Pervasive and Mobile Computing 102 (2024) 101937

Contents lists available at ScienceDirect

Pervasive and Mobile Computing

journal homepage: www.elsevier.com/locate/pmc

Analysis of micro- vs. macro-flows management in QKD-secured edge computing

Claudio Cicconetti [*], Marco Conti, Andrea Passarella

IIT, National Research Council, Pisa, Italy

A R T I C L E I N F O

Keywords:
Quantum key distribution
Edge computing
Distributed resource allocation

A B S T R A C T

Quantum Key Distribution (QKD) holds the promise of a secure exchange of cryptographic material between applications that have access to the same network of QKD nodes, interconnected through fiber optic or satellite links. Worldwide several such networks are being deployed at a metropolitan level, where edge computing is already offered by the telco operators to customers as a viable alternative to both cloud and on-premise hosting of computational resources. In this paper, we investigate the implications of enabling QKD for edge-native applications from a practical perspective of resource allocation in the QKD network and the edge infrastructure. Specifically, we consider the dichotomy between aggregating all the applications on the same source–destination path vs. adopting a more flexible micro-flow approach, inspired from Software Defined Networking (SDN) concepts. Our simulation results show that there is a fundamental trade-off between the efficient use of resources and the signaling overhead, which we managed to diminish with the use of suitable hybrid solutions.

1. Introduction

Quantum Key Distribution (QKD) is a technology that allows two nodes that can transfer quantum bits (qubits) from one to another to exchange a sequence of bits that cannot be spoofed by a third party with access to the classical/quantum channels used. Such a sequence can be used in a straightforward manner as a symmetric key to encrypt private communications between the two nodes. Several protocols exist for this purpose, the first and most famous being BB84 [1]. Unlike other applications of the Quantum Internet [2], which are still in their infancy, QKD is mature enough for production deployments on commercial networks (see Table 1 in [3]). The most common medium used for QKD is optical fibers, which makes it possible to reuse the existing infrastructure deployed for classical networks [4]. Conveniently, quantum and classical communications can even share the same cable, though with some mutual performance degradation [5]. Significant public and private investments are being done all over the world to deploy quantum communication infrastructures and enable QKD in the near-term (and lay the foundations for the Quantum Internet in the long-term), e.g., the EuroQCI joint initiative between the European Commission and the European Space Agency.

An example of a QKD network is illustrated in Fig. 1: with current technology, the secure exchange of keys is limited to a single hop, which means that end-to-end security can only be achieved if all the intermediate nodes in the QKD network can be fully trusted, i.e., they behave honestly and have not been tampered, and in fact are called *trusted repeaters*. There are several ways in which the link-local secure keys can be combined by the trusted repeaters towards the end applications [6]. The research community is working on several alternatives to overcome this limitation in the future, including Measurement-Device independent (MDI) stations [5] and Quantum Repeaters (QRs) [7].

[*] Corresponding author.
E-mail addresses: c.cicconetti@iit.cnr.it (C. Cicconetti), m.conti@iit.cnr.it (M. Conti), a.passarella@iit.cnr.it (A. Passarella).

https://doi.org/10.1016/j.pmcj.2024.101937
Received 18 December 2023; Received in revised form 26 April 2024; Accepted 13 May 2024
Available online 16 May 2024

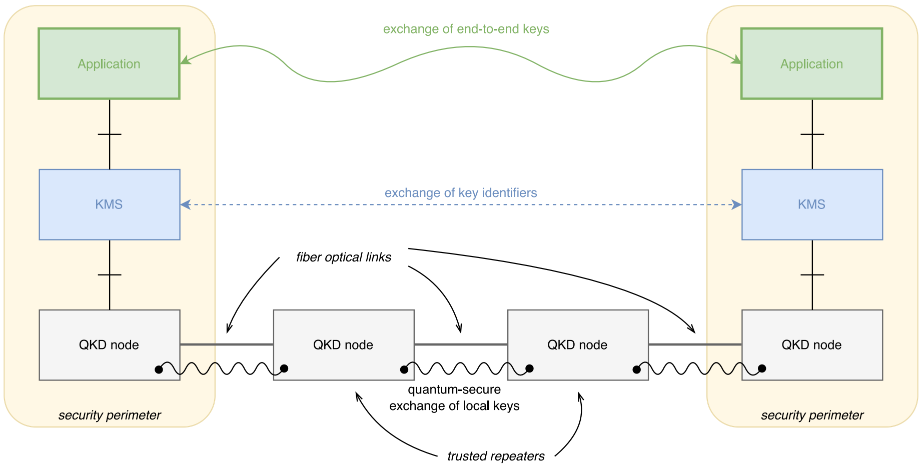

Fig. 1. Example of a QKD network with two trusted repeaters interconnecting the QKD nodes within the respective security perimeters of two applications wishing to exchange secure keys for end-to-end encryption. Each Key Management System (KMS) mediates the transfer of the keys between the application and QKD devices, and they interact with one another to exchange required pieces of information, such as the key identifiers (see Section 3). Unconditional security is only provided between pairs of QKD nodes, whereas end-to-end security relies on the intermediate ones being *trusted*.

To have practical impact on the market, QKD will have to be integrated within the existing communication and computing infrastructures. We believe that edge computing [8] provides the perfect environment for early deployments because (i) many applications, especially in the Internet of Things (IoT) domain, have strict privacy requirements; and (ii) since quantum network links, with optical fibers, are relatively short (realistically within 20-30 km), most of the infrastructures under construction lie within a metropolitan area, though with long-term plans to join these "islands" via free-space satellite links [9]. A practical use case we have implemented is the secure computation offloading of processing-intensive tasks in digital health applications [10]. The target scenario involves a hospital or care center where the patients are monitored through body sensors that generate highly-sensitive personal data to be processed through Machine Learning (ML) pipelines, which require specialized computation infrastructure that is provided by the telco operator. By using QKD it is possible to provide security guarantees, including forward secrecy, which increases the appeal and acceptance of this solution in the market.

Such a scenario, where an application requires not only computational resources but also QKD capacity in the underlying network to encrypt the communications between the user and the edge node, creates several new research challenges related to the joint optimization from the point of view of both the QKD network operator and the edge service operator. We carried out a preliminary study in [11], where we formally defined the *QKD/Edge Online Admission Problem (QEOAP)*. In brief, it consists in determining whether a new application with given Quality of Service (QoS) characteristics can be accepted and, if yes, (i) which of the edge nodes should serve it and (ii) which path should be used between the user and the edge node. We refer the interested reader to the sections II and IV of [11] for a detailed illustration of the system model and related assumptions, which are fully inherited in this work. In the paper we proposed three alternative solutions, depending on the interaction model between the QKD network and edge service operators, which we have shown to be superior to baseline approaches such as random or static allocation. However, the previous work did not consider an important aspect: it was assumed that path selection does not incur any cost, hence each application could be assigned an individual path. This might not be feasible or efficient in practice due to protocol constraints and overhead.

The main contribution of this work is to study two opposed approaches related to path selection in a QKD network: *macro-flow aggregation*, where all the applications from the same user node to the same edge node follow the same path, and *micro-flow service*, where each application can, indeed, have its own path, but signaling incurs some cost because of the network configuration required.

The remainder of the paper is structured as follows. The related work is surveyed in Section 2. In Section 3 we introduce the terminology of QKD systems and illustrate three standard protocols defined by the ETSI QKD group to provide the reader with a practical view on the main concepts used in this work. In Section 4 we describe the system model, formalize the resource management problem in mathematical terms, and address it with two classes of solutions relying on macro-flow aggregation vs. micro-flow service. The performance evaluation, based on simulations, is reported in Section 5. Finally, Section 6 concludes the paper and discusses some open research directions.

