Multimedia Tools and Applications (2023) 82:25585–25614
https://doi.org/10.1007/s11042-023-14349-6

# A fuzzy delay-bandwidth guaranteed routing algorithm for video conferencing services over SDN networks

Jianhu Gong[1] · Amin Rezaeipanah[2]

Received: 22 September 2021 / Revised: 25 March 2022 / Accepted: 2 January 2023 /
Published online: 23 January 2023

## Abstract
Video conferencing is one of the advanced technologies for users that allows online communication despite long distances. High quality communication and ongoing support for the principles of video conferencing service that can be achieved through Software-Defined Networking (SDN). SDN is a new architecture for computer networks that separates the control plane from the data plane to improve network resources and reduce operating costs. All routing decisions and control mechanisms are made by a device called a controller. Traffic engineering can be well implemented in SDN because the entire network topology is known to the controller. Considering SDN features, user requests can be dynamically routed according to current network status and Quality of Service (QoS) requirements. In general, the purpose of SDN routing algorithms is to maximize the acceptance rate of user requests by considering QoS requirements. In this literature, most routing studies to provide satisfactory video conferencing services have focused solely on bandwidth. Nevertheless, some studies have considered both delay and bandwidth constraints. In this paper, a Fuzzy Delay-Bandwidth Guaranteed Routing (FDBGR) algorithm is proposed that considers both delay and bandwidth constraints in routing. The proposed fuzzy system is based on rules that can postpone requests with high resource demands. Also, the purpose of the FDBGR is to distribute the network workload evenly for all requests, where this is done by maintaining the capacity to accept future requests. The combination of conventional routing algorithms and SDN provides remarkable improvements in mobility, scalability and the overall performance of the networks. Simulations are performed on different scenarios to evaluate the performance of the FDBGR compared to state-of-the-art methods. Besides, FDBGR has been compared with

✉ Jianhu Gong
gjhlunwen2021@163.com

Amin Rezaeipanah
amin.rezaeipanah@gmail.com

[1] School of Data and Computer Science, Guangdong Peizheng College, Guangzhou 510830, People's Republic of China

[2] Department of Computer Engineering, University of Rahjuyan Danesh Borazjan, Bushehr, Iran

a number of most related previous works such as H-MCOP, MH-MCOP, QoMRA, QROUTE and REDO based on criteria such as number of accepted requests, average path length, energy consumption, load balancing, and average delay. The simulation results clearly prove the superiority of the proposed algorithm with an average delay of 48 ms in different topologies for video conferencing applications.

**Keywords** Routing algorithm · SDN · Delay · Bandwidth · Video conferencing service · Fuzzy system

## 1 Introduction

Today, video constitutes a large amount of Internet traffic [4]. This is because video-based communication is becoming an inseparable part of our life today. Also, it has been the prevailing solution for companies, and organizations to have their staff telecommute with the worldwide outbreak of COVID-19. Therefore, although the demand for low delay and high-quality video is growing more and more, current video streaming solutions often suffer from impairments such as startup delays and/or playback stalls, which consequently deteriorate the Quality of Experience (QoE) of users [36]. Thus, a variety of approaches has been proposed and followed by networking academia and industry to meet stringent QoE requirements. Among all, Software-Defined Networking (SDN) has been successful and is the most promising approach in shaping next-generation networks.

SDN makes centralized routing decisions by employing a logically centralized controller that can adaptively configure the forwarding elements. The centralized controller forms and maintains an online global knowledge of the network status, i.e., all the switches, interconnecting links, and all monitoring metrics such as links utilization and delay. SDN is well suited to enable the optimization of flow routing [37]. This is because SDN brings an environment in which a network programmer would be able to configure all the forwarding elements in the network from a single point of control. This has been realized by separating the hardware (Data Plane) and the software (Control Plane) from each other. Data plane would remain on forwarding elements and merely deals with packets forwarding based on the forwarding entries in the forwarding element's flow table which are set by the controller(s). Control plane is the piece of software that is detached from forwarding elements and executed in a central controller. So, the network programmer would be able to program each forwarding element in a granular manner based on the required information for performing node-to-node packet forwarding and ensure that the packet reaches the destination.

