$ and $\mathbb{S}^s_\tau$ and starts the next round till no more swappable candidates can be chosen.

An example of **Algorithm 3** is illustrated in Fig. 3(d). In this case, there exists $N = 4$ nodes (from u_1 to u_4) and 5 entangled pairs. In the first round, a swappable candidate is selected on each node in **Algorithm 3**. As a result, ($e^1_{1,2}$, $e^1_{2,3}$) and ($e^1_{2,3}$, $e^1_{3,4}$) are selected. Then, the controller performs **Algorithm 2**, and since the above two swappable candidates are in conflict (as they use the same entangled pair $e^1_{2,3}$), only one swappable candidate is allowed. Let us assume that **Algorithm 2** allows u_3 to perform swapping through using $e^1_{2,3}$ and $e^1_{3,4}$, while $e^1_{1,2}$ is left into the next round. In the second round, the controller chooses the candidate ($e^1_{1,2}$, $e^2_{2,3}$) and allows it to be performed. Finally, since there are no swappable candidate left, **Algorithm 3** exits, and $e^2_{1,2}$ is reserved for the next time slot. At a result, ($e^1_{2,3}$, $e^1_{3,4}$) and ($e^1_{1,2}$, $e^2_{2,3}$) is allowed to perform entanglement swapping.

D. Computational Complexity Analysis

Although it is difficult to estimate the average time complexity of **Algorithm 3**, we can get the upper bound of the algorithm's running time in the worst case. HSA can be considered acceptable whenever this upper bound can be acceptable. Simulation in the Section VI shows that even in the strictest scenarios, the average time complexity of the algorithm is much better than its worst-case runtime complexity discussed in this section.

*Lemma 1: The worse-case runtime complexity of **Algorithm 2** is $O(N)$.*

Proof: **Algorithm 2** is a dynamic programming algorithm, which traverses every quantum node $u_i \in \mathbb{U}$ in line 2, so its time complexity is $O(N)$. When entangled pairs exist between every adjacent node, the upper bound is reached. □

*Lemma 2: The main loop (in line 1 of **Algorithm 3**) needs to be executed at most $m \cdot \lceil \frac{N}{2} \rceil$ times.*

Proof: It is mainly because $|C|$ is at least 1, where $|C|$ is the cardinality of C (first defined in line 1). Otherwise, the algorithm will jump out of the outer loop and exit. It also means that at least one entanglement swapping needs to be determined each round before **Algorithm 2** is executed.

When m and N are given, the maximum number of entanglement swapping that can be performed in a time slot is $m \cdot \lceil \frac{N}{2} \rceil$. This bound will be reached when the quantum memory is fully used, and all entangled pairs between adjacent nodes are generated. Consequentially, the main loop runs at most $m \cdot \lceil \frac{N}{2} \rceil$ times in the worst cases. □

*Proposition 4: In the worst case, the runtime complexity of **Algorithm 3** is $O(m \cdot N^4)$.*

Proof: When considering the runtime complexity of the inner loop in line 3, the algorithm traverses each quantum nodes and find two entangled pairs e_1 and e_2. For u_j, the algorithm traverses from u_1 to u_{j-1} and looks for the entangled pairs with the longest distance. Then, it traverses from u_{j+1} to u_N to find another entanglement. It is obvious that the runtime complexity for all nodes is

$$\sum_{i=1}^{N}((i-1)+(N-i)) = N \cdot (N-1).$$

From Lemma 2, the maximum number of execution of **Algorithm 2** is $m \cdot \lceil \frac{N}{2} \rceil$. From Lemma 1, the complexity of **Algorithm 2** is $O(N)$. Therefore, the worst runtime complexity is

$$O(m \cdot N \cdot N \cdot (N-1) \cdot N) = O(m \cdot N^4). \tag{6}$$

□

Discussion: Since N are small constants (even the number of hops in Internet is usually less than 128), the total runtime complexity is acceptable. Further experiments in Section VI-C show that average time complexity of the algorithm is $O(m \cdot N^{2.109})$ in a relatively strict scenario.

VI. Numeric Simulation and Evaluation

This section conducts simulations and investigates the performance and runtime complexity of HSA under different settings. Expressly, we set up different scenes given N, m, p_{swap} and p_{gen}, and then we calculate EDR after a fixed number of time slots as the criterion for the performance of algorithms. We also perform simulations to evaluate the average runtime of our algorithm.

A. Experiment Environment and Baselines

The results of the simulations are obtained from commodity hardware (precisely, a single logical processor of an Intel i5-8259U @ 2.3 GHz and 8GB 2133 MHz RAM). In our evaluations, we fix the time slot to 1000 rounds. We find that EDR has been stable in 1000 time slots and can be used to evaluate the algorithm. More time slots will only increase the computational cost of simulations without any further advantages.

