Reliability-Aware Flow Distribution Algorithm in SDN-enabled Fog Computing for Smart Cities

Muhammad Ibrar, Lei Wang, Nadir Shah, Ori Rottenstreich, Gabriel-Miro Muntean, and Aamir Akbar

***Abstract*—To improve the performance of a smart cities, integration of two emerging technologies, namely Fog Computing (FC) and Software-Defined Networking (SDN), has been proposed and is gaining momentum. This integrated architecture is called SDN-based FC for Internet-of-Things (IoT) applications and is expected to meet the requirements of these applications (i.e., easy manageability, high scalability, reliability, and low latency). Existing traffic engineering approaches proposed for SDN-based FC for IoT compute the route between an IoT device and fog server subject to some Quality of Service (QoS) constraints. However, these approaches ignore the link reliability level in the route computation process. Unlike them, this paper proposes a Reliability-Aware Flow Distribution Algorithm (RAFDA) and two associated optimization algorithms called Reactive Reliability-Aware Heuristic Algorithms (RRAHA-1 and RRAHA-2), which distribute the flows on the links based on the links' reliability levels subject to additional constraints like traffic load on the link, bandwidth allocation, link utilization, and end-to-end delay. The proposed algorithms minimize the impact of link failure occurrences on the ongoing time-critical flows (applications/services) of smart cities. The proposed algorithms, evaluated using both real network traces and simulations, outperform existing approaches in terms of performance for delay-sensitive services in smart cities.**

***Index Terms*—Fog computing, IoT, SDN, link failure, reliability, smart cities.**

I. Introduction

A smart city consists of a large number of smart things, also known as Internet-of-Things (IoT) devices (e.g., appliances in smart homes, wearable devices, smart vehicles, laptops, and smart grids) [1]. The IoT devices are embedded with sensors, allowing them to monitor the environmental information and transmit it to the concerned objects (i.e., users or the remote cloud) in real-time via the Internet for further analysis. The IoT devices produce large amounts of data which needs to be retrieved, processed, and eventually stored in real-time [2]. Although massive computing resources are available for processing of this data in the cloud and/or data centers, when many IoT devices transmit large amounts of data via the Internet, the performance of data access is affected. For example, this results in congestion in the core/backbone network, which is in turn causes lower Quality of Service (QoS) levels to IoT device services (e.g. longer end-to-end delays, lower throughput). To address this issue, edge computing was introduced, with a profound positive impact on smart cities services [3]. In particular, edge computing technologies such as Fog Computing (FC) [4] and Mobile Edge Computing (MEC) [5] have emerged as promising solutions to the resource constraints IoT devices. FC and MEC enable IoT devices access large amounts of storage and computation resources, while also reducing the latency.

Ensuring path reliability in smart cities for IoT applications including in contexts such as tactile internet, healthcare, and autonomous vehicles is crucial because these applications demand persistent communications to facilitate uninterrupted high-quality services. Unfortunately, it is widely noted, including in a Cisco report [6], that IoT devices produce large amounts of data traffic, which may cause network congestion [7]. Congestion in the network is such severe and growing a problem that it has the potential to paralyze the whole network activity. Similarly, faults occur frequently in large networks (i.e., links go down almost every 30 minutes [8]–[10]), affecting negatively the supported services.

In order to address diverse network challenges, including congestion, a centralized networking approach employing Software-Defined Networking (SDN) was proposed. A SDN-based architecture is a viable solution to provide enhanced communication services to IoT devices in a FC smart cities context. In smart cities, due to the limited operating expenses (OPEX) and capital expenditure (CAPEX) [13, 38], the number of fog nodes should be much less than the number of IoT devices. Therefore, the IoT devices connected to the fog are served in a multi-hop manner. The applications of resource constraint IoT devices increase day by day, and IoT devices offload time-critical tasks to the fog servers in smart cities and in doing so, they need reliable paths. By reliable path, we mean a path that has very low chances of link failure. A link failure can occur due to many factors related to network connection, configuration, power, software, hardware, and incident events [8, 9]. Studies show that in a communication network, the incidence of communication link failures is higher than router/switch failures, and they significantly impact network performance. Consequently, link failure chances increase as the network size increases. Therefore, this paper focuses on communication link failures in SDN-based smart cities architecture. SDN decouples the control plane from the data plane. It enables control delegation from all forwarding devices to a controller, a logically centralized entity [11] and makes

M. Ibrar and Lei Wang are with the School of Software, Dalian University of Science and Technology, China. (e-mail: mibrar@mail.dlut.edu.cn, lei.wang@dlut.edu.cn)

N. Shah is with Department of Computer Science, COMSATS University Islamabad, Wah Campus, Pakistan. (e-mail: nadirshah82@gmail.com)

Ori Rottenstreich is with Department of Computer Science and the Department of Electrical Engineering of the Technion. (e-mail: ori.rot@gmail.com)

G.-M. Muntean is with School of Electronic Engineering, Dublin City University, Ireland (e-mail: gabriel.muntean@dcu.ie)

A. Akbar is with Department of Computer Science, Abdul Wali Khan University Mardan, Pakistan. (e-mail: amirakbar@awkum.edu.pk)

§ L. Wang is corresponding author (Dalian University of Technology, China).

the data plane programmable [12]. This makes easier network management. Although SDN offers numerous advantages, the limited Ternary Content Addressable Memory (TCAM) of SDN switches adversely affects the network performance [13]. Considering the TCAM constraint, proactive installation of multiple paths for a given flow in each switch is not feasible in a large network [14]. Additionally, if the path is reliable, then there is no need to install multiple paths for a given flow [15]. Therefore, when a link gets disconnected, a reactive approach is most appropriate.

Several existing traffic engineering approaches proposed for SDN-based FC for IoT improve network performance given diverse constraints in terms of available bandwidth on individual links, energy consumption, end-to-end delay, link utilization, and load balancing [16], [17]. However, these approaches distribute the number of flows on the links without considering the links' reliability levels [9], [15]. Apart from the reliability level, the IoT applications have also several QoS requirements. Therefore, the path from source to destination also needs to satisfy multiple constraints simultaneously, such as traffic load, bandwidth, link utilization, and delay. However, in conventional routing, decisions are made based only on a single or two metrics. In this paper, we used a SDN architecture to distribute the flows based on the links' reliability levels (minimizing the impact of link failures on the services), subject to additional constraints. Therefore, the proposed approach assigns more flows to a more reliable link and vice versa, while considering other constraints related to traffic load, bandwidth, link utilization, and delay. This is the novelty of our paper and was not covered in any other works. Additionally, we proposed heuristic algorithms to compute a reliable path between a source and the destination because there is no efficient (polynomial) exact solution for the multi-constrained (MCP) selection problem.

Addressing the limitations of existing works and motivated by the importance of reliability, this paper proposes a new Reliability-Aware Flow Distribution Algorithm (RAFDA) for a SDN-based FC for IoT architecture. RAFDA enables flow distribution based on link reliability, while also considering traffic load, bandwidth allocation, link utilization, and end-to-end delay. Specifically, RAFDA defines the reliability of a link as the frequency of down-time and failure-time of the link in a given time interval.

The major contributions of this paper are as follows:

1) A reliable path for task offloading in SDN-based FC for IoT applications is identified in order to minimize the impact of a link failure on the services (i.e., delay, potential loss, congestion, and extra energy consumption); this is especially important for mission-critical and delay-sensitive IoT applications.
2) RAFDA is introduced to distribute the flows based on the links' reliability levels, subject to additional constraints (i.e., load balancing, bandwidth allocation, link utilization, and delay). The proposed approach assigns more flows to more reliable links and vice versa.
3) Two Reactive Reliability-Aware Heuristic Algorithms (RRAHA-1 and RRAHA-2) are proposed enabling the SDN controller to compute a reliable path to maximize network performance and minimize the computation time while rerouting the affected flow, end-to-end delay, and improve link utilisation.
4) Considering TCAM capacity limitations, it employs a reactive approach with low associated processing to recover from link failure based on the fact that the SDN controller calculates the best path for a flow. When the path/link at a switch gets disconnected, the switch starts the recovery procedure by asking the SDN controller to compute and install a new best path for each flow affected by disconnection.

The rest of the paper is organized as follows. Section II discusses related works and identifies their benefits and limitations. Section III explains the problem statement through an example scenario. The proposed solution, including the related mathematical modeling is introduced in Section IV. Section V discusses the experimental results. Finally, Section VI concludes the paper and describes some potential future research directions.

II. Related Works

This section discusses some existing approaches relevant to the work presented in this paper, as follows.

Managing link failure is more significant in SDN-based FC to ensure the high reliable and uninterrupted services to the IoT devices because of the time-critical characteristics of most applications of IoT devices [18]–[20]. These time-critical IoT applications need a reliable path for data transmission because if a link failure occurs, the service will be affected severely. Additionally, IoT devices generate traffic continuously, which needs reliable communication links that have a low probability of failure to achieve QoS. In our previous works [9], [21], the SDN controller uses the K-nearest neighbor algorithm to estimate the reliability level of the links. For different types of IoT applications, the controller computes the best path. However, to minimize the impact of the link failure with the rapidly increasing demand of IoT applications, solutions to enhance the reliability must account for network dynamics. None of the proposed solutions distribute the flows based on link reliability levels, increasing the burden on the SDN controller to react every time there is a failure and enable flow redistribution. This, in turn, adds extra delay to the flow.

In an SDN-based IoT network, the SDN controller must handle link failures and restore the communication path for seamless delivery of IoT applications. Thorat *et al.* proposed the Immediate Controller Dependent (ICoD) and Local Immediate (Lim) approaches to minimizes the effect of link failures in a SDN-IoT network [22]. When a link failure occurs in the proposed model, the associated switch sends a notification to the SDN controller. The SDN controller generates a group message to notify the associated switch. Therefore, a single message can reroute "n" flows. Additionally, the proposed model minimizes the use of TCAM memory of OpenFlow switches using the group entry feature. However, to achieve low latency and minimize the impact of link failure, the authors must consider path the reliability factor, number of flows on the path, utilization ratio, and bandwidth.

TABLE I: Existing Literature: Summary

Existing Literature	SDN	Smart City	Link Failure	Path Reliability	Flows Distribution	Delay	Traffic Load	Bandwidth	Link Utilization	FC
[9], [21]	✓	×	✓	✓	×	✓	×	✓	×	✓
[22]	✓	✓	✓	×	×	✓	×	✓	×	✓
[11], [14], [23]–[27]	✓	×	✓	×	×	✓	✓	✓	×	×
[28]–[32]	✓	✓	✓	✓	×	✓	×	✓	×	✓
Proposed Solution *(RAFDA)*	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Unfortunately, the research literature on SDN-based FC for IoT applications does not address well link failures. Recent studies which discuss link failure in SDN architectures include [11], [14], [23]–[27].

Chu *et al.* proposed a scheme that takes advantage of SDN and ensures traffic reachability in single link failure scenarios [23]. The proposed scheme installs proactively multiple backup paths and redirects the traffic through pre-configured IP tunnels. The proposed solution obtains good results. However, proactive computation of multiple backup paths is not practical due to TCAM constraints. Link fault protection is also a critical aspect. A solution called Hybrid-Hie [11] is proposed for both computation time and configuration overhead minimization, e.g., flow table entries and IP tunnels, in a link fault case. Hybrid-Hie reroutes fast the traffic on pre-computed backup paths when the default path is not available. Network layer failure, e.g., switch or link, leads to packets loss, congestion, end-to-end delay increase, and network efficiency reduction. Fast recovery of network layer failure plays a paramount role in the network [25]. In this context, Carmelo *et al.* proposed a scheme called Stateful Programmable faIlure DEtection and Recovery (SPIDER) [26]. SPIDER uses a stateful data plane to monitor link status periodically and reroute the traffic in case of link failure.