2. Related work

In this section we provide a short survey on the scientific literature relevant to our work, which covers QKD networks and edge computing. The latter has been very popular in the last decade and there are countless prior works, thus it would be daunting as much as pointless to summarize them all here. Instead, we limit ourselves to referring the interested reader to a recent survey paper [8] and highlighting the following:

Key point: While some of the studies in the literature of edge computing can be considered influential for the topic of this paper, to the best of our knowledge there are no previous works that consider the allocation of processing resources in an edge infrastructure that includes QKD, which has distinguishing features compared to classical networks. These include: limited resources (as the technology is in its infancy), unstructured topology (because of the maximum length of a single hop), constant rates (driven by the applications' security requirements).

In the remainder of this section we address the more relevant literature on QKD, which is, in general, by far more modest. In [12] the authors noted some similarities between QKD networks with Relay Nodes (RNs) and Mobile Ad-Hoc Networks (MANETs): limited one-hop distance, capacity inversely proportional to distance, hop-by-hop routing, links unreliable (due to lack of key material and possibly to the public channel becoming congested [13]). Thus, they proposed a distributed geographic reactive routing model to minimize the number of routing packets and achieve high-level scalability, which was compared to OSPFv2 with simulations. *Key difference: In their work it is assumed that resource allocation in the QKD network is disjoint from the upper layers, which we also consider in Section 4 below. However, in this paper we do not delve into the details of how to realize routing as in the work [12], which can thus be considered complementary to ours.*

A layered architecture of a complete scalable QKD system is illustrated in [14], where the authors propose to assign routes in the QKD network by solving a maximum concurrent multi-commodity flow problem, which maximizes the throughput assigned to selected pairs of users. *Key difference: The approach proposed is elegant and relatively efficient, because well-known good approximations exist for that problem, but ours is different because we have fixed demands, coming from the joint allocation of edge computing applications.*

In [15] a key-as-a-service framework is illustrated, which provides keys in a reactive virtualization-based manner, while a secret-key-aware routing method is investigated in [16], which proposes a dynamic forecast mechanism for the residual local key availability at each link. *Key difference: Both these studies are complementary to ours, where the availability of QKD (and computing) resources is checked before admission. A comparison between on-demand and reservation-based schemes is left for future work.*

The optimized deployment of a QKD network is also an active area of research. In [17] the authors proposed a strategy to find the optimal placement of QKD nodes, as well as the optimal working points of their links. In particular, the results indicate that it is generally best to operate links at their peak capacity rather than increasing their span while sacrificing link rate. On the other hand, most of the existing works assume that direct links exist between nodes, but in practice passive optical switches can also be used to create physical communication paths in an optical QKD network [18]. In this respect, the authors of [19] proposed a cost-oriented mathematical formulation of the problem of deploying QKD in a fiber optical transport network, while satisfying the secret-key rate demands of all the requests, to be solved by heuristic algorithms. A generalization of this problem has been recently addressed by Zhang et al. [20] to jointly select the path, channel, key-rate, and time-slot assignment, also deciding if a trusted repeater or optical bypass should be used to interconnect nodes without a direct link. *Key difference: In this study we assume that the topology, physical or logical through passive optical networks, has been pre-provisioned and its characteristics, in terms of the capacities of links that interconnect QKD nodes, are constant throughout the lifetime of the resource management process. The integration of the results illustrated here with a dynamically changing topology is left for future investigation.*

Finally, we mention that a complete stack for the integration of QKD in a cloud-native environment was proposed in [21], where the authors focused on the interfaces between the main components, such as QKD devices and the KMS. This work confirms the interest of the research community on the use and optimization of QKD systems as an integral part of upcoming telecommunications/computing infrastructures, as also corroborated by the authors of [22], who presented realistic use cases from a telco provider's perspective.

3. Background

In this section we provide the reader with basic QKD terminology and background information on the European Telecommunications Standards Institute (ETSI) QKD standard specifications that are relevant to the core contribution (Section 4). The interested reader can find a comprehensive illustration of the topic in the survey [23]. Even if the paper contribution is abstract, and it does not depend on the specific standard/framework adopted, we think it is beneficial to position it concretely with respect to existing technologies, to best understand the design choice and assess the long-term impact of the findings. In particular, we refer to ETSI standards because the industry study group on QKD was founded in 2008, that is earlier than many others active today, and it has since become a reference for many scientific studies and implementation efforts, such as the integration of QKD in OpenSSL carried out within the Pan-European Quantum Internet Hackathon 2019, in Delft (Netherlands).[1] The set of specifications mentioned in this section are reported in Table 1 and they will be referred in the following for brevity only by their number, i.e., QKD 004, QKD 014, and QKD 015. **QKD 014** defines an Application Programming Interface (API) that allows an application (APP) to query its Key Management System (KMS).[2] via HTTP commands, as illustrated in Fig. 2. Assuming that the initiative is taken by APP-A, the latter will request a fresh key towards APP-B. If a key is available, the KMS will reply with the key material, i.e., the sequence of secret bits that will also become available to APP-B, and an identifier of the key (key ID), which is used to distinguish among multiple sequences that may be valid in a given time frame between A and B. Such identifier has to be transferred to APP-B, so that it may also retrieve the *same* key material from *its* KMS. Only afterwards, the two applications can use the shared secret to encrypt/decrypt communications, e.g., with AES-256 or any other symmetric-key algorithm. Depending on the specific application, the same key

[1] https://github.com/brunorijsman/openssl-qkd

[2] The name of this element changes slightly in the literature, and even across the same family of standards. For instance, in 014 it is called Key Management Entity (KME) Throughout the paper, we always adopt the name KMS for consistency reasons.

Table 1
ETSI QKD standard specifications considered as reference in this work.

Specification	Version	Date	Title
ETSI GS QKD 014	V1.1.1	2019–02	Protocol and data format of REST-based key delivery API
ETSI GS QKD 004	V2.1.1	2020–08	Application Interface
ETSI GS QKD 015	V2.1.1	2022–04	Control Interface for Software Defined Networks

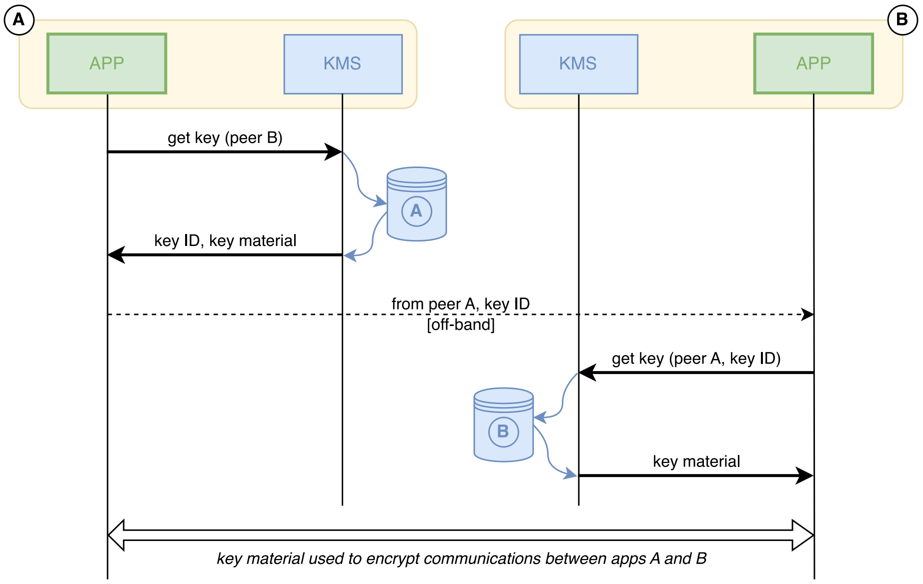

Fig. 2. QKD 014: basic interactions between the applications and their respective KMSs to retrieve a single key used to encrypt future communications. The KMS offers a REST interface, that is HTTP is used by the APP for the 'get key' command. The key has to be refreshed periodically until the communication remains active. The dashed arrow represents the transfer from APP-A to APP-B of the identifier of the key to be used by the latter to query its KME; such interaction happens off-band, i.e., it is left unspecified by the standard.

can be reused for a number of encryptions or until an expiration time, then the procedure must be repeated. The APP and KMS must be in the same security parameter, otherwise the transfer of the key could be compromised and jeopardize the security of the overall system.