We compare the proposed algorithm with other trivial algorithms, including Ordered Swapping Algorithm (OSA) and Binary Tree Swapping Algorithm (BTSA) [34]. In OSA, the entanglement swapping is carried out from the source to the destination hop by hop. Specifically, node u_2 will perform swapping to distribute $e_{1,3}$, followed by node u_3 perform swapping with $e_{1,3}$ and $e_{3,4}$ to distribute $e_{1,4}$ in the next time slot. For multi-tunnel models (i.e., $m > 1$), each tunnel performs entanglement swapping independently in each time slot. While in BTSA, entangled pairs of the same distance perform swapping in pairs. Specifically, the odd-numbered entangled pairs (for example, the i-th entangled pair) perform swapping with the next entangled pair (i.e., $i+1$-th entangled pair). If the consumed entangled pairs are n hop, the new generated entangled pair will be $2n$ hop. As a result, BTSA uses $\log_2 N$ rounds to distribute an end-to-end entangled pair on a N hop path. Fig. 4 shows an example that a quantum path of five nodes (from u_1 to u_5). u_2 and u_4 performs swapping firstly to distribute $e_{1,3}$ and $e_{3,5}$. Then, u_3 performs swapping to distribute the proposed remote entangled pair $e_{1,5}$. Both OSA and BTSA are stop-and-wait algorithms. When a necessary entangled pair is not present, the algorithm will stop and wait until it is generated.

In this paper, we propose a heuristic algorithm called HSA. However, the contribution function f in HSA has not yet been determined. In the evaluations, we use two heuristic views to select f and also constitute two candidates, HSA$_g$ and HSA$_m$, respectively. In HSA$_g$, the algorithm tends to perform more

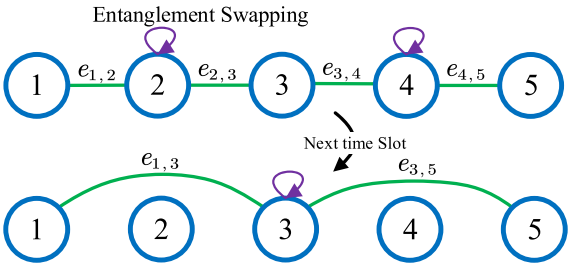


Fig. 4. Example of BTSA algorithm on a $N = 5$ path.

entanglement swappings in each time slot. To achieve this goal, we use a constant function $f_g(e_{i,j}, e_{j,k}) = 1$. In this case, M_τ indicates the number of entanglement swapping allowed to be performed in time slot τ.

While in HSA$_m$, we prefer to choose entangled pairs with longer distance (i.e., the distance between two nodes) to participate in swapping and because they usually have low fidelity and are easy to decoherence. Therefore, the contribution function in HSA$_m$ is

$$f_m(e_{i,j}, e_{j,k}) = \max\{j - i, k - j\}. \tag{7}$$

B. Simulation Results

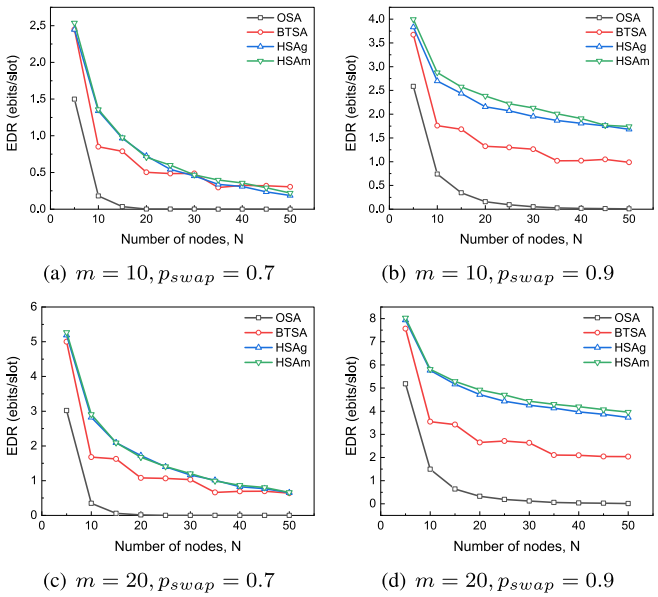
To evaluate the performance of those algorithms, we use the averaged entanglement distribution rate (EDR) in 1000 time slots as an evaluation indicator. Another feasible indicator is the time it takes for the first entanglement distribution. However, in the experimental evaluation, we do not adopt it because entangled pairs need to be continuously distributed in most quantum network scenarios.