With the growing miniaturization of sensors, computers, and mobile phones, smart city applications are increasingly relying on communication technologies and many of them require reliable connectivity [28]. Smart cities applications not only demand congestion-free routes, high speed, and availability, but time-critical applications also need a reliable path between source and destination pair in today's networks. Generally, a reliable path in smart cities relies on wired network switch or link availability [29]–[31]. However, the term reliability is still subject of many different interpretations, although overlapping to a large extent. According to the Cambridge dictionary[1], reliability is "the quality of being able to be trusted or believed because of working or behaving well". However, in the context of a smart city communication system, the more technical definition of the American Society for Quality (ASQ)[2] was prefered: "the probability that a product, system or service will perform its intended function adequately for a specified period of time, or will operate in a defined environment without failure." Therefore, in this work, the reliability of a link is defined as the frequency of down-time and failure-time of the link in a given time interval in a SDN-based smart city.

Link failure problems can significantly impact reliable data transmission due to internal or external events [29]. In the presence of link failure, it is challenging to fulfil the QoS requirements of delay-sensitive smart city applications such as internet telephony and video streaming. AlZoman *et al.* proposed a smart city resilient system to provide QoS to smart city applications in the presence of link failure [29]. SCRS consists of the following modules: (a) TopoDiscovery module enables the SDN controller to obtain the status information about switches and links, (b) FailureDetection module detects the switch or link failure, (c) FastDivRoute computes the alternative path for the disturbed flows due to the link failure, (d) trafficQoS module priorities the disturbed flows based on the flows requirement (i.e., voice-over-IP (VoIP) and video conferencing) and available link capacity, and (e) RulesGenerator installs the flow rules for disturbed flows based on the status of a smart city. However, the proposed approach has not considered the link failure status while computing the alternative path for the disturbed flows and has not been designed for a large smart network. Harir et al. [32] proposed a protection-based approach to handle the link failure problem. In the proposed model, the authors examined the link status by using three univariate parameters, packet loss, latency, throughput. To minimize the latency delay, the proposed protection model installed backup paths if it realized that the failure probability of a link is high. Probability density functions are used to distinguish between functional and non-functional links. Multipath routing schemes have been proposed to improve the performance of SDN-based smart in both cities [31] and [33].

Table I summarizes the major aspects focused on existing work on link failure in SDN-based systems. This analysis shows that there is an important research gap for reliability-aware flow distribution in smart cities in the presence of link failure. The difference between our model and the state of the art in this area is as follows:

- In case of failure, the SDN-based related works redirect the data flows on an alternative shortest path using a reactive or proactive flow installation process. These approaches do not consider the path reliability level when computing the alternative path. As our proposed model assigns data flows based on path reliability levels, this is a novelty of our approach.
- Current approaches reroute the flows after a failure has occurred and are applied to the affected flows only. In contrast, our approach distributes the number of data flows to a path based on path reliability in advance of

[1]Cambridge Online Dictionary. Available online: https://dictionary.cambridge.org/dictionary/english/reliability (Access: 3 Nov. 2021)

[2]ASQ Reliability Definition. Available online: https://asq.org/quality-resources/reliability (Access: 3 Nov. 2021)

any potential failure. This means that a more reliable path receives many data flows, and a less reliable path handles a lower number of flows. This approach minimizes the potential impact of link failures and involves all flows.
- Table I indicates the parameters considered by the existing research papers when computing a path for a data flow. Our proposed approach considers all the parameters listed, which no other related work does.
- Computing a path based on link reliability, load balancing, and link utilization is a complex optimization problem. To solve it, we proposed two Reactive Reliability-Aware Heuristic Algorithms (RRAHA-1 and RRAHA-2), enabling the SDN controller to compute a reliable path to maximize network performance and minimize the computation time and end-to-end delay, and improve link utilization, while rerouting the affected flows. RRAHA-1 and RRAHA-2 algorithms are novel and have very good computational complexity.

Following the discussion of the related works, it is evident that for identification of efficient routes from source to destination in a smart environment context, there is a need to consider the link reliability levels subject to additional constraints, including load balancing, link capacity, and link utilization. The significance of this statement is illustrated through an example scenario in Section III.

III. Problem Statement

This section illustrates the significance of the problem addressed in this paper (i.e., link reliability consideration in route computation) via an example. Consider five IoT devices *(s1, s2, s3, s4, and s5)* connected with different switches via Access Points (AP) to communicate with a Fog Server (FS) located at switch G, as shown in Fig. 1(a). The traffic generated by each IoT device is shown in Fig. 1(a). There are seven OpenFlow enabled switches and a fog server at Fog layer (i.e., A, B, C, D, E, F, and G), which are connected to the network with links whose costs are expressed in terms of distance, and each has a bandwidth of 10Mbps. Let us assume that the SDN controller calculates the shortest path based on link costs. The path followed by the flows of each IoT device is shown in Fig. 1(a). This produces congestion on link D-G because all IoT devices share the common competitive link D-G. The total traffic flow on the link D-G is 19Mbps, which is greater than the available 10Mbps, which is the maximum utilization of the links in the given network.

This scenario leads to network congestion and increases end-to-end delay for time-critical applications. To minimize the congestion, some schemes have been proposed in SDN, like Fibbing [23]. When congestion occurs on any forwarding path, Fibbing redirects the traffic flow to other links having low traffic. For each destination, Fibbing uses a Directed Acyclic Graph (DAG) to compute multiple paths. After computing multiple paths to a destination, Fibbing places virtual fake nodes and links by modifying the forwarding path at each physical router. After applying the Fibbing approach in Fig. 1(a), the congestion can be avoided, as shown in Fig. 1(b). However, this approach does not consider the link reliability, which leads to the following problem. In this scenario, seven flows of 1Mbps share the common competitive link D-G. Thus, the data rate of seven flows is less than the capacity of the D-G link. However, if the link D-G has low reliability and fails, then seven flows are disrupted by running the recovery procedures [14]. It takes a long time for seven flows to recover. Additionally, retransmission of flows due to link failure also determines extra energy consumption of energy-constrained IoT devices. To address this problem, we should compute the paths such that the number of flows is assigned to each link based on the link reliability level, as shown in Fig. 1(b). By using this technique, if the link D-G fails, then only three flows are disrupted. This reduces the pressure on the SDN controller since only three flows are disrupted, as shown in Fig. 1(b).

Finally, this paper focuses on addressing major challenges related to employing link reliability in the efforts to improve the performance of SDN-based IoT networks. Here are these challenges and how they are addressed:

- **How to compute the link reliability?** We proposed a new method for measuring link reliability which was described in Bootstrapping (Section IV-A2).
- **How to use link reliability to improve network performance?** Our proposed model assigns data flows based on path reliability levels. This is based on the fact that more reliable paths will fail less and therefore will have a positive impact on a higher number of flows.
- **When would link reliability be used?** Unlike existing approaches which react to link failures only, our approach distributes a higher number of data flows to more reliable paths in advance of any failure. This minimizes the impact of a link failure on flows when a failure occurs.
- **What other parameters are considered along with link reliability?** Our proposed solution employs other aspects along link reliability, including bandwidth, traffic load, delay and link utilisation. This increases the overall performance of the network.
- **How to recover in the fastest manner from failure considering reliability?** This paper introduces two Reactive Reliability-Aware Heuristic Algorithms (RRAHA-1 and RRAHA-2) which enable the SDN controller to compute a reliable path while maximizing network performance and minimizing the computation time.

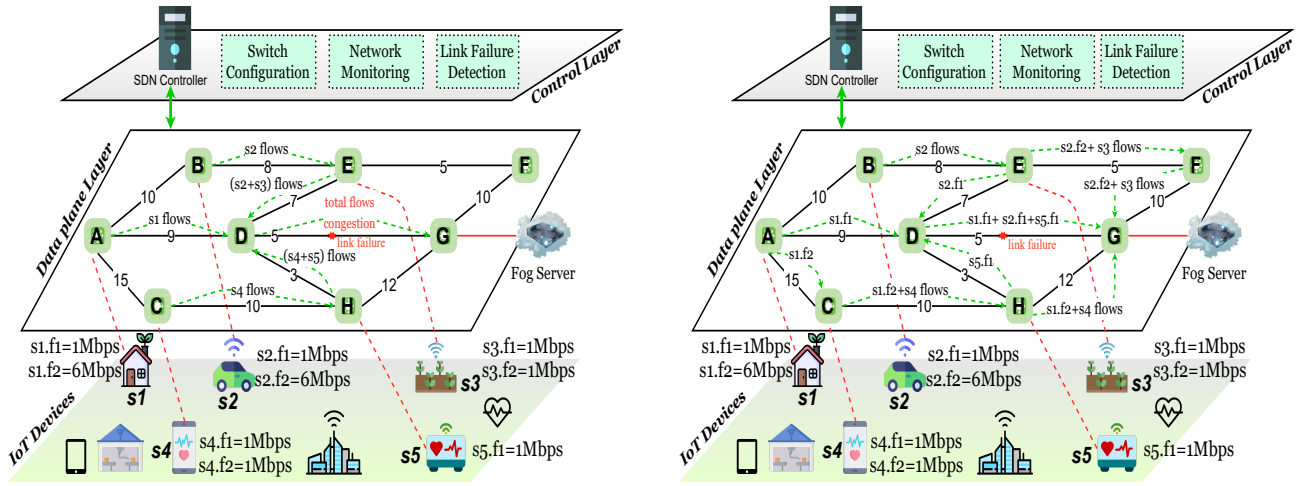
The proposed approach, the Reliability-Aware Flow Distribution Algorithm is detailed in section IV.

IV. The Proposed Reliability-Aware Flow Distribution Algorithm (RAFDA)

In this section, we formulate our proposed problem and algorithm, RAFDA, by describing its details. Next we propose two optimization algorithms to solve the proposed problem.

A. Problem Formulation

As discussed in Section III, we propose RAFDA to distribute data flows from IoT devices to the FS over different links/paths based on the links' reliability levels subject to additional constraints like traffic load on the link, bandwidth allocation, and link utilization. Unlike existing works which



(a) Congestion and Link Failure Scenario

(b) Avoid Congestion through Flows Distribution and Link Failure Scenario

Fig. 1: Communication of IoT devices with a Fog Server in a smart city scenario

just react to failure, the proposed solution uses an innovative approach and distributes the flows on paths according to their reliability levels, trying to prevent failures from affecting flow activity as much as possible. A SDN-based FC architecture is considered.

The SDN-based FC infrastructure contains both network devices (i.e., OpenFlow enabled switches/routers) denoted as V_o and fog server nodes (i.e., fog enabled WiFi/switches/routers/servers) denoted as FS. The infrastructure is modeled as an undirected graph $G = (V,\ L)$, where V is the set of nodes, and L represents the set of links/edges that connect the nodes. Each link $l_j \in L$ has a reliability level denoted by $0 \leq \Re_{l_j} \leq 1$ and capacity signified as $\varphi_{l_j} > 0\ bytes/sec$. The associated notations are explained in subsection IV-A1. The proposed model has the following four components - the bootstrap process, flow set-up process, packet forwarding process, and link failure process, as shown in Fig. 2. The functionality of each component is explained in the following subsections.

1) Notations: This sub-section defines the notations used in this paper.