QKD 014 implicitly assumes that the KMSs have a buffer of keys pre-established to be returned on demand to the applications. All the QKD devices continuously generate secret keys on all their QKD links, but such keys are *local* to the two nodes (refer to Fig. 1 in Section 1), whereas applications request keys that are *end-to-end*, i.e., that are valid between a specific pair of nodes anywhere in the network. *But how can KMS-A know that APP-A should cache keys to reach APP-B because such request will arrive in the future?* One possibility is to use reservations, which is particularly enticing with edge computing applications, since QKD network reservation can be combined with the procedure to provision computational resources on edge nodes, which have to be carried out anyway before the service can be activated. **QKD 004** can be used precisely for this purpose, since it defines a protocol for an application to create a session of secret key exchanges with another application, subject to QoS parameters, such as the minimum key rate. Briefly, the protocol consists of three phases: a negotiation phase during which the session is established and the QoS parameters are consolidated; an active phase during which the key buffers are kept up-to-date by periodically pulling new key material associated to a given 'key stream ID'; and, finally, the ending phase where the session is terminated and the reserved resources are freed.

QKD 014/QKD 004 do not specify the algorithms for path selection, or other allocation/optimization of resources. However, the way the interactions are defined implies an architecture that inspires from classical networks, where routing happens as a background process influenced by topology changes and capacity reservations and produces an allocation of paths between any two possible nodes. On the other hand, **QKD 015** puts forward an architecture that adheres to Software Defined Networking (SDN) concepts, as illustrated in Fig. 3. The key difference is that applications request the creation of *flows*, by borrowing SDN terminology, with identified source, destination, and QoS requirements, which are then managed in a centralized manner by the SDN controller, who has a complete view on the topology and active flows. This approach has been already demonstrated in a metropolitan QKD network in the city of Madrid, Spain [24].

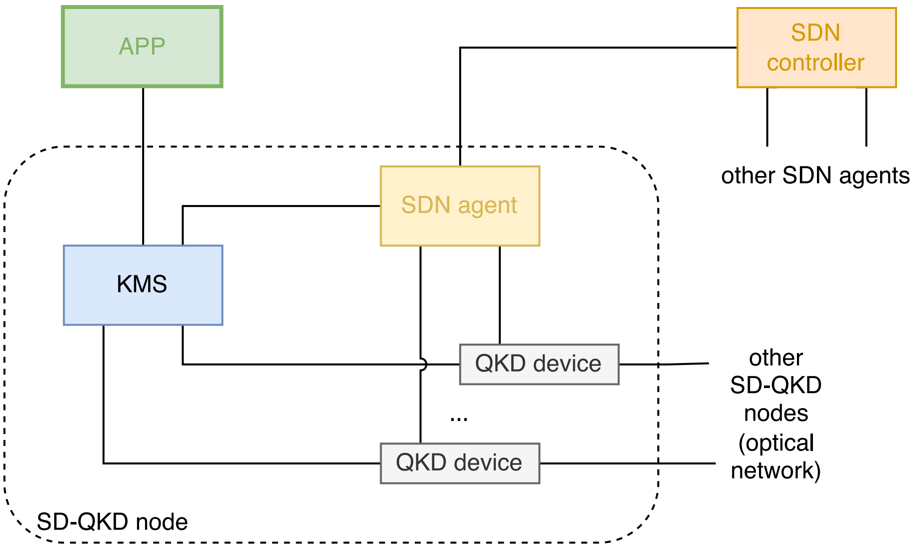

Fig. 3. QKD 015: architecture of a SD-QKD node, which consists of a KMS, which mediates interaction with the applications, and QKD devices, which interconnect through a physical layer with the underlying optical network, controlled by system-wide SDN controller through a local SDN agent.

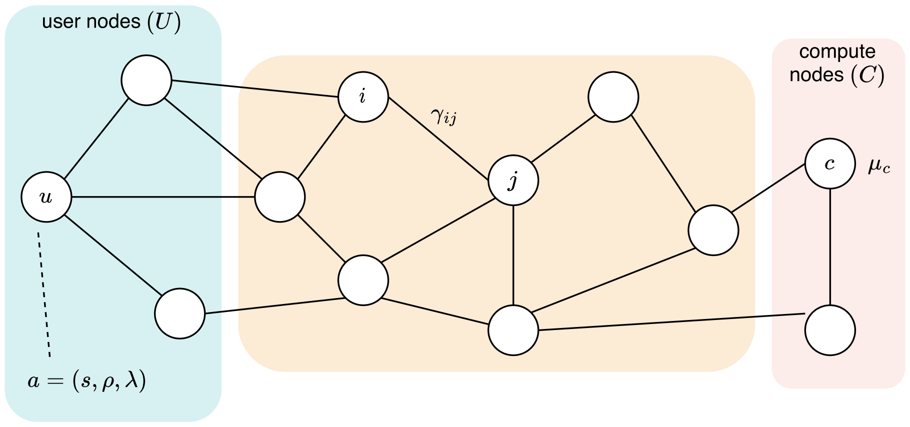

Fig. 4. The model used in this paper. The QKD capacity between two adjacent QKD nodes (i, j) is γ_{ij}, in terms of the secret key rate. The processing capacity of a compute node c is μ_c. An application a requesting admission is characterized by its source node (s), minimum key rate (ρ), and computational load required (λ).

4. Resource management

In this section we illustrate the main contribution of this work, which is a framework for the allocation of resources in a QKD network (see Section 3) and edge computing infrastructure, following a macro-flow aggregation or micro-flow service model, which will be evaluated in the next section (Section 5).

4.1. System model

We illustrate the system model with the help of the example in Fig. 4. The QKD devices are modeled as vertices in a graph $G(V, E)$, where the edges are the links in the QKD network. Without loss of generality, in V we distinguish $U \subset V$ as the **user nodes**, i.e., nodes that can host applications from different customers, and $C \subset V$ as the **compute nodes**, i.e., the edge servers that are made available by the telco operator for edge computing applications. An application is defined by the user node $s \in U$ and the QoS requirements: minimum key rate ρ, also called QKD network capacity, and computational load λ. The system resources are: the QKD capacity of the links, γ_{ij} for the edge between nodes $i, j \in V$, in terms of the secret key rate, and the processing capacity of compute nodes, μ_c for node $c \in C$.