On the contrary, in a traditional network, the data plane and control plane in all the forwarding elements are highly coupled and are not detachable. So, the network programmer is in charge of configuring every single forwarding element. That becomes more adverse once a failure in the routing policy occurs and it needs to be updated. It becomes nearly impossible in large-scale topologies to configure every single forwarding element. Also, the routing decision is made in a distributed manner and every device maintains and advertises its necessary routing information in order to travel the flow of packets between different sources and destinations. In this case, the knowledge of the topology of the entire network is not placed in a central point. Also, Traffic Engineering (TE) efforts, specifically meeting QoS constraints in routing decisions for various traffics, become more complicated. These challenges are addressed by SDN [21].

SDN paradigm, in a sense, is known as a successor of MPLS networks in terms of providing TE mechanisms. The critical issue in MPLS networks is to (re)-configure Label-Switched Paths (LSPs), particularly in cases of dynamic network conditions. Hence, the SDN controller with the global and online knowledge of the whole network condition would be able to rapidly respond to the network changes [28]. SDN controller decides on a path, being able to consider various metrics such as utilized delay and bandwidth, and dictates every single forwarding element in the network according to the current network condition to traverse a flow of packets from a given pair of source-destination. Figure 1 shows a general schema of SDN paradigm.

Traffic delay and throughput (or bandwidth) are two key QoS metrics and are highly interrelated in such a way that excessive bandwidth utilization of a path, which might end up congesting links, would cause excessive delays, and consequently no further data can flow [6, 18]. Although the higher bandwidth is desired, network delay also needs to be considered simultaneously [3, 7]. Therefore, finding the most proper path for incoming traffic requests in such a way that guarantees these two QoS metrics, namely delay and bandwidth, is highly desired especially in multimedia environment [5, 12]. The goal in deciding on a routing path is to minimize the probability of rejecting future requests by refraining from selecting critical links across the network. To do so, future bandwidth demands are met by reducing current routing delays. Thus, the process of deciding on a routing path requires guaranteeing not only the traffic bandwidth, but also the delay.

In this paper, we develop an SDN-based solution for establishing routing paths in a way that guarantee both delay and bandwidth of traffics. In this paper, we assume that flow packets, requiring path establishment process, are coming one in a time, and subsequent requests are not known. However, information about the residual capacity (bandwidth) of links are updated constantly and in an online manner, which is realized by exchanging, as an example, OpenFlow statistics messages exchanging between SDN controller and forwarding elements [26]. Then, this information is collected within SDN controller and utilized during further path selection process. To more scrutiny our proposed routing algorithm, we defer routing requests for delay- and bandwidth-intensive traffics by developing a fuzzy filtering system on the SDN controller. Our approach mainly aims at jointly minimizing delay and maximizing accepting incoming flows following the approach in [31]. Our approach also desires a uniform distribution of traffic loads across the network.

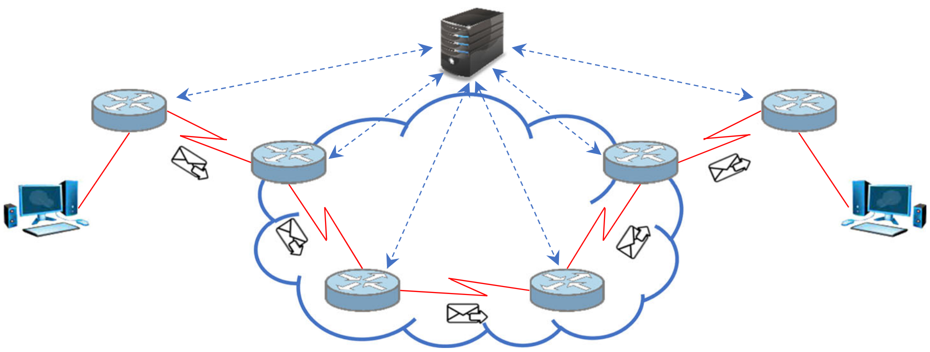


**Fig. 1** The general schema of SDN paradigm

The main contribution of this paper is as follows:

- Designing an SDN-based fuzzy delay and bandwidth guaranteed routing algorithm
- Jointly considering delay and bandwidth constraints in the process of routing in SDN-based networks
- Developing a predicting system applying fuzzy rules to postpone requests with high resource demands

The rest of this paper is organized as follows: Section 2 discusses a literature review. Section 3 describes the general idea of the routing algorithm in the SDN controller. Our proposed routing algorithm is elaborated on Section 4. The experimental results and comparisons are reported in Section 5. Finally, Section 6 is dedicated to conclusions and future directions.