First, we consider the performance at different node scales. The parameter p_{gen} fits

$$p_{gen} = 10^{-\gamma \cdot D/10},$$

where D is the distance between two nodes and loss rate γ is 0.02 [45], [46]. Considering a quantum network, we give priority to deploy more quantum nodes to reduce the distance between nodes and ensure the success probability of entanglement generation p_{gen}. Therefore, we set p_{gen} to 0.9 and examine the performance on a path with up to 50 nodes. We fix the success probability of entanglement swapping p_{swap} to 0.7 or 0.9, and the memory size of $m = 10$ or $m = 20$. The results of the simulations are shown in Fig. 5.

Overall, HSA$_m$ has a better performance in most scenarios, followed by HSA$_g$. These two algorithms achieve higher EDR than the two baseline algorithms. OSA is almost infeasible in large-scale networks because it is difficult to distribute remote entangled pairs. In addition, we also observe the following phenomena. First, EDR decreases with the number of nodes increasing in all algorithms. Averaged EDR is 4.001 ebits/slot when $N = 5$, whereas it is 1.757 ebits/slot when $N = 50$ in HSA$_m$, which is shown in Fig. 5(b). Second, HSA$_g$ and HSA$_m$ are better than OSA and BTSA, for the curve of both HSA algorithms is above the baseline curves as a whole. It can be further observed that the performance of OSA drops



(a) $m = 10, p_{swap} = 0.7$ (b) $m = 10, p_{swap} = 0.9$

(c) $m = 20, p_{swap} = 0.7$ (d) $m = 20, p_{swap} = 0.9$

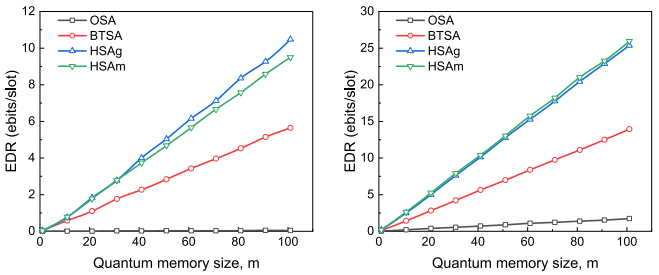
Fig. 5. Averaged EDR with different node scales.

drastically as N increases. It is almost impossible to distribute remote entangled pairs reliably. For example, when averaged EDR reaches 3.037 ebits/slot for $N = 5$ while it drops to 0.005 ebits/slot when $N = 15$ in Fig. 5(c). Third, EDR in BTSA drops as a wave shape, which significantly degrades when N is an integer power of 2. It is because it takes one more round to distribute remote entangled pairs when N exceeds these integer powers of 2 (i.e., 2, 4, 8 …). As a result, each distribution of entangled pairs requires an additional time slot.

In the next experiments, we fix the network scale and then examine the impact of the quantum memory size on EDR. We fix the number of quantum nodes to 20 and then evaluate the averaged EDR when m increases from 1 to 100. As shown in Fig. 6, with the increase of the memory size, EDR grows approximately linearly. However, with the increase of each qubit memory, EDR increases faster in HSA$_g$ and HSA$_m$ than BTSA and OSA, which shows the proposed algorithm can make better use of the limited size of the quantum memory. To be more specific, Fig. 6(a) illustrate averaged EDR when $p_{swap} = 0.7$ and $p_{gen} = 0.9$. The EDR increment of each qubit of the quantum memory is 0.1061 ebits/slot in HSA$_g$, which is approximately 87.76% higher than 0.05651 in BTSA. In Fig. 6(b), it is 0.2537 (0.2587) in HSA$_g$ (HSA$_m$) and is 83.31% (86.92%) higher than 0.1384 in BTSA respectively.

With the development of quantum devices, the probability of entanglement generation p_{gen} and entanglement swapping p_{swap} will increase. On the other hand, the quantum memory size m may also increase on quantum nodes. Therefore, to evaluate EDR in a wider scene, we conduct the following experiments to evaluate EDR when p_{gen} and p_{swap} vary. Consider a quantum path with 20 nodes and $m = 10$, and we use the following indicator to illustrate the advantage of the algorithm HSA$_m$:

$$Adv_{\mathrm{HSA}_m} = \frac{EDR_{\mathrm{HSA}_m} - \max\left(EDR_{\mathrm{OSA}}, EDR_{\mathrm{BTSA}}, EDR_{\mathrm{HSA}_g}\right)}{\max\left(EDR_{\mathrm{OSA}}, EDR_{\mathrm{BTSA}}, EDR_{\mathrm{HSA}_g}\right)},$$



(a) $N = 20, p_{swap} = 0.7$ (b) $N = 20, p_{swap} = 0.9$

Fig. 6. Averaged EDR under different quantum memory sizes.