The system model was originally defined in our previous work [11], with the only difference that the capacity on the classical network was also considered a system resource subject to constrained allocation. In this work we consider only the QKD network

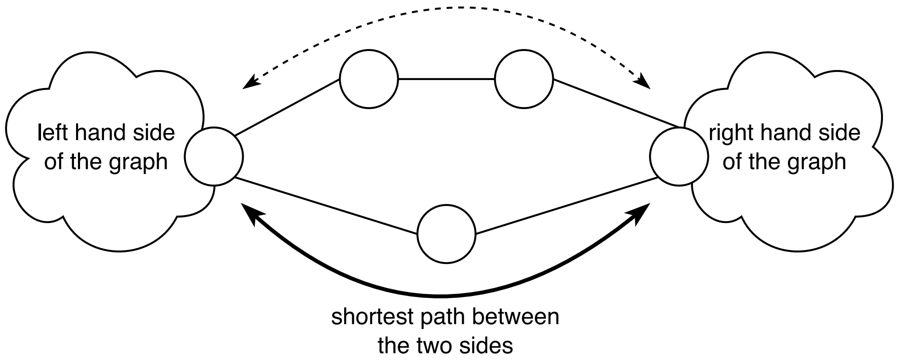
left hand side of the graph
right hand side of the graph
shortest path between the two sides

Fig. 5. Example of a graph where a path (the top one with a dashed line) is never used by policy 014.

capacity because, due to the emerging nature of this technology, it is much more likely to be a performance bottleneck than classical network capacity in practical deployments. In the same work we defined the QKD/Edge Online Admission Problem (QEOAP) as that of determining whether a given application $a = (s, \rho, \lambda)$ can be admitted, which is iff:

- there is a non-empty set of compute nodes C' with residual computational capacity $\mu_c \geq \lambda, c \in C'$; and,
- there is at least a processing node $\bar{c} \in C'$ with a path in the QKD network such that all the intermediate links (i, j) have sufficient residual capacity to support the required key rate, i.e., $\gamma_{ij} \geq \rho$.

In [11] we proposed three different heuristics to solve QEOAP, depending on who takes the admission decision: the QKD network operator (policy SPF), the edge computing service operator (policy BF), or a third-party (policy ANY). The simulation results have shown that all the policies have comparable performance, in terms of the admission ratio, which was the key metric, and they all exceed significantly the performance of simpler comparison algorithms, i.e., random and location-based selection of the compute node. However, they all assumed that each application can be assigned its micro-flow in the QKD network at no cost and independent from all the other active applications. In this work, we build on the previous results obtained but progress beyond this simplistic approximation, as discussed hereafter.

4.2. Policies

We argue that the most important design choice is whether QKD routing should adopt a classical IP or SDN-inspired approach. With classical IP, there is a single route between a user and a compute node, and such paths are pre-computed during the network setup or as part of a sporadic reconfiguration process. We call this approach **014** because it matches well the ETSI QKD 014 specifications (see Section 3), where applications asynchronously request keys for a specific destination assuming that there is a corresponding buffer available at their KMS. Therefore, when an application requests to be admitted, the only choice is which compute node with sufficient residual processing capacity will handle it, provided that the known path from the user node has sufficient residual QKD network capacity. Since there might be multiple such compute nodes, like in [11] and following best practices from Operations Research, we propose to break ties by selecting the compute node that leaves the smallest residual, which is known as 'best-fit' algorithm. Formally, we can define the Policy 014 as follows.

Policy 014. Assume that $\mathcal{P}_1$ contains all the shortest paths from $u \in U$ to $c \in C$ and let $G' = (V, E')$ be the graph with the residual QKD network capacities left by the active applications and $\{\mu_c\}$ the corresponding vector of residual processing capacities. Let $a = (s, \rho, \lambda)$ be a new application requesting to be admitted. First, find:

$$\forall c \in C : \gamma_c = \min_{(i,j) \in p_1(s,c)} \gamma_{ij}. \tag{1}$$

Then, select the target compute node $\bar{c}$ as follows:

$$\bar{c} = \underset{c \in \{c : \gamma_c \geq \rho\}}{\operatorname{argmin}} \left\{ \mu_c - \lambda | \mu_c \geq \lambda \right\}. \tag{2}$$

The application a is admitted iff $\bar{c}$ is found, in which case it is allocated the path $p_1(s, \bar{c}) \in \mathcal{P}_1$.

In the policy above, we assume shortest path routing because this strategy is advocated generally by the research community for QKD (e.g., in [19]). Ideally, with policy 014 the routing paths can be constructed once and for all based only on the topology, which means that the signaling overhead due to the their reconfiguration at run-time is null. However, this can lead to underutilization of the QKD network, since there may be paths that are never used, regardless of the current set of active flows or any other conditions, see for example Fig. 5. To overcome this shortcoming, we propose a generalization of 014 that reconfigures the routing of paths to exploit unused resources.

Policy 014 (K-generalized). Assume that $\mathcal{P}_1$ contains all the shortest paths from $u \in U$ to $c \in C$, and $\mathcal{P}_2$ contains all the second-shortest paths, and so on until $\mathcal{P}_K$. Let us call $p_1(s,c)$ the primary path from s to c; all other $K-1$ paths $p_2(s,c),\ldots,p_K(s,c)$ are called secondary paths. Let A be the set of active applications. The other assumptions are the same as policy 014. First, search for $\bar{c}$ as in the Policy 014 above, i.e., by looking only into the primary paths: if found, then use it. Otherwise, for each compute node c, find ρ_c, which is the total QKD network capacity from user node s that has to be migrated from the current primary path to a secondary path on each disjoint link:

$$\forall c \in C, \rho_c = \sum_{a \in A | c = c_a} \rho_a. \tag{3}$$

Then, let I_{ijsc} be an indicator function of whether link (i,j) is contained in the primary path from s to c:

$$I_{ijsc} = \begin{cases} 0 & (i,j) \in p_1(s,c) \\ 1 & (i,j) \notin p_1(s,c) \end{cases}, \tag{4}$$

and select all the feasible secondary paths $\mathcal{F}$ as follows:

$$\mathcal{F} = \left\{ c \in C, k \in \{2,\ldots,K\} : \forall (i,j) \in p_k(s,c), \gamma_{ij} \geq \left(\rho + I_{ijsc}\rho_c\right)\right\}. \tag{5}$$

Finally, select the target compute node $\bar{c}$ and secondary path $\bar{k}$ as follows:

$$(\bar{c},\bar{k}) = \operatorname*{argmin}_{(c,k)\in\mathcal{F}} \left\{ \mu_c - \lambda | \mu_c \geq \lambda \right\}. \tag{6}$$

The application a is admitted iff $(\bar{c},\bar{k})$ is found, in which case it is allocated to compute node c along the path $p_{\bar{k}}(s,\bar{c})$, which becomes the new primary path for all applications between s and c, including those already active. The number of hops that have to be reconfigured is:

$$\left| p_1(s,\bar{c}) \oplus p_{\bar{k}}(s,\bar{c}) \right|, \tag{7}$$

that is the set of links that belong only to the former primary path or to the new primary path.

We call the above 'K-generalized' because with $K = 1$ it collapses to the plain policy 014 defined earlier in this section, the main difference being that with $K \geq 2$ the routing paths are subject to reconfiguration at run-time. The higher the value of K, the broader will be the search for alternatives in secondary paths. In addition to increasing the execution time upon admission, choosing a higher value of K also increases the chance of injecting paths longer than the shortest one, which consume more QKD network resources.

On the opposite end of the spectrum, we find SDN-inspired approaches, where each application is assigned a dedicated micro-flow configured with an individual network configuration, as enabled by the ETSI QKD 015 specifications , for instance . For this reason, we call the next policy 015, formally defined as follows.

Policy 015. Let $G' = (V, E')$ be the graph with the residual QKD network capacities left by the active applications and $a = (s, \rho, \lambda)$ a new application requesting to be admitted. First, create a temporary graph $\tilde{E}$ containing only the links that can support the new application, i.e., $(i,j) \in \tilde{E}$ if $\gamma_{ij} \geq \rho$. Then, select $\tilde{C} \subseteq C$ as the subset of compute nodes that have enough residual capacity to host the new application , i.e., $c \in \tilde{C}$ if $\mu_c \geq \lambda$. Let us call $p(s,c)$ the shortest path from s to $c \in \tilde{C}$ in $(V, \tilde{E})$. Finally, select the compute node $\bar{c}$ as follows:

$$\bar{c} = \operatorname*{argmin}_{c\in\tilde{C}} \left\{ \mu_c - \lambda \right\}, \tag{8}$$

for which the number of hops that have to be configured in the QKD network is $|p(s,\bar{c})|$.