- Sets
  - $V_o = \{s_1, ..., s_n\}$, set of OpenFlow enabled switches.
  - $FS = \{v_1, ..., v_m\}$, set of fog servers or destination node.
    - v is considered a fog server or a destination node, $v \in FS$.
  - $V = \{V_o \cup FS\}$.
  - $E_d = \{u_1, ..., u_z\}$, set of edge/source devices.
    - u is considered an edge/source device $u \in E_d$.
  - $L = \{l_1, ..., l_y\}$, set of links that connect the nodes to maintain connectivity. The link is defined by a set of ordered pairs of distinct nodes, i.e., $l_j = (u, v)\ |u$ and $v \in V$. $|L|$ shows the number of links of set L.
    - $\Re_{l_j}^{t_i}$, represents the reliability level of a link l_j at time slot t_i, $((0 \leq \Re_{l_j}^{t_i} \leq 1),\ l_j \in L)$.
    - $\mathbb{DF}_{l_j}^{t_i}$, represents the number of failures (frequency) that occurred for the link l_j during time slot t_i
    - $\mathbb{DT}_{l_j}^{t_i}$, represents the down duration for the link l_j during time slot t_i
    - $\lambda_{l_j}^{t_i}$, is traffic load λ (in terms of bytes) on link $l_j \in L$ in time slot t_i.
    - $\beta_{l_j}^{t_i}$, is the bandwidth allocation of $l_j \in L$ (in term of bytes).
    - $\mu_{l_j}^{t_i}$, represents the link l_j utilization during time slot t_i.
    - φ_{l_j}, is the maximum capacity of $l_j \in L$.
    - $\kappa_{l_j}^{t_i}$, is the threshold value of the link l_j reliability level.
    - T shows the total simulation time.
  - $F = \{f_1, ..., f_q\}$ set of flows, $f \in F$.
    - $f(d)$, represents the data rate of f in bytes/s, $((0 < f(d) \leq N)$ in bytes)).
    - $f(d)l_j$, is the data rate of f on link l_j.
  - $P = \{p_1, ..., p_r\}$, is the set of paths from the edge device to the fog server $p_k = (u, v)$, where $p_k \in P$
- Decision Variables
  - $f_{l_j} \in \{0, 1\}$, if the flow f passing through the link l_j then $f_{l_j} = 1$, otherwise $f_{l_j} = 0$
  - $\varphi_{l_j} \geq 1$, is an integer value (expressed in bytes)
  - $\kappa_{l_j} > 0$, is the threshold value of $\Re_{l_j}^{t_i} > 0$

2) Bootstrap Process: When the network is switched-on, OpenFlow protocol Hello, Feature-Request and Feature-Reply [34] messages are exchanged between switches and the controller in order for the controller to get the network view[3]. In the proposed work, the Feature-Reply message of every switch contains along with other information the $\mathbb{DT}_{l_j}^{t_i}$ and $\mathbb{DF}_{l_j}^{t_i}$ (which represents the down duration and number of failures occurred for the link l_j during time slot t_i), $\sum_{i=1}^{n} f_{l_j}^{t_i}$ (which represents the number of flows on each link during

[3]https://opennetworking.org/wp-content /uploads/2014/10/openflow-switch-v1.5.1.pdf

a specific time slot t_i, where $f \in F$), $\lambda_{l_j}^{t_i}$ in bytes (this represents traffic load on each link in time slot t_i) , φ_{l_j} in bytes (which represents the link capacity), and $\mu_{l_j}^{t_i}$ (which is the link utilization during the specific time slot t_i). We assume that such information of a link can be obtained by router/switch, for example, using an approach like the one reported in [35]. The controller stores this information along with the topology of the network.

In the SDN-based FC, when the SDN controller receives the Feature-Reply message after the specific time interval t_i, e.g., t_i = 10 seconds(s), then the SDN controller computes the reliability level for the link l_j (i.e., $\Re_{l_j}^{t_i}$) based on Eq. (1). In Eq. (1), from the values of $\mathbb{DT}_{l_j}$ and $\mathbb{DF}_{l_j}$, the controller defines the reliability level of the link l_j in the time slot t_i. When the downtime duration of a link is greater than t_i, then $\Re_{l_j} = 0$. If the number of failures is zero, e.g., $\mathbb{DF}_{l_j}^{t_i} = 0$, then the $\Re_{l_j}$ is maximum (i.e., $\Re_{l_j} = 1$). Otherwise, $\Re_{l_j}$ is computed, as shown in Eq. (1).

$$\Re_{l_j}^{t_i} = \begin{cases} 1 & \iff \mathbb{DF}_{l_j} = 0 \\ \frac{(1-\mathbb{DT}_{l_j}/t_i)}{\sum_{t_i \in T} \mathbb{DF}_{l_j}} & \iff \mathbb{DF}_{l_j} > 0 \text{ and } \mathbb{DT}_{l_j} \leq t_i \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

3) Flow Set-up Process: In this work, based on a SDN-based FC-based architecture, when a data packet arrives, the switch matches it against the flow rules from its flow table. If a match is found, the packet is forwarded accordingly. Otherwise, the switch forwards the packet header to the controller and waits for the flow rules. When the controller receives the packet header, it performs the process of identification of a reliable path according to the updated network view and policies and installs the flow rules in the flow tables of the switches along the path.

In this paper, the flow f is represented by a triple $\Psi = (u, v, f(d))$. In this triple, u and v are the edge device and destination node respectively, and $f(d) : u \rightarrow \mathbb{R}^+$ represents flow f's data rate of the edge device u ($\mathbb{R}^+$ indicates a positive real number). The SDN controller computes a set of feasible paths P_f for flow f (where $P_f \subseteq P$) and selects a path $p_k^f \in P_f$ for f from all the feasible paths that satisfy the requirements (described in details in Section IV-A4). Moreover, we assume that a path $p_k^f \in P_f$ for flow f from u to v, $(s.t., u \neq v)$, is represented as a sequence of links $L_{p_k^f} \subseteq L = \{l_1, l_2, ..., l_{y-1}, l_y\}$, where $\{l_1, l_2, ...l_{y-1}\} \in L_{p_k^f}$, $l_j = (s_i, s_{i+1}) | s_i, s_{i+1} \in V_o$, and $i = 1, 2, 3, ...n$.

In the proposed model, we make some assumptions. (a) the controller has updated network G statistical information about the links, (b) the proposed algorithm is invoked for every new flow f arrival event at the controller, (c) active flows are considered only, and (d) the controller is functional. Next, in the proposed model, the controller computes the reliable path based on the objective function subject to the additional constraints, as follows.

4) Objective Function: As already indicated, the main goal of the proposed RAFDA is to assign flows to paths based on their reliability (see Eq. (2)), subject to additional constraints.

$$objective \quad min(f) : min \left(\frac{\sum_{i=1}^{q} \sum_{j=1}^{y} f_{l_j}^{t_i} \Re_{l_j}^{t_i}}{|L|} \right), \quad (2)$$

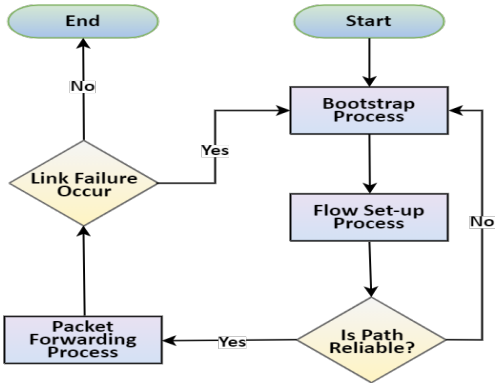


Fig. 2: Flow-Chart of the Proposed RAFDA Model

Eq. 2 assigns the number of flows to each link based on its reliability level ($\Re_{l_j}$), where indices q, y, and $|L|$ represent the total number of flows, total number of links, and number of links of set L, respectively. As a path consists of links, we consider that the reliability level of a path is represented by the reliability level of the bottleneck link along the path. This is as when a link fails or gets disconnected, the desire is that a minimum number of flows shall be disturbed. In order to minimize the impact of link failure, one of the primary objectives of the proposed RAFDA is to minimize the maximum utilization of reliable links. Eq. (2) shows the main objective of the proposed model, subject to additional constraints:

The additional constraints are as follows:

1) The minimum reliability level of each link in the flow path should be maximized (see Eq. (3)).
2) The traffic load should be minimum on each link - expressed in bytes (see Eq. (4), Eq. (5), and Eq. (6)).
3) To avoid the congestion, the bandwidth allocation for all flows on the links should be less than the maximum capacity of the links - expressed in bytes (see Eq. (7)).
4) To avoid under-utilization of the link, the link utilization should be maximized: $0 < \mu_{l_j}^{t_i} \leq 1$ (see Eq. (8)).
5) The delay of a reliable path should be minimum (see Eq. (10)).

The SDN controller updates frequently its status by collecting network statistical information from the switches. Each link l_j in the network G, which can be used to transmit data, has the reliability level $\Re_{l_j}^{t_i}$ and available bandwidth $\beta_{l_j}^{t_i}$. We assume that there is a set of flows F and each $f \in F$ requires a reliable path satisfying the capacity demand of the flow for data delivery.

In the context of this paper, IoT devices in smart cities generate flows with different data rates, and the communication links are heterogeneous in terms of bandwidth capacity (φ_{l_j}). The bandwidth demand of f in time slot t_i is denoted by $\Upsilon_f^{t_i}$.

In general, due to limited capacity φ_{l_j} and low-reliability level $\Re_{l_j}$ of the link, not all flows can be forwarded on the same path. In the proposed model, the established path needs to satisfy the following constraints:

(a) Path's reliability ($\Re_{l_j}^{t_i}$) **Constraint:** The proposed RAFDA algorithm distributes the numbers of flows on the

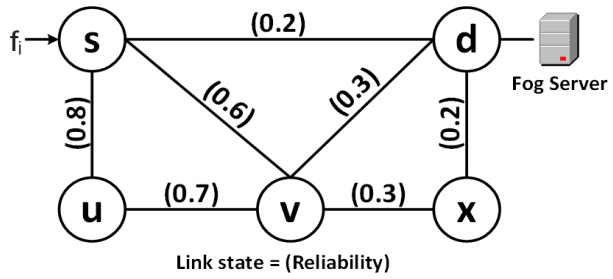


Fig. 3: A network (G) showing the reliability value of each link($\Re_{l_j}^{t_i}$)

links such as to maximise the minimum link reliability level, as indicated by Eq. (3):

$$max(\Re_{l_j}^{t_i}) = max\left(min \sum_{p_k^f \in P_f} \sum_{l_j \in L_{p_k^f}} \Re_{l_j}\right) \quad (3)$$

$$\forall\ \Upsilon_f \leq \varphi_{l_j}, f_{l_j} \leq \varphi_{l_j}, \Re_{l_j} > \kappa_{l_j}, f_{l_j} > 0$$

Eq. (3) refers to the minimum reliability level of each link l_j in the path p_k^f. This addresses the concern raised by the report in [23], [36], that any link goes down almost every 30 minutes.