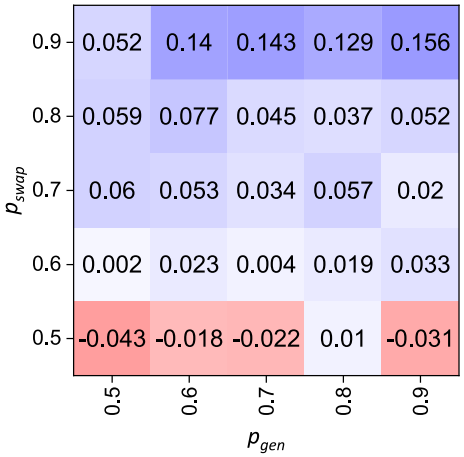


Fig. 7. This is the heat map that shows the advantage of HSA$_m$. Darker blue indicates a greater advantage compared to other three algorithms.

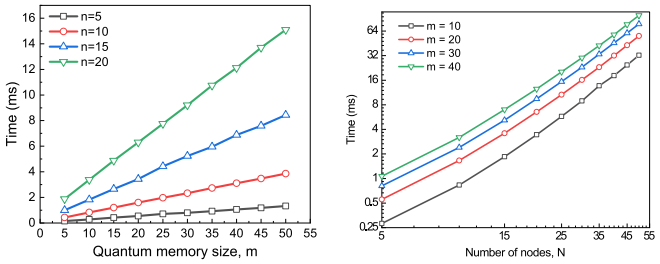
where EDR_{HSA_m} is the averaged EDR of HSA$_m$, so as other variables. We set p_{gen} and p_{swap} between 0.5 and 1, and form the heat map in Fig. 7.

In most scenarios where $p_{swap} \geq 0.6$, HSA$_m$ has advantages compared with two baselines and even HSA$_g$. When p_{swap} is smaller than 0.5, the probability of distributing remote entangled pairs is almost close to 0, and so is EDR. From another perspective, p_{gen} has little influence on EDR under the same conditions. The advantage between algorithms is relative constant, which indicates that HSA$_m$ performs better in a wild scene. With the development of quantum information technology, the advantage of the proposed algorithm will increase.

C. Estimate the Average Runtime complexity of HSA

We also evaluate the average time complexity based on the following experiments. We execute the algorithm under a variable node scale N and the quantum memory size m. Note that our simulations are implemented based on Python 3 and run on a single-threaded processor. The following experimental data does not represent the absolute performance of the algorithm. However, it is possible to use its relative performance to explore the runtime complexity of HSA.

In the first experiment, we set both p_{gen} and p_{swap} to be 0.9. We evaluate HSA$_m$'s runtime in different node scale settings. More specific, we set N to be 5, 10, 15 and 20,



(a) Time v.s. memory size (m) (b) Time v.s. nodes number (N)

Fig. 8. Evaluate the averaged runtime of HSA$_m$.

TABLE II
POWER OF N AND CORRELATION COEFFICIENT

m	10	20	30	40
x [1]	2.108523	2.035238	2.020934	1.998706
R [2]	0.995596	0.996497	0.996619	0.997248

[1] x is the exponent of N, which measures the progressive performance of the algorithm
[2] R represents the correlation coefficient

and then measure the execution time of the algorithm for the quantum memory size from $m = 5$ to $m = 50$ as shown in Fig. 8(a). Experimental results show that HSA$_m$'s averaged runtime complexity is proportional to m, which is consistent with Eq. (6). In addition, we find that p_{gen} and p_{swap} have no significant effect on the complexity in additional experiments.

Then we examine how N affects the algorithm runtime complexity. We construct a path with $m = 10, 20, 30$ and 40. The influence of the number of nodes on the runtime complexity of HSA$_m$ is discussed. The variable N varies from 5 to 50 to evaluate the increase in runtime. The simulation results are presented in Fig. 8(b) in the form of a double logarithmic curve. Since the curve approximates a straight line in the logarithmic graph, we consider the time t a polynomial function of N (i.e., $t = C \cdot N^x$) and the low-order terms are negligible, we can perform a linear regression to the double logarithmic curve. Its slope can be viewed as the exponent x for the constraint

$$x = \frac{\ln(t) - \ln(C)}{\ln(N)}$$

holds, where both x and $\ln(C)$ are obtained directly from the linear regression. The result is shown in TABLE II. When $m = 10$, we get the scope is 2.109 while it is 1.999 when $m = 40$. The scope even slightly drops when the memory size is larger, which means HSA can get higher progressive performance with a larger quantum memory size. In all settings, the average runtime complexity of the algorithm is less than $O(N^{2.109})$, which is far better than its worst bound $O(N^4)$. We also calculate the correlation coefficient, which is above 0.995 in every setting. Our assumption that t is a polynomial function about N is acceptable. We can also observe that the correlation coefficient gets close to 1 when m increases.