The policy 015 is the same as BF in [11], which does not consider the QKD network configuration overhead. To improve this aspect, we propose below a variation of the policy that preliminarily checks if the new application can be aggregated to one already active, which would require no configuration at all in the QKD network.

Policy 015 (with reuse). Let A be the set of applications already active, which contains tuples $a = (s_a, \rho_a, \lambda_a, c_a, p_a)$. The other assumptions are the same as policy 014. If $\exists a \in A$ s.t.:

$$s_a = s \wedge \mu_{c_a} \geq \lambda \wedge \min_{(i,j)\in p_a} \gamma_{ij} \geq \rho, \tag{9}$$

then allocate a to c_a along path p_a. Otherwise, proceed with policy 015.

Table 2
Number of hops to configure with new application accepted for the policies in Section 4.2.

Policy	Number of hops
$014 \| k = 1$	0
$014 \| k \geq 2$	0 if the path is found in $\mathcal{P}_1$, otherwise number of disjoint links between the primary path and the secondary path selected
015	Length of the path selected
015| reuse	0 if the path of an active application is used, otherwise see 015

Table 3
Worst-case time complexity of the policies in Section 4.2.

Policy	Number of operations
014 (prep)	$\mathcal{O}(UCKV(E + V \log V))$
014 (exec)	$\mathcal{O}(CDK \log CK)$
015 (exec)	$\mathcal{O}((E + C) + C(E + V \log V) + C \log C)$
015 (exec-reuse)	$\mathcal{O}(AD)$

Table 4
Characteristics of the network topologies used in the simulations.

Topology	Num links	Min degree	Max degree	Diameter
Regular network	314–538	1–6	11–23	3–5
Large network	3376–4170	1–2	15–28	6–9

4.3. Analysis

We conclude the section by analyzing some static properties of the policies defined above, whose run-time performance will be assessed in the next section through extensive simulation. First, in Table 2 we report the number of hops that the QKD network operator has to configure whenever a new application is accepted to enforce the path that was selected. Second, in Table 3 we show the worst-case time complexity of the policies, where D is the diameter of the network, i.e., the length of the maximum path that can be found; in the table, for better readability we use the name of a set to indicate its cardinality, e.g., V in the table is the *number* of nodes in the network. The complexity analysis assumes that the shortest path is found using Dijkstra's algorithm (or Yen's algorithm for the k shortest paths) using a minimum heap implementation. The complexity for the 014 policy is divided in two rows: 'prep' is for the preparation of the data structures with shortest paths from all user nodes to all compute nodes, which is done once and for all, while 'exec' is the cost incurred at every new application arriving. On the other hand, the 015 policy does not have a preparatory state. The 'exec-reuse' applies to the step performed to try and aggregate the new application to an active flow, which is optional.

5. Performance evaluation

In this section we evaluate the performance of the policies 014 and 015 defined in Section 4. We first illustrate the methodology adopted (Section 5.1) and then we report and discuss the results found (Section 5.2).

5.1. Methodology, assumptions, and tools

Due to the lack of large-scale QKD networks in production, we evaluated the performance using simulation with QueeR[3] (tag `v1.7`, experiment 007), which we maintain as part of our research activities and was used already for studies on traffic engineering of entanglement-based quantum networks. The software is open source and shipped with simulation scripts and artifacts for full reproducibility.

The network topology is generated using the Waxman model [25], which is widely adopted in the performance evaluation of quantum networks without a regular structure (e.g., [12]). Accordingly, in our simulations the nodes are uniformly distributed in a plane with maximum inter-node distance L and probability of two nodes being connected $p = \beta \cdot e^{-d/\alpha L}$, where d is the distance (in km) between the nodes. The α, β, and L parameters determine the properties of the random networks generated; in a real deployment, they are related to the provisioning choices made by the network operator with regard to the density of links (increasing with α and β) and the trade-off between connectivity and capacity (L). In this paper, we only consider two topologies with the characteristics in Table 4: a *regular network*, which is representative of a typical medium metropolitan-size deployment, with 50 nodes, $\alpha = \beta = 0.4$; a *large network*, which targets a larger deployment, with 500 nodes, $\alpha = \beta = 0.125$. In both topologies

[3] https://github.com/ccicconetti/queer

Table 5
Metrics used in the performance evaluation in Section 5.2.

Metrics	Description
Blocking probability	Probability that an application is rejected
Residual network capacity	Ratio between the network capacity left unused and the total capacity available in the network
Residual processing capacity	Ratio between the processing capacity left unused and the total processing capacities
Path length	Number of hops between the user node where the application originates and the compute node to which it is assigned
Allocation execution time	Time it took the admission control and allocation algorithm to run on the server used for the simulation
Signaling overhead	Number of links that need to be configured to establish the path between the user and compute nodes to admit a given application

it is $L = 100$ km, because after such a distance the link rate becomes negligible; the user and edge nodes are selected randomly. The study of how the α, β, and L parameters affect the system performance is deferred to future work, as in this paper we focus on the comparison between the opposite design approaches 014 vs. 015. The rate of QKD links is assumed to decrease exponentially with distance, which is reasonable due to attenuation/absorption effects: $\gamma = \gamma^{\max} e^{-d\eta}$. We set $\gamma^{\max} = 133\ kb/s$ and $\eta = 0.0609/km$ to match the experimental results obtained in [26], where the authors evaluated the effectiveness of a time-interleaving technique to enable the co-propagation of quantum and classical channels in the C-band with minimal mutual interference.

For the workload, we adopted a trace-driven approach starting from a dataset of traces obtained in 2020 from a production system and publicly released by Microsoft [27]. The dataset contains a log of read/write backend activities of user applications, which we transformed into a workload matching our required input as follows: the application load λ is equal to the event rate in the trace (ranging from 0.008 to a capped value of 1) and the QKD capacity requested by the application ρ is the common logarithm of the traffic rate (values range from 1 kb/s to 8 kb/s). The trace transformation tool is publicly available on GitHub[4] (tag `dataset-002`).

We adopted a Monte Carlo approach: for each scenario we ran a number of independent runs , each with a random topology and a random arrival process of applications requesting admission. Every replication was evaluated through a flow-level simulation with duration equal to that of the original trace and a 15% initial warm-up period, handling the arrival and departure of applications as events. The simulator does not model the transmission of individual qubits or packets. Each application in the trace has the same probability of being selected as the next one to request admission and the admission of each application is evaluated independently according to one of the algorithms defined in Section 4.[5] If the application is blocked, then we simply keep track of the event for the purpose of collecting the simulation metrics; otherwise, if the application is accepted, then its resources are reserved on the edge node selected and along the corresponding path. The arrival of applications follows a Poisson distribution, and each accepted application remains in the system for a duration drawn from an exponential r.v. with average equal to the lifetime read from the trace. In the plots in Section 5.2 we report the average values from all such independent runs , whose number (5000) is such that a relatively small variance was measured (not shown in the plots for better readability). The performance was measured according to the metrics in Table 5.

5.2. Results

We begin our analysis with two scenarios in a regular network topology with increasing load: *processing-bound*, with relatively scarce compute resources (i.e., nodes are assigned a random $\mu \in [3, 7]$), and *network-bound*, with plentiful compute capabilities (i.e., $\mu = 100$ for all nodes) but reduced network capacity (i.e., $\gamma^{\max} = 66.5$ kb/s, which is half the peak capacity used elsewhere) . In the following we report the results in pairs of plots for the two scenarios to showcase the impact on several system metrics (i.e., blocking probability, average path length, residual processing/network capacity, and signaling overhead, respectively) in two limit conditions where the scarce resource is computation vs. network. However, we will not compare them in quantitative terms with one another because they have been run in different conditions. In Fig. 6 we show the blocking probability, which increases with the load in all cases, as expected. As can be seen, $014|k = 1$ performs worse than the others in both scenarios, because of the static allocation of user-compute pairs to a path, which does not allow to adapt to changing conditions. With $k = 3$ the performance improves thanks to the dynamic reconfiguration of the routing paths. The best performance is achieved by policy 015, with negligible impact of the reuse option.