Fig. 3 illustrates the proposed model for a network G with 5 OpenFlow switches and 7 links. The labels on the links represent the reliability level of each link (i.e., $l(s,d)$ has a reliability level $\Re_{l(s,d)}$ = 0.2). Suppose for a flow f, the SDN controller calculates the reliable path from the switch s towards the fog server, located at switch d. There are five 5 possible paths for flow f distribution in G:

- p_1 = $\{l(s,d)\}$ = $\{\Re_{l(s,d)} = 0.2\}$
- p_2 = $\{l(s,v),\ l(v,d)\}$ = $\{\Re_{(s,v)} = 0.6,\ \Re_{(v,d)} = 0.3\}$
- p_3 = $\{l(s,v),\ l(v,x),\ l(x,d)\}$ = $\{\Re_{(s,v)} = 0.6,\ \Re_{(v,x)} = 0.3,\ \Re_{(x,d)} = 0.2\}$
- p_4 = $\{l(s,u),\ l(u,v),\ l(v,d)\}$ = $\{\Re_{(s,u)} = 0.8,\ \Re_{(u,v)} = 0.7,\ \Re_{(v,d)} = 0.3\}$
- p_5 = $\{l(s,u),\ l(u,v),\ l(v,x),\ l(x,d)\}$ = $\{\Re_{(s,u)} = 0.8,\ \Re_{(u,v)} = 0.7,\ \Re_{(v,x)} = 0.3,\ \Re_{(v,d)} = 0.2\ \}$

The goal is to identify the most reliable path, defined by the fact that the minimum reliability level of all its links is the highest. In G, paths p_2 and p_4 are the most reliable because their minimum reliability levels of their links (i.e., 0.3) is the highest. Therefore, by using our proposed approach, the SDN controller would select either path p_2 or p_4 for flow f because both paths meet the criteria. However, the SDN controller checks the other constraints, as described next.

(b) Traffic load $(\lambda_{l_j}^{t_i})$ **Constraint:**

Our proposed approach also attempts to minimize the traffic load (expressed in terms of bytes) on the links. The traffic load λ of the flows passing through the link l_i is collected in $\lambda^{t_i} = \left\{\lambda_{l_j,f_i}^{t_i}, ..., \lambda_{l_j,f_q}^{t_i}\right\}$ in time slot t_i by the controller in our proposed approach. Therefore, the flow f forwarding decision on the link is based on traffic load on the link $\lambda_{l_j}^{t_i}$ subject to the capacity constraint φ_{l_j} as shown in Eq. (4).

$$0 < \sum_{p_k^f \in P_f} \sum_{l_j \in L_{p_k^f}} \lambda_{l_j,f} \Upsilon_f \leq \varphi_{l_j}, \quad \forall\ l_j \in p_k^f, f_{l_j} > 0 \quad (4)$$

In the proposed model, we distribute the traffic load (expressed in terms of number of bytes) on the links, as shown in Eq. (5). Eq. (5) was introduced to minimize the impact of link failure while fulfilling the flows' bandwidth requirements and therefore considers both links' reliability levels and flows' bandwidth demands. Eq. (5) guarantees that traffic load (expressed in terms of bytes) distributes equally on the links. Additionally, the traffic load passing through the link should follow Eq. (6).

$$min(\lambda_{l_j}^{t_i}) = min\left(\lambda \sum_{j=1}^{y} 1\left\{l_j \in p_f\right\} \frac{\Re_{l_j}^{t_i} \Upsilon_f}{|F|} \leq \varphi_{l_j}\right) \quad (5)$$

$$f_{l_j} \sum_{f} \lambda_{f_{l_j}} \leq \varphi_{l_j}, \quad \forall f, l_j \quad (6)$$

$$\forall\ l_j \in p_k^f,\ \Re_{l_j} > \kappa_{l_j}\ ,\ \Upsilon_f \leq \varphi_{l_j}$$

(c) Bandwidth Allocation $(\beta_{l_j}^{t_i})$ **Constraint:** When a flow is assigned to the path at time t_i, the path's spare capacity is reduced. More specifically, when a new flow arrives in the network and is assigned to a path (which consists of a sequence of links), the algorithm changes the residual capacity of all links along the path. This also determines changes in the reliable path conditions for the new flows. Therefore, bandwidth allocation to f on path $p_k^f \in P_f$ is represented as a mapping $f(d) : p_k^f \rightarrow \mathbb{R}^{''}$, where $\mathbb{R}^{''}$ is the non-negative real number set. Bandwidth allocation on path p_k is $\sum_{l_j \in p_k^f : p_k^f \in P_f} f(d)(p_k^f) \leq \varphi_{l_j}$. The shared capacity constraints are satisfied for any link $l_j \in p_k^f$ in a path at time t_i, as shown in Eq. (7).

$$max(\beta_{l_j}^{t_i}) = max\left(\sum_{t_i, p_k^f \in P_f} \sum_{l_j \in p_k^f} f(d) l_j \leq \varphi_{l_j}, \forall l_j \in p_k^f\right) \quad (7)$$

$$\forall\ \Upsilon_f < \beta_{l_j}^{t_i}, \mu_{l_j} < 1, (0 < f(d) \leq N\ in\ bytes),\ p_k^f \in L$$

Traffic load $\lambda_{l_j}^{t_i}$ should be less or equal to bandwidth allocation $\beta_{l_j}^{t_i}$ (expressed in terms of bytes) in time slot t_i on each link.

(d) Link Utilization $(\mu_{l_j}^{t_i})$ **Constraint:** Suppose $f(d)l_j$ indicates the data rate of flow f (expressed in terms of bytes) carried by the link $l_j \in p_k^f$ is $f(d)l_j = \sum_{i=0}^{q} f_{l_j}$ $(f(d)l_j)$ at time t_i. The link utilization $\mu_{l_j}^{t_i}$ can be calculated from aggregated flows data rate in specific time slot t_i, as shown in Eq. (8).

$$min(\mu_{l_j}^{t_i}) = min\left(\sum_{t_i, p_k^f \in P_f} \sum_{l_j \in p_k^f} \left\{\frac{f(d)l_j}{\varphi_{l_j}}\right\}\right) \quad (8)$$

$$\forall\ p_k^f \in L, \mu_{l_j}^{t_i} \leq \varphi_{l_j}^{t_i}, f_{l_j} > 0$$

The desire is to select those links which have a minimum utilization ratio $\mu_{l_j}^{t_i}$, indicating the amount of traffic that is currently (or can be) transported by link l_j at the moment t_i.

(e) Delay $(\varepsilon_{l_j}^{t_i})$ **Constraint:** To calculate the end-to-end delay of a path, we employ the $\varepsilon(.)$ function in the proposed work. The delay associated with a link depends on the traffic

load on that link (i.e., $\lambda_{l_j}^{t_i}$), available bandwidth ($\beta_{l_j}^{t_i}$), and total link capacity ($\varphi_{l_j}^{t_i}$), and can be expressed as follows.

$$\varepsilon_{p_k}^{t_i} = \sum_{t_i, l_j \in p_k} \varepsilon_{l_j}^{t_i}, \tag{9}$$

where, $\varepsilon_{l_j}^{t_i}$ is the end-to-end delay of l_j link. In terms of delay components, we consider the processing Pro delay, transmission Tx delay, and propagation Pg delay in the proposed work. The SDN controller obtains the links status information to compute end-to-end delay of a path for the time-critical applications, as follows.

$$min(\varepsilon_{p_k}^{t^i}) = min\left(\sum_{t_i, f \in F} \varepsilon^{t_i}(Tx + Pg + Pro)\right). \tag{10}$$

Algorithm Description: Based on the objective function presented in Eq. (2) and subject to the additional constraints described in Eq. (3), Eq. (5), Eq. (7), Eq. (8), and Eq. (10), the SDN controller computes the reliable path in the given network G for a flow f. Let suppose u is the IoT device generating $f \in F$ and $v \in FS$. The feasible path set for $p_k^f \in P_f$ where $P_f \subseteq P$ regarding u and v, indicated by $p_{k(u,v)}^f$ is shown in Eq. (11).

- Reliable and feasible path calculation, where χ_{l_j} represents the feasibility of the path which is based on a $min(f)$ on the path p_k, $min(\lambda)$, $min(\mu)$, $min(\varepsilon)$, and $max(\beta)$ (expressed in terms of number of bytes).

$$p_{k(u,v)}^f = \frac{(\Re_{l_j})(\chi_{l_j})}{\sum_{p_k=1}^{r}(\Re_{l_j})(\chi_{l_j})}, \quad \forall l_j \in p_k^f \tag{11}$$

Algorithm 1 describes the steps involved in the computation of the reliable path based on our objective function and constraints. Algorithm 1 distributes the flows on the links based on the links' reliability levels subject to the additional constraints in order to achieve the objective. When a data packet arrives, the switch matches it against the flow rules present in the flow table. If the flow rule is present in the flow table, then the packet forwards/drop it according to the flow rule (lines 1 to 4). Otherwise, the switch forwards the digest packet to the controller. Based on the network's global status (as explained in IV-A2), the SDN controller checks for those paths that have the lowest number of flows by using Eq. 2. The controller computes the links' reliability level and selects those paths in which the minimum reliability of links is maximum (Line 7) by using Eq. (3). After completing the reliable paths P selection, Algorithm 1 uses the Check_Const function for the remaining constraints (lines 18 to 26). Based on IoT flow demand, Algorithm 1 uses Eq. (5) and computes paths with minimum traffic, and traffic demand should be less the link capacity. High link utilization also causes delay; therefore, the RAFDA targets the paths with low utilisation links (see Eq. (8)) and low end-to-end delays (see Eq. (10)). Finally, the controller computes and installs the flow rules for the given flow on a reliable path subject to the constraints (Line 15) and updates the network status (Line 16).

Unfortunately, the proposed computation of a reliable path for f based on the mentioned constraints is NP-hard as is a multi-path routing with bandwidth and delay constraints problem, which is NP-hard [37], [38]. Consequently, the number of possibilities for choosing the path and the time complexity of Algorithm 1 are increasing exponentially, so we solve this problem by proposing two heuristic algorithms, which will be described in subsection IV-B.

Algorithm 1: Reliability-Aware Flow Distribution Algorithm (RAFDA)

Input: $G(V, L)$, $\Re_{l_j}^{t_i}$, $\lambda_{l_j}^{t_i}$, $\beta_{l_j}^{t_i}$, $\mu_{l_j}^{t_i}$, $\varepsilon_{l_j}^{t_i}$
Output: return reliable and feasible path

```
1  f = flow
2  if (rule-match = True) then
3      forwarded/drop f as per flow table entry
4      return;
5  else
6      for path in paths(src, dst) do
           // compute the reliable paths:
7          path = Networkx.reliable_paths(G, src, dst)
            using Eq. (3)
8          if Check_Const(path, f) then
9              flag = 1
10             rel_path = path
11         else
12             rel_path = path - 1
13         end
14     end
15     install the flow rules on rel_path for f
16     update the network G status
17 end
18 Function Check_Const(path, f):
       // check the constraints
19     for (l_i,l_j) in path do
20         compute path using Eq. 11
21         if Const satisfied then
22             return True
23         else
24             return False
25         end
26     end
```

After computing the reliable path based on the objective as mentioned earlier and constraints, the controller installs the path by setting up the flow rules at the switches along the path. This is achieved by using Flow-Mod and Packet-Out messages of the OpenFlow protocol. Moreover, in order to avoid the inconsistency behavior and packet loss in the network [39], the controller installs the flow rules in reverse order at the switches along the path in our proposed approach. Reverse order means that the controller installs the flow rules on the traversed path from the last switch to the first switch (.i.e, $\{v_{d-1}, ..., v_2, v_s\} \in V$). After this, the switch forwards the data packet as will be described in subsection IV-A5.

5) Packet Forwarding Process: When a flow rule is received at a switch, then the switch forwards any relevant data packet according to the rule. The switch also stores the flow rule so that if subsequent data packets arrive, then the switch

will forward the packets according to the flow rule without consulting the controller. This reduces the traffic overhead at the controller. In the packet forwarding process, if a switch receives the packet and in the meanwhile the forwarding path goes down, then the switch will initiate the link failure process, which will be explained in subsection IV-A6.