In these two experiments, we set both p_{gen} and p_{swap} to 0.9. Since the complexity is related to the number of

existing entangled pairs on the path, with more entangled pairs exist, the more time HSA consumes. We explore a scenario where both p_{gen} and p_{swap} are very close to 1 so that the number of entangled pairs is relatively big to achieve a strict result. The experiments show that even in this scenario, the increase in the running time of HSA is limited and acceptable.

D. Discussion: the Close Form of M_τ and f

So far, the exact form of M_τ remains flexible. We choose two metrics heuristically in the experiments and form the corresponding algorithm HSA_g and HSA_m. According to the experimental results, HSA_m works better in most games than HSA_g and baselines as well, while HSA_g wins a few games. For example, we can see the line of HSA_g is slightly above the line of HSA_g in Fig. 6(a).

According to the experimental results, there may be no simple form of M_τ that enables the algorithm to achieve the best EDR in every scenario. For example, when p_{swap} is relatively tiny, BTSA works fine because it performs only necessary entanglement swappings to avoid entanglement decoherence while it cannot fully use the quantum memory in the opposite scenarios. While in most cases, HSA_m achieves a better EDR. On the other hand, our current understanding of quantum networks is not sufficient. Especially for the value of p_{gen}, p_{swap}, and m may change significantly with the development of our knowledge and quantum devices. Therefore, it may be unnecessary to decide the close form of the instant optimal goal M_τ and the swapping contribution function $f(\cdot)$ for a specific setting. This work can be left for further studies. However, as demonstrated in the experiment, HSA_m has advantages in the most common scenarios.

VII. Conclusion

To improve the performance of the quantum network, we focus on the remote end-to-end entanglement distribution problem over paths in which quantum routers have limit-sized memories. In our design, we consider the imperfection of quantum operations, the limited memory size, and the vulnerability of entangled pairs.

The solution to our proposed design is based on an entanglement distribution framework. Under this framework, our generation algorithm works greedily and attempts to use quantum memory fully. Also, we introduce a deterministic polynomial-time algorithm to make entanglement swapping decisions. Here, the swapping algorithm uses a heuristic to divide the original NP-complete problem into several sub-problems that a dynamic programming algorithm can easily solve. The evaluations show that our algorithm achieves a high distribution rate level and fully uses the limited size of quantum memory. On the other hand, the proposed algorithm has a polynomial-time upper bound, which makes it an acceptable solution in a large-scale quantum network.

References

[1] H. J. Kimble, "The quantum internet," *Nature*, vol. 453, no. 7198, pp. 1023–1030, Jun. 2008.

[2] H. De Riedmatten, I. Marcikic, J. A. W. Van Houwelingen, W. Tittel, H. Zbinden, and N. Gisin, "Long-distance entanglement swapping with photons from separated sources," *Phys. Rev. A, Gen. Phys.*, vol. 71, May 2005, Art. no. 050302.

[3] Z. Li *et al.*, "Building a large-scale and wide-area quantum internet based on an OSI-alike model," *China Commun.*, vol. 18, no. 10, pp. 1–14, Oct. 2021.

[4] A. S. Fletcher, P. W. Shor, and M. Z. Win, "Channel-adapted quantum error correction for the amplitude damping channel," *IEEE Trans. Inf. Theory*, vol. 54, no. 12, pp. 5705–5718, Dec. 2008.

[5] A. K. Ekert, "Quantum cryptography based on Bell's theorem," *Phys. Rev. Lett.*, vol. 67, no. 6, pp. 661–663, Aug. 1991.

[6] C. H. Bennett, G. Brassard, and N. D. Mermin, "Quantum cryptography without Bell's theorem," *Phys. Rev. Lett.*, vol. 68, pp. 557–559, Feb. 1992.

[7] J. Preskill, "Quantum computing in the NISQ era and beyond," *Quantum*, vol. 2, pp. 79–99, Aug. 2018.

[8] P. W. Shor, "Algorithms for quantum computation: Discrete logarithms and factoring," in *Proc. 35th Annu. Symp. Found. Comput. Sci.*, 1994, pp. 124–134.

