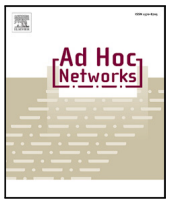
Ad Hoc Networks 172 (2025) 103803

Contents lists available at ScienceDirect

Ad Hoc Networks

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



Reliability and bandwidth aware routing in SDN-based fog computing for IoT applications

Parisa Valizadeh [1], Mohammad Hossein Yaghmaee [*], Yasser Sedaghat

Computer Engineering Department, Ferdowsi University of Mashhad (FUM), Mashhad, Iran

ARTICLE INFO

Keywords:
SDN
Bandwidth
Reliability
IoT applications
Fog computing

ABSTRACT

Software-Defined Networking (SDN) and fog computing are pivotal in supporting computationally intensive tasks within Internet of Things (IoT) applications, enhancing efficiency and reliability. However, many IoT applications are constrained by communication paths prone to link failures, necessitating robust fault tolerance techniques to ensure reliable traffic flow. In particular, real-time IoT applications demand stringent reliability and bandwidth requirements (constraints), which are challenging to meet simultaneously. Although previous research has investigated SDN-based routing to improve reliability, developing a routing algorithm that satisfies both reliability and bandwidth constraints remains an NP-hard problem. In this paper, we propose two novel routing algorithms: Reliability Aware Bandwidth constrained Routing (RABR) and Reliability and Bandwidth Constrained Routing (RBCR), specifically designed for SDN-enabled environments. Our approach prioritizes service reliability while meeting strict reliability and bandwidth criteria. The proposed solution integrates several phases, including reliability aware and bandwidth constrained path routing and flow duplication through parallel/hybrid and sequential routing methods. Furthermore, we introduce a greedy heuristic algorithm, implemented by the SDN controller with an efficient time complexity. Simulation results demonstrate that our algorithm surpasses state-of-the-art approaches in critical metrics such as reliability, reliability-bandwidth success rate, and Runtime. As such, our solution emerges as a robust choice for SDN-enabled IoT environments.

1. Introduction

The Internet of Things (IoT) has witnessed rapid growth, with billions of interconnected devices generating massive amounts of data. This surge in network traffic has led to congestion and decreased Quality of Service (QoS), particularly for real-time IoT applications that demand high reliability and low latency [1]. To mitigate these issues, fog computing has emerged as a powerful complement to IoT, offering enhanced computational capabilities, reduced latency, and more effective traffic management [2,3]. Real-time applications such as tactile internet and autonomous vehicles further demand ultra-reliable and low-latency communication [4], placing even greater strain on traditional network infrastructures, which often struggle with high latency and congestion.

Software-Defined Networking (SDN) architectures present a promising solution for managing these performance demands in highly dynamic environments [5]. However, one critical issue persists: the resilience of SDN-based networks to link failures, which occur more frequently than router or switch failures and significantly degrade performance [6]. In large-scale networks, link failures lead to frequent disruptions in IoT data flows, impacting the reliability of applications that require consistent and timely data transmission [7].

IoT devices generate massive amounts of data traffic, leading to severe network congestion, which can amplify the risk of failures. This growing problem has the potential to paralyze overall network activity. Moreover, Networks operating in harsh or resource-constrained conditions are particularly susceptible to frequent faults due to factors such as limited fault-tolerance mechanisms, high traffic loads, and physical infrastructure limitations in large-scale networks. Studies have reported that, in certain scenarios, links may fail as frequently as every thirty minutes [8,9], severely impacting the quality of supported services.

Hence, a fast recovery fault tolerance technique is essential for rerouting affected traffic flows swiftly and maintaining high reliability in real-time applications [10]. Current recovery approaches, such as reactive and proactive schemes, are either too slow or impractical for large-scale deployments [6]. Reactive methods involve recalculating paths post-failure, leading to delays, while proactive methods, though faster, are costly in terms of switch storage and computational power [11,12].

* Corresponding author.
E-mail addresses: valizadeh@mail.um.ac.ir (P. Valizadeh), hyaghmae@um.ac.ir (M.H. Yaghmaee), y_sedaghat@um.ac.ir (Y. Sedaghat).

https://doi.org/10.1016/j.adhoc.2025.103803
Received 1 October 2024; Received in revised form 4 February 2025; Accepted 17 February 2025
Available online 25 February 2025

Existing SDN-based fog computing traffic engineering solutions prioritize QoS parameters like delay and throughput but overlook link reliability, despite its necessity for applications requiring ≥99.999% ("five-nines") reliability, such as telemedicine and autonomous vehicles [13–15]. Current approaches falter in congested or failure-prone networks [16,17]. Reliability needs vary by application criticality, user expectations, and SLAs: mission-critical systems (e.g., healthcare) demand five-nines reliability, while non-critical applications tolerate 95%–99%. These thresholds align with standards like IEC 62304 (medical devices) [18], FAA regulations (aviation), and ITU-T recommendations, which balance technical feasibility and industry needs. For instance, the U.S. Department of Energy enforces 99.99% reliability for smart grid AMI networks [19].

In addition, existing methods [16,17,20], which concentrate exclusively on link reliability metrics, may fall short in scenarios where individual flows demand exceptionally high reliability. Such constraints may not be met by a single path, even if it is considered the most reliable. Consequently, these approaches frequently fail to meet the comprehensive reliability requirement of real-time IoT applications in SDN-based fog computing environments.

Flow duplication is an effective way to increase the reliability of flow, however, it has a few challenges. For example, finding suitable path sets to duplicate flow through them is challenging due to the calculation of the reliability of hybrid path sets with shared and it is inadequately addressed in prior research which is going to be addressed in this paper.

Considering bandwidth constraints alongside reliability is essential for IoT applications, as it addresses both resource limitations and the time-sensitive nature of these systems.

To address these critical gaps, we propose the Reliability and Bandwidth Constrained Routing (RBCR) algorithm and its extension, Reliability-Aware Bandwidth Constrained Routing (RABR). RBCR simultaneously considers both reliability and bandwidth constraints by employing parallel, hybrid, and sequential routing methods, thereby improving flow reliability and maintaining uninterrupted service for critical IoT applications. In contrast, RABR prioritizes reliable routing while meeting strict bandwidth constraints.

Although energy efficiency and time constraints are not explicitly addressed in this work, the proposed algorithms inherently contribute to both aspects through reliable path selection and efficient flow management. By prioritizing reliability, the approach minimizes link failures and retransmissions, reducing energy overhead and ensuring continuous data flow even in unreliable network conditions. Additionally, efficient bandwidth management guarantees sufficient resources for IoT traffic, further lowering energy consumption and mitigating delays caused by congestion or insufficient capacity. These optimizations collectively enhance network performance, reduce end-to-end latency, and support the timing requirements of time-sensitive IoT applications.

RBCR algorithm proactively distributes network flows along the most reliable paths, utilizing flow duplication to further enhance reliability in real-time (time-sensitive) IoT environments. Additionally, we introduce the Sum of Disjoint Products (SDP) method, a novel technique for precisely calculating reliability across hybrid path sets, which is a feature of RBCR.

The SDP technique is a robust and efficient approach for calculating system reliability, overcoming the limitations of traditional methods such as state enumeration, graph transformation, the Inclusion-Exclusion (I-E) principle, the decomposition method and binary decision diagrams [21]. SDP offers superior numerical stability and accuracy, making it particularly well-suited for medium to large-scale systems. By preventing overcounting and effectively managing overlapping paths and dependencies, SDP ensures precise reliability evaluation while maintaining computational efficiency.

Both RBCR and RABR are tailored for real-time IoT environments, such as smart cities and healthcare, where high reliability and low latency are critical. By mitigating link failure effects, our approach meets the stringent constraints of these applications, providing a comprehensive solution for maintaining network resilience and performance.

While both algorithms aim to optimize reliability and bandwidth, they differ significantly in their approach, computational complexity, and practical applicability. RABR selects the single most reliable path that meets the bandwidth constraints for a given flow. Its low computational complexity makes it well-suited for environments with moderate reliability requirements and limited computational resources, as it evaluates individual paths without considering path combinations or duplication. In contrast, RBCR employs a more advanced strategy that integrates both parallel/hybrid and sequential routing methods. By utilizing the SDP method, it calculates the combined reliability of multiple paths, ensuring stringent reliability and bandwidth requirements are met even in the presence of failures or congestion. However, this enhanced fault tolerance comes at the cost of higher computational complexity compared to RABR. Given its robustness, RBCR is designed for mission-critical IoT applications such as healthcare, smart grids, and autonomous systems, where reliability is paramount and failure is not an option.

To the best of our knowledge, this is the first work to address both reliability and bandwidth constraints in real-time IoT applications within SDN-based fog computing environments.

The key contributions of this paper are as follows:

- Two routing algorithms are proposed: (1) RBCS, which considers reliability and bandwidth constraints simultaneously, and (2) RABR, which is a reliability-aware bandwidth constraint.
- Integrating parallel/hybrid and sequential routing methods enhances flow reliability and ensures continuous service for real-time IoT applications.
- Novel application of the Sum of Disjoint Product (SDP) method for precise reliability calculations in hybrid path sets with shared links.
- Development of a time-efficient heuristic for path identification for the proposed algorithms that significantly reduce the time complexity.
- Comprehensive performance evaluation demonstrating the superior ability of the proposed algorithms to meet the stringent reliability and bandwidth constraints of real-time IoT applications.

The remainder of the paper is organized as follows: Section 2 reviews existing literature, distinguishing our work from previous research to highlight the novelty of our contributions. Section 3 presents the network model and precisely defines the problem we aim to address. The proposed RBCR and RABR algorithms are presented in Section 4. The experimental results and simulations are illustrated in Section 5. Finally, we conclude the paper by summarizing our findings and discussing potential avenues for future research in Section 6.

2. Related works

Network performance is often compromised by node and link failures. While various recovery methods have been proposed for link failures in SDN, the issue remains particularly challenging in SDN-based fog computing for IoT applications [22]. A key challenge in traffic engineering is effectively responding to underlying link or node failures.

Recovery methods generally fall into two categories: reactive and proactive approaches. Reactive approaches involve the controller installing backup paths for affected flows upon detecting a failure. For instance, Wang et al. [23] developed a two-stage SDN algorithm for rapid link failure recovery, meeting QoS standards but potentially incurring delays due to controller intervention. Conversely, proactive approaches deploy backup paths in advance to preempt link failures. Yang et al. [24] proposed a proactive approach for hybrid SDN networks, utilizing pre-configured IP tunnels to swiftly redirect traffic and

Table 1
Comparison of existing literature and our proposed approach.