## 2 Literature review

The QoS-based routing is a kind of routing mechanism in which the flow path is determined based on available resources and QoS requirements [16]. So far, various QoS-based routing algorithms have been proposed, regardless of their context and technology [14, 22, 27, 33]. A high performance optimal dynamic routing algorithm with QoS guarantee has been developed in [13] for communication networks. They developed an Optimal Dynamic Unicast Multi-channel QoS Routing algorithm (ODUMR) based on Constrained Based Routing (CBR) and label switching technology. ODUMR is a novel dynamic framework that converts a static routing algorithm to dynamic routing algorithms.

The information of input-output pairs is exploited in Minimum Interference Routing Algorithm (MIRA) to determine feasible paths [15]. Basically, MIRA attempts to determine the paths with minimum interference with any path that is potentially critical for satisfying subsequent demands. The main shortage of MIRA is that it only concentrates on guaranteeing bandwidth and is unable to guarantee the constraints like delay and hop counts. MIRA also selects longer paths to exclude critical links. The Minimum Delay and Maximum Flow Algorithm (MDMF) algorithm prevent causing network bottlenecks by distributing the load across the network [31]. MDMF mirrors the remaining bandwidth of links in their weight and dynamically adjusting the weights leads to selecting the shortest path.

In another study, OGUL et al. have investigated the quality characteristics of MPLS-TE/FRR networks with additional directions of traffic transmission lines [25]. They applied the nodal tensor method to solve the problem of searching for the quality characteristics in routing across the MPLS-TE/FRR networks. The quality parameters of the network and the additional transmission traffic criterion are extracted from the minimum packet delivery time to guarantee balanced load of the network. Mesbahi and Dahmouni designed an innovative efficient multipath routing algorithm on the MPLS networks [24]. The goal of this algorithm is to guarantee QoS for the delay and the delay variations (jitter) through formulating multiple nonlinear objective functions to guarantee link utilization as well as traffic flow routing.

Rischke, Justus, et al. developed a Reinforcement Learning (RL) solution for direct flow routing in SDN and called their solution QR-SDN [29]. They subtly modeled RL states, actions and rewards on the network and run the RL agent within the SDN controller. They applied a classic tabular RL approach that directly represents the routing paths of individual flows in a state-action pairs space. This approach also enables multiple routing paths between a pair of given sources (ingress) and destination (egress) forwarding element. They only consider flow latency as their routing decision metric. Also, a limitation of this work is its scalability, meaning that the state-actions table size would exponentially grow with increasing the number of network nodes.

In [8], a Deep Reinforcement Learning (DRL) approach for routing in SDN has been developed. Since a very narrow information are employed on the process of making routing decisions in traditional routing protocols, leading to lack of (or at least slow) adapting to variations in traffic and involving QoS metrics during routing path selection, the authors proposed the Deep Reinforcement Learning and Software-Defined Networking Intelligent Routing (DRSIR) mechanism. DRL is employed because classic RL-based solutions usually suffer from storage and time challenges when dealing with large action and state spaces. DRSIR considers path-state metrics to provide proactive, efficient, resilient, and intelligent routing in such a way that be able to adapt to dynamic traffic changes.

In [1], a reinforcement learning-based dynamic routing algorithm selection method for SDN has been proposed. REDO takes advantage of conventional routing algorithm capabilities to be applied on the flow routing within a SDN environment and in a centralized manner. REDO defines and attempts to maximize the RL agent's reward function based on a set of centralized routing algorithms (MHA, WSP, SWP, MIRA) to dynamically select the most appropriate routing algorithm. REDO solution evaluation experiments have been implemented on Mininet emulator, Open vSwitch switches, Floodlight SDN controller. The main shortcoming is that Large-scale topologies with complex scenarios have not been addressed in this study.

Barakabitze et al. developed a novel QoE-Centric SDN-based multipath routing algorithm for multimedia services over 5G networks, namely QoMRA [2]. QoMRA attempts to improve user's QoE and network resources utilization by employing Multi-Path Transmission Control Protocol (MPTCP) and Segment Routing (SR) in SDN-based networks with concentration on delivering multimedia services over 5G networks. They implemented their QoE-centric Multipath Routing Algorithm (QoMRA) on an SDN source routing platform using POX SDN controller, which is an old controller that is not supporting newer versions of OpenFlow, over Mininet emulator and carried out some experiments on Dynamic Adaptive video Steaming over HTTP (DASH) applications over various network conditions.