6) Link Failure Process: In the proposed approach, a reactive rerouting process is used for any link failure-disrupted flows. In order to reduce the amount of communication and processing in the proposed work, the SDN controller stores information about the flow status. When a link failure occurs, the affected OpenFlow switch notifies the controller using a *Port-Down* message. Upon receiving the *Port-Down* message, the SDN controller finds the flows that will be affected due to the link failure by examining the status of the stored flow at the controller. Consequently, the affected installed flows are removed from the switches [40]. When the affected OpenFlow switch $S_u \in V_o$ notifies the controller C, the response time of the controller to disturbed flows is based on the number/frequency of the disturbed flows. The frequency of disturbed flows f reported by S_u in time slot t_i is represented by $DF_f^{S_u}$. Therefore, the load on controller $\mathbb{L}(C(t_i))$ is shown in Eq. (12).

$$\mathbb{L}(C(t_i)) = \bigcup_{S_u \in V_o} DF_f^{S_u}(t_i) \tag{12}$$

Eq. (12) shows that a link with high number of failed flows creates load on the controller. As a result, rerouting a large number of disrupted flows takes a longer time, and consequently increases the end-to-end delay for packet delivery, affecting system performance.

B. Reactive Reliability-Aware Heuristic Algorithms (RRAHA)

This section introduces two algorithms as heuristic solutions to the NP-hard optimisation problem set as an objective in Section IV-A3. The problem is to identify the most appropriate path based on links' reliability levels given a set of performance-related constraints.

1) Reactive Reliability-Aware Heuristic Algorithm-1 (RRAHA-1): The goal of RRAHA-1 is to enable data flow forwarding in case of a link failure on the most reliable path, while minimizing both the number of disturbed flows and the computation time. RRAHA-1 is presented in Algorithm 2 and is described as follows.

When the switch receives flow f from the IoT device, it matches the flow against the installed flow rules. If a flow rule is present, the switch forwards, or drops the flow (Lines 2-3). If the flow rule is not present in the associated switch, the switch forwards the flow to the SDN controller for route computation (Line 5). Initially, the path set is empty (Line 6). RRAHA-1 algorithm computes all possible paths between the source and destination pair for a given flow f and associates them with reliability values (Line 7). Among these paths, the algorithm RRAHA-1 selects RPP% (reliable path percentage) with the highest reliability values by using Eq. (1) (Line 8). Starting from these paths with the highest reliability values, RRAHA-1 identifies the best path after applying the other constraints as described in Section IV-A3. This reduces significantly the computation time, as not all the paths are checked for the other constraints. Note that if the subset of paths is very small e.g. 1, then all of the paths identified will be considered in Algorithm 2.

Algorithm 2: Reactive Reliability-Aware Heuristic Algorithm (RRAHA-1)

Input: $G(V, L)$, $\Re_{l_j}^{t_i}$, $\lambda_{l_j}^{t_i}$, $\beta_{l_j}^{t_i}$, and $\mu_{l_j}^{t_i}$
Output: return reliable and feasible path

1 $f =$ flow
2 **if** *(rule-match = True)* **then**
3 forwarded/drop f as per flow table entry
4 return;
5 **else**
6 P_f = Ø, where $P_f \subseteq P$ // initially path set P_f is empty
// run the algorithm to find all possible paths P_f (loop free paths) for f
7 $P_f \leftarrow$ Networkx.reliable_paths(G, src, dst)
// take top 10% of reliable paths for f, in which min links reliability is max. (Eq. (3)
8
$max(min \sum_{p_k^f \in P_f} \sum_{l_j \in p_k^f} \Re_{l_j})$
$p_k^f \leftarrow p_k^f \subseteq \{P_f\}$
// check p_k^f paths constraints
9
$feasible \;\; p_k^f \leftarrow p_k^f \subseteq \left\{p_k^f\right\}$
repeat $step - 9$ until the path is offered by p_k^f is feasible for f
10 **end**
11 install the flow rules on rel_path for f
12 update the network G status

2) Reactive Reliability-Aware Heuristic Algorithm-2 (RRAHA-2): The goal of RRAHA-2 is to enable data flow forwarding in case of a link failure on the most reliable path, while both minimizing the underutilized links in the network and computation time. RRAHA-2 forwards the rerouted flows on the less utilized paths. RRAHA-2 is presented in Algorithm 3 and is described as follows.

RRAHA-2 computes all possible paths based on utilization level between the source and destination pair for a given flow f (Line 6). RRAHA-2 selects UPP (underutilized path percentage) with the lowest utilization ratio by using Eq. (8) (Line 7) in Algorithm 3. Using this UPP, the algorithm computes the best path after applying the other constraints, described in Section IV-A3. This minimizes the computation time as not all the paths are checked against the other constraints. Again, note that if the UPP is very small, then all of the paths identified will be considered in Algorithm 3. In Algorithm

2 and Algorithm 3, RPP and UPP were both set to 10% to reduce the computational time, as desired.

Algorithm 3: Reactive Reliability-Aware Heuristic Algorithm (RRAHA-2)

Input: $G(V, L)$, $\Re_{l_j}^{t_i}$, $\lambda_{l_j}^{t_i}$, $\beta_{l_j}^{t_i}$, and $\mu_{l_j}^{t_i}$
Output: return reliable and feasible path

1 $f =$ flow
2 **if** *(rule-match = True)* **then**
3 forwarded/drop f as per flow table entry
4 `return;`
5 **else**
`// run the algorithm to find all underutilized-paths for` f
6 $P_f \leftarrow$ Networkx.underutilized_paths(G, src, dst)
`// take top 10% of underutilized-paths for` f
7 $p_k^f \leftarrow p_k^f \subseteq \{P_f\}$
`// check` p_k^f `paths constraints`
8 $max(min \sum_{p_k^f \in P_f} \sum_{l_j \in p_k^f} \Re_{l_j})$
$feasible \;\; p_k^f \leftarrow p_k^f \subseteq \{P_f\}$
repeat $step - 4$ until the path is offered by p_k^f is feasible for f
9 **end**
10 install the flow rules on for f
11 update the network G status

C. Computational Complexity of Algorithms

This section analyzes the computation complexity of the proposed heuristic algorithms, detailed in Algorithm 1, Algorithm 2, and Algorithm 3.

Algorithm 1: The SDN controller obtains updated links' status, including link failure frequency, number of flows, and traffic load, from the network, as discussed in Section IV-A2. Based on the obtained information, the SDN controller computes the reliable and feasible path for a given flow f. Suppose the total number of paths between a pair of source and destination are P. The computational time for selecting the reliable paths among all paths is $\mathcal{O}(P)$. The reliability level of all links in the reliable paths should be greater than a threshold value (i.e., $\Re_{l_j}^{t_i} > \kappa_{l_j}$). Consequently, each link in a path can only be selected k times for flow distribution (until the threshold value is reached). Thus, the $Networkx.reliable\_paths$ function takes $\mathcal{O}(kP \times log(kP))$ (see Algorithm 1, line 7). After completing the reliable paths P selection, Algorithm 1 uses the Check_Const function for remaining constraints m for each path of the total number of reliable paths P. Therefore, the Check_Const function takes $\mathcal{O}(mP)$, while m signifies the constraints of the a path (see Algorithm 1, line 19-24). The time complexity for the other statements in the algorithm is constant, i.e., $\mathcal{O}(c)$. Therefore, for F number of flows, the worst-case complexity of Algorithm 1 is $\mathcal{O}(F \times kP \times log(kP) + mP + c) \approx \mathcal{O}(F \times kP \times log(kP) + mP)$.

Algorithm 2 and 3: The selection process of reliable and feasible paths of Algorithm 2 and Algorithm 3 is the same as in Algorithm 1. The major differences are that Algorithm 2 first computes the top RPP percentage of reliable paths, and Algorithm 3 first selects among the top UPP underutilized paths between source and destination. Algorithm 2 and Algorithm 3 involve sorting all P paths based on reliability and utilization ratios, respectively. Therefore, the complexity of Algorithm 2 and Algorithm 3 is the same $\approx \mathcal{O}(F \times P \times log(P) + mP)$.

V. Experimental Results

In order to validate our proposed algorithms (i.e., RAFDA, RRAHA-1, and RRAHA-2) for SDN-based FC, simulations are conducted in two environments based on Mininet (small network) and real network traces (large network) [41], with a small and large number of flows, respectively. We compare our proposed algorithms (RAFDA, RRAHA-1, and RRAHA-2) with the Un-Reliability Model (U-RM) shortest path (based on link cost) [13] and Smart City Resilient System (SCRS) [29]. In [13], the authors select the optimal path in terms of link utilization, delay, and SDN rules-capacity for IoT task offloading. SCRS [29] utilized SDN architecture to minimize the impact of a link failure on the time-critical application of smart cities and computes the alternative path based on application requirements. Box plots (i.e., box-and-whisker plots) are used to illustrate comparatively the experimental results. For each scenario, a box is drawn by connecting the lower quartile, median, and upper quartile. Finally, the whiskers from the box show the lower and upper extremes of the scenario. Additionally, the black line in a box depicts the mean value of the scenario. All the simulations are performed 10 times, and the duration of each scenario is $1000 - seconds$, giving credibility to the analysis of the proposed algorithms. Additionally, we use UDP flows in both real network traces and Mininet.

A. Results of Real Network Traces (Large Network)

Real network traces are available at [41] for an anonymous topology of the local large-scale campus network. The topology [41] contains 415 switches, 1062 links, and 1024 source/edge nodes. In the real network traces, we have modified the capacity of the communication link randomly (i.e., $[10, 100]$Mbps), and each link in the topology annotated with link's reliability level using Eq. (1).

We select various numbers of source and destination pairs (i.e., 200 to 1000) from the topology using the Python random function. These IoT devices generate flows randomly with different data rates in the $[1, 25]$Mbps interval and transmit these flows continuously (for 1000s). Each flow include between [5, 15] data packets, each from a different IoT device to a different fog server [38]. We are not considering the controller overhead in these scenarios, but we assume the out-of-band communication model for the switch to controller communication.

Assessment parameters: In performance evaluation, we examine three parameters,
(i) Congested links in the network (i.e., when the utilization ratio of the link is greater than link capacity ($\mu_{l_j}^{t_i} > \varphi_{l_j}$)). More specifically, if the controller computes a path for a flow using an algorithm (i.e., either RAFDA, or RRAHA-1, or RRAHA-2, SCRS or U-RM) and the bandwidth demand of the flow exceeds the path bandwidth due to some links, we consider the links in the paths as congested links in the simulation.
(ii) The number of disturbed flows in case of link failure. More specifically, when a link, say l_j fails, the number of flows the controller needs to recompute alternative paths for is considered.
(iii) The number of underutilized links in the network. In the simulation, a link is considered as an underutilized link if its utilization ratio is less than 50% ($\mu_{l_j}^{t_i} < 50\%$).

Solutions and Scenarios We compare the assessment parameters (i.e., congested links, number of disturbed flows, and underutilized links) when the four algorithms are employed (i.e., RAFDA, RRAHA-1, RRAHA-2, SCRS [29], and U-RM scheme [13]) in diverse scenarios characterised by various number of flows and diverse link capacity. Detail explanation is given in subsection V-A1 and subsection V-A2.

1) Results When Varying the Number of Flows: The proposed RAFDA, RRAHA-1, and RRAHA-2 algorithms and SCRS and U-RM for SDN-based FC are evaluated using real network traces [41] with different number flows in order to show their efficiency.