[9] S.-K. Liao *et al.*, "Satellite-to-ground quantum key distribution," *Nature*, vol. 549, pp. 43–47, Sep. 2017.

[10] J.-G. Ren *et al.*, "Ground-to-satellite quantum teleportation," *Nature*, vol. 549, pp. 70–73, Sep. 2017.

[11] H. J. Briegel, W. Dür, J. I. Cirac, and P. Zoller, "Quantum repeaters: The role of imperfect local operations in quantum communication," *Phys. Rev. Lett.*, vol. 81, pp. 5932–5935, Dec. 1998.

[12] N. Sangouard, C. Simon, H. De Riedmatten, and N. Gisin, "Quantum repeaters based on atomic ensembles and linear optics," *Rev. Modern Phys.*, vol. 83, no. 1, pp. 33–80, Mar. 2011.

[13] R. Van Meter and J. Touch, "Designing quantum repeater networks," *IEEE Commun. Mag.*, vol. 51, no. 8, pp. 64–71, Aug. 2013.

[14] S. Shi and C. Qian, "Concurrent entanglement routing for quantum networks: Model and designs," in *Proc. ACM Conf. Appl., Technol., Archit. Protocols Comput. Commun. (SIGCOMM)*, 2020, pp. 62–75.

[15] M. Pant *et al.*, "Routing entanglement in the quantum internet," *npj Quantum Inf.*, vol. 5, no. 1, pp. 25–36, Dec. 2019.

[16] C. Li, T. Li, Y.-X. Liu, and P. Cappellaro, "Effective routing design for remote entanglement generation on quantum networks," *npj Quantum Inf.*, vol. 7, no. 1, pp. 10–22, 2021.

[17] S. Pirandola, "End-to-end capacities of a quantum communication network," *Commun. Phys.*, vol. 2, no. 1, pp. 51–61, Dec. 2019.

[18] L. Gyongyosi and S. Imre, "Decentralized base-graph routing for the quantum internet," *Phys. Rev. A, Gen. Phys.*, vol. 98, no. 2, Aug. 2018, Art. no. 022310.

[19] K. Chakraborty, D. Elkouss, B. Rijsman, and S. Wehner, "Entanglement distribution in a quantum network: A multicommodity flow-based approach," *IEEE Trans. Quantum Eng.*, vol. 1, pp. 1–21, 2020.

[20] C. Godsil and G. F. Royle, *Algebraic Graph Theory*, vol. 207. New York, NY, USA: Springer, 2001.

[21] R. Bellman, "Dynamic programming," *Science*, vol. 153, no. 3731, pp. 34–37, Jul. 1966.

[22] M. Sniedovich, *Dynamic Programming: Foundations and Principles*. Boca Raton, FL, USA: CRC Press, 2010.

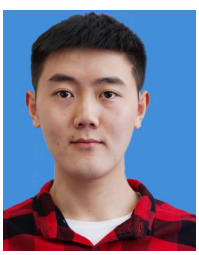
[23] A. S. Cacciapuoti, M. Caleffi, F. Tafuri, F. S. Cataliotti, S. Gherardini, and G. Bianchi, "Quantum internet: Networking challenges in distributed quantum computing," *IEEE Netw.*, vol. 34, no. 1, pp. 137–143, Jan. 2019.

[24] C. H. Bennett, G. Brassard, C. Crépeau, R. Jozsa, A. Peres, and W. K. Wootters, "Teleporting an unknown quantum state via dual classical and Einstein-Podolsky-Rosen channels," *Phys. Rev. Lett.*, vol. 70, no. 13, pp. 1895–1899, Mar. 1993.

[25] J. Illiano, M. Caleffi, A. Manzalini, and A. S. Cacciapuoti, "Quantum internet protocol stack: A comprehensive survey," 2022, *arXiv:2202.10894*.

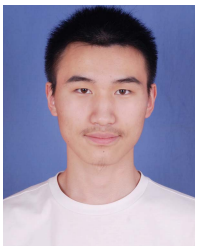
[26] L.-M. Duan, M. D. Lukin, J. I. Cirac, and P. Zoller, "Long-distance quantum communication with atomic ensembles and linear optics," *Nature*, vol. 414, no. 6862, pp. 413–418, Nov. 2001.

