2024 IEEE International Conference on Communications; Selected Areas in Communications: Quantum Communications and Information Technology

A Distributed Routing Protocol Based on Key Reservation in Quantum Key Distribution Networks

Jiaqi Wu*, Lutong Chen*, Jing Zhang†‡, Zixuan Huang†, Zhonghui Li*, Jian Li*, Kaiping Xue*, Nenghai Yu*
* School of Cyber Science and Technology, University of Science and Technology of China (USTC), Hefei, 230027, China
† Institute of Space Integrated Ground Network, Hefei, Anhui 230088, China
‡ Science Island Branch of Graduate School, USTC, Hefei, Anhui 230031, China
§ Corresponding Author: J. Li (lijian9@ustc.edu.cn), K. Xue (kpxue@ustc.edu.cn)

***Abstract*—Nowadays, Quantum Key Distribution (QKD) has garnered widespread attention due to its ability to provide symmetric secret keys with information-theoretic security in point-to-point communication. Additionally, key relay technology has been introduced to complete end-to-end key distribution between remote parties by consuming keys in the intermediate links. The routing problem of selecting paths for key relay technology becomes crucial as it directly impacts the network's performance. In this paper, we focus on addressing the routing problem for a specific scenario to serve applications with real-time requirements. Real-time is a critical necessity for supporting various applications, such as video chatting and calling. To satisfy real-time requirements and achieve Quality of Service (QoS) provision to guarantee the completion time, we introduce a distributed routing protocol called Distributed Routing Protocol Based on Key Reservation (Q-RoKR). It reserves keys in advance along the selected path, thereby satisfying real-time requirements. We also propose a priority-awareness mechanism to address resource competition and make efficient use of keys in links. Extensive experiments demonstrate that our protocol effectively meets the real-time requirements and significantly improves throughput. Furthermore, our key efficiency approaches the optimal bound when compared to other comparison schemes.**

***Index Terms*—Quantum Network, Quantum Key Distribution, Routing Algorithm, Key Reservation.**

I. Introduction

Secure communication in classical networks is a primary focus of researchers. Currently, it heavily relies on asymmetric encryption algorithms, such as RSA [1], which derive their security from computational complexity assumptions. However, the advent of quantum computing has introduced a significant threat to the security of classical communication systems [2], as quantum computers possess unparalleled computational power. Since the first quantum protocol showed its feasibility [3], Quantum Key Distribution (QKD) can distribute secret keys between communicating parties to achieve unconditional secure communication in theory, by utilizing unique features of quantum mechanics. Accordingly, numerous QKD networks have been established worldwide, including the first DAPRA network [4], SECOQC in Europe [5], Tokyo QKD network in Japan [6], Beijing-Shanghai in China [7], and others.

In QKD networks, distributing secret keys between remote parties is typically achieved through key relay, where end-to-end keys are transmitted hop-by-hop. At each hop, the key relay encrypts the end-to-end keys by utilizing point-to-point keys on the links. However, the current point-to-point quantum key coding rate is relatively low due to inherent photon loss in quantum channels and susceptibility to noise-induced disturbances [8]. Furthermore, the performance of QKD decreases when employing more trusted relays to extend the QKD distance [9], [10]. As a result, the routing problem, *i.e.,* selecting appropriate key relay paths, emerges as a critical challenge in QKD networks. Overcoming it can enhance the performance of remote quantum key distribution in networks.

Recently, several routing schemes have been proposed for QKD networks, and most of these schemes are primarily based on the network's status for decision-making and treat all requests equally. This approach overlooks the fact that different applications have diverse Quality of Service (QoS) requirements, and meeting the requirements has the potential to enhance network performance. The real-time requirement is critical and common for various applications, such as video chatting and video meetings, that the network completes the end-to-end key distribution before a specified deadline to ensure uninterrupted and smooth communication. Recognizing the need for QoS provision, Mehic *et al.* [11] introduced a routing protocol called GPSRQ, but real-time was just a criterion for classifying applications and was not considered concretely. It desires to develop a routing scheme specifically tailored to meet real-time requirements in QKD networks.