In Fig. 4, as the number of flows is increasing, the number of congested links also increases in all approaches: RAFDA, RRAHA-1, RRAHA-2, SCRS, and U-RM. However, in the proposed approaches, the number of congested links is smaller than that recorded in the SCRS and U-RM approaches. It is mainly because the proposed algorithms distribute the flows on the links based on the links' reliability levels subject to additional constraints. From the results, we observe that there are more congested links in SCRS than in our approaches because SCRS computes the delay-aware paths for the flows and does not consider the flow distribution based on links' reliability level and link utilization during link failures. U-RM considers the shortest path based on the number of hops. Therefore, the number of links congested in this scheme is higher. The congestion ratio in RAFDA (Algorithm 1) is much lower than that of RRAHA-1 (Algorithm 2) and RRAHA-2 (Algorithm 3) because RAFDA checks all the reliable paths subject to the additional constraints, unlike RRAHA-1 and RRAHA-2 which use a subset of paths only.

Furthermore, the number of congested links in RRAHA-2 is less than that recorded in RRAHA-1 because RRAHA-2 selects a subset of the underutilized links first by using Eq. (8) and then checks the remaining constraints. RRAHA-1 selects some of the paths (e.g., 10%) with the highest reliable values by using Eq. (1), then the algorithm computes the best path after applying the other constraints, as described in Section IV-A3. Although RRAHA-1 finds the most reliable path for a given flow f, it may select a congested path, which is not desirable. RRAHA-1 and RRAHA-2 reduce the computation time in comparison with RAFDA, as unlike RAFDA they

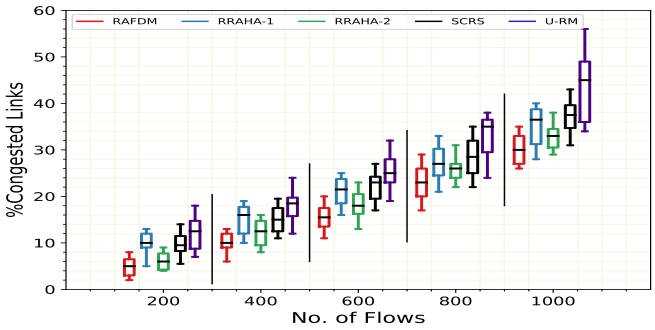


Fig. 4: %Congested Links Vs. No. of Flows

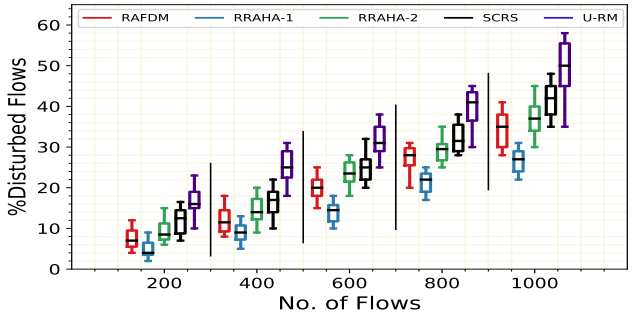


Fig. 5: %Disturbed Flows Vs. No. of Flows

do not check all the paths for the remaining constraints. RAFDA reduces the congestion in the network as compared to RRAHA-1, RRAHA-2, SCRS, and U-RM. In U-RM scheme, the percentage of congested links in network has reached about 60%, the percentage of congested links in SCRS is almost 43%, and in the proposed scheme RAFDA, the number of congested links is about 37%, in algorithm RRAHA-1 is around 44%, and in algorithm RRAHA-2, the number of congested links is almost 40% when the number of flows has increased to 1000.

Next, we analyze the number of disturbed flows, in case a link gets disconnected, using RAFDA, RRAHA-1, RRAHA-2, SCRS, and U-RM algorithms. The simulation results presented in Fig. 5 show that RRAHA-1 performed better than RAFDA, RRAHA-2, SCRS, and U-RM as the number of flows increases. The number of disturbed flows of IoT/edge devices in RRAHA-1 is less than those recorded for RAFDA, RRAHA-2, SCRS, and U-RM. The reason is as follows. RRAHA-1 first selects only a subset of the most reliable paths and then checks the remaining constraints. The main purpose of the proposed RRAHA-1 algorithm is to minimize the number of disturbed flows in case of link failure. The results show that RRAHA-1 performs better and has less variation in the number of disturbed flows. In RRAHA-2, the number of disturbed flows is larger (but not higher than U-RM) because it selects some of them (e.g., 10%) underutilized links, and then checks the reliability level and other constraints of the underutilized links. Therefore in RRAHA-2, the more reliable path might not be found, which leads to link failure and may disturb flows. From Fig. 4, we can analyze that congestion ratio in RRAHA-1 is higher than RAFDM and RRAHA-2. The reason is RRAHA-1 computes the most reliable path for a flow, it may select a congested path, which is not desirable.

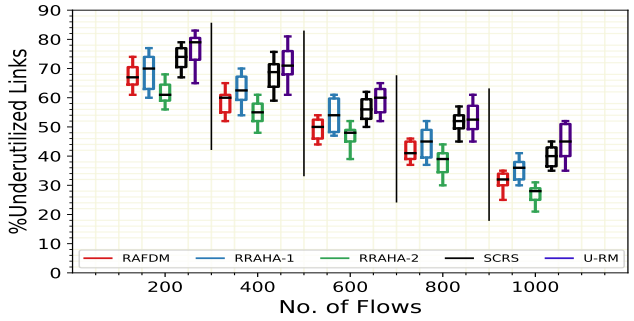


Fig. 6: %Underutilized links Vs. No. of Flows

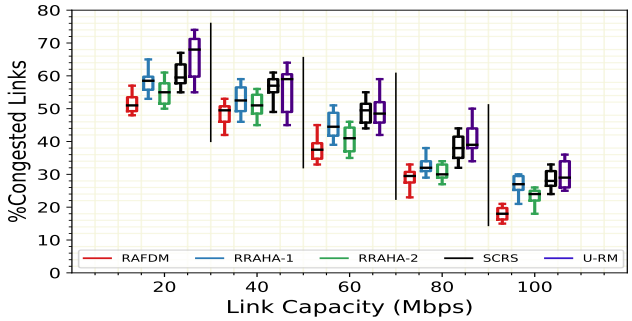


Fig. 7: %Congested Links Vs. Link Capacity (Mbps)

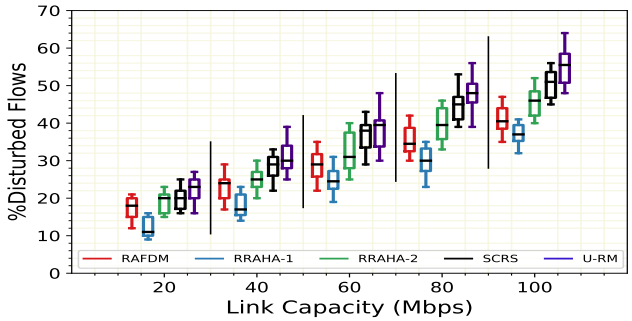


Fig. 8: %Disturbed Flows Vs. Link Capacity (Mbps)

However, the selection of the most reliable path minimizes the number of disturbed flows, as shown in Fig. 5. Unlike U-RM and SCRS, in RAFDA, RRAHA-1, and RRAHA-2 based on reliability level, flows are assigned to links, which decreases the disturbance of the flow in the network. In the proposed algorithms, when the controller receives a link down notification, then it re-routes the flows of the failed link based on the reliability level of links. Consequently, in RRAHA-1, the number of disturbed flows is almost 32%, in RAFDA, the number of disturbed flows is 43%, in RRAHA-2, the number of disturbed flows is almost to 47%, but in SCRS and U-RM, the variation of number of disturbed flows is very high, almost 49% and 58%, when the number of flows has increased to 1000, respectively.

In Fig. 6, the number of underutilized links is decreasing in all approaches (i.e., RAFDA, RRAHA-1, RRAHA-2, SCRS, and U-RM), as the number of flows increases. However, in the proposed approaches, the number of underutilized links is less than that of the SCRS and U-RM approaches. The main reason is that the proposed algorithms also consider the link utilization constraint (Eq. (8)) along with other constraints in the flow forwarding process, as described in Section IV-A3. In RRAHA-2, the number of underutilized links is much lower than that recorded for RAFDA and RRAHA-2, as its main goal was to maximize the link utilization ratio in the network. The number of underutilized links in RAFDA is less than that of RRAHA-1, as the latter selects some of the most reliable paths (e.g., 10%), and after that it selects the best path after applying the other constraints. RRAHA-2 first focuses on the most reliable paths then checks the utilization constraint. Therefore, in RRAHA-2, the underutilized links are almost 40%, in RRAHA-1, the underutilized links are about 32%, in RAFDA - almost 47%, in SCRS, the underutilized links are almost 47%, and in U-RM model, the percentage of disturbed flows nears 56%. The SCRS and U-RM algorithms forward the flows based on the delay and shortest path, respectively, and overlooks the reliability level and the link utilization of the path. Therefore, in the alternative schemes, the number of congested links, number of disturbed flows, and percentage of underutilized links are high compared to the three proposed algorithms in all scenarios.

2) Results When Varying Links' Capacity: In this scenario, the capacity of communication links is varied (i.e., 20Mbps, 40Mbps, 60Mbps, 80Mbps, and 100Mbps) and the total number of generated flows in the network is 800.

The goal is to investigate the performance of the five algorithms, RAFDA, RRAHA-1, RRAHA-2, SCRS, and U-RM under various communication link capacities. As the communication links capacity increases in all approaches, the number of congested links decreases, as shown in Fig. 7. However, the simulation results show that RAFDA performs better than RRAHA-2, RRAHA-1, SCRS, and U-RM. In RAFDA, congested links account for about 22%, in RRAHA-1, congested links are almost 32%; in RRAHA-2, congested links are almost 28%; in SCRS, congested links are almost 34%; and in the U-RM algorithm, congested links are almost 40% shown in Fig. 7.

Figure 8 presents the number of distributed flows, in case link get down, when employing RAFDA, RRAHA-1, RRAHA-2, SCRS, and U-RM algorithms in turn. In RRAHA-1, the number of disturbed flows is less than that of SCRS, U-RM, RRAHA-1, and RAFDA.

The results of real network traces show that RAFDA and RRAHA-2 decrease the congestion and increase the link utilization as compared to RRAHA-1. On the other hand, RRAHA-1 performs better than RAFDA and RRAHA-2 in case of a link failure scenario and reduces the number of disturbed flows in the network. The results also show that the proposed algorithms perform better than SCRS and U-RM and balance in terms of several flows per link. Furthermore, the simulation results show that SCRS and U-RM exhibits detrimental behavior when the traffic load (flows/link) increases. The proposed algorithms RAFDA, RRAHA-1, and RRAHA-2 are largely unaffected when the traffic load (flows/link) increases, while SCRS and U-RM scales poorly, as shown in Fig. 7, and Fig. 8.