[27] K. Goodenough, D. Elkouss, and S. Wehner, "Optimizing repeater schemes for the quantum internet," *Phys. Rev. A, Gen. Phys.*, vol. 103, no. 3, Mar. 2021, Art. no. 032610.
[28] B. Julsgaard, J. Sherson, J. I. Cirac, J. Fiurášek, and E. S. Polzik, "Experimental demonstration of quantum memory for light," *Nature*, vol. 432, no. 7016, pp. 482–486, Nov. 2004.
[29] S. Pirandola, "Capacities of repeater-assisted quantum communications," 2016, *arXiv:1601.00966*.
[30] Y. Zhao and C. Qiao, "Redundant entanglement provisioning and selection for throughput maximization in quantum networks," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, May 2021, pp. 1–10.
[31] S. Zhang, S. Shi, C. Qian, and K. L. Yeung, "Fragmentation-aware entanglement routing for quantum networks," *J. Lightw. Technol.*, vol. 39, no. 14, pp. 4584–4591, Apr. 28, 2021.
[32] E. Shchukin, F. Schmidt, and P. Van Loock, "Waiting time in quantum repeaters with probabilistic entanglement swapping," *Phys. Rev. A, Gen. Phys.*, vol. 100, no. 3, 2019, Art. no. 032322.
[33] L. Jiang, J. M. Taylor, N. Khaneja, and M. D. Lukin, "Optimal approach to quantum communication using dynamic programming," *Proc. Nat. Acad. Sci. USA*, vol. 104, no. 44, pp. 17291–17296, 2007.
[34] N. K. Bernardes, L. Praxmeyer, and P. van Loock, "Rate analysis for a hybrid quantum repeater," *Phys. Rev. A, Gen. Phys.*, vol. 83, no. 1, Jan. 2011, Art. no. 012323.
[35] W. Dai, T. Peng, and M. Z. Win, "Optimal remote entanglement distribution," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 3, pp. 540–556, Mar. 2020.
[36] J. W. Pan, C. Simon, Č. Brukner, and A. Zeilinger, "Entanglement purification for quantum communication," *Nature*, vol. 410, no. 6832, pp. 1067–1070, 2001.
[37] W. Dür, H.-J. Briegel, J. I. Cirac, and P. Zoller, "Quantum repeaters based on entanglement purification," *Phys. Rev. A, Gen. Phys.*, vol. 59, no. 1, pp. 169–181, Jan. 1999.
[38] A. Dahlberg *et al.*, "A link layer protocol for quantum networks," in *Proc. ACM Conf. Appl., Technol., Archit. Protocols Comput. Commun. (SIGCOMM)*, 2019, pp. 159–173.
[39] G. Vardoyan, S. Guha, P. Nain, and D. Towsley, "On the stochastic analysis of a quantum entanglement distribution switch," *IEEE Trans. Quantum Eng.*, vol. 2, pp. 1–16, 2021.
[40] J. Hartmanis, "Computers and intractability: A guide to the theory of NP-completeness (M. R. Garey and D. S. Johnson)," *Siam Rev.*, vol. 24, no. 1, p. 90, 1982.
[41] S. Brand, T. Coopmans, and D. Elkouss, "Efficient computation of the waiting time and fidelity in quantum repeater chains," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 3, pp. 619–639, Mar. 2020.
[42] P. Berman and T. Fujito, "On approximation properties of the independent set problem for low degree graphs," *Theory Comput. Syst.*, vol. 32, no. 2, pp. 115–132, Mar. 1999.
[43] Y. Liu, J. Lu, H. Yang, X. Xiao, and Z. Wei, "Towards maximum independent sets on massive graphs," in *Proc. VLDB Endowment*, 2015, pp. 2122–2133.
[44] R. Faudree, E. Flandrin, and Z. Ryjáček, "Claw-free graphs—A survey," *Discrete Math.*, vol. 164, nos. 1–3, pp. 87–147, 1997.
[45] S. Pirandola, R. Laurenza, C. Ottaviani, and L. Banchi, "Fundamental limits of repeaterless quantum communications," *Nature Commun.*, vol. 8, p. 15043, Apr. 2017.
[46] S. Pirandola *et al.*, "Advances in quantum cryptography," *Adv. Opt. Photon.*, vol. 12, no. 4, pp. 1012–1236, 2020.



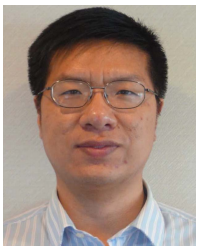
Lutong Chen (Graduate Student Member, IEEE) received the bachelor's degree from the School of Cyber Science and Technology, University of Science and Technology of China (USTC), in 2020, where he is currently pursuing the Ph.D. degree in information security. His research interests include quantum networking and network security.