In this paper, we focus on delivering high-quality QKD services to requests that demand real-time requirements. To achieve this goal, we present a distributed routing protocol named Distributed Routing Protocol Based on Key Reservation (Q-RoKR). This protocol intelligently leverages secret keys present on the network links, satisfying the real-time demands of requests. The routing algorithm gives priority to meeting real-time requirements while also attempting to enhance key efficiency. By employing flexible key reservation in advance along the selected path for a given request, we ensure exclusive key utilization for real-time key relay technology. Furthermore, we design a priority-awareness mechanism to address resource competition during key reservations, leading to enhanced network performance. We conduct extensive experiments utilizing SimQN [12], a discrete event-based quantum network simulation platform. The Q-RoKR protocol, along with two novel comparison schemes, has been implemented and evaluated.

978-1-7281-9054-9/24/$31.00 ©2024 IEEE

The experimental results demonstrate the availability and effectiveness of our protocol for serving real-time requests. The contributions of the paper are the following:

- We propose a complete and available routing protocol based on flexible key reservations in links and fill the gap in current routing schemes about the scenario that requests have specific real-time requirements.
- We design a priority-awareness mechanism according to the requirements of requests to deal with resource competition and make reasonable use of precious keys.
- We realize our protocol and two comparison schemes on SimQN and the results show that our protocol is superior to comparison schemes in service rate and throughput significantly while effectively using keys in links.

This paper is organized as follows. Section II introduces related works. Section III presents the system model and problem statement. In Section IV, we propose the Q-RoKR routing protocol for real-time requirements, and we conduct experiments in Section V. Section VI concludes the paper.

II. Related Work

Numerous routing schemes have been proposed to select paths for distributing end-to-end secret keys between communicating parties. Nevertheless, most of them only consider the network status. In [13], Tanizawa *et al.* designed a metric of the available secret keys in links to quantify the links and then used it to instruct path selection. Yang *et al.* [14] used the average key generation rate and status of key pools to calculate the link cost metric and thus selected the path with the lowest cost. Similarly, Yao *et al.* [15] designed the recovery capacity indicator with more information, including key consumption rate, while Chen *et al.* [16] considered the length of the path when calculating the cost. However, they select the path according to the network status directly, which can not promise to serve requests with specific requirements.

We also notice that several schemes have been proposed with a focus on meeting application requirements and delivering a certain level of QoS. Chen *et al.* [17] introduced the concept of setting application priorities in advance, indicating the relative importance of requests, along with the total key requirement and the required key update rate. Similarly, Cheng *et al.* [18] categorized requests into three groups based on their distinct real-time requirements. They employed key reservations for requests with the highest priority while adopting hop-by-hop queuing for other requests. However, their scheme lacks a mechanism to address recovery and resource competition. Motivated by the urgent need to design a routing scheme for QoS provision and recognizing the interconnection between public and quantum channels [19], a novel QoS model and routing protocol were proposed in [11] to ensure QoS assurance, albeit without providing a detailed approach to satisfying a requirement strictly. In conclusion, current schemes can not handle real-time requirements precisely.

III. System Model and Problem Statement

In this section, we present the layered architecture of a typical QKD network and some notations used in our protocol. We also describe and analyze the problem.

A. System Model

We adopt a typical layered QKD network architecture due to its precise delineation of different layers [20]. This architecture has three distinct layers from bottom to top: 1) a quantum layer to generate symmetric keys in point-to-point ways, 2) a key management layer to manage keys, and 3) a network control layer to control the entire network, including the routing model. Applications that request end-to-end keys are in an upper service layer, and there are interfaces between it and three QKD network layers to interact, including sending or receiving requests and end-to-end keys distributed.

The QKD network serves as a secure key provider. A request in QKD networks is represented by a four-element tuple denoted as $R = \{S, D, K, DDL\}$, consisting of the source node S, the destination node D, key requirement K indicating the desired number of end-to-end keys, and a real-time constraint DDL which specifies the maximum time allowed for the completion of the end-to-end key distribution process. When an application located at node S generates a request R, it transmits this request to the routing model situated within the network control layer. Then, the routing model selects a viable path P for the given request R. Once a suitable path P is determined, it is promptly forwarded to the key management layer. Subsequently, all nodes along the path P communicate to facilitate key relay technology and distribute the end-to-end keys within the specified time constraint. Finally, the application can retrieve the distributed keys from the key manager and employ them to establish a secure tunnel with the destination node D.