Study	SDN	Reliability	Bandwidth constraint	Reliability constraint	Handling link failures	Flow duplication	FC
[7,22–25]	Yes	Recovery-based methods	No	No	Post-failure	No	No
[26]	Yes	High-risk links	No	No	Post-failure	No	No
[27]	Yes	Multipath	No	No	Post-failure	No	No
[28]	Yes	Link reliability-based backup paths	No	No	Post-failure	No	No
[29]	Yes	No	No	No	Congestion-focused	No	Yes
[30,31]	Yes	No	No	No	Delay-focused	No	Yes
[17,20]	Yes	Link reliability prediction (k-NN)	Yes	No	Proactive fault tolerance	No	Yes
[16]	Yes	Max-Min path reliability	Yes	No	Proactive fault tolerance	No	Yes
[32]	No	Product of link reliabilities	Yes	No	Proactive fault tolerance	No	No
Our work	**Yes**	**Product + SDP**	**Yes**	**Yes**	**Proactive fault tolerance**	**Yes (P/H + Seq)**	**Yes**

coordinate multiple backup paths. Similarly, P4Resilience, designed for SDN with P4 switches, employs packet header encapsulation to efficiently store backup path information and prevent loops during multi-link failures [7].

With the proliferation of IoT applications, there is an increased demand for reliable communication solutions that meet diverse constraints. Thorat et al. [25] addressed IoT network reliability in SDN by using VLAN-enabled flow labeling to aggregate flow rules, conserving switch memory in large-scale deployments. Similarly, in [22], an SDN-based system is proposed to maintain QoS during link failures in smart city networks by rerouting traffic through alternative paths. However, they did not consider the link failure state when computing alternative paths for disrupted flows.

Seddiqi et al. [26] proposed an SDN link failure management approach that preemptively deploys backup paths by identifying high-risk links. Another method, MPResiSDN [27], introduces a multipath routing scheme to ensure uninterrupted data transmission even during path failures. However, these approaches do not incorporate proactive parallel/hybrid or sequential flow duplication, nor do they consider both reliability and bandwidth constraints—key goals of this paper (see Table 1).

In IoT networks, link failures can severely degrade service performance, particularly for applications that generate high traffic volumes. Establishing reliable paths with low failure probabilities is crucial to managing such traffic efficiently. Raza et al. [28] proposed a method for calculating link reliability and installing minimal flow rules for backup paths based on the reliability of the primary path. Although this method improves network reliability by using multiple backup paths, it does not simultaneously consider both reliability and bandwidth, a gap that our paper addresses.

Most existing methods focus on rerouting affected flows post-failure without considering link reliability during initial path selection. For real-time IoT applications that require highly reliable communication, network operators must prioritize both reliability and bandwidth—an issue this paper directly tackles.

Recent studies have investigated the integration of SDN and fog computing to enhance performance in real-time IoT applications. In [29], a routing protocol is proposed to optimize data transmission in IoT networks by leveraging mobile edge computing. This protocol focuses on improving energy efficiency while mitigating network congestion in IoT environments. Similarly, [30,31]introduce the Fog Node Placement Problem (FNPP), which seeks to minimize latency between hosts and fog nodes. However, these approaches do not incorporate reliability or bandwidth constraints in their routing strategies, limiting their effectiveness in ensuring stable and high-performance data transmission. In [17,20], methods using the k-nearest neighbor algorithm were proposed to predict link reliability in SDN-enabled IoT-fog architectures. However, due to the dynamic nature of IoT applications, frequent link failures can still degrade network reliability. Addressing this issue requires adaptive solutions that respond to fluctuating network conditions, which is a core focus of our proposed work.

In [16], a reliable flow distribution method for SDN-enabled fog computing in smart cities was introduced, focusing on maximizing the minimum link reliability across all links. However, approaches that rely solely on the minimum link reliability, as seen in [16,17,20], often lack the precision needed in more complex networks. Alternatively, calculating path reliability as the product of individual link reliabilities, as suggested in [32], can yield more accurate reliability measures, particularly in diverse network topologies. This serial reliability approach, employed in our work, offers a more adaptable and comprehensive measure for dynamic environments.

A common limitation across most of these studies [16,17,20] is their failure to account for the specific reliability constraint of real-time IoT applications. Selecting paths based solely on link reliability metrics may fall short in scenarios where exceptionally high reliability is required for individual flows—constraints that may not be met by a single path, even if it is the most reliable. Our paper addresses this gap by considering both reliability and bandwidth in real-time IoT scenarios within SDN-based fog computing environments.

Moreover, calculating the reliability of hybrid path sets with shared links presents a significant challenge that has been inadequately addressed in previous research. To address this, we employ the Sum of Disjoint Product (SDP) method, which allows for more accurate reliability calculations in hybrid path sets—an additional contribution of this paper.

Our work further introduces a novel methodology that integrates parallel/hybrid and sequential routing with flow duplication, significantly improving network resilience. Therefore, we propose two algorithms: Reliability and Bandwidth-Constrained Routing (RBCR) and an extension of it, Reliability-Aware Bandwidth-Constrained Routing (RABR). These algorithms are designed to meet the stringent reliability and bandwidth constraints of real-time IoT applications, ensuring effective performance under diverse network conditions. By proactively addressing potential network disruptions and optimizing resource allocation, our approach provides a robust foundation for the next generation of IoT services in fog computing environments.

3. Network model and problem statement

In an IoT fog network, the fog servers continuously receive traffic from IoT end nodes. The SDN controller is responsible for choosing a reliable path set for a certain flow while considering the reliability and bandwidth constraint. Let $F = \{f_1, f_2, f_3, \ldots, f_n\}$ represent the set of flows used in the network. Each IoT flow $f_i \in F$ is represented by a triple $(\theta_i, \eta_i, \beta_i)$, where θ_i denotes the IoT device acting as the source of a specific flow, η_i denotes the fog server acting as the destination of the flow, and $\beta_i > 0$ represents the bandwidth constraint of the flow.

To facilitate the routing of these flows, the network topology is represented as a directed graph $G = (S, L)$, where S represents the collection of all SDN-based switches and fog servers and L represents the collection of links connecting them. Each link $e \in L$ has a capacity of $\zeta(e) > 0$ bytes/sec and a reliability value of $R_e(t)$ at time period t. The link reliability R_e is calculated for each link that is defined in Eq. (1), where λ is the failure rate of the link.

$$R_e(t) = \exp(-\lambda t) \tag{1}$$

The failure rate λ represents the probability that the link will fail over a specific time period. It is defined in Eq. (2) where LNF_e is the total number of link failures over the observation time period T.

$$\lambda = \frac{\text{LNF}_e}{T} \tag{2}$$

The SDN controller generates a list of paths for each flow $f_i \in F$ defined in Eq. (3). At each time t, the reliability of each path R_p in the network is defined in Eq. (4).

$$A_{f_i} = \{p^1_{f_i}, p^2_{f_i}, \ldots, p^n_{f_i}\} \tag{3}$$

$$R_p(t) = \prod_{e \in p} R_e(t) \tag{4}$$

The reliability constraint of network service f_i is indicated between a range of zero to one (i.e., $\Gamma_{f_i} \in [0, 1]$). This reliability constraint (Γ) represents the minimum acceptable level of reliability for the network service f_i.

3.1. Problem statement of the proposed algorithms

In this subsection, we present the problem statement of the proposed algorithm, which considers both reliability and bandwidth constraints. For each flow (f_i), we select a forwarding path set from all available paths (A_{f_i}) between the source–destination pair, taking into account the reliability (Γ_{f_i}) and bandwidth (β_i) constraints of the flow f_i.

We use path set reliability to evaluate the selected path set for each flow, ensuring it meets or exceeds a user-defined reliability constraint. Additionally, we also consider bandwidth constraints to guarantee that the paths can support the required flow capacity without causing network bottlenecks. The optimization problem is defined in Eqs. (5)–(9). Our objective, defined in Eq. (5), is to maximize the reliability value of flow f_i, subject to the constraints defined in Eqs. (6)–(9). The RABR algorithm focuses solely on bandwidth constraints (i.e., removing the reliability constraint in Eq. (7)).

$$\text{Maximize} \quad R_{f_i}(t), \quad \forall f_i \in F \tag{5}$$

$$\text{Subject to:} \quad \zeta(e) \geq 0, \quad \forall e \in L \tag{6}$$

$$R_{f_i}(t) = Y\left(\sum_{p \in A_{f_i}} X_{f_i}.R_p\right) \geq \Gamma_{f_i}, \quad \forall f_i \in F \tag{7}$$

$$\sum_{f_i \in F} \delta_{f_i}(e)\beta_i \leq \zeta(e), \quad \forall e \in L \tag{8}$$

$$\delta_{f_i}(e), X_{f_i} \in \{0, 1\}, \quad \forall e \in L, \forall p \in A_{f_i}, \forall f_i \in F \tag{9}$$

Eq. (6) ensures that the residual capacity of link e is non-negative. Eq. (7) guarantees that the flow reliability meets or exceeds the reliability threshold, as Y defines reliability based on the selected path set $X_{f_i}.R_p$ and its corresponding reliability. Eq. (8) ensures that the aggregate bandwidth used by all flows sharing a link does not exceed the link's capacity. Finally, Eq. (9) defines binary variables: $\delta_{f_i}(e)$, which indicates whether link e is selected to route flow f_i, and X_{f_i}, determining if path p is chosen for flow f_i.

Table 2 shows some of the frequently used notations in this paper.

4. Proposed algorithms

In this section, we elaborate on the proposed RBCR and RABR algorithms. First, we formulate the flow reliability constraint as an input constraint for the algorithm. Next, we incorporate bandwidth constraints to account for both reliability and bandwidth constraints simultaneously. To further reduce the time complexity of the proposed algorithm, we propose a greedy heuristic approach that allows the algorithm to operate in polynomial time. Fig. 1 illustrates the proposed system architecture based on SDN. In this architecture, IoT application requests are transmitted to the SDN controller through an SDN switch infrastructure. The RBCR and RABR modules, along with other controller modules, are responsible for determining the optimal path set that satisfies the users' constraints. Once the optimal path set is identified, it is applied to the SDN switches to route the traffic to its destination, which in this study is a fog server.