[4] https://github.com/ccicconetti/support/tree/master/Dataset/002_Lifecycles

[5] For the policy 014, we consider two cases: $K = 1$, which is the special case when the allocation of paths is performed as part the network provisioning and is never changed at run-time, and $K = 3$, which we have found to provide a good trade-off between performance improvement and complexity/signaling overhead in preliminary experiments not included in Section 5.2.

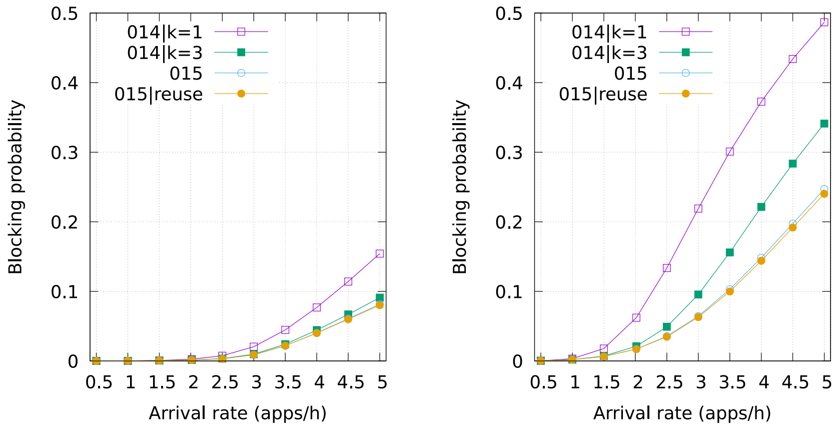

Fig. 6. Regular network, increasing load, processing-bound (left) vs. network-bound (right): blocking probability.

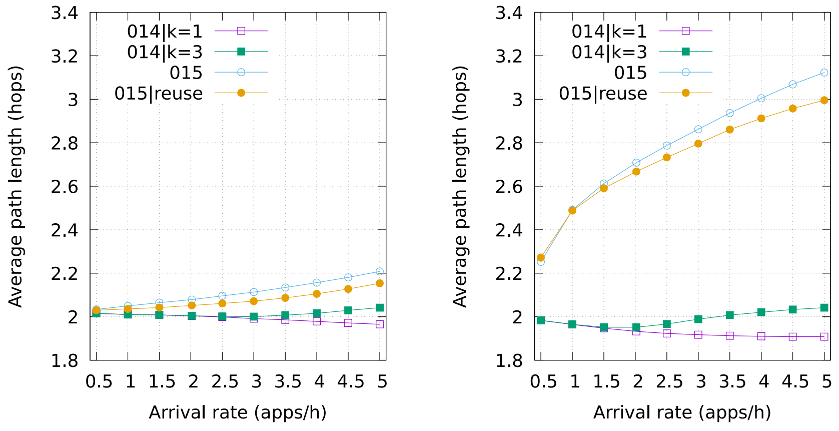

Fig. 7. Regular network, increasing load, processing-bound (left) vs. network-bound (right): average path length.

The low blocking probability with policy 015 is achieved at the cost of using longer paths, as shown in Fig. 7: while the average path length with the 014 policies is about 2 hops at both low and high loads, the same metric increases significantly with the load using a 015 policy. In particular, the effect is more prominent without the reuse option, which has the effect of reinforcing existing paths before unexplored, possibly longer, ones are selected. The average path length with $014|k = 1$ decreases with the load: as more and more requests are being rejected, those with a shorter path have a higher chance of being accepted, which creates a biased allocation. In addition to consuming more network capacity, longer paths also incur a higher latency of the QKD protocol, which can negatively affect delay-sensitive applications. This however does not impact on the results presented in the paper, as it focuses on the long-term provisioning of resources for applications, thus handling *average* rates.

The residual bottleneck resources, compute vs. network capacity, are shown in Fig. 8. The behavior in the two cases is very different: in a processing-bound scenario (left side) the residual capacity at a high load is very small, i.e., all the policies are able to consume effectively a large fraction of the bottleneck resource. This is especially true for 015 policies , which leave a little more of 10% of resources unused with a blocking probability of less than 10%. On the other hand, in a network-bound scenario, the residual network capacity remains high with all the policies and even at high loads. This observation is consistent with our assumptions that the QKD network is unplanned and it suggests that an accurate network provisioning is required to provide a good utilization of the network resources.

We now show the signaling overhead in Fig. 9. The case $014|k = 1$ is not reported since it does not require any configuration of the network except for the setup of the initial flows between user and compute nodes. As expected, the policy 015, originally proposed in [11], incurs the highest cost in terms of signaling, because no provisions are made in that sense. The reuse option, instead, reduces significantly the network reconfiguration overhead, which becomes comparable with $014|k = 3$ in the processing-bound scenario.

In Fig. 10 we report the blocking probability in a sensitivity analysis where we keep the arrival rate constant and equal to 3.6 apps/h and draw randomly $\mu \in [3, 7]$ while increasing the number of compute nodes $|C|$ (with 10 user nodes, left) and user nodes $|U|$ (with 10 compute nodes, right). In both plots, all the curves decrease with increasing $|C|$ or $|U|$: this is straightforward for $|C|$, since adding compute nodes results in more resources available in the system; for $|U|$, the decreasing blocking probability

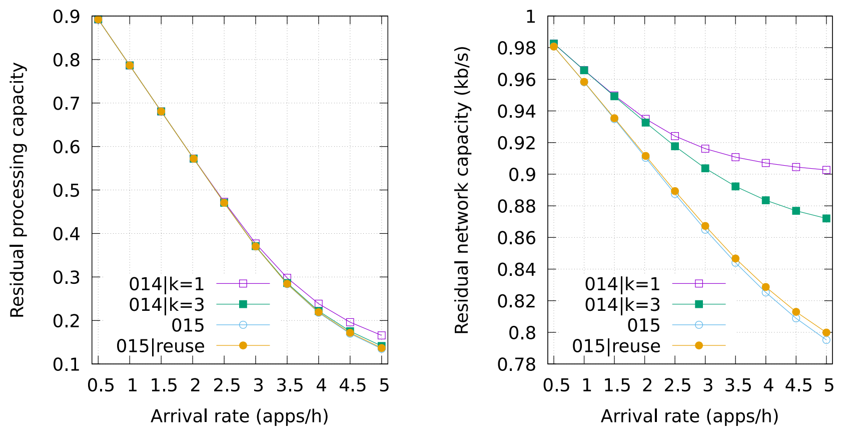

Fig. 8. Regular network, increasing load, processing-bound (left) vs. network-bound (right): residual processing capacity (left) and residual network capacity (right).

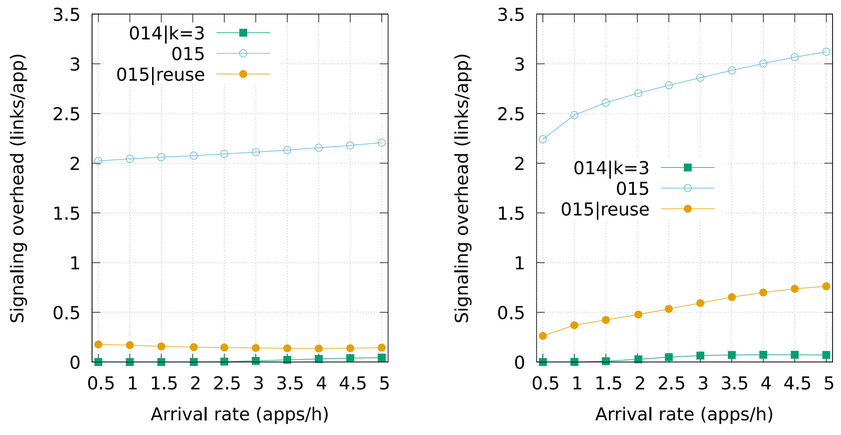

Fig. 9. Regular network, increasing load, processing-bound (left) vs. network-bound (right): signaling overhead.