The nodes along a given path P are categorized into two classes based on their respective positions: the source node and the reservation nodes. Source nodes are responsible for initiating requests, managing the routing procedures, and communicating with other nodes to apply key reservations. On the other hand, reservation nodes refer to both intermediate nodes and destination nodes. Their primary role is to accept or decline key reservations. One of the challenges here arises from the fact that point-to-point keys are generated between two adjacent nodes, necessitating the cooperation of both nodes involved in the key reservation process to avoid conflicts. To address this challenge, we adopt a straightforward yet effective principle: each reservation node is assigned the responsibility of handling the key reservation for the predecessor link. For instance, the destination node D should be responsible for managing the key reservation for its preceding link, *i.e.,* the last link in the path P.

B. Problem Statement

Real-time requirements refer to the transmission finish time from the source node to the destination node successfully being limited. The transmission time can mainly be divided into

two parts: the time required for end-to-end key establishment and the data transmission delay. Also, the time spent on end-to-end key distribution is the primary factor due to the low key generation rate. In this context, we ignore data transmission time as it is irrelevant to the routing protocol, and we consider the time consumption during key distribution to satisfy real-time requirements. Similar to the classic networks, the time on end-to-end key establishment t_{total} could be calculated by

$$t_{total} = t_{rout} + t_{queue} + t_{proc} + t_{propa}, \tag{1}$$

where t_{rout} is the delay from routing, t_{queue} is the delay for queuing in intermediate nodes, t_{proc} is the time for processing by keys at two ends of each link and t_{propa} is the propagation delay along the links. Among four elements, t_{proc} and t_{propa} usually depend on the physical device's performance, which is beyond our consideration. Moreover, t_{queue} has a great association with the key generation rate in links. As a result, the primary goal is to guarantee key provisions within time constraints by routing.

IV. Routing Protocol

Motivated by the insufficiency of existing routing schemes in addressing the specific real-time requirement scenarios, we propose a novel Distributed Routing Protocol Based on Key Reservation (Q-RoKR) for QKD networks. Our protocol aims to resolve the challenge of distributing an adequate number of end-to-end keys within the specified time requirements. Additionally, considering the scarcity of QKD keys, we introduce a priority-awareness mechanism to effectively manage resource competition and optimize key utilization on the links.

A. Overview

In order to deal with real-time requirements, we first design a key management model to manage keys on each link from a time perspective. That is, it reports whether a request can be finished before its DDL by comprehensively considering the number of keys in the key pool and the key generation rate. The key management model is presented in Section IV-B.

Next, we introduce the Q-RoKR protocol. It checks multiple paths in the order of the number of hops. Then, it leverages the key management model to ensure the key supplement can meet the DDL requirement, and contains three stages, as shown in Fig. 1: 1) The source node selects a path between S and D and sends reservation requests to all reservation nodes; 2) The reservation nodes attempt to reserve keys in the links based on the required K keys before DDL and provide feedback to the source node regarding the feasibility of the service and 3) Subsequently, the source node makes decisions based on all the feedback received. Thus, we can ensure that all links are capable of providing K QKD keys before DDL. Consequently, if all links within the selected path P successfully reserve keys for the request R, the end-to-end keys between S and D can be established via key relays prior to the arrival of the deadline. We illustrate the Q-RoKR protocol in Section IV-C.

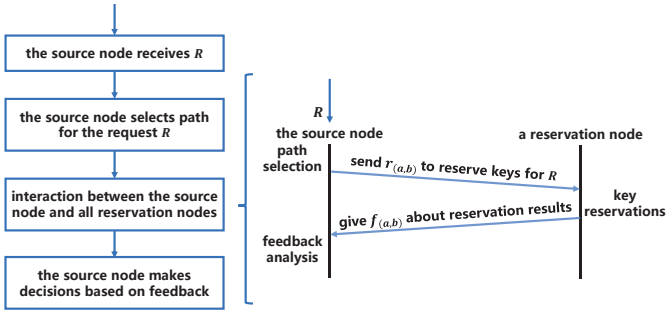


Fig. 1. The overview of Q-RoKR protocol and the key reservation procedure between the source node and a reservation node.