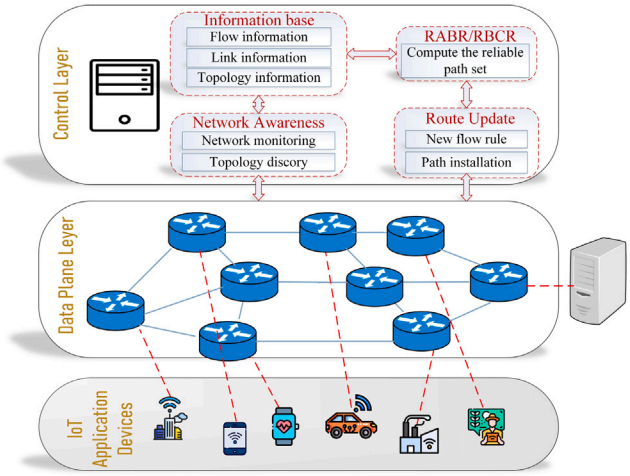


Fig. 1. The proposed system architecture.

Table 2
Frequently used notations.

Symbol	Description
f_i	Flow in the network
A_{f_i}	Set of paths for flow f_i
β_i	Bandwidth constraint of flow f_i
Γ_f	Reliability constraint of flow f_i
$\zeta(e)$	Bandwidth capacity of link e
LNF_e	Number of link failure
$R_e(t)$	Reliability of a link
$R_p(t)$	Reliability of a path
$R^p_{f_i}(t)$	Reliability in parallel flow duplication
$R^H_{f_i}(t)$	Reliability in hybrid flow duplication
$R^S_{f_i}(t)$	Reliability in sequential flow duplication
N_f	The number of sequential duplicated flows

4.1. Phase 1: Reliability aware and bandwidth constrained path routing

First, we address the challenges of ensuring reliable flow transmission through the network while meeting the specified reliability constraints. The proposed RBCR algorithm consists of three phases to achieve this: (1) reliability aware and bandwidth-constrained path routing (RABR algorithm), (2) duplication methods using parallel or hybrid routing, and (3) sequential routing with flow duplication. In the following subsections, we will discuss each phase of the proposed algorithm in detail. Fig. 2 provides an overview of the algorithm.

In this phase, the proposed algorithm involves the selection of the optimal path for each flow, considering the reliability of each link in the network. When the controller receives a request from the data plane to calculate a reliable path set for each flow f_i, it computes the reliability of candidate path sets C_{f_i} from the source device θ_i to the fog server η_i in a time t, using Eq. (4). Subsequently, the controller selects the most reliable path for the specific flow, as defined in Eq. (10). Later, in Section 4.5, we propose a pathfinder algorithm that returns the k-most reliable path sets that satisfy the bandwidth requirement to calculate candidate paths. The reason is to reduce the time complexity of the proposed algorithm.

$$R_{f_i}(t) = \max_{p \in C_{f_i}} \left(R_p(t)\right) \tag{10}$$

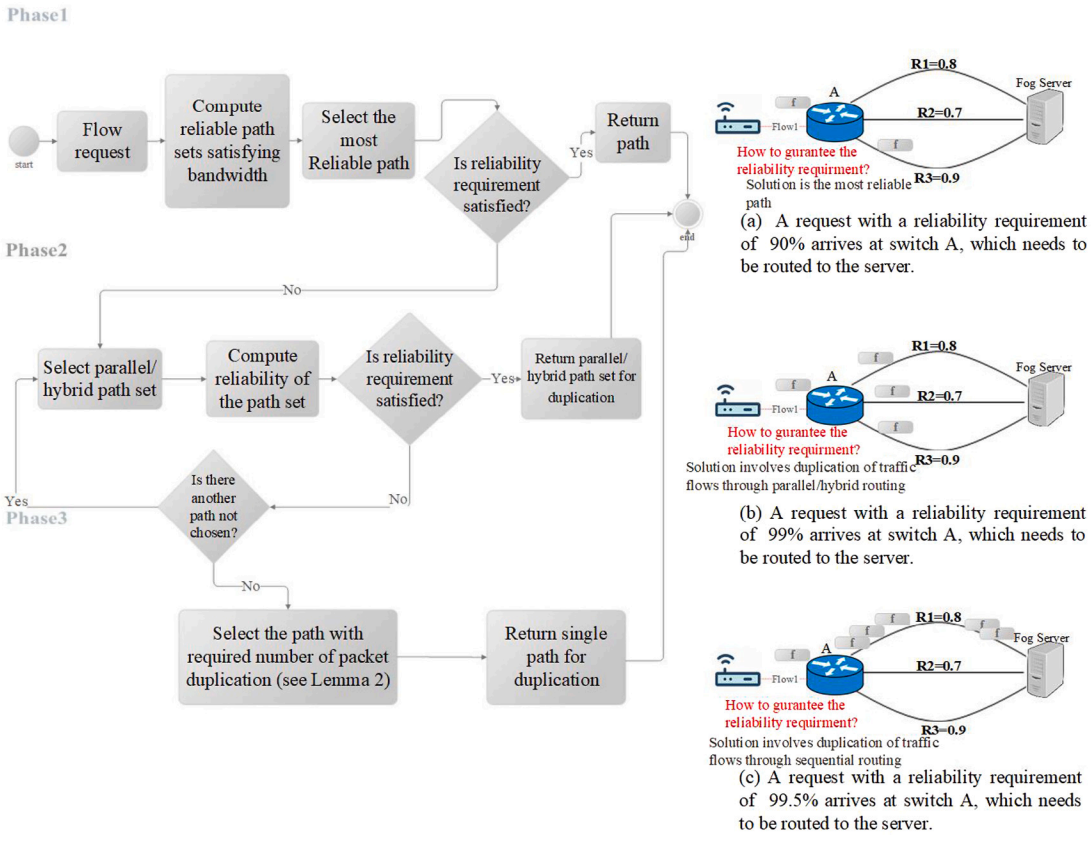


Fig. 2. Illustration of the proposed algorithm.

To ensure that the reliability constraint of flow f_i is met, it is essential that the reliability value of the optimal path is sufficient to meet the specified reliability constraint (i.e., $R_{f_i}(t) \geq \Gamma_{f_i}$). Otherwise, the proposed duplication methods presented in the following are adopted.

4.2. Phase 2: Parallel/hybrid routing with flow duplication

It is possible that the initially selected most reliable path for flow f_i does not meet the strict reliability constraint. In such cases, a set of additional paths is selected to duplicate the flow, aiming to fulfill the reliability constraint. To select parallel or hybrid paths for flow f_i, the algorithm first sorts all candidate paths in descending order of their reliability values, excluding the most reliable path (selected in phase 1). Starting with the path of lowest reliability, each path is incrementally checked to the parallel or hybrid path set. This approach helps to distribute network traffic across multiple paths, reducing the likelihood of overloading the more reliable ones.

For each path, the algorithm recalculates the combined reliability of the path set, including the most reliable path from phase 1. If the reliability of the path set meets the required threshold, that path is chosen for duplicating the flow. If not, the algorithm proceeds to the next path in the sorted list.

If none of the paths meet the reliability constraint during the initial round of selection, indicating that the stricter reliability criterion is not met, the algorithm adds the second most reliable path from the list to the path set. If the reliability constraint is still not satisfied, the algorithm proceeds to add a third path, and so forth. This iterative process continues until a path set is found that meets the reliability constraint or until the maximum number of rounds — equal to the number of candidate paths between the source and destination — is reached. It is important to note that this phase is only applicable if there is more than one path from the source to the destination.

4.2.1. Proposed parallel/hybrid path set reliability calculation

Once the parallel/hybrid path set has been identified, the controller duplicates the flow on the selected path set. Therefore, we consider two cases to determine the reliability of the parallel/hybrid paths. In the first case, the proposed method uses parallel routing, where all paths in the path set are link-disjoint. We create a set of Y paths from all candidate paths between the source–destination pair $Y(f_i) \subseteq C_{f_i}, Y \neq \emptyset$ for each flow f_i. The reliability of the specific flow f_i in parallel flow duplication routing in the link-disjoint scenario is denoted as $R^P_{f_i}(t)$ and is defined in Eq. (11).

$$R^P_{f_i}(t) = 1 - \prod_{p \in Y} (1 - R_p(t)) \tag{11}$$

For paths sharing links or nodes, dependencies arise, making Eq. (11) inaccurate. To address this, we propose using the SDP method, which accounts for overlapping links and ensures accurate reliability calculations in networks with shared links [33].

The second scenario involves hybrid routing, where some paths share one or more links. In this case, the reliability of the paths in the set is affected by the reliability of the shared links, making the calculation more complex. Failures in the shared links can impact the reliability of multiple paths in the set, increasing the probability of network failure. For this reason, the I-E principle, the SDP method and the decomposition method are some of the current precise approaches for addressing connection reliability. Because of its straightforward premise and ease of computation, the SDP technique has been widely employed among them to calculate system resiliency, especially in network reliability analysis in real-world networks where the number of paths Y is restricted [34].

The I-E Principle is a key technique in probability and reliability theory, used to calculate the probability of system success by accounting for overlapping probabilities. While effective for simple systems, it is not practical for complex networks due to the need to calculate

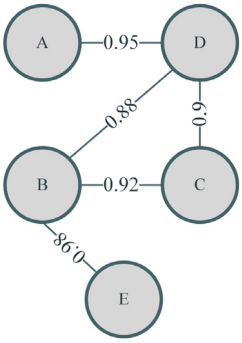


Fig. 3. Network topology with the link-shared scenario (known as hybrid routing).

$(2^Y - 1)$ terms for Y paths. In contrast, SDP requires only Y terms, making it more efficient for complex systems. Consequently, SDP is often preferred for accurately calculating reliability in intricate network configurations [35].

The SDP method expresses path set reliability as a sum of mutually exclusive events. SDP is defined in Eq. (12), where $\bar{p}_i$ denotes the complement (failure) of path p_i, i.e., path p_i fails.

$$R^H_{f_i}(t) = R_{p_1}(t) + R_{(p_2 \cap \bar{p}_1)}(t) + R_{(p_3 \cap \bar{p}_2 \cap \bar{p}_1)}(t) + \cdots + R_{(p_Y \cap \bar{p}_{Y-1} \cap \ldots \cap \bar{p}_1)}(t) \tag{12}$$

To illustrate the application of the SDP method, we present an example to demonstrate the path set's reliability, in which certain paths can contain at least one shared link. Suppose for a flow $f = (A \to E)$, the links (A, D) and (B, E) are shared links in two paths p_1 and p_2, as shown in Fig. 3.

In the following, we illustrate the steps of calculation of reliability of p_1 and p_2 with shared links.