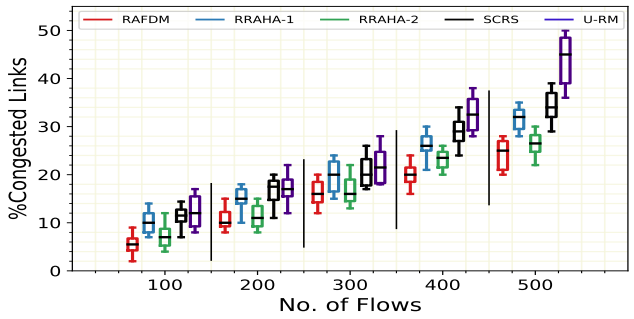


Fig. 9: %Congested Links Vs. No. of Flows

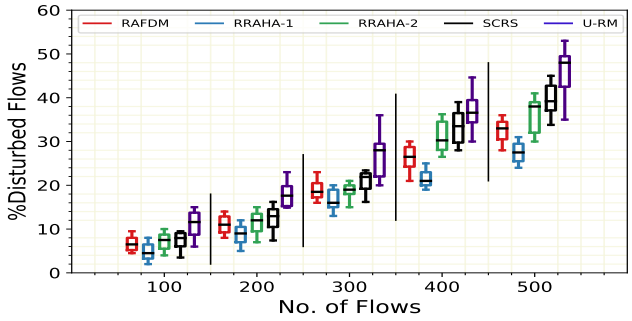


Fig. 10: %Disturbed Flows Vs. No. of Flows

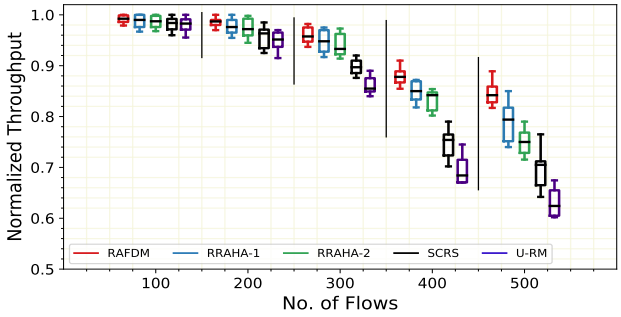


Fig. 11: Normalized Throughput Vs. No. of Flows

B. Mininet Results (Small Network)

A POX controller[4] is used in our simulations. We evaluate the proposed algorithms RAFDA, RRAHA-1, and RRAHA-2 under fixed topology size (10 switches), hosts are 50, and by varying the number of flows (i.e., $100, 200, 300, 400, 500$) to investigate the efficiency of the proposed algorithms. The flows are generated randomly with different data rates 10Mbps, 15Mbps, and 20Mbps. The capacity of a communication link is constant (100Mbps).

In the simulation, we consider the battery capacity of an IoT device set to *1000 J* [42], its transmitting power is *60 mW* [43], and and task size (average) is 450KB [44]. Consequently, bandwidth of wireless channel is *20 MHz* [45] and Noise Power is *-100 dB* [45]. When the IoT device ($u \in E_d$) offloads task (k) to a fog server, the energy required to transmit the data defines the IoT device's energy consumption. The SDN controller uses the southbound API like Provisioning of Wireless AP (CAPWAP) or Simple Network Management Protocol (SNMP) [44] to obtain information about data rate and transmission power. In this proposed work, $E_u^{Tx} = P_u^{Tx}\xi_u^{Tx}$ gives the required energy for task (k) offloading, where E_u^{Tx} represents the energy consumption of IoT device for transmitting the task, P_u^{Tx} is the transmitting power of IoT device (u) and ξ_u^{Tx} signifies the time taken to transmit the task to the associated device.

We examine the following performance metrics:

1) congested links in the network.
2) the number of disturbed flows in case of the links' failure.
3) normalized throughput of the network in case of the links' failure and normalized throughput is defined as the ratio of received packets over sent packets.
4) the end-to-end delay for data packets.
5) average energy consumption of the IoT devices.

We investigate the performance of the three proposed algorithms RAFDA, RRAHA-1, and RRAHA-2 with respect to a number of flows and compare them with the SCRS and U-RM algorithms. In Fig. 9, as the number of flows is increasing, the number of congested links also increases in all approaches. However, in the proposed algorithms, the number of congested links is less than that of the SCRS and U-RM approaches. It is mainly because the proposed algorithms distribute the numbers of flows on the links based on the links' reliability levels subject to additional constraints (i.e., the traffic load on the link, bandwidth allocation, and utilization ratio of the link) as discussed in Section IV-A3. Additionally, RAFDA performs better than RRAHA-1 and RRAHA-2 because RAFDA checked all possible paths between the source and destination for any given flow of f. RRAHA-2 algorithm selects some of the paths (e.g., 10%) with lower utilization ratio by using Eq. (8). After that using these 10% paths of lower utilization ratio then computes a feasible path after applying the other constraints. In RRAHA-1 algorithm congestion ratio is higher in comparison with those of RAFDA and RRAHA-2 because he algorithm finds (e.g., 10%) paths of maximum reliability then check the remaining constraints. Therefore, it can be said that the proposed algorithms reduce the congestion in the network as compared to SCRS and U-RM. In U-RM, congested links in network reached 50%, and variation in the results is high because U-RM computed the unreliable shortest path, unlike the proposed algorithms. In SCRS, there are fewer congested links than in U-RM and more than in the proposed approaches because SCRS distributes the flows based on QoS requirements like bandwidth and delay. In RAFDA the number of congested links is up to 28%, in RRAHA-1 the number of the congested links is almost 37%, in RRAHA-2 the number of congested links is almost 31%, in SCRS the number of congested links is almost 39%, when the number of flows increases to 500.

In order to analyze the behavior of the proposed algorithms RAFDA, RRAHA-1, and RRAHA-2 in a link failure scenario, we experimented with observing the number of flows disturbance by calculating the reliability level to all the links randomly in the network using Eq. (1). The simulation results show that our proposed algorithms perform better in comparison with SCRS and U-RM, as shown in Fig. 10.

[4]POX. http://www.noxrepo.org/pox/about-pox/

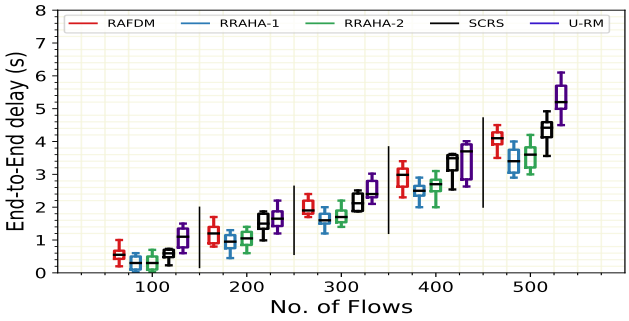


Fig. 12: End-to-End delay (s) Vs. No. of Flows

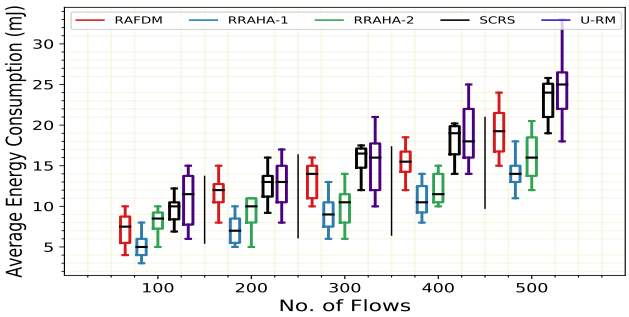


Fig. 13: Average Energy Consumption (mJ) Vs. No. of Flows

The number of disturbed flows in the proposed algorithms is much less than that in the SCRS and U-RM schemes. Unlike SCRS and U-RM, all proposed algorithms distribute the flows on the links based on reliability level and other constraints, as discussed in Section IV-A3. Thus, proposed algorithms assign a lower number of flows to less reliable links. In turn, this decreases the number of flows disturbance when a link gets disconnected. In the proposed algorithms, when the controller receives a link down notification, then it re-routes the flows of the failed link based on the reliability level of links. Additionally, RRAHA-1 performs better than RAFDA and RRAHA-2 algorithms because RRAHA-1 uses paths of high level of reliability, then the RRAHA-1 computes a feasible path after applying the other constraints for the flow f as discussed in Algorithm 2. Therefore, in RRAHA-1, the number of disturbed flows is much lower than those experienced in U-RM, SCRS, RRAHA-2, and RAFDA.

In order to comparatively assess the performance of the proposed algorithms RAFDA, RRAHA-1, RRAHA-2 and existing solutions (i.e., SCRS and U-RM) in terms of data transport, we illustrate the normalized throughput (i.e., received packets divided by the number of sent packets) in Fig. 11. The results show that the proposed algorithms have the highest normalized throughput, followed by SCRS and U-RM. When the number of flows increases in the network, the network bandwidth is no longer sufficient to accommodate all flows in the case of link failure, and, as expected, the throughput of all approaches decreases. However, the normalized throughput of the proposed approaches is higher than those recorded in SCRS and U-RMS cases. This is mainly because the proposed algorithms distribute the flows based on links' reliability level, as discussed in Section IV-A3, and therefore, the impact of link failure is lower than that of the existing solutions. Additionally, the normalized throughput of RAFDA is much higher than those of RRAHA-1 and RRAHA-2 because RAFDA also considers the additional constraints, along with the reliability of the paths.

The proposed algorithms in this paper are designed to provide a reliable path between IoT device and fog server pair in order to minimize the congestion and disturbed flows in the network rather than provide the shortest path. The number of congested links and disturbed flows in a network increases the end-to-end delay.

In the proposed work, we also examine the average energy consumption per IoT device in the SDN-based FC architecture. The average energy consumption increases as the number of flows increases in the network, as shown in Fig. 13. However, in the proposed algorithms, energy consumption is much lower thank that of the SCRS and U-RM model. The main reason is that the proposed algorithms compute a reliable path. Therefore, the number of retransmissions is lower and reducing retransmissions leads to energy saving. The results show that link failure and congestion increases the average energy consumption of IoT devices. When using RAFDA, RRAHA-1, and RRAHA-2, the number of congested links and disturbed flows is much lower than when SCRS and U-RM are employed. The number of disturbed flows in the proposed algorithms (RRAHA-1 and RRAHA-2) is lower as compared to that of U-RM. The SCRS increases the performance of smart city applications in terms of congestion, end-to-end delay, and energy consumption by considering link failure compared to the U-RM approach. The SCRS provides priority to time-critical applications during link failure. However, unlike the proposed algorithms, the SCRS does not distribute the time-critical flows based on the link's reliability level and additional path constraints, as discussed in Section IV-A3. Furthermore, Fig. 12 shows that the performance of the proposed algorithms is better than that of SCRS and U-RM in terms of end-to-end delay as the number of flows in the network increases due to the following two reasons. First, the number of congested links is lower in the proposed algorithms as compared to that of SCRS and U-RM, as shown in Fig. 12. Second, the proposed algorithms assign a flow f to the link based on the reliability level. Therefore, the chances of link failure are lower in proposed algorithms as compared to those in U-RM. However, RAFDA has a smaller number of congested links (see Fig. 9), but it has a greater average end-to-end delay (shown in Fig. 12) because the RAFDA algorithm searches for all alternative links, while the other two algorithms (RRAHA-1 and RRAHA-2) perform search for 10% alternative paths only. This is because when a link fails, then it takes some time for the switch to consult the controller, to re-compute the shortest path, to install the shortest path on the switches along the path, and redirect the flow on the newly computed shortest path. The results show that proposed algorithms RAFDA, RRAHA-1, and RRAHA-2 are consistent in improving the performance when compared to the SCRS and U-RM model in terms of number of congested links, number of disturbing flows, and end-to-end delay performance metrics.

C. Discussion

Time, message, and computation cost are essential components the proposed algorithm's overhead when computing a link reliability level in the SDN-based smart cities environment. The overall time complexity of Algorithm-1 is $\approx \mathcal{O}(F(kP) + mP)$ and Algorithm 2 and Algorithm 3 time complexity is $\approx \mathcal{O}(F \times P(P) + mP)$, as discussed in section IV-C. Additionally, message complexity is related to the total number of extra control messages exchanged to compute the final status of the underlying communication links. To obtain the current status of the links, the controller already exchanges control information with OpenFlow switches, so there is no need to exchange additional messages to obtain the status of the links. Consequently, the controller can monitor the traffic in the network, and check the quality of the links, so again there is no need to exchange any additional control messages. Computing link reliability, congested links, disturbed flows, and underutilized links and determining a link's final status are carried out in the controller without exchanging additional control messages. So, the message complexity (MC) of the intended algorithms is MC $\approx \mathcal{O}(1)$. Finally, the computation cost is associated with the operations needed for the link failure management and reliability-aware flows distribution calculation. The proposed solution's computation cost is $\sum_{j=1}^{y} s_{\times} \triangle$, where $\triangle$ is the computation cost of a link's status. Consequently, to check the additional constraints (m) of a link, the computation cost is $m \times \sum_{j=1}^{y} s_j \times \vartheta$, where ϑ is the unit computation cost. It is worth noticing that the computation cost of the proposed algorithms can be controlled (and kept low) easily by employing advanced programming techniques like multiple threads and parallel processing.