B. Key Management Model

We construct the key management model starting from a trivial ideal to evaluate the number of keys in a link by multiplying the key generation rate and the period of generation time. Thus, based on the history of the QKD link performance, it is possible to evaluate the time used to produce a given number of keys. Consider a link $l_{(a,b)}$ that connects node a and node b. Let $v_{(a,b)}$ be the QKD key generation rate. We mark the moment $tag_{(a,b)}$ as the time boundary when all existing requests are serviced, as shown in Fig. 2, where the two black arrows stand for the time axis, the blue block shows the keys reserved for requests and the yellow block indicates keys numbered K. It means that for incoming requests with a $DDL \geq tag_{(a,b)} + \frac{K}{v_{(a,b)}}$, $l_{(a,b)}$ can serve them and update $tag_{(a,b)}$. In other words, $tag_{(a,b)}$ also stands for the moment when $l_{(a,b)}$ could start providing keys for those requests arriving newly.

Based on the above key management model, we shift the focus from direct management of keys to indirect management by adjusting the time boundary. When a new request is received, it updates $tag_{(a,b)}$ and evaluates whether $tag_{(a,b)}$ satisfies the DDL requirement. If so, $l_{(a,b)}$ is available for the incoming request, and reserve keys for it.

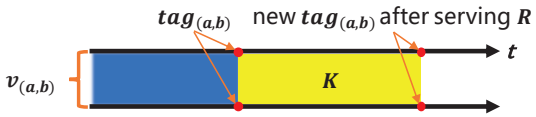


Fig. 2. Key management model to evaluate the most recent available time.

Due to the capacity of key pools, $tag_{(a,b)}$ has the minimum value

$$min_{(a,b)} = t_c - \frac{cap_{(a,b)}}{v_{(a,b)}}, \tag{2}$$

where $cap_{(a,b)}$ is the capacity of the key pool and t_c is the current moment. We must check $tag_{(a,b)}$'s validation, and update it by $min_{(a,b)}$ when it is less than $min_{(a,b)}$ to avoid key overflow, *i.e.,* the key pool could not store all keys.

C. Q-RoKR Protocol Design

In this section, we present the Q-RoKR protocol. It selects an optimal path that satisfies both the desired quantity K and the real-time DDL requirements.

1) Path Selection: We consider the utilization of either a link-state or distance-vector-based routing protocol to perform continuous calculations of routing tables for all nodes. Specifically, for a given node a, its routing table comprises multiple paths that are arranged in ascending order based on the number of hops required. However, we limit our consideration to paths within a certain number of hops to reduce the consumption of keys in links on long paths, due to the scarcity of keys.

Upon receiving a request R from applications, node a proceeds with the following steps. Firstly, it selects the first path from its routing table as the aimed path P. Subsequently, it examines the time boundaries locally, denoted as tag on all links along path P. If all of these boundaries meet the DDL, node a initiates the process of reserving keys by sending a reservation request to all reservation nodes. Inversely, if the DDL can not be met, node a selects the next path from the routing table as the new aimed path and repeats the above procedure. When all available paths fail to serve, or DDL is reached, the routing module notifies the application of the unsuccessful routing attempt for the request R.

2) Key Reservation and Priority-Awareness Mechanism: Without loss of generality, we describe the interaction between a source node and any reservation node b along a path to accomplish key reservations in detail here, since all reservation nodes behave the same.

Considering the situation that two reservation requests $r_{(a,b)}$ (for $R = \{S, D, K, DDL\}$) and $r^{'}_{(a,b)}$ (for $R' = \{S^{'}, D^{'}, K^{'}, DDL^{'}\}$) arrives at the node b, where $K < K^{'}$ and $DDL < DDL^{'}$. If the intuitive strategy of First-In-First-Out (FIFO) is adopted, b reserves keys for $R^{'}$ and R in order of arrival. However, it is possible that $l_{(a,b)}$ could not satisfy R due to its strict DDL after providing lots of keys for $R^{'}$ with low real-time requirement $DDL^{'}$. However, if it serves R first, it may provide keys for both requests successfully.

Clearly, both the real-time and key quantity requirements should be considered coordinately. We put forward a priority-awareness mechanism to get a proper service order to achieve high performance accordingly. Specifically, we give priority to real-time requirements and prefer to reserve keys for those with strict real-time requirements to serve requests as many as possible. Second, to enhance throughput, we prefer to serve requests with higher key quantity requirements whenever they demand the same real-time requirement. Finally, to avoid low-priority requests starve, Q-RoKR adopts priority promotion to upgrade the final priority Pr of $r_{(a,b)}$ by

$$Pr \leftarrow \frac{Pr}{DDL - t_c}. \tag{3}$$

It depends on the urgency of the request.