Step 1: Calculate $R(p_1)$

For path p_1, which traverses $A \to D \to C \to B \to E$, the reliability is given by:

$$R(p_1) = R_{AD} \cdot R_{DC} \cdot R_{CB} \cdot R_{BE}$$

Substituting the values:

$$R(p_1) = 0.95 \times 0.90 \times 0.92 \times 0.98 = 0.7688$$

Step 2: Calculate $R(p_2 \cap \bar{p}_1)$

For path p_2, which shares some links with p_1 and traverses $A \to D \to B \to E$, considering the failure of p_1, the reliability is:

$$R(p_2 \cap \bar{p}_1) = R_{AD} \cdot R_{DB} \cdot R_{BE} \cdot (1 - R_{DC} \cdot R_{CB})$$

Substituting the values:

$$R(p_2 \cap \bar{p}_1) = 0.95 \times 0.88 \times 0.98 \times (1 - 0.90 \times 0.92) = 0.1441$$

Step 3: Calculate total reliability $R^H_f(t)$

Combining paths p_1 and p_2, the total reliability is:

$$R^H_f(t) = R(p_1) + R(p_2 \cap \bar{p}_1)$$

Substituting the values:

$$R^H_f(t) = 0.7688 + 0.1441 = 0.9129$$

Thus, the total reliability of the flow f is 0.9129, indicating a 91.29% probability of successful operation. The effects of link failures on time-critical flows can be minimized by optimizing the reliability of the paths and shared links.

Now, we provide proof that the parallel path increases the overall reliability of the flows.

Table 3
Time complexity comparison for Y paths and m links. Theoretical worst-case vs. best-case.

Method	Worst-case	Best-case
SDP	$O(2^Y \cdot m)$	$O(Y \cdot (m - s))$
I-E	$O(2^Y \cdot m)$	$O(2^Y \cdot m)$
Decomposition	$O(2^m)$	$O(2^m)$

Lemma 1. *Increasing the number of parallel paths enhances network reliability.*

Proof. one with j parallel paths and another with $j+1$ parallel paths. Let R_j and R_{j+1} denote the reliability of these networks, respectively. Define P'_{j+1} as the probability that the $(j+1)$th path is functional.

The reliability of a network with j parallel paths is given Eq. (11). So for a network with $j+1$ parallel paths, the reliability R_{j+1} is defined in Eq. (13).

$$R_{j+1} = 1 - \prod_{i=1}^{j+1}(1 - R_i) \tag{13}$$

We can separate the $(j+1)$th path from the product:

$$R_{j+1} = 1 - \prod_{i=1}^{j}(1 - R_i)(1 - P'_{j+1})$$

The difference in reliability between R_{j+1} and R_j is:

$$R_{j+1} - R_j = \prod_{i=1}^{j}(1 - R_i)\left[P'_{j+1}\right]$$

Thus, the final expression for the difference in reliability is defined in Eq. (14) which validating the additional reliability introduced by a new parallel path.

$$R_{j+1} - R_j = (1 - R_1)(1 - R_2)\ldots(1 - R_j)P'_{j+1} \tag{14}$$

Since $0 < P'_{j+1} < 1$, $R_{j+1} > R_j$. Thus, increasing the number of parallel paths improves network reliability. Using multiple paths and duplication methods boosts network resilience, reduces packet loss, and minimizes serialization delays. However, if the strict reliability constraints are still not met by the parallel duplication method, sequential routing with packet duplication can be employed to further enhance the overall reliability of the network.

4.2.2. Time complexity comparison of reliability calculation methods

The SDP method is often recognized as superior than the I-E principle and the Decomposition method for reliability analysis in real-world networks. The subject of this paper discusses the theoretical and practical benefits of SDP, focusing on its efficiency and scalability in modern network designs such as SDNs. Table 3 compares the time complexity of Y pathways and m links relevant to paths under both worst-case and best-case scenarios.

For Y paths sharing s links, SDP's complexity reduces to $O(Y \cdot (m-s))$, demonstrating polynomial scaling with respect to Y. This contrasts with I-E, which remains $O(2^Y \cdot m)$ regardless of shared links [36].

Furthermore, in extremely dense and large-scale networks, the computational overhead of the SDP method can become a bottleneck. To mitigate this, we propose complementary strategies like Hierarchical SDP in which by modularizing the network into z clusters (e.g., subnetworks or administrative domains) [37], we reduce the complexity from $O(m + z \log m)$. This approach exploits locality in fog/SDN architectures, where clusters operate semi-independently, and reliability calculations are parallelized across clusters.

To ensure computational efficiency while maintaining accuracy, we incorporate the following optimization techniques:

- Pruning Techniques: Filter out infeasible or unreliable paths early in the process, reducing unnecessary computations.
- Iterative Hybrid Path Selection: Evaluate paths in descending order of reliability, terminating the selection process as soon as the required reliability threshold is met.

These strategies enable our approach to balance accuracy with computational feasibility, making it well-suited for dynamic and resource-constrained networking environments.

4.3. Phase 3: Sequential routing with flow duplication

Sequential routing transmits flow along a single path at a time, which is effective for simple topologies with a limited number of paths. While this method does not offer the fault tolerance of parallel routing, it can still be beneficial for maintaining performance. In sequential routing, flows are duplicated and sent through the same path, preferably the one with the highest bandwidth. This approach enhances reliability by ensuring packets are not lost.

To implement sequential routing, an SDN switch sends an initial flow along with N_f duplicate flows to increase the probability of successful delivery. Ideally, all duplicate flows should reach the destination. If all flows fail, the probability of this occurring can be calculated using Eq. (15).

$$F^S_{f_i}(t) = (1 - R_p(t))^{N_f} \tag{15}$$

Assuming failures occur independently of each other and the flows, the probability of at least one flow's successful delivery can be computed in Eq. (16).

$$R^S_{f_i}(t) = 1 - (1 - R_p(t))^{N_f} \tag{16}$$

Lemma 2. *Sequential routing can enhance reliability and meet the reliability constraint when* $R_{parallel/hybrid} < \Gamma$.

Proof. Consider sending N_f duplicate flow over a single path $R_p(t)$. The reliability of this duplication can be expressed in Eq. (17).

$$R_{\text{serial}} = 1 - (1 - R_p(t))^{N_f} \tag{17}$$

To ensure that the reliability constraint of the flow is satisfied, we need to have $R_{\text{serial}} \geq \Gamma_{f_i}$.

To achieve this, we can select an appropriate value of N_f such that Eq. (18), which can be rewritten as Eq. (19).

$$(1 - R_p(t))^{N_f} \leq (1 - \Gamma_{f_i}) \tag{18}$$

$$R_p(t)^{N_f} \geq \Gamma_{f_i} \tag{19}$$

Therefore, we can calculate the minimum value of N_f required to satisfy the reliability constraint in Eq. (20).

$$N_f = \left\lceil \log \Gamma_{f_i} / \log R_p(t) \right\rceil \tag{20}$$

4.4. Heuristic algorithm

In this section, we present our proposed heuristic algorithms to address the aforementioned NP-hard problem by determining the optimal reliable path set for flow f_i, which satisfies the objective function in Eq. (5) and adheres to the constraints in Eqs. (6)–(9). To achieve this, we introduce a time-efficient heuristic algorithm (Algorithm 1) that efficiently computes a reliable routing path set while ensuring that both the reliability and bandwidth constraints of the flows are met. The RBCR is a greedy heuristic that utilizes the K-Most Reliable Paths with Bandwidth Constraints (KMRPB) algorithm (Algorithm 2). KBRPB returns the k-most reliable paths between the source and the destination. Pathfinder algorithms typically use the BFS algorithm to find all available paths between the source and the destination. However, we limit our search space to only paths that meet our objectives and constraints, which significantly reduces computational complexity. Without this optimization, considering all possible paths would result in prohibitive time complexity, as the problem would scale exponentially.

Algorithm 1: Reliability and Bandwidth Constrained Routing (RBCR)

Input: f_i: flow, Γ_{f_i}: reliability constraint, β_i: bandwidth constraint, K: max candidate paths , θ_i: source, η_i: destination
Output: Reliable path set

1 $C_{f_i} \leftarrow$ KMRPB(θ_i,η_i,k,β_i)
2 **if** $|C_{f_i}| == 0$ **then**
3 | **return** $\emptyset$
4 **end**
5 $p^m_{f_i} \leftarrow C_{f_i}[0]$ // Phase 1
6 **if** $R_{p^m_{f_i}} \geq \Gamma_{f_i}$ **then**
7 | **return** $p^m_{f_i}$
8 **end**
9 $PC_{f_i} \leftarrow C_{f_i} \setminus \{p^m_{f_i}\}$ // Phase 2
10 $PHP \leftarrow \{p^m_{f_i}\}$ // PHP:Parallel/hybrid path set
11 **for** $r \leftarrow 1$ **to** $k - 1$ **do**
12 | **foreach** $p^\mu_{f_i} \in PC_{f_i}$, *in reverse order* **do**
13 | | **if** $(PHP \cup \{p^\mu_{f_i}\}) \geq \Gamma_{f_i}$ **then**
14 | | | **return** $PHP \cup \{p^\mu_{f_i}\}$
15 | | **end**
16 | **end**
17 | $PHP \leftarrow PHP \cup \{PC_{f_i}[k-r]\}$
18 | $PC_{f_i} \leftarrow PC_{f_i} \setminus \{PC_{f_i}[k-r]\}$
19 **end**
20 $p_{f_i} \leftarrow$ Get the highest bandwidth // Phase 3
21 $N_f \leftarrow \left\lceil \log \Gamma_{f_i} / \log R_{p_{f_i}} \right\rceil$, Eq. (20)
22 **if** $(p_{f_i})^{N_f} \geq \Gamma_{f_i}$ **then**
23 | **return** $(p_{f_i})^{N_f}$
24 **end**

The RBCR algorithm builds on the strategies outlined in Lemmas 1 and 2 to ensure the required reliability threshold is met. Upon receiving flow f_i as input, the algorithm first computes C_{f_i} using the KMRPB algorithm, which identifies the top K most reliable paths that meet the bandwidth constraints for each flow $f_i \in F$. The KMRPB algorithm optimizes for both reliability and bandwidth, returning the set of k paths that best satisfy these criteria, thereby reducing computational complexity by limiting the search space to only the most viable paths. C_{f_i} is sorted in descending order of reliability.

If no path satisfies the bandwidth constraint, the algorithm returns an empty set (Line 3), signaling that no feasible solution exists and prompting a reassessment of network parameters.

The algorithm proceeds by selecting the most reliable path from C_{f_i} that satisfies both the reliability and bandwidth constraints (Line 5). If this path meets the constraints, it is returned as the reliable path set (Lines 5–8). If not, the algorithm moves to Phase 2, where it attempts to enhance reliability by constructing a set of parallel/hybrid paths, as outlined in Lemma 1 (Section 4.2). Notably, the RABR algorithm

encompasses these lines (5–8), while this and the subsequent phases specifically pertain to the RBCR algorithm.