Varyani et al. [34] proposed a QoS-based routing scheme for Software-Defined Overlay Networks called QROUTE. QROUTE attempts to satisfy multiple QoS constraints in software-defined overlay networks. Overlay link QoS characteristics are changing fast; so, current routing algorithms cannot adapt to the fast-changing overlay link QoS characteristics due to their high route computation time. QROUTE also saves storage on forwarding elements by reduces the forwarding entries in the data plane. QROUTE has been evaluated on a P4-BMv2 switches testbed controlled by the ONOS SDN controller using P4Runtime protocol.

Korkmaz and Krunz proposed a multi-constrained optimal path selection algorithm to simultaneously meet multiple QoS metrics while maintaining high utilization of network resources [17]. They proposed a heuristic algorithm for the most general form of the form of QoS-based routing. However, it is yet unable to properly perform in the presence of inaccurate routing state information and suffer from high computational complexity to operate online. Similarly, Feng et al. proposed both heuristic and exact algorithms for QoS routing with involving multiple constraints [11]. In fact, they provided an extension on [17] by significantly improving its performance through executing the Dijkstra routing algorithm a few more times. However, this also slightly increased its time complexity.

Table 1 summarizes the study of the papers discussed. The first column denotes the year of publication and authors. The second and third columns indicate the model's name and the main objective of the paper, respectively. The fourth column describes the main idea of the routing algorithm, and the final column indicates the drawbacks of the algorithm.

## 3 Background

Assume $(s, d)$ is a tuple within LSP over SDN that shows the source and destination forwarding elements. We assume that path setup requests enter the network one at a time, and there is no information regarding the upcoming requests arrival time in advance. SDN controller utilizes information about the available bandwidth capacity of links during performing routing algorithms. It also applies other monitoring metrics including delays (queuing and propagation delays) to meet the delay constraints. In this paper, we modeled the delay by employing the Latency-Rate servers (LR-servers) model [32]. Hence, following this section, we briefly describe the interference, LR-servers, minimum delay, and maximum flow concepts before going through the main idea of our routing algorithm.

Our SDN controller, with thank of its global view, attempts to refrain from selecting the paths for a given source and destination pair of forwarding elements $(s, d)$, with higher interference with other already selected paths. In this paper, we have not only considered bandwidth but also involved delay for the process of interference avoidance. In other words, the minimum interference in the selected path between $s$ and $d$ forwarding elements if we reach maximum number of the minimum potential between other source and destination pairs of forwarding elements. Accordingly, before defining the potential of a link and a path, two concepts maximum flow $\theta(s, d)$ and minimum delay $d(s, d)$ between a pair of source and destination forwarding elements $(s, d)$ need more clarification.

### 3.1 Delay model

In SDN network, two kinds of delays are incurred for each flows including the data plane delay and control plane delay. The former is the time elapsed in the data plane to forward packets of a flow on links. The latter is also known as the flow installation delay which is the time elapsed from sending a flow installation request to the controller to responding to this message and having flows installed on the forwarding element.

*Data plane delay:* We applied the model presented by Stiliadis and Varma for formulating the data plane delay [32]. They demonstrated that many scheduling algorithms follow the model they presented called LR-servers model. They presented the general formulation of maximum delay of paths within the networks. In more details, we assume forwarding elements as LR-servers in the network, and the maximum data plane delay of the given path is equivalent to $d_{DP}$ and is calculated by Eq. (1).

$$d_{DP} = \frac{t-R}{t-r} \cdot \frac{b}{r} + \sum_{(i,j)\in P} \left( \frac{M}{R} + \frac{M_{ij}^{m}}{C_{ij}} + prop_{ij} \right) \tag{1}$$

Where, $r$ and $t$ represent the traffic rate and the maximum traffic rate respectively. $b$ indicates the traffic burstiness rate, and $M$ symbolizes the maximum length of packets in the flow and $M_{ij}^{m}$ is the maximum length of packets in the flow, which passes through the link that