While receiving $r_{(a,b)}$, Q-RoKR puts it into the waiting queue $Q_{(a,b)}$ of $l_{(a,b)}$ first so that it can handle multiple requests with more efficiency at once. It waits for at the latest $T_{(a,b)}$ to collect key reservation requests. To avoid waiting too long to be provided with sufficient keys, we calculate the latest distribution moment $T_{(a,b)}$ to limit the waiting time of $r_{(a,b)}$ in $Q_{(a,b)}$ as

$$T_{(a,b)} = \begin{cases} \max(t_c, DDL - \frac{K}{in_{(a,b)}}) & in_{(a,b)} > 0, \\ t_c & in_{(a,b)} \leq 0. \end{cases} \tag{4}$$

Here, $in_{(a,b)}$ is the available key increasing rate of $l_{(a,b)}$:

$$in_{(a,b)} = v_{(a,b)} - out_{(a,b)}, \tag{5}$$

and $out_{(a,b)}$ is the key consumption rate of $l_{(a,b)}$ estimated by average moving speed of $tag_{(a,b)}$ roughly:

$$out_{(a,b)} = \begin{cases} v_{(a,b)} \times \frac{tag_{(a,b)} - t_s}{t_c - t_s} & t_c > t_s, \\ 0 & t_c = t_s, \end{cases} \tag{6}$$

where t_s stands for the starting moment. When the link load is low, the design of (4) realizes key reservations for requests in a batch manner to serve more requests efficiently while hoping to deal with requests as soon as possible to send quick feedback when the link load goes high.

Algorithm 1: Priority-awareness Key Reservation

Input: $Q_{(a,b)} = \{r^1_{(a,b)}, \cdots, r^n_{(a,b)}\}, T^i_{(a,b)}$;
Output: $f^k_{(a,b)}, \ \forall k \in \{1, 2, ..., n\}$;
if $r^i_{(a,b)}$ *for* R^i *has not been dealt with* **then**
  Sort all $r^k_{(a,b)}$ by K^k in ascending order;
  Sort all $r^k_{(a,b)}$ by DDL^k in descending order;
  for $k = 1$ *to* n **do**
    $Pr^k \leftarrow \frac{k}{DDL^k - t_c}$;
  end
  Sort all $r^k_{(a,b)}$ by Pr^k in decreasing order;
  while $Q_{(a,b)} \neq \varnothing$ **do**
    Pop out $r^k_{(a,b)}$ with the highest priority;
    Get $tag_{(a,b)}$ and $v_{(a,b)}$;
    $pre_{(a,b)} \leftarrow tag_{(a,b)} + \frac{K^k}{v_{(a,b)}}$;
    if $pre_{(a,b)} \leq DDL^k$ **then**
      $tag_{(a,b)} \leftarrow pre_{(a,b)}$;
      Put R^k into $Succ_{(a,b)}$;
    end
    Send $f^k_{(a,b)}$ about the reservation;
  end
end

Once arriving at $T_{(a,b)}$, Q-RoKR checks if it has tried to reserve keys for $r_{(a,b)}$. If not, we should deal with all reservation requests in $Q_{(a,b)}$ at once: order requests by priority, reserve keys for them in order, and finally send feedback about key reservation results to the associated source nodes. Specifically, we sort all reservation requests by key requirements in ascending order and then sort them by real-time requirements in descending order. Then, we use (3) to calculate their final priorities and sort them by priorities in descending order. Next, Q-RoKR reserves keys for those requests in order. It pops the request with the highest priority, predicts the time boundary $pre_{(a,b)}$ and examines if $pre_{(a,b)}$ is within the real-time requirement DDL: if so, node b sends feedback about the successful reservation to the associated

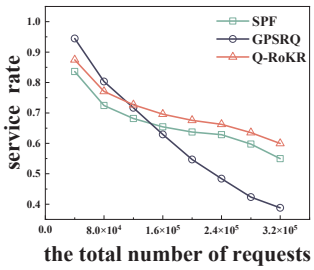


Fig. 3. Service rate vs requests.