In Phase 2, RBCR iteratively explores the candidate paths from C_{f_i}. It calculates the overall reliability of each path set for parallel duplication, beginning with the least reliable path from the set of parallel candidate paths (PC) (Line 9). If the combined reliability meets the required threshold, the path is chosen for flow duplication and added to the Parallel/Hybrid Paths (PHP) set (Line 14). Otherwise, the next path is evaluated. If no path satisfies the reliability constraint (Lines 12–15), the algorithm adds the most reliable path from PC (Line 17) and removes it from the set (Line 18). This process continues until the reliability threshold is met or all paths have been evaluated. This phase is only applicable when multiple paths exist between the source and destination.

If parallel routing fails to meet the reliability threshold in Phase 2, the algorithm adapts by moving to sequential routing with packet duplication, as outlined in Lemma 2 (Lines 20–24) in Phase 3.

To optimize network performance, the algorithm selects the highest bandwidth path (Line 20), ensuring that the most capable paths are prioritized. It then calculates the required number of sequential duplicate packets (N_f) using Eq. (20) (Line 21) to further optimize resource usage. Ultimately, the algorithm identifies and returns the path that satisfies both the bandwidth and reliability constraints, providing an optimal solution for reliable data flow through calculated duplication strategies.

4.5. K-most reliable paths with bandwidth constraints algorithm

We propose the K-Most Reliable Paths with Bandwidth Constraints (KMRPB) Algorithm, inspired by Yen's K-Shortest Path Algorithm. This algorithm computes the K most reliable paths between a source node (IoT device) θ_i and destination node (Fog server) η_i in a network while ensuring each path meets a specified bandwidth constraint β_i.

The KMRPB algorithm (Algorithm 2) operates as follows:

1. Initialize an empty set A for storing the K most reliable paths and a priority queue B for managing candidate paths (Line 1).
2. Compute the initial most reliable path satisfying the bandwidth constraint using a modified Dijkstra's algorithm (Lines 2–6).
3. While $|A| < K$ and B is not empty, iteratively generate candidate paths (Line 8):
   (a) Extract the most reliable path $S_{current}$ from B (Line 9).
   (b) For each node s_{dev} in $S_{current}$ (except the destination), create a deviation (Line 10):
      i. Remove links (edges) of the root path R_{root} from the graph. (Line 13)
      ii. Find a new most reliable path S_{dev} from s_{dev} to η_i destination (Line 14).
      iii. If S_{dev} exists, combine it with R_{root} to form S_{total} (Line 15).
      iv. If S_{total} meets the bandwidth constraint and is not in A, add it to B and potentially to A (Lines 17–22).
4. Return the set A of K most reliable paths (line 26).

The algorithm efficiently narrows the solution space by focusing only on paths that are feasible under the given bandwidth and reliability constraints. This approach is particularly valuable in network applications where both reliability and bandwidth constraints are critical factors in path selection, offering network operators flexibility in routing decisions and enhancing overall network performance and resilience.

Algorithm 2: KMRPB Algorithm

Input: $G = (S, L)$, θ_i: source, η_i: destination, K, β_i:bandwidth constraint
Output: Set of K most reliable paths from θ_i to η_i with bandwidth $\geq \beta_i$

```
1  A ← ∅; B ← PriorityQueue() ;
2  S_initial ← MostReliablePath(G, θ_i, η_i, β_i) ;
3  if S_initial exists then
4  |  A ← A ∪ {S_initial};
5  |  B.push(S_initial);
6  end
7  G_original ← G;
8  while |A| < K and not B.isEmpty() do
9  |  S_current ← B.pop() ;
10 |  for s_dev ∈ S_current except η_i: destination do
11 |  |  R_root ← SubPath(S_current, θ_i, s_dev) ;
12 |  |  G ← G_original ;
13 |  |  RemoveLinks(G, R_root) ;
14 |  |  S_dev ← MostReliablePath(G, s_dev, η_i, β_i) ;
15 |  |  if S_dev exists then
16 |  |  |  S_total ← R_root + S_dev;
17 |  |  |  if PathBandwidth(S_total) ≥ β_i and S_total ∉ A then
18 |  |  |  |  B.push(S_total);
19 |  |  |  |  if |A| < K then
20 |  |  |  |  |  A ← A ∪ {S_total};
21 |  |  |  |  end
22 |  |  |  end
23 |  |  end
24 |  end
25 end
26 return A
```

4.6. Time complexity analysis of the algorithms

The RBCR algorithm utilizes the KMRPB algorithm as a subroutine. To analyze the combined time complexity, we will examine each algorithm individually.

The time complexity of KMRPB which is based on Yen's algorithm is $O(K|S|(|L|+|S|\log|S|+K|S|))$ where K is the number of paths, $|S|$ is the number of nodes, and $|L|$ is the number of links in the graph. The main complexity of RABR is the KMRPB complexity.

The RBCR algorithm consists of three phases. The initial pathfinding which is dominated by the KMRPB has the same time complexity. Then, the time complexity of the Path Combination phase is as follows:

- The nested loops iterate up to K times each, resulting in a complexity of $O(K^2)$.
- For link-disjoint paths, path combination and reliability checking are assumed to have a time complexity of $O(1)$.
- For non-link-disjoint paths (where paths share links), path combination and reliability checking have a time complexity of $O(Y \cdot (m-s))$.

Flow duplication performs a constant number of operations $O(1)$.

The overall time complexity of RBCR is dominated by the KMRPB call in the first phase. The subsequent phases add relatively minor computational overhead. Therefore, the combined time complexity is expressed as follows:

1. For link-disjoint paths:
$$O(K|S|(|L|+|S|\log|S|+K|S|))+O(k^2)$$
2. For non-link-disjoint paths:
$$O(K|S|(|L|+|S|\log|S|+K|S|))+O(k^2\cdot Y\cdot(m-s))$$

Here, the first term corresponds to Phase 1, which is dominated by the KMRPB algorithm. The second term reflects the complexity of the Path Combination phase, where the computation becomes more intensive when paths share common links.

It is important to note that, the RBCR algorithm may terminate early if a suitable path or combination of paths is found, potentially reducing the actual runtime. The value of K in RBCR is typically small and fixed, which can be considered a constant factor in practice. The KMRPB algorithm is called only once, regardless of the subsequent phases in RBCR.

In practice, the efficiency of this combined approach depends heavily on the network topology, the reliability constraints, and the bandwidth constraints. While the worst-case complexity appears high, for many real-world networks with well-behaved structures, the actual performance may be significantly better.

Space complexity remains $O(K|S|)$, primarily for storing the K paths returned by KMRPB.

5. Performance evaluation

We used Mininet, a network emulator, and the Floodlight SDN controller to evaluate the proposed algorithm's performance [38]. To simulate flow requests in the network, we utilized the Distributed Internet Traffic Generator (D-ITG) and Iperf tool [39]. To ensure that the traffic generated was representative of real-world traffic patterns, packets were generated using a uniform distribution method. The OpenFlow 1.3.1 protocol was employed in OpenFlow switches to direct traffic flows, establish several paths, and manage bandwidth via its meter function. We calculate the reliability of each path set based on long-term measurements of the operational and failure times of each network link. These reliability values are regularly updated with the most recent measurements to ensure accuracy. This dynamic approach guarantees the practical reliability and effectiveness of the proposed algorithm.

5.1. Setup and evaluation metrics

The simulation experiments were designed to evaluate the performance of the proposed algorithm under various network conditions, such as various reliability and bandwidth constraints, network connectivity, and fault rate. We conducted the tests on a Windows 11 PC equipped with an Intel Core i7 3630 processor and 8 GB of RAM.

5.1.1. Real-network evaluation

To rigorously evaluate the proposed algorithms, we employed a real network topology consisting of 38 switches and various network connectivity (150 to 740 links) and similar flow generators like these studies [17,20]. This topology was simulated to represent a hybrid SDN-based Fog Computing (FC) environment, incorporating 50 IoT devices, 5 fog servers, and a cloud server. Additionally, we randomly select various numbers of source and destination pairs (ranging from 50 to 200) within the network. For each IoT flow, the bandwidth demand was generated randomly from various ranges [1–25, 25–50, 50–75, 75–100] Mbps to simulate diverse traffic loads [16]. Each flow is transmitted from a different IoT device to a different fog server. In addition, we have considered different failure rates from $[0.0001, 0.005]$ based on the [40,41]. The capacity of each link in the topology was standardized to 100–200 Mbps to reflect realistic network conditions [16,17,20]. Fog servers are randomly selected in random and distant positions from each other.

Reliability requirements vary across applications, ranging from ≥99.999% for mission-critical systems to below 99% for non-critical systems [19,42]. These standards form the basis of our evaluation.

We conduct a comparative analysis of the proposed RBCR and RABR algorithms and RAFDA, a state-of-the-art Reliability aware algorithm [16] under comparable network conditions.

The following are the metrics evaluated under these experiments.

5.1.2. Reliability and bandwidth success rate of the flows

One of the metrics to evaluate the performance of flows is the Success Rate (SR), which considers both reliability and bandwidth constraints. We define a flow as satisfying the reliability constraint if its reliability surpasses a given threshold Γ (e.g., $\Gamma = 0.99$, corresponding to 99% reliability). Additionally, a flow satisfies the bandwidth constraint if the available bandwidth of its path meets or exceeds the flow's requested bandwidth. The Success Rate (SR) is defined as the proportion of flows that meet both the reliability and bandwidth constraints relative to the total number of flows (N_{flows}) in the network. This is expressed mathematically in Eq. (1):

$$SR = \frac{1}{N_{\text{flows}}} \sum_{i=1}^{N_{\text{flows}}} \mathbb{I}(R_{f_i} \geq \Gamma) \cdot \mathbb{I}(\zeta \geq \beta_i) \tag{21}$$

where $\mathbb{I}(\cdot)$ is the indicator function, equaling 1 when the condition is true and 0 otherwise. This formulation provides a comprehensive measure of network performance, accounting for both the reliability and bandwidth constraints of each flow.