VI. Conclusions

In large scale SDN-based FCs for smart cities, efficient and reliable routing is needed to handle the traffic engineering problems, such as congestion avoidance, fault tolerance, and load balancing. Data traffic from an IoT device (an edge user) to the corresponding FS or (FS to FS) needs a reliable and congestion-free path to minimize the end-to-end delay. The exiting proposed approaches do not consider the link reliability level in computing the path from IoT/edge device to the fog server in SDN-based FC. In the paper, we propose a new algorithm, called RAFDA and some optimized algorithms RRAHA-1 and RRAHA-2, that distributes the numbers of flows on the links based on the links' reliability levels subject to additional constraints traffic load on the link, link capacity, and link utilization. The simulation results of the proposed algorithms clearly show how they reduce congestion, end-to-end delay, and number of flows disturbed when a link failure occurs in comparison to an existing approach. Future work considers extending the proposed algorithms by employing machine learning in a multi-controller scenario.

Acknowledgments

G.M. Muntean would like to acknowledge the Science Foundation Ireland (SFI) support for the Insight SFI Centre for Data Analytics (grant number 12/RC/2289_P2). The work was supported by the "National Natural Science Foundation of China" grant no. 61902052, "National Key Research and Development Plan" grant no. 2017YFC0821003-2, "Science and Technology Major Industrial Project of Liaoning Province" grant no. 2020JH1/10100013, "Dalian Science and Technology Innovation Fund" grants no. 2019J11CY004 and 2020JJ26GX037, and "Fundamental Research Funds for the Central Universities" grants no. DUT20ZD210 and DUT20TD107.

References

[1] C. Tang, X. Wei, C. Zhu, Y. Wang, and W. Jia, "Mobile vehicles as fog nodes for latency optimization in smart cities," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 9, pp. 9364–9375, 2020.

[2] M. Li, P. Si, and Y. Zhang, "Delay-tolerant data traffic to software-defined vehicular networks with mobile edge computing in smart city," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 10, pp. 9073–9086, 2018.

[3] C. Lin, G. Han, X. Qi, M. Guizani, and L. Shu, "A distributed mobile fog computing scheme for mobile delay-sensitive applications in sdn-enabled vehicular networks," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 5, pp. 5481–5493, 2020.

[4] Y. Liu, H. Zhang, K. Long, H. Zhou, and V. C. M. Leung, "Fog computing vehicular network resource management based on chemical reaction optimization," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 2, pp. 1770–1781, 2021.

[5] C. Lin, G. Han, X. Qi, M. Guizani, and L. Shu, "A distributed mobile fog computing scheme for mobile delay-sensitive applications in sdn-enabled vehicular networks," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 5, pp. 5481–5493, 2020.

[6] V. N. I. Cisco, "Global mobile data traffic forecast update, 2015–2020 white paper," *Document ID*, vol. 958959758, 2016.

[7] P. Zhang, C. Wang, G. S. Aujla, N. Kumar, and M. Guizani, "Iov scenario: Implementation of a bandwidth aware algorithm in wireless network communication mode," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 12, pp. 15 774–15 785, 2020.

[8] M. Ibrar, L. Wang, G.-M. Muntean, A. Akbar, N. Shah, and K. R. Malik, "PrePass-Flow: A machine learning based technique to minimize ACL policy violation due to links failure in hybrid SDN," *Computer Networks*, vol. 184, p. 107706, 2021.

[9] M. Ibrar, L. Wang, G. M. Muntean, J. Chen, N. Shah, and A. Akbar, "IHSF: An intelligent solution for improved performance of reliable and time-sensitive flows in hybrid SDN-based FC IoT systems," *IEEE Internet of Things Journal*, vol. 8, no. 5, pp. 3130–3142, 2021.

[10] A. Akbar, M. Ibrar, M. A. Jan, A. K. Bashir, and L. Wang, "SDN-enabled adaptive and reliable communication in IoT-Fog environment using machine learning and multiobjective optimization," *IEEE Internet of Things Journal*, vol. 8, no. 5, pp. 3057–3065, 2021.

[11] X. Jia, Y. Jiang, and J. Zhu, "Link fault protection and traffic engineering in hybrid SDN networks," in *IEEE INFOCOM WKSHPS*, 2018, pp. 853–858.

[12] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese *et al.*, "P4: Programming protocol-independent packet processors," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 87–95, 2014.

[13] S. Misra and N. Saha, "Detour: Dynamic task offloading in software-defined fog for iot applications," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 5, pp. 1159–1166, 2019.

[14] P. Thorat, S. Jeon, and H. Choo, "Enhanced local detouring mechanisms for rapid and lightweight failure recovery in OpenFlow networks," *Computer Communications*, vol. 108, pp. 78–93, 2017.

[15] O. Hohlfeld, J. Kempf, M. Reisslein, S. Schmid, and N. Shah, "Guest editorial scalability issues and solutions for software defined networks," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 12, pp. 2595–2602, 2018.

[16] V. R. Tadinada, "Software defined networking: Redefining the future of internet in IoT and cloud era," in *IEEE International Conference on Future Internet of Things and Cloud*, 2014, pp. 296–301.

[17] P. K. Sharma, M.-Y. Chen, and J. H. Park, "A software defined fog node based distributed blockchain cloud architecture for IoT," *IEEE Access*, vol. 6, pp. 115–124, 2017.

[18] L. M. Vaquero and L. Rodero-Merino, "Finding your way in the fog: Towards a comprehensive definition of fog computing," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 5, pp. 27–32, 2014.

[19] L. M. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner, "A break in the clouds: Towards a cloud definition," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 1, pp. 50–55, 2008.

[20] Y. Xiao and M. Krunz, "QoE and power efficiency tradeoff for fog computing networks with fog node cooperation," in *IEEE INFOCOM*, 2017, pp. 1–9.

[21] A. Akbar, M. Ibrar, M. A. Jan, A. K. Bashir, and L. Wang, "Sdn-enabled adaptive and reliable communication in iot-fog environment using machine learning and multiobjective optimization," *IEEE Internet of Things Journal*, vol. 8, no. 5, pp. 3057–3065, 2020.

[22] P. Thorat, S. Singh, A. Bhat, V. L. Narasimhan, and G. Jain, "SDN-enabled IoT: ensuring reliability in IoT networks through software defined networks," in *Towards Cognitive IoT Networks*. Springer, 2020, pp. 33–53.

[23] C.-Y. Chu, K. Xi, M. Luo, and H. J. Chao, "Congestion-aware single link failure recovery in hybrid SDN networks," in *IEEE INFOCOM*, 2015, pp. 1086–1094.

[24] V. Muthumanikandan and C. Valliyammai, "Link failure recovery using shortest path fast rerouting technique in SDN," *Wireless Personal Communications*, vol. 97, no. 2, pp. 2475–2495, 2017.

[25] P. Thorat, R. Challa, S. M. Raza, D. S. Kim, and H. Choo, "Proactive failure recovery scheme for data traffic in software defined networks," in *IEEE NetSoft Conference and Workshops (NetSoft)*, 2016, pp. 219–225.

[26] C. Cascone, D. Sanvito, L. Pollini, A. Capone, and B. Sanso, "Fast failure detection and recovery in SDN with stateful data plane," *International Journal of Network Management*, vol. 27, no. 2, p. e1957, 2017.

[27] Y.-D. Lin, H.-Y. Teng, C.-R. Hsu, C.-C. Liao, and Y.-C. Lai, "Fast failover and switchover for link failures and congestion in software defined networks," in *IEEE International Conference on Communications (ICC)*, 2016, pp. 1–6.

[28] N. Ahmed, A. Roy, A. Mondal, and S. Misra, "SDN-based link recovery scheme for large-scale Internet of Things," in *IEEE 22nd Int. Conference on High Performance Switching and Routing (HPSR)*, 2021, pp. 1–6.

[29] R. AlZoman and M. J. 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.

[30] N. Tcholtchev and I. Schieferdecker, "Sustainable and reliable information and communication technology for resilient smart cities," *Smart Cities*, vol. 4, no. 1, pp. 156–176, 2021.

[31] S. L. Aljohani and M. J. Alenazi, "MPResiSDN: Multipath resilient routing scheme for SDN-enabled smart cities networks," *Applied Sciences*, vol. 11, no. 4, p. 1900, 2021.

[32] H. Seddiqi and S. Babaie, "A new protection-based approach for link failure management of software-defined networks," *IEEE Transactions on Network Science and Engineering*, 2021.

[33] P. K. Singh, S. Sharma, S. K. Nandi, and S. Nandi, "Multipath tcp for v2i communication in sdn controlled small cell deployment of smart city," *Vehicular communications*, vol. 15, pp. 1–15, 2019.

[34] G. Bianchi, M. Bonola, A. Capone, and C. Cascone, "OpenState: programming platform-independent stateful OpenFlow applications inside the switch," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 2, pp. 44–51, 2014.

[35] P. Megyesi, A. Botta, G. Aceto, A. Pescapé, and S. Molnár, "Challenges and solution for measuring available bandwidth in software defined networks," *Computer Communications*, vol. 99, pp. 48–61, 2017.

[36] L. Suresh, J. Schulz-Zander, R. Merz, A. Feldmann, and T. Vazao, "Towards programmable enterprise WLANS with odin," in *workshop on Hot topics in software defined networks*, 2012, pp. 115–120.

[37] S. Misra, G. Xue, and D. Yang, "Polynomial time approximations for multi-path routing with bandwidth and delay constraints," in *IEEE INFOCOM*, 2009, pp. 558–566.

[38] R. Yu, G. Xue, and X. Zhang, "Application provisioning in fog computing-enabled internet-of-things: A network perspective," in *IEEE INFOCOM*, 2018, pp. 783–791.

[39] M. Yu, J. Rexford, M. J. Freedman, and J. Wang, "Scalable flow-based networking with DIFANE," *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 4, pp. 351–362, 2010.

[40] M. Hussain and N. Shah, "Automatic rule installation in case of policy change in software defined networks," *Telecommunication Systems*, vol. 68, no. 3, pp. 461–477, 2018.

[41] D. Levin, M. Canini, S. Schmid, F. Schaffert, and A. Feldmann, "Panopticon: Reaping the benefits of incremental SDN deployment in enterprise networks," in *Annual Technical Conf.*, 2014, pp. 333–345.

[42] M. Chen and Y. Hao, "Task offloading for mobile edge computing in software defined ultra-dense network," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 3, pp. 587–597, 2018.

[43] A. Yousefpour, G. Ishigaki, R. Gour, and J. P. Jue, "On reducing IoT service delay via fog offloading," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 998–1010, 2018.

[44] M. Ibrar, L. Wang, G.-M. Muntean, N. Shah, A. Akbar, and K. I. Qureshi, "SOSW: scalable and optimal nearsighted location selection for fog node deployment and routing in sdn-based wireless networks for iot systems," *Annals of Telecommunications*, pp. 1–11, 2021.

[45] T. X. Tran and D. Pompili, "Joint task offloading and resource allocation for multi-server mobile-edge computing networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 1, pp. 856–868, 2018.