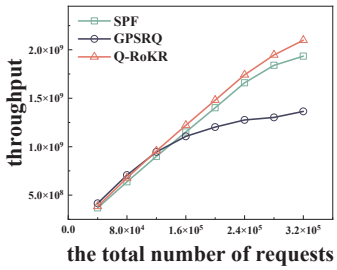


Fig. 4. Throughput vs requests

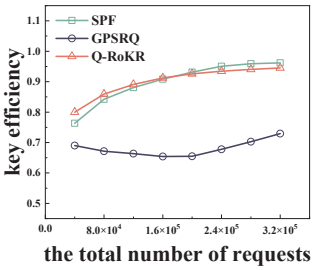


Fig. 5. Key efficiency vs requests

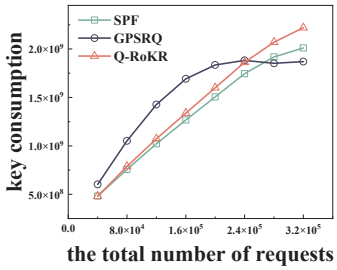


Fig. 6. Key consumption vs requests

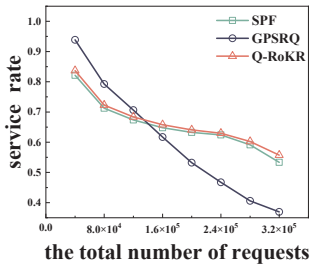
source node, update time boundary to $tag_{(a,b)} = pre_{(a,b)}$ and store the request R; if not, inform the source node of the failed reservation in $l_{(a,b)}$. The details of the key reservation are shown in Alg. 1.

3) Feedback Analysis: Node a receives feedback $f_{(a,b)}$ from node b and makes different decisions according to it. If $f_{(a,b)}$ shows that $l_{(a,b)}$ has reserved keys numbered K for R successfully and P is still the aimed path for R, then it is stored for the moment. Next, a checks if all links have reserve keys, and if so, the key relay starts immediately, and end-to-end keys will be provided for the application to keep communication in traditional networks securely. However, if P can not meet the request's requirement, node a would send a cancellation request to node b at once.

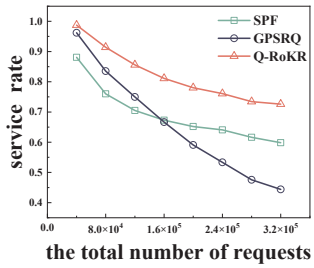
In another case, if $f_{(a,b)}$ shows that $l_{(a,b)}$ failed to reserve keys, node a updates local $tag_{(a,b)}$ first: if $tag_{(a,b)}$ is zero, set $tag_{(a,b)}$ to the value in $f_{(a,b)}$. Otherwise, update $tag_{(a,b)}$ only when the value in $f_{(a,b)}$ is larger. Second, node a sends a cancellation request to all other reservation nodes (other than b) to notify them to release the reserved keys if it has not received any failed reservation feedback before receiving $f_{(a,b)}$. Meanwhile, node a selects a next aimed path P' along the routing table and begins a new key reservation on P'.

V. Experiments

To evaluate the generalized performance, we conduct our experiments in a random topology using the Waxman algorithm [21]. We also implement the Shortest Path First (SPF) algorithm and the GPSRQ protocol [11] as the baselines. The entire simulation and implementation are conducted in the SimQN platform [12]. In our simulations, we utilize the BB84 protocol [3] to generate secret keys at a rate of 10 kbps. All nodes equip a key pool capacity of 20 Mbit, ensuring adequate storage of keys for the routing process.



(a) High real-time requirements.



(b) Low real-time requirements.

Fig. 7. The service rate in different real-time requirements.

A. Parameter Settings

To simulate a practical QKD networks scenario, we perform the simulation by setting up 50 nodes in a square area of 1.2×10^5 m^2, and locations of nodes and links are generated randomly. The simulation lasts for 30 seconds, and the information about R, including S, D, K, and DDL, is generated randomly as well. Specifically, the value of K follows a normal distribution with a mean of 1.0×10^3 or 1.0×10^4 bit, and DDL ranges from 0 to 30 seconds. The number of requests in every simulation ranges from 4×10^4 to 3.2×10^5 and increases by 4×10^4 every time. Note that an application can continuously generate multiple requests whenever it needs. Last, the simulation is repeated 50 times, and average values are taken as the final results to conclude the general performance of routing schemes.