5.1.3. Average reliability

The average reliability of all flows is computed by taking the mean reliability across all network flows, as shown in Eq. (22). In this equation, the reliability for each flow is calculated using the corresponding reliability equations.

$$\text{Average Reliability} = \frac{1}{N_{\text{flows}}} \sum_{i=1}^{N_{\text{flows}}} R_{f_i} \tag{22}$$

5.1.4. Flow duplication rate

We define Flow Duplication Rate (FDR) as the ratio of additional data transmitted to ensure reliability to the total data transmitted, as shown in Eq. (23). In this equation, D_{extra} represents the additional data transmitted for reliability (e.g., redundant packets), while D_{total} is the total amount of data transmitted, including this extra data.

$$FDR = \frac{D_{\text{extra}}}{D_{\text{total}}} \tag{23}$$

By evaluating the proposed algorithm with these metrics, we can assess its effectiveness in enhancing the reliability and resiliency of SDN-based fog computing for IoT applications, as well as overall network performance.

5.2. Results and discussion

This section presents the results of different experiments under various conditions. To assess the performance of the proposed algorithm, we compared it against the previously mentioned algorithms using several performance metrics, including the success rate, average reliability, and flow duplication rate. To ensure the robustness of our results, each parameter was computed by averaging the outcomes from 20 independent experiments for each iteration.

Furthermore, studies were conducted using various configurations. For the baseline parameters, we used a reliability constraint of 0.999, network connectivity of 0.5, and a bandwidth constraint range of [1–25].

5.2.1. Algorithms runtime of the evaluated algorithms

In this section, we conducted experiments to evaluate the running time of the proposed algorithms in comparison to the state-of-the-art, to showcase their efficiency. Fig. 4 illustrates the running times of our proposed algorithm, with various extensions, alongside the state-of-the-art, in a network with 0.5 link connectivity, 38 nodes, and 100 flow transmissions. A key contribution of this work is the development of a runtime-efficient routing algorithm integrated into the proposed RBCR approach.

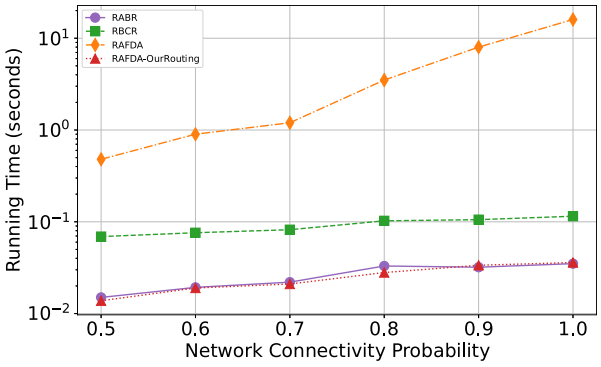


Fig. 4. Algorithms running time comparison.

Based on the figure, our proposed algorithm outperforms the RAFDA algorithm in terms of running time (fast runtime). We also integrated our routing algorithm KRMPB with the RAFDA algorithm to demonstrate its effectiveness in reducing the time complexity of the algorithms.

RAFDA's routing algorithm identifies all possible paths and filters them based on reliability and bandwidth constraints, but this increases its computational complexity, especially in highly connected or large networks. Calculating all possible paths between source and destination before selecting the top k paths can lead to spending time on paths that will later be discarded due to failing bandwidth constraints and low reliability, adding to its overall runtime. As a result, RAFDA's running time increases exponentially as the network size grows.

The proposed routing algorithm KMRPB, based on Yen's algorithm, identifies the most reliable and bandwidth-satisfying paths, making it more time-efficient. Unlike RAFDA, RBCR and RABR focus on paths that satisfy key constraints early in the process and prune invalid paths to save computation time.

The algorithms RAFDA with our routing and RABR show better runtime performance. RBCR includes duplication methods for reliability, while RABR selects the most reliable paths from the top k paths. The RAFDA algorithm has the slowest running time, mainly due to its non-optimized routing algorithm.

5.2.2. Reliability and bandwidth success rate of the flows

In this section, we evaluate the reliability and bandwidth success rates of the flows (SR) under various configurations, including reliability constraints (Γ_{f_i}), network connectivity, and bandwidth constraints of the flows.

Fig. 5 illustrates the SR performance of the algorithms under different reliability constraints with Network Connectivity of 0.5 with K-Path: 5, bandwidth: 1–25. The results indicate that RBCR achieves the highest SR due to its duplication methods that fulfill reliability constraints. RABR attains a higher SR compared to RAFDA because it selects the most reliable and bandwidth-satisfied paths. Conversely, RAFDA initially selects all possible paths and then filters the k-most reliable paths. Among these k paths, it subsequently selects the path that satisfies the bandwidth constraints. Consequently, none of these k-most reliable paths may meet the bandwidth constraints. In contrast, our proposed RBCR simultaneously identifies the k-most reliable and bandwidth-satisfying paths. Additionally, RAFDA's strategy of selecting paths based on the maximum of the minimum link reliability may result in paths being rejected more frequently when the reliability constraint is stringent, leading to a lower SR.

Fig. 6 shows that RBCR consistently achieves the highest SR across varying levels of network connectivity. As connectivity increases, the number of paths between the source and destination also increases, providing potential solutions for flow duplications to meet the reliability constraints of the flows. Consequently, RBCR achieved 100% SR

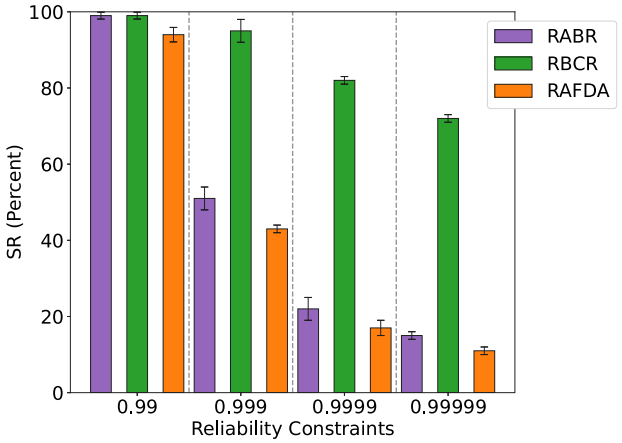


Fig. 5. Success rate in different reliability constraints of the flows.

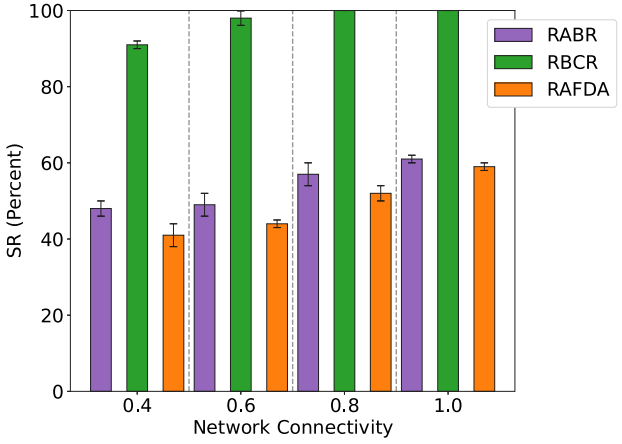


Fig. 6. Success rate in different network connectivity.

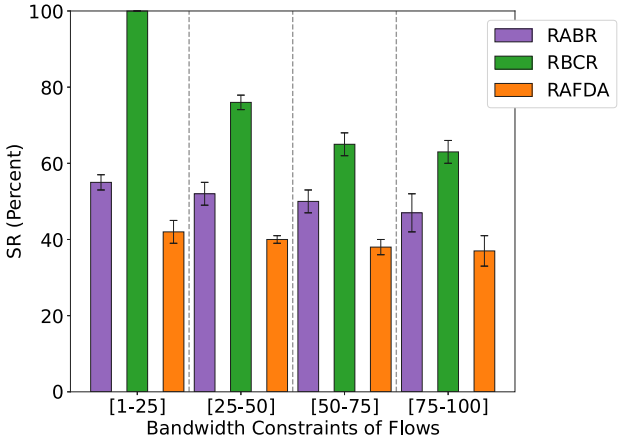


Fig. 7. Success rate in different bandwidth range constraint of the flows.

satisfaction in network connectivity of (≥0.6). RABR achieves a moderate SR, positioned between RAFDA and RBCR. RAFDA exhibits the lowest SR due to the aforementioned reasons. As network connectivity increases, the SR of all algorithms improves because the likelihood of finding more bandwidth-reliable paths increases.

Fig. 7 demonstrates the RSF with Network Connectivity of 0.5 with K-Path:5 in different bandwidths. That RBCR leads the highest SR across various flow bandwidth constraints. However, as the bandwidth

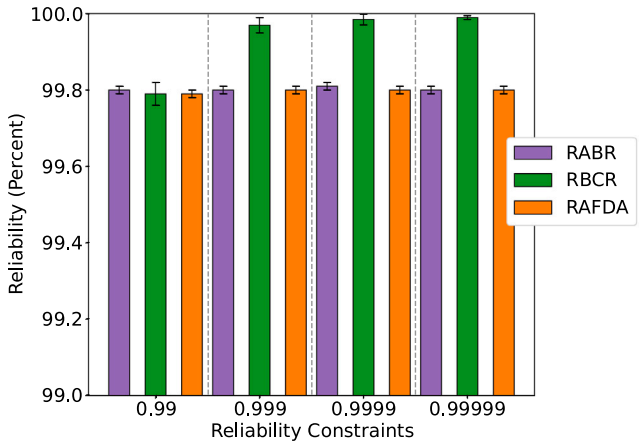


Fig. 8. Average reliability in different reliability constraints of the flows with Network Connectivity of 0.5.

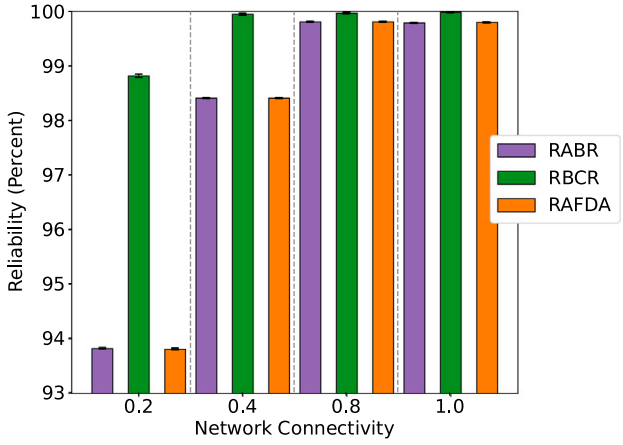


Fig. 9. Average reliability in different network connectivity of the flows.