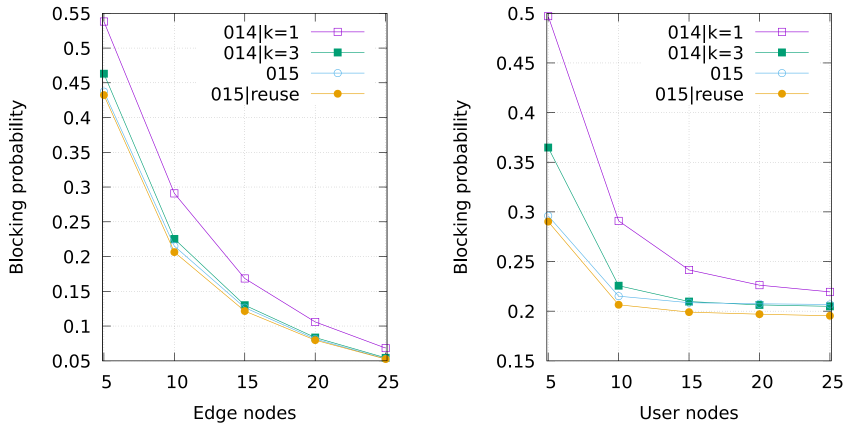

Fig. 10. Regular network, varying number of compute $|C|$ (left) and user $|U|$ (right) nodes: blocking probability.

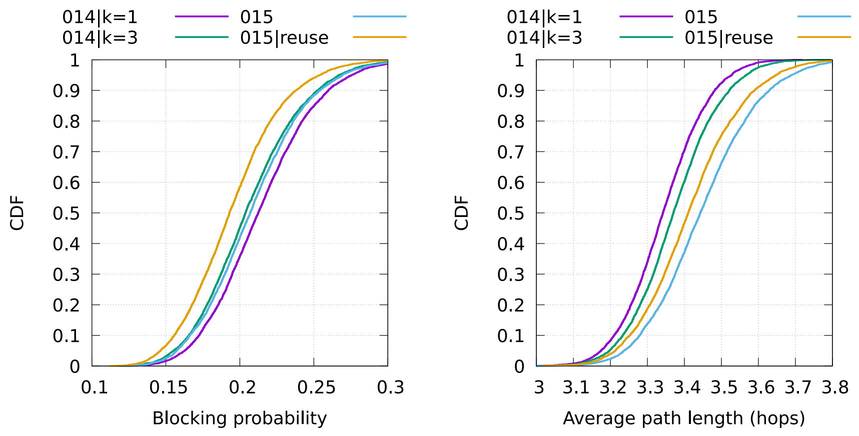

Fig. 11. Large network: cumulative distribution function of the blocking probability (left) and the average path length allocation execution time (right).

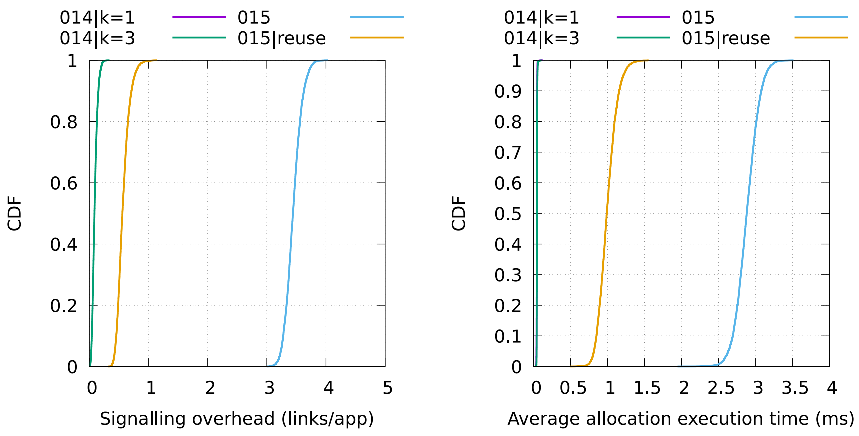

Fig. 12. Large network: cumulative distribution function of the signaling overhead (left) and the allocation execution time (right).

is because it is easier to find empty paths/links if the users are spread over multiple nodes. In both cases, policy 015 is confirmed to perform best, with $014|k = 3$ not so distant, especially with a high number of compute/user nodes.

We conclude the analysis by reporting the results obtained in a large scenario, with 10 compute nodes, 45 user nodes, and $\mu \in [3, 7]$. The Cumulative Distribution Function (CDF) of the blocking probability obtained in the multiple independent runs is shown in the left part of Fig. 11, where the 015|reuse curve is significantly on the left of the others, despite the longer paths selected compared to the 014 policies (see right part of Fig. 11). Interestingly, the 015 policy without reuse suffers from a higher blocking probability than that with reuse, possibly because of the even longer path lengths.

Finally, a further trade-off between aggregated (014) and micro-flows (015) exists in terms of the signaling overhead and allocation execution time, which are reported in Fig. 12. On the one hand, $014|k = 3$ incurs a negligible overhead of path reconfiguration; on the other hand, the overhead increases significantly for policy 015 and it is, for each new application, $[0.5, 1]$ hops with reuse, $[3, 4]$ hops without. Finally, the allocation procedure is also more complex, in terms of the number of operations that have to be performed. In the right part of Fig. 12 we report the average allocation times obtained in our experiments, which have been run on a dual-socket Intel Xeon Platinum 8164 with 512 GB RAM. The exact numbers will change depending on a number of factors (implementation, compilation optimizations, hardware, etc.) but from the plot it is clear that policies 015 are much more expensive than policies 014, and this is only partially compensated by the reuse option.

6. Conclusions

In this paper we have studied the problem of the allocation of resources in a QKD network to serve applications running in an edge computing infrastructure. More specifically, we have focused on the problem of a practical realization of the path allocation through one of two standards within the ETSI ISG QKD family: QKD 004/014, which naturally leads to assigning aggregated macro-flows for the same source/destination nodes, and QKD 015, which is inspired from SDN concepts and foresees the allocation of micro-flows to applications. For each class, we have proposed heuristic joint resource allocation algorithms building on prior results and best practices in the literature, aimed at assessing the fundamental trade-offs between the two competing design choices. We have carried out an extensive performance evaluation campaign using dynamic simulation of random-generated network topologies and trace-driven workload composition. The results have shown that using micro-flows can significantly increase the system utilization/decrease the blocking probability, compared to using aggregated macro-flows. However, this came at the cost of a higher signaling overhead, due to the frequent network configuration, and a more complex allocation algorithm. Furthermore, micro-flows tend to have longer paths, which can have additional practical disadvantages. Within each class: the pre-allocation of $K > 1$ paths with policy 014 brought significant advantages, compared to $K = 1$, at a very limited cost; in policy 015, reusing the path of active flows proved to be valuable since it reduced significantly the signaling overhead without a noticeable loss in terms of other performance metrics.