B. Simulation Results

We calculate the service rate $rate_{succ}$ as the ratio of requests served in time to all requests in the network. Throughput key_{out} is another metric indicating the number of end-to-end keys established in the network. To express usage efficiency of keys in links, we also design key efficiency index key_{eff} and calculate it as

$$key_{eff} = \frac{key_{out}}{key_{in}}, \tag{7}$$

where key_{in} is the total number of keys consumed for key relay technology. In theory, due to the path for routing consisting of no less than one link, key_{eff} has a range of $(0, 1]$ and its optimal value is 1, which means that the number of the end-to-end keys is the same as consumed in the links. In addition, we record the service rate in different real-time requirements to show the tendency of $rate_{succ}$ in different real-time requirements as the number of requests increases.

As Fig. 3 and Fig. 4 show, since Q-RoKR guarantees key provision for R before DDL arrives once P is selected by key reservation mechanism as well as reasonable order calculated by priority-awareness mechanism, Q-RoKR is superior to SPF and GPSRQ in $rate_{succ}$ and brings bigger key_{out}. Due to path selection hop by hop, GPSRQ could find longer paths with more keys to use and thus serve a little more requests than Q-RoKR at first. However, as the number of requests increases, our protocol could satisfy more requests and bring more throughput thanks to making reasonable use of keys in links. Besides, when more requests are generated in QKD

networks, GPSRQ even serves fewer requests and thus brings less throughput than SPF since serving a request successfully consumes a lot of keys on average because of its routing mechanism for QoS guarantee.

As shown in Fig. 5, due to the hop restriction and reasonable utilization of keys in links, Q-RoKR finds short paths, and the key efficiency of Q-RoKR is higher than that in SPF and is near the optimal value. As it serves more requests, Q-RoKR has to find longer paths due to limited keys in links and its key_{eff} becomes lower compared to SPF. In GPSRQ, key_{eff} is lower than that in our protocol, and it decreases first because of selecting a longer path and then increases gradually since that distance becomes critical as information about keys becomes similar due to large key consumption for many requests. In short, our protocol brings more throughput by using keys in links efficiently. As Fig. 6 shows, it is reasonable that Q-RoKR consumes more keys to serve more requests than comparison schemes. In addition, GPSRQ uses a lot of keys since it selects longer paths at first, and its key_{in} becomes lower than that in SPF and our protocol just because of the low service rate.

The service rate in different real-time requirements is shown in Fig. 7. In general, we conclude that we serve more requests in high and low real-time requirements, as shown in Fig. 7(a) and Fig. 7(b) since Q-RoKR adopts key reservation and priority-awareness mechanism to make use of keys more reasonably. Meanwhile, when real-time requirement looses, the proportion of requests served grows bigger. However, GPSRQ does provide service for a few more requests with high real-time requirements because it could find longer paths with sufficient keys for requests in the beginning.

VI. Conclusion

In this paper, we proposed a routing protocol, Q-RoKR, to provide QoS guarantees in QKD networks, specifically for those requests that impose real-time requirements on key relay technology. Recognizing the advantages of distributed routing in large-scale networks, the Q-RoKR protocol adopts a distributed routing scheme. This approach ensures both flexibility and efficient utilization of keys in network links while prioritizing the satisfaction of real-time requirements raised by requests. The Q-RoKR protocol incorporates a key reservation mechanism to effectively manage real-time requirements and reduce the wastage of keys that are commonly observed in previous hop-based distributed routing schemes. Furthermore, the wastage of keys can even be avoidable if the capacity of key pools is large enough to match the usage of keys while it still exists in former schemes due to the failure to meet real-time requirements. By considering both key quantity and real-time requirements, the priority-awareness mechanism enhances throughput and optimizes the utilization efficiency of keys. Extensive experiments prove the overall network performance by effectively balancing the trade-off between meeting real-time requirements and maximizing key utilization.