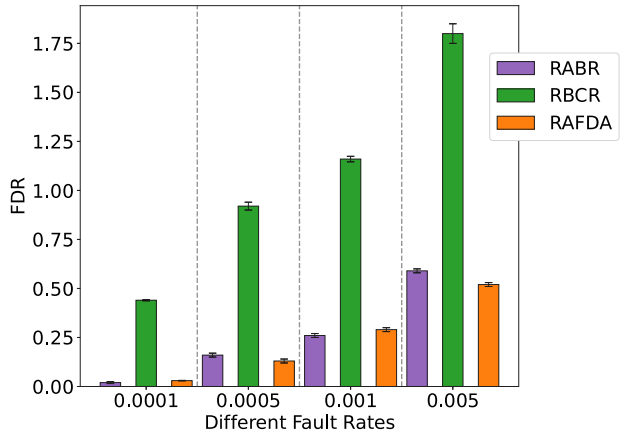


Fig. 10. FDR of the evaluated algorithms in different network fault rates.

constraints increase, the SR of RBCR declines. This is because lower link capacities limit the ability to perform sufficient flow duplication, whether in parallel or sequential forms. Additionally, both RABR and RAFDA exhibit a slight decrease in SR as the bandwidth constraints of the flows increase.

5.2.3. Reliability of the evaluated algorithms

In this part, we evaluate the average reliability of the algorithms in different configurations. Fig. 8 shows the average reliability of the algorithms in different reliability constraints. All the compared algorithms achieve higher average reliability. However, RBCR leads to the most average reliability because of the duplication method. RBCR consistently excels in terms of average reliability across various reliability constraints due to its ability to duplicate paths. Utilizing multiple routes ensures that flows achieve the reliability constraint. RABR delivers moderate average reliability. It selects the most reliable paths from the top k paths without path duplication, which limits its performance compared to RBCR. However, it generally outperforms RAFDA. When reliability constraints are lower, RABR performs well, as it can choose highly reliable paths from available options without needing duplication, achieving average reliability close to that of RBCR.

RAFDA generally exhibits the lowest average reliability among the evaluated algorithms. It employs a max–min reliability approach, selecting paths based on the minimum link reliability. Overall, RAFDA and RABR do not show significant differences in average reliability, as neither incorporates methods to satisfy reliability constraints, although they are reliability-aware. This often results in fewer flows meeting higher reliability constraints, particularly in the absence of path duplication in scenarios with lower reliability constraints.

Fig. 9 illustrates the average reliability of the algorithms across different levels of network connectivity. As network connectivity increases, the RBCR algorithm demonstrates superior average reliability compared to both RAFDA and RABR. All the compared algorithms exhibit improved performance with increased network connectivity (i.e., a higher number of possible paths). The highest reliability for RBCR is largely attributable to its ability to leverage proposed duplication methods (sequential or parallel/hybrid) to satisfy reliability constraints, whereas RAFDA and RABR are limited to selecting a single path, thereby limiting their ability to enhance reliability.

5.2.4. The flow duplication rate (FDR) of the evaluated algorithms

Fig. 10 highlights that the FDR of RBCR is higher than that of other algorithms, owing to its use of parallel, hybrid, or sequential duplication methods. By transmitting multiple copies of data across different paths to ensure reliability constraints of IoT applications, RBCR increases the FDR compared to RABR and RAFDA, both of which rely on a single path and thus exhibit lower FDR. These results emphasize the trade-off between reliability and efficiency. While RBCR achieves the highest reliability, it incurs a higher FDR, whereas RAFDA and RABR offer lower FDR at the expense of reduced reliability.

The FDR of RBCR is higher, particularly in scenarios where the network experienced higher faults (i.e., $\lambda \geq 0.001$). It is because the algorithm had to increase the number of duplications to ensure the flows continued to meet the reliability constraints as demonstrated in Section 5.2.2.

The increased FDR in RAFDA and RABR is due to their dependence on re-transmitting flows when delivery to the destination fails due to network issues. In contrast, the higher FDR in RBCR is a result of duplication, not re-transmission. Re-transmission leads to delays and deadline violations, which can be very detrimental and costly for real-time IoT applications.

5.2.5. Overall comparison

Table 4 presents a comprehensive comparison of the RBCR, RABR, and RAFDA algorithms across several critical performance metrics. Runtime Efficiency: RABR demonstrates the fastest run-time at 0.026 s, followed by RBCR at 0.092 s. RAFDA lags significantly behind with a runtime of 5.01 s. This substantial difference in execution speed could be critical in real-time applications where rapid decision-making is essential.

In terms of Success Rates: RBCR consistently outperforms both RABR and RAFDA across all success rate metrics, where for Success

Table 4
Overall comparison of the algorithms.

Metrics/Algs	RBCR	RABR	RAFDA
Runtime	0.092 s	0.026 s	5.01 s
SR-NC	97.25%	53.75%	49%
SR-RC	88%	46.75%	41.25%
SR-BC	80%	51%	39.25%
R-RC	99.933%	99.803%	99.797%
R-NC	99.886%	98.308%	98.307%

Rate under different Network Connectivity (SR-NC), RBCR achieves a remarkable 97.25% success rate, nearly doubling RABR (53.75%) and RAFDA (49%). With various Reliability Constraints (SR-RC), RBCR maintains a high 88% success rate, while RABR and RAFDA drop to 46.75% and 41.25%, respectively. Under Bandwidth Constraints (SR-BC), RBCR still leads at 80%, with RABR following at 51% and RAFDA at 39.25%.

These results suggest that RBCR is significantly more effective at satisfying various network constraints compared to its counterparts.

In terms of reliability, all three algorithms perform well, but RBCR maintains a slight edge. For example, under Reliability Constraints (R-RC), RBCR achieves 99.933% reliability, higher than RABR (99.803%) and RAFDA (99.797%). Under different Network Connectivity (R-NC), RBCR again leads with 99.886% reliability, compared to RABR and RAFDA, both around 98.3%.

While the differences in reliability are small, they could be significant in large-scale networks or critical applications where even minor improvements in reliability are valuable. Overall, RBCR demonstrates superior performance in success rates and reliability across all scenarios. RAFDA, while competitive in reliability, struggles with success rates and runtime efficiency. These results suggest that RBCR offers the best performance in networks with real-time IoT applications where high success rates and reliability are very important.

6. Conclusions and future works

This paper presents two algorithms: RBCR and RABR. RBCR is tailored for real-time applications with stringent reliability constraints, where violating time requirements incurs significant costs. In contrast, RABR is more suitable for time-sensitive applications, where slight delays are tolerable. Both algorithms are designed specifically for Software-Defined Networks (SDNs) in Fog Computing environments and effectively address the dual challenges of meeting reliability and bandwidth constraints for IoT applications. The RBCR algorithm operates in three phases: (1) reliability and bandwidth-constrained path selection, (2) flow duplication through parallel/hybrid routing, and (3) sequential routing with flow duplication to meet reliability requirements. A greedy pathfinder is also introduced to enhance time efficiency while maintaining high performance.

Simulation results demonstrate that RBCR outperforms the state-of-the-art RAFDA algorithm in terms of reliability, success rate (flows satisfying reliability and bandwidth constraints), and time efficiency (run-time). RBCR achieves success rates up to twice those of existing solutions, coupled with highly efficient execution times, making it ideal for applications requiring optimized reliability and bandwidth.

While flow duplication significantly enhances network reliability, its inherent redundancy introduces trade-offs with bandwidth efficiency particularly in dense network environments. Emerging dynamic mitigation approaches like Adaptive Flow Duplication could optimize this balance even under high-traffic conditions. Future work should integrate temporal constraints for time-sensitive IoT domains (e.g., industrial automation) and expand evaluations to energy/delay metrics. Machine learning-enhanced coordination could further refine reliability-efficiency balances.

CRediT authorship contribution statement

Parisa Valizadeh: Writing – review & editing, Writing – original draft, Software, Methodology. **Mohammad Hossein Yaghmaee:** Supervision. **Yasser Sedaghat:** Validation, Supervision.

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

Data will be made available on request.

References

[1] J. Jiang, Z. Li, Y. Tian, N. Al-Nabhan, A review of techniques and methods for IoT applications in collaborative cloud-fog environment, in: D. Wanchun (Ed.), Secur. Commun. Netw. 2020 (2020) 8849181, http://dx.doi.org/10.1155/2020/8849181.

[2] A. Taghinezhad-Niar, J. Taheri, Security , reliability , cost , and energy-aware scheduling of real-time workflows in compute-continuum environments, IEEE Trans. Cloud Comput. 12 (3) (2024) 954–965, http://dx.doi.org/10.1109/TCC.2024.3426282.

[3] Y. Asghari Alaie, M. Hosseini Shirvani, A.M. Rahmani, A hybrid bi-objective scheduling algorithm for execution of scientific workflows on cloud platforms with execution time and reliability approach, J. Supercomput. 79 (2) (2023) 1451–1503.

[4] K. Liu, W. Quan, N. Cheng, W. Wu, Z. Xu, L. Guo, D. Gao, H. Zhang, Reliable PPO-based concurrent multipath transfer for time-sensitive applications, IEEE Trans. Veh. Technol. 72 (10) (2023) 13575–13590, http://dx.doi.org/10.1109/TVT.2023.3277712.

[5] L. Csikor, M. Szalay, G. Rétvári, G. Pongrácz, D.P. Pezaros, L. Toka, Transition to SDN is HARMLESS: Hybrid architecture for migrating legacy ethernet switches to SDN, IEEE/ACM Trans. Netw. 28 (1) (2020) 275–288, http://dx.doi.org/10.1109/TNET.2019.2958762.

[6] N. Khan, R. bin Salleh, A. Koubaa, Z. Khan, M.K. Khan, I. Ali, Data plane failure and its recovery techniques in SDN: A systematic literature review, J. King Saud Univ. - Comput. Inf. Sci. 35 (3) (2023) 176–201, http://dx.doi.org/10.1016/j.jksuci.2023.02.001.

[7] Z. Li, Y. Hu, J. Wu, J. Lu, P4resilience: Scalable resilience for multi-failure recovery in SDN with programmable data plane, Comput. Netw. 208 (2022) 108896, http://dx.doi.org/10.1016/j.comnet.2022.108896.

[8] F. Tang, I. Haque, ReMon: A resilient flow monitoring framework, in: 2019 Network Traffic Measurement and Analysis Conference, TMA, 2019, pp. 137–144, http://dx.doi.org/10.23919/TMA.2019.8784521.

[9] M. Ibrar, L. Wang, G.-M. Muntean, A. Akbar, N. Shah, K.R. Malik, PrePass-Flow: A machine learning based technique to minimize ACL policy violation due to links failure in hybrid SDN, Comput. Netw. 184 (2021) 107706, http://dx.doi.org/10.1016/j.comnet.2020.107706.