Kaiping Xue (Senior Member, IEEE) received the bachelor's degree from the Department of Information Security, University of Science and Technology of China (USTC), in 2003 and the Ph.D. degree from the Department of Electronic Engineering and Information Science (EEIS), USTC, in 2007. From May 2012 to May 2013, he was a Post-Doctoral Researcher at the Department of Electrical and Computer Engineering, University of Florida. He is currently a Professor with the School of Cyber Science and Technology, USTC. His research interests include next-generation internet architecture design, transmission optimization, and network security. He is an IET Fellow. He serves on the Editorial Board fot several journals, including the IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING (TDSC), the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS (TWC), and the IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT (TNSM). He has also served as a (Lead) Guest Editor for many reputed journals/magazines, including IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS (JSAC), *IEEE Communications Magazine*, and *IEEE Network*.

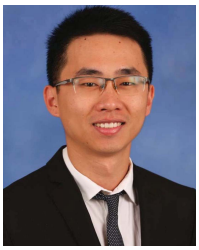


Jian Li (Member, IEEE) received the B.S. degree from the Department of Electronics and Information Engineering, Anhui University, in 2015, and the Ph.D. degree from the Department of Electronic Engineering and Information Science (EEIS), University of Science and Technology of China (USTC), in 2020. From November 2019 to November 2020, he was a Visiting Scholar at the Department of Electronic and Computer Engineering, University of Florida. He is currently a Post-Doctoral Researcher with the School of Cyber Science and Technology, USTC. His research interests include wireless communications, satellite networks, and future internet architecture.

Nenghai Yu received the B.S. degree from the Nanjing University of Posts and Telecommunications, Nanjing, China, in 1987, the M.E. degree from Tsinghua University, Beijing, China, in 1992, and the Ph.D. degree from the Department of Electronic Engineering and Information Science (EEIS), University of Science and Technology of China (USTC), Hefei, China, in 2004. He is currently a Professor with the Department of Electronic Engineering and Information Science, USTC, and the School of Cyber Science and Technology, USTC. He is also the Executive Dean of the School of Cyber Science and Technology, USTC, and the Director of the Information Processing Center, USTC. He has authored or coauthored more than 130 papers in journals and international conferences. His research interests include multimedia security, multimedia information retrieval, video processing, and information hiding.



Ruidong Li (Senior Member, IEEE) received the B.E. degree from Zhejiang University, China, in 2001, and the Doctor of Engineering degree from the University of Tsukuba in 2008. He is currently an Associate Professor with the College of Science and Engineering, Kanazawa University, Japan. Before joining Kanazawa University, he was a Senior Researcher at the Network System Research Institute, National Institute of Information and Communications Technology (NICT). His current research interests include future networks, big data networking, blockchain, information-centric network, the Internet of Things, network security, wireless networks, and quantum internet. He is a member of IEICE. He is the Founder and the Chair of the IEEE SIG on big data intelligent networking and IEEE SIG on intelligent Internet edge and the secretary of the IEEE Internet Technical Committee. He also serves as the Chair for conferences and workshops, such as IWQoS 2021, MSN 2020, BRAINS 2020, ICC 2021 NMIC Symposium, ICCN 2019/2020, NMIC 2019/2020, and organized the special issues for the leading magazines and journals, such as *IEEE Communications Magazine*, *IEEE Network*, and IEEE TRANSACTIONS ON NETWORK SCIENCE AND ENGINEERING (TNSE).



Jianqing Liu (Member, IEEE) received the B.Eng. degree from the University of Electronic Science and Technology of China in 2013 and the Ph.D. degree from The University of Florida in 2018. He is currently an Assistant Professor with the Department of Computer Science, NC State University. His research interest is wireless communications and networking, security, and privacy. He received the U.S. National Science Foundation Career Award in 2021. He was also the Recipient of several best paper awards including the 2018 Best Journal Paper Award from IEEE Technical Committee on Green Communications and Computing (TCGCC).

Qibin Sun (Fellow, IEEE) received the Ph.D. degree from the Department of Electronic Engineering and Information Science (EEIS), University of Science and Technology of China (USTC), in 1997. He is currently a Professor with the School of Cyber Science and Technology, USTC. His research interests include multimedia security, network intelligence and security, and so on. He has published more than 120 papers in international journals and conferences.

Jun Lu received the bachelor's degree from Southeast University in 1985 and the master's degree from the Department of Electronic Engineering and Information Science (EEIS), University of Science and Technology of China (USTC), in 1988. He is currently a Professor with the School of Cyber Science and Technology and the Department of EEIS, USTC. His research interests include theoretical research and system development in the field of integrated electronic information systems. He is an Academician of the Chinese Academy of Engineering (CAE).