Acknowledgment

This work is supported in part by Innovation Program for Quantum Science and Technology under Grant No. 2021ZD0301301, Anhui Initiative in Quantum Information Technologies under Grant No. AHY150300, the National Natural Science Foundation of China under grant No. 62201540, and Youth Innovation Promotion Association of the Chinese Academy of Sciences (CAS) under Grant No. Y202093.

References

[1] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[2] C. Gidney and M. Ekerå, "How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits," *Quantum*, p. 433, 2021.

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

[4] C. Elliott, D. Pearson *et al.*, "Quantum cryptography in practice," in *Proceedings of the 2003 ACM SigCOMM*, 2003, pp. 227—–238.

[5] M. Peev, C. Pacher *et al.*, "The SECOQC quantum key distribution network in Vienna," *New Journal of Physics*, p. 075001, 2009.

[6] M. Sasaki, M. Fujiwara *et al.*, "Field test of quantum key distribution in the Tokyo QKD Network," *Optics Express*, vol. 19, no. 11, pp. 10 387––10 409, 2011.

[7] W. Kozlowski and S. Wehner, "Towards large-Scale quantum networks," in *Proceedings of the Sixth ACM International Conference on Nanoscale Computing and Communication*, 2019, pp. 1–7.

[8] Y. Cao, Y. Zhao, Q. Wang *et al.*, "The evolution of quantum key distribution networks: On the road to the qinternet," *IEEE Communications Surveys & Tutorials*, vol. 24, no. 2, pp. 839–894, 2022.

[9] M. Wang, J. Li, K. Xue *et al.*, "A segment-based multipath distribution method in partially-trusted relay quantum networks," *IEEE Communications Magazine*, vol. 61, no. 12, pp. 184–190, 2023.

[10] C. Elliott, "Building the quantum network*," *New Journal of Physics*, vol. 4, no. 1, p. 46, 2002.

[11] M. Mehic, P. Fazio, S. Rass *et al.*, "A novel approach to quality-of-service provisioning in trusted relay Quantum Key Distribution Networks," *IEEE/ACM Transactions on Networking*, vol. 28, no. 1, pp. 168–181, 2020.

[12] L. Chen, K. Xue, J. Li, N. Yu, R. Li, Q. Sun, and J. Lu, "SimQN: a network-layer simulator for the quantum network investigation," *IEEE Network*, pp. 1–8, 2023, DOI:10.1109/MNET.130.2200481.

[13] Y. Tanizawa, R. Takahashi, and A. R. Dixon, "A routing method designed for a quantum key distribution network," in *Proceedings of the 2016 Eighth International Conference on Ubiquitous and Future Networks*, 2016, pp. 208–214.

[14] C. Yang, H. Zhang, and J. Su, "Quantum key distribution network: Optimal secret-key-aware routing method for trust relaying," *China Communications*, vol. 15, no. 2, pp. 33–45, 2018.

[15] J. Yao, Y. Wang, Q. Li, H. Mao, A. A. Abd El-Latif, and N. Chen, "An efficient routing protocol for quantum key distribution networks," *Entropy*, vol. 24, no. 7, p. 911, 2022.

[16] L. Chen, M. Zhao *et al.*, "ADA-QKDN: A new quantum key distribution network routing scheme based on application demand adaptation," *Quantum Information Processing*, vol. 20, no. 309, pp. 1–22, 2021.

[17] L. Chen, Z. Zhang, M. Zhao, K. Yu, and S. Liu, "APR-QKDN: A quantum key distribution network routing scheme based on application priority ranking," *Entropy*, vol. 24, no. 11, p. 1519, 2022.

[18] X. Cheng, Y. Sun, and Y. Ji, "A QoS-supported scheme for quantum key distribution," in *Proceedings of the 2011 International Conference on Advanced Intelligence and Awareness Internet*, 2011, pp. 220–224.

[19] M. Mehic, O. Maurhart *et al.*, "Analysis of the public channel of quantum key distribution link," *IEEE Journal of Quantum Electronics*, vol. 53, no. 5, pp. 1–8, 2017.

[20] "ITU-T Y.3800 overview on networks supporting quantum key distribution," 2019, accessed on Feb. 2024. [Online]. Available: https://handle.itu.int/11.1002/1000/13990

[21] B. Waxman, "Routing of multipoint connections," *IEEE Journal on Selected Areas in Communications*, vol. 6, no. 9, pp. 1617–1622, 1988.