[10] P. Valizadeh, A. Taghinezhad-Niar, A fault tolerant multi-controller framework for SDN DDoS attacks detection, Int. J. Web Res. 5 (1) (2022) 1–7, http://dx.doi.org/10.22133/ijwr.2022.345927.1119.

[11] R. Mohammadi, S. Akleylek, A. Ghaffari, A. Shirmarz, Automatic delay-sensitive applications quality of service improvement with deep flows discrimination in software defined networks, Clust. Comput. 26 (1) (2023) 437–459, http://dx.doi.org/10.1007/s10586-022-03729-6.

[12] D. Tang, Z. Zheng, K. Li, C. Yin, W. Liang, J. Zhang, FTOP: An efficient flow table overflow preventing system for switches in SDN, IEEE Trans. Netw. Sci. Eng. (2023) 1–13, http://dx.doi.org/10.1109/TNSE.2023.3297650.

[13] A.J. Kadhim, S.A.H. Seno, J.I. Naser, J. Hajipour, DMPFS: Delay-efficient multicasting based on parked vehicles, fog computing and SDN in vehicular networks, Veh. Commun. 36 (2022) 100488, http://dx.doi.org/10.1016/j.vehcom.2022.100488.

[14] G. Ding, J. Yuan, G. Yu, Y. Jiang, Two-timescale resource management for ultrareliable and low-latency vehicular communications, IEEE Trans. Commun. 70 (5) (2022) 3282–3294, http://dx.doi.org/10.1109/TCOMM.2022.3162366.

[15] M. Hosseini Shirvani, Y. Ramzanpoor, Multi-objective QoS-aware optimization for deployment of IoT applications on cloud and fog computing infrastructure, Neural Comput. Appl. 35 (26) (2023) 19581–19626.

[16] M. Ibrar, L. Wang, N. Shah, O. Rottenstreich, G.-M. Muntean, A. Akbar, Reliability-aware flow distribution algorithm in SDN-enabled fog computing for smart cities, IEEE Trans. Veh. Technol. 72 (1) (2023) 573–588, http://dx.doi.org/10.1109/TVT.2022.3202195.

[17] M. Ibrar, L. Wang, G.-M. Muntean, J. Chen, N. Shah, A. Akbar, IHSF: An intelligent solution for improved performance of reliable and time-sensitive flows in hybrid SDN-based FC IoT systems, IEEE Internet Things J. 8 (5) (2021) 3130–3142, http://dx.doi.org/10.1109/JIOT.2020.3024560.
[18] A.-C. Hauschild, R. Martin, S.C. Holst, J. Wienbeck, D. Heider, Guideline for software life cycle in health informatics, Iscience 25 (12) (2022).
[19] S. Xu, Y. Qian, R.Q. Hu, On reliability of smart Grid Neighborhood Area networks, IEEE Access 3 (2015) 2352–2365, http://dx.doi.org/10.1109/ACCESS.2015.2502250.
[20] A. Akbar, M. Ibrar, M.A. Jan, A.K. Bashir, L. Wang, SDN-enabled adaptive and reliable communication in IoT-fog environment using machine learning and multiobjective optimization, IEEE Internet Things J. 8 (5) (2021) 3057–3065, http://dx.doi.org/10.1109/JIOT.2020.3038768.
[21] N. Wang, T.-z. Tian, J.-t. He, C.-z. Zhang, J. Yang, Transmission reliability evaluation of wireless sensor networks considering channel capacity randomness and energy consumption failure, Reliab. Eng. Syst. Saf. 242 (2024) 109769.
[22] R. AlZoman, M.J.F. Alenazi, Exploiting SDN to improve QoS of smart city networks against link failures, in: 2020 Seventh International Conference on Software Defined Systems, SDS, 2020, pp. 100–106, http://dx.doi.org/10.1109/SDS49854.2020.9143878.
[23] L. Wang, L. Yao, Z. Xu, G. Wu, M.S. Obaidat, CFR: A cooperative link failure recovery scheme in software-defined networks, Int. J. Commun. Syst. 31 (10) (2018) e3560, http://dx.doi.org/10.1002/dac.3560.
[24] Z. Yang, K.L. Yeung, SDN candidate selection in hybrid IP/SDN networks for single link failure protection, IEEE/ACM Trans. Netw. 28 (1) (2020) 312–321, http://dx.doi.org/10.1109/TNET.2019.2959588.
[25] P. Thorat, S. Singh, A. Bhat, V. Lakshmi Narasimhan, G. Jain, SDN-enabled IoT: Ensuring reliability in IoT networks through software defined networks, in: M.A. Matin (Ed.), Towards Cognitive IoT Networks, Springer International Publishing, Cham, 2020, pp. 33–53.
[26] H. Seddiqi, S. Babaie, A new protection-based approach for link failure management of software-defined networks, IEEE Trans. Netw. Sci. Eng. 8 (4) (2021) 3303–3312, http://dx.doi.org/10.1109/TNSE.2021.3110315.
[27] J. Vidyakant, A. Ghazali, B. Appasani, C. Ravariu, P.A. Srinivasulu, Reliability analysis of smart grid networks incorporating hardware failures and packet loss, Rev. Roum. Sci. Tech. - Ser. Electrotech. Énergétique 65 (2020) 245–252.
[28] S.M. Raza, S. Ahvar, R. Amin, M. Hussain, Reliability aware multiple path installation in software-defined networking, Electronics 10 (22) (2021) 2820.
[29] M.L. Foko Sindjoung, M. Velempini, V. Kengne Tchendji, ARPMEC: an adaptive mobile edge computing-based routing protocol for IoT networks, Clust. Comput. 27 (7) (2024) 9435–9450, http://dx.doi.org/10.1007/s10586-024-04450-2.
[30] J.L. Herrera, J. Galán-Jiménez, L. Foschini, P. Bellavista, J. Berrocal, J.M. Murillo, QoS-aware fog node placement for intensive IoT applications in SDN-Fog scenarios, IEEE Internet Things J. 9 (15) (2022) 13725–13739, http://dx.doi.org/10.1109/JIOT.2022.3143948.
[31] J.L. Herrera, J. Galán-Jiménez, P. Bellavista, L. Foschini, J. Garcia-Alonso, J.M. Murillo, J. Berrocal, Optimal deployment of fog nodes, microservices and SDN controllers in time-sensitive IoT scenarios, in: 2021 IEEE Global Communications Conference, GLOBECOM, 2021, pp. 1–6, http://dx.doi.org/10.1109/GLOBECOM46510.2021.9685995.
[32] B. Fan, H. Tan, Y. Li, Critical link identification algorithm for power communication networks in SDN architecture, Int. J. Crit. Infrastruct. Prot. 40 (2023) 100584.
[33] M.A. Mahmood, W.K. Seah, I. Welch, Reliability in wireless sensor networks: A survey and challenges ahead, Comput. Netw. 79 (2015) 166–187.
[34] X.-Z. Xu, Y.-F. Niu, C. He, A minimal path-based method for computing multistate network reliability, Complexity 2020 (1) (2020) 8060794.
[35] A. Dâmaso, N. Rosa, P. Maciel, Reliability of wireless sensor networks, Sensors 14 (9) (2014) 15760–15785.
[36] W.-C. Yeh, An improved sum-of-disjoint-products technique for symbolic multi-state flow network reliability, IEEE Trans. Reliab. 64 (4) (2015) 1185–1193, http://dx.doi.org/10.1109/TR.2015.2452573.
[37] L. Xing, A. Shrestha, QoS reliability of hierarchical clustered wireless sensor networks, in: 2006 IEEE International Performance Computing and Communications Conference, 2006, pp. 6 pp.–646, http://dx.doi.org/10.1109/.2006.1629464.
[38] B. Lantz, B. Heller, N. McKeown, A network in a laptop: Rapid prototyping for software-defined networks, in: Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks, 2010, p. 19, http://dx.doi.org/10.1145/1868447.1868466.
[39] A. Botta, A. Dainotti, A. Pescapé, A tool for the generation of realistic network workload for emerging networking scenarios, Comput. Netw. 56 (15) (2012) 3531–3547, http://dx.doi.org/10.1016/j.comnet.2012.02.019.
[40] X. Yang, H. Xu, J. Liu, C. Qian, X. Fan, H. Huang, H. Wang, Achieving high reliability and throughput in software defined networks, Comput. Netw. 197 (2021) 108271.
[41] F. Zhang, Y. Chen, H. Lu, Y. Huang, Network-aware reliability modeling and optimization for microservice placement, 2024, arXiv preprint arXiv:2405.18001.
[42] J. Park, S. Samarakoon, H. Shiri, M.K. Abdel-Aziz, T. Nishio, A. Elgabli, M. Bennis, Extreme ultra-reliable and low-latency communication, Nat. Electron. 5 (3) (2022) 133–141.

Parisa Valizadeh received her B.Sc. and M.Sc. degrees in Computer Engineering from the University of Tabriz, Tabriz, Iran in 2016 and 2019, respectively. Her research interests include Software-Defined Networking, Internet of Things, Network Management, and Quality of Services. She is now a Ph.D. candidate at the Ferdowsi University of Mashhad, Mashhad, Iran.

Mohammad Hossein Yaghmaee received his B.S. degree from Sharif University of Technology, Tehran, Iran in 1993, and M.S. and Ph.D degrees in communication engineering from Tehran Polytechnic (Amirkabir) University of Technology in 1995 and 2000, respectively. He is a full professor at the Computer Engineering Department, Ferdowsi University of Mashhad (FUM). His research interests are in Smart Grid Communication, Internet of Things (IoT), Software Defined Networking (SDN) and Network Function Virtualization (NFV). November 1998 to July1999, he was with Network Technology Group (NTG), C&C Media research labs., NEC corporation, Tokyo, Japan, as a visiting research scholar. From September 2007 to August 2008, he was with the Lane Department of Computer Science and Electrical Engineering, West Virginia University, Morgantown, USA as the visiting associate professor. Currently, he is with the Electrical and Computer Engineering (ECE) department of the University of Toronto (UoT) as the visiting professor.

Yasser Sedaghat received the M.Sc. and Ph.D. degrees in Computer Engineering from Sharif University of Technology, Tehran, Iran, in 2006 and 2011, respectively. He is an Associate Professor with the Department of Computer Engineering, Ferdowsi University of Mashhad (FUM), Mashhad, Iran. He has established and has been one of the chairs of the Dependable Distributed Embedded Systems (DDEmS) Laboratory, FUM, since 2012. His current research interests include Dependable Embedded Systems and Networks, Reliable Software Design, Embedded Operating Systems, and FPGAbased Designs.