**Table 1** Study of various papers discussed

| Reference | Model | Main objective | Idea | Drawbacks |
|---|---|---|---|---|
| [31] | MDMF | minimum delay and maximum flow routing in MPLS networks | Reducing delay by increasing bandwidth in each iteration to meet subsequent bandwidth demands. | Increased complexity due to iteration process in guaranteeing end-to-end path delay |
| [13] | ODUMR | Optimal Dynamic Unicast Multichannel QoS Routing algorithm | a new dynamic framework which transforms static routing algorithms to dynamic routing algorithms to deal with failing nodes and links | Unable to properly deal with flows with loops |
| [15] | MIRA | MPLS routing decisions with minimal interference | Decide on upcoming LSP requests in such a way that have the minimum interference with other LSPs to meet bandwidth demands | Only considering bandwidth may lead to select longer paths for refraining from selecting critical links, while other metrics such as delay and hop counts have been ignored |
| [25] | MPLS-TE/ FRR | MPLS traffic engineering with fast reroute | Solving the problem of considering QoS characteristics in routing across the MPLS-TE / FRR networks | Considering additional directions of traffic transmission lines |
| [24] | – | multi-path routing scheme in an MLPS network | QoS (delay and jitter) guarantee for real-time and interactive traffic flows | Only considering delay and jitter in their routing decision, while links utilization is also important |
| [29] | QR-SDN | SDN-based dynamic flow routing using RL | Modeling the SDN environment into RL states, actions, and rewards and propose a flow routing solution to minimize latency | Only considering latency and exponential growth of state-action table with increasing the number of network nodes |
| [8] | DRSIR | Dynamic Routing with considering QoS metrics | Applying DRL to perform routing in SDN in a way that minimizes latency and loss ratio as well as prioritizing paths with large available bandwidth for making intelligent routing decisions | Lack of considering multi-path and traffic-aware routing |
| [1] | REDO | A reinforcement learning-based dy-namic routing algo-rithm selection method for SDN. | REDO decides on the conventional routing algorithm to be applied on the traffic flows within a SDN environment | Large-scale topologies with complex scenarios have not been addressed in this study |
| [2] | QoMRA | A QoE-based Multipath Routing Approach for Multimedia Services over SDN-Based 5G Networks | Improving user's QoE and network resources utilization using MPTCP and SR in SDN-based networks focusing on de-livering multimedia ser-vices over 5G networks | Implemented on POX SDN controller, which is an old SDN controller that does not support newer versions of OpenFlow |

**Table 1** (continued)

| Reference | Model | Main objective | Idea | Drawbacks |
|---|---|---|---|---|
| [34] | QROUTE | A QoS-based Routing Scheme for Software-Defined Overlay Networks | Satisfying multiple fast-changing QoS constraints in software-defined overlay networks | Single point of failure challenge because of employing a single-controller architecture |
| [17] | H_MCOP | Multi-constrained Optimal Path Selection | An efficient heuristic algorithm for the most general form of QoS-based routing. | Unable to properly perform in the presence of inaccurate routing state information and yet suffer from high computational complexity to operate online |
| [11] | MH_MCOP | Heuristic and Exact Algorithms for QoS routing | Improving the performance of [17] by executing the Dijkstra routing algorithm a few more times | This approach suffers from high time complexity |

interconnects the forwarding elements $i$ and $j$. Analogously, $C_{ij}$ and $prop_{ij}$ in turn identify the bandwidth capacity and the propagation delay for the link that interconnects the forwarding elements $i$ and $j$. Lastly, $R$ determines the minimum bandwidth allocated to the flow over the path.

According to Eq. (1), there is a high correlation between the maximum delay and the allocated bandwidth to each link in such a way that the more the allocated bandwidth to each link of the requested path increases, the path delay decreases.