The work presented can be extended in many directions, including: extension of the algorithms to the case of a dynamically changing topology, due to new nodes/links being added or node/link failure or dynamic reconfiguration of the light paths in a passive optical network; integration with L2/L3 QKD routing protocols; generalization to untrusted relay nodes; implementation and performance evaluation in a real QKD infrastructure; design of provisioning solutions that can significantly improve the utilization of the QKD network resources compared to what has been obtained with random topologies.

CRediT authorship contribution statement

Claudio Cicconetti: Software, Methodology, Investigation, Data curation, Conceptualization, Visualization, Writing – original draft. **Marco Conti:** Writing – review & editing, Funding acquisition, Supervision. **Andrea Passarella:** Funding acquisition, Supervision, Writing – review & editing.

Declaration of competing interest

The authors declare that they have **no known competing financial interests or personal** relationships that could have appeared to influence the work reported in this paper.

Data availability

The code used in the paper is publicly available on GitHub. A link to the repository is included in the paper.

Acknowledgments

The work of A. Passarella was partially supported by project RESTART (PE00000001) under the MUR National Recovery and Resilience Plan funded by the European Union – NextGenerationEU. The work of C. Cicconetti was partially supported by project SERICS (PE00000014) under the MUR National Recovery and Resilience Plan funded by the European Union – NextGenerationEU. The work was partly funded by the European Commission under Grant Agreement No 101091408 (QUID project). This document reflects only the authors' view and the European Commission is not responsible for any use that may be made of the information it contains.

References

[1] C.H. Bennett, G. Brassard, Quantum cryptography: Public key distribution and coin tossing, Theoret. Comput. Sci. 560 (2014) 7–11.
[2] L. Gyongyosi, S. Imre, Advances in the quantum internet, Commun. ACM 65 (8) (2022) 52–63.
[3] P.-Y. Kong, Challenges of routing in quantum key distribution networks with trusted nodes for key relaying, IEEE Commun. Mag. (2023) 1–7.
[4] D. Bacco, I. Vagniluca, B. Da Lio, N. Biagi, A. Della Frera, D. Calonico, C. Toninelli, F.S. Cataliotti, M. Bellini, L.K. Oxenløwe, A. Zavatta, Field trial of a three-state quantum key distribution scheme in the florence metropolitan area, EPJ Quantum Technol. 6 (1) (2019) 5.
[5] R.C. Berrevoets, T. Middelburg, R.F.L. Vermeulen, L.D. Chiesa, F. Broggi, S. Piciaccia, R. Pluis, P. Umesh, J.F. Marques, W. Tittel, J.A. Slater, Deployed measurement-device independent quantum key distribution and bell-state measurements coexisting with standard internet data and networking equipment, Comm. Phys. 5 (1) (2022) 1–8.
[6] ITU-T, Quantum Key Distribution Networks – Key Management, Tech. Rep. Y.3803, 2020.
[7] F. Rozpędek, R. Yehia, K. Goodenough, M. Ruf, P.C. Humphreys, R. Hanson, S. Wehner, D. Elkouss, Near-term quantum-repeater experiments with nitrogen-vacancy centers: Overcoming the limitations of direct transmission, Phys. Rev. A 99 (5) (2019) 052330, arXiv:1809.00364.
[8] Q. Luo, S. Hu, C. Li, G. Li, W. Shi, Resource scheduling in edge computing: A survey, IEEE Commun. Surv. Tutor. 23 (4) (2021) 2131–2165.
[9] J. Yin, et al., Satellite-based entanglement distribution over 1200 kilometers, Science 356 (6343) (2017) 1140–1144.
[10] C. Cicconetti, M. Conti, E. Lella, P. Noviello, G.D. Paduanelli, A. Passarella, E. Storelli, A prototype for QKD-secure serverless computing with ETSI MEC, in: 2023 IEEE International Conference on Smart Computing, SMARTCOMP, IEEE, Nashville, TN, USA, 2023, pp. 189–190.
[11] C. Cicconetti, M. Conti, A. Passarella, Qkd@Edge: Online admission control of edge applications with QKD-secured communications, in: 2023 IEEE International Conference on Smart Computing, SMARTCOMP, IEEE, Nashville, TN, USA, 2023, pp. 65–73.
[12] M. Mehic, P. Fazio, S. Rass, O. Maurhart, M. Peev, A. Poppe, J. Rozhon, M. Niemiec, M. Voznak, A novel approach to quality-of-service provisioning in trusted relay quantum key distribution networks, IEEE/ACM Trans. Netw. 28 (1) (2020) 168–181.
[13] M. Mehic, O. Maurhart, S. Rass, D. Komosny, F. Rezac, M. Voznak, Analysis of the public channel of quantum key distribution link, IEEE J. Quantum Electron. 53 (5) (2017) 1–8.
[14] P.K. Tysowski, X. Ling, N. Lütkenhaus, M. Mosca, The engineering of a scalable multi-site communications system utilizing quantum key distribution (QKD), Quantum Sci. Technol. 3 (2018) 024001.
[15] Y. Cao, Y. Zhao, J. Wang, X. Yu, Z. Ma, J. Zhang, KaaS: Key as a service over quantum key distribution integrated optical networks, IEEE Commun. Mag. 57 (5) (2019) 152–159.

[16] C. Yang, H. Zhang, J. Su, Quantum key distribution network: Optimal secret-key-aware routing method for trust relaying, China Commun. 15 (2) (2018) 33–45.
[17] R. Alléaume, F. Roueff, E. Diamanti, N. Lütkenhaus, Topological optimization of quantum key distribution networks, New J. Phys. 11 (7) (2009) 075002.
[18] K. Azuma, K. Tamaki, H.-K. Lo, All-photonic quantum repeaters, Nature Commun. 6 (1) (2015) 6787.
[19] Y. Cao, Y. Zhao, J. Wang, X. Yu, Z. Ma, J. Zhang, Cost-efficient quantum key distribution (QKD) over WDM networks, J. Opt. Commun. Netw. 11 (6) (2019) 285–298.
[20] Q. Zhang, O. Ayoub, A. Gatto, J. Wu, F. Musumeci, M. Tornatore, Routing, channel, key-rate, and time-slot assignment for QKD in optical networks, IEEE Trans. Netw. Serv. Manag. 21 (1) (2024) 148–160.
[21] I. Pedone, A. Atzeni, D. Canavese, A. Lioy, Toward a complete software stack to integrate quantum key distribution in a cloud environment, IEEE Access 9 (2021) 115270–115291.
[22] V. Lopez, A. Pastor, D. Lopez, A. Aguado, V. Martin, Applying QKD to improve next-generation network infrastructures, in: 2019 European Conference on Networks and Communications, EuCNC, 2019, pp. 283–288.
[23] Y. Cao, Y. Zhao, Q. Wang, J. Zhang, S.X. Ng, L. Hanzo, The evolution of quantum key distribution networks: On the road to the qinternet, IEEE Commun. Surv. Tutor. 24 (2) (2022) 839–894.
[24] V. Martin, A. Aguado, J.P. Brito, A.L. Sanz, P. Salas, D.R. Lopez, V. Lopez, A. Pastor-Perales, A. Poppe, M. Peev, Quantum aware SDN nodes in the madrid quantum network, in: ICTON'19.
[25] B. Waxman, Routing of multipoint connections, IEEE J. Sel. Areas Commun. 6 (9) (1988) 1617–1622.
[26] J. Wang, B.J. Rollick, Z. Jia, H. Zhang, B.A. Huberman, Time-interleaving enabled co-propagation of QKD and classical channels over 100-km fiber with 10-dbm classical launch power, 2023, arXiv:2304.13828 [quant-ph].
[27] F. Romero, G.I. Chaudhry, I. Goiri, P. Gopa, P. Batum, N.J. Yadwadkar, R. Fonseca, C. Kozyrakis, R. Bianchini, Faa$t: A transparent auto-scaling cache for serverless applications, ACM SoCC (2021).