*Control plane delay:* We used the delay model presented in [9] for formulating the control plane (or flow installation) delay. The flow installation delay comprises three distinct components: (a) propagation delay between switch and controller, (b) processing delay of the SDN controller, and, (c) inter-controller synchronization delay. Flow installation delay is calculated according to the Eq. (2). Here, the first term indicates the propagation delay between switch $u$ and its controller $v$, considering the shortest path between them. The second term shows the processing delay within the controller which are modeled as M/M/1 queues [9]. The last one, is the inter-controller synchronization delay which is expressed as the weight of the Steiner tree connecting all controllers in multi-controller use cases.

$$d_{CP} = \frac{\sum_{v\in V} d_u^v\, z_u^v(t)}{c} + \frac{1}{\mu - \sum_{u\in V} \lambda_u\, z_u^v(t)} + \frac{\sum_{e\in\varepsilon} w_e\, q_e(t)}{c} \tag{2}$$

Where, $V$ indicates the set of forwarding elements, $d_u^v$ is the shortest path between nodes $u$ and $v$, $z_u^v(t)$ is a Boolean to determine the mapping between the controller and the forwarding element at time $t$, $\mu$ is the service rate of the controller, and $\lambda_u$ indicates the average control load generated by switch $u$. Also, $w_e$ and $q_e(t)$ are weight of link and a Boolean variable to determine the link $e$ of the Steiner tree of controllers at time $t$ respectively. Plus, $c = 2 \times 10^8$m/s represents the signal propagation speed in fiber.

*End-to-end delay:* Finally, the overall delay would be the sum of the data plane and the control plane delays and form Eq. (3).

$$d_{(s,d)} = d_{DP} + d_{CP} \tag{3}$$

## 3.2 Minimum delay

The minimum delay of a given pair of source and destination forwarding elements is described by the minimum delay that the traffic flow faces between forwarding elements. In simpler words, the minimum delay of a given source and destination pair is the lower bound for the overall delay of links participating in the routing path. Our algorithm minimizes interference by minimizing the delay of the newly routed path and the paths for other previously established paths. Minimizing the weighted sum of the minimum delays is shown on Eq. (4).

$$min_{\forall paths} \sum_{(s,d) \in P} \lambda_{sd} d(s,d) \tag{4}$$

Where, $\lambda_{sd}$ indicates the weight of the pair of source and destination $(s,\ d)$ and are often configured by the network administrator.

## 3.3 Maximum flow

Maximum flow (or maxflow) remarks the maximum bandwidth request among paths between a pair of source and destination forwarding elements that SDN controller accepts by calculating the remaining bandwidth capacity in that graph [23]. Our algorithm minimizes maxflow interference between newly arrived path routing requests from a given source to a given destination forwarding element and other previously routed paths. We render this as a MAX-MIN-MAX problem and to our best of knowledge is well tailored to the problem of maximizing minimum maxflow. Therefore, our SDN controller selects the new path which maximizes the weighted sum of maximum flows of all source and destination pairs according to the equation presented in Eq. (5).

$$max_{\forall paths} \sum_{(s,d) \in P} \gamma_{sd} \theta(s,d) \tag{5}$$

Where, $\gamma_{sd}$ indicates the weight of the pair of source and destination $(s,\ d)$. Practically, these weights are often configured by the network administrator.

## 4 Proposed algorithm

The proposed FDBGR algorithm has been developed to improve routing to provide video conferencing services on SDN. One of the main challenges in SDN is to provide routing requests with guaranteed delay and bandwidth so that routing is done for future requests with minimal interference. The FDBGR seeks to increase the number of routed requests by temporarily postponing high-resource requests (for example, low maximum delay and high bandwidth). Routed requests are considered as accepted requests. In the literature, most routing studies focus only on bandwidth, while ignoring delay. Given the importance of both delay and bandwidth factors in providing video conferencing services, FDBGR introduces a fuzzy delay-bandwidth guaranteed routing algorithm to improve the SDN-based routing process. Delays in a route include propagation delays and queue delays. The propagation delay is the physical property of each link in the path and its value is assumed to be constant. Here, the LR-servers model is used to calculate the maximum delay (sum of propagation delay and queue delay) for each link. In addition, requests are entered online and future requests are unknown, so FDBGR routers dynamically. The flowchart of the proposed routing algorithm is presented in Fig. 2.

The routing process is performed on the network graph topology undirected $G(V, E, C, D, P)$, where $V$ is the network nodes, $E$ is the network links, $C$ is the residual bandwidth of the links, $D$ is the propagation delay of physical links, and $P$ is the potential ingress - egress pairs. Let $N$ and $M$ be the number of nodes and the number of links in the $G$ network, respectively. Also, $e_{ij} \in E$ is the link between nodes $i$ and $j$, $c_{ij} \in C$ is the residual bandwidth in $e_{ij}$, and $d_{ij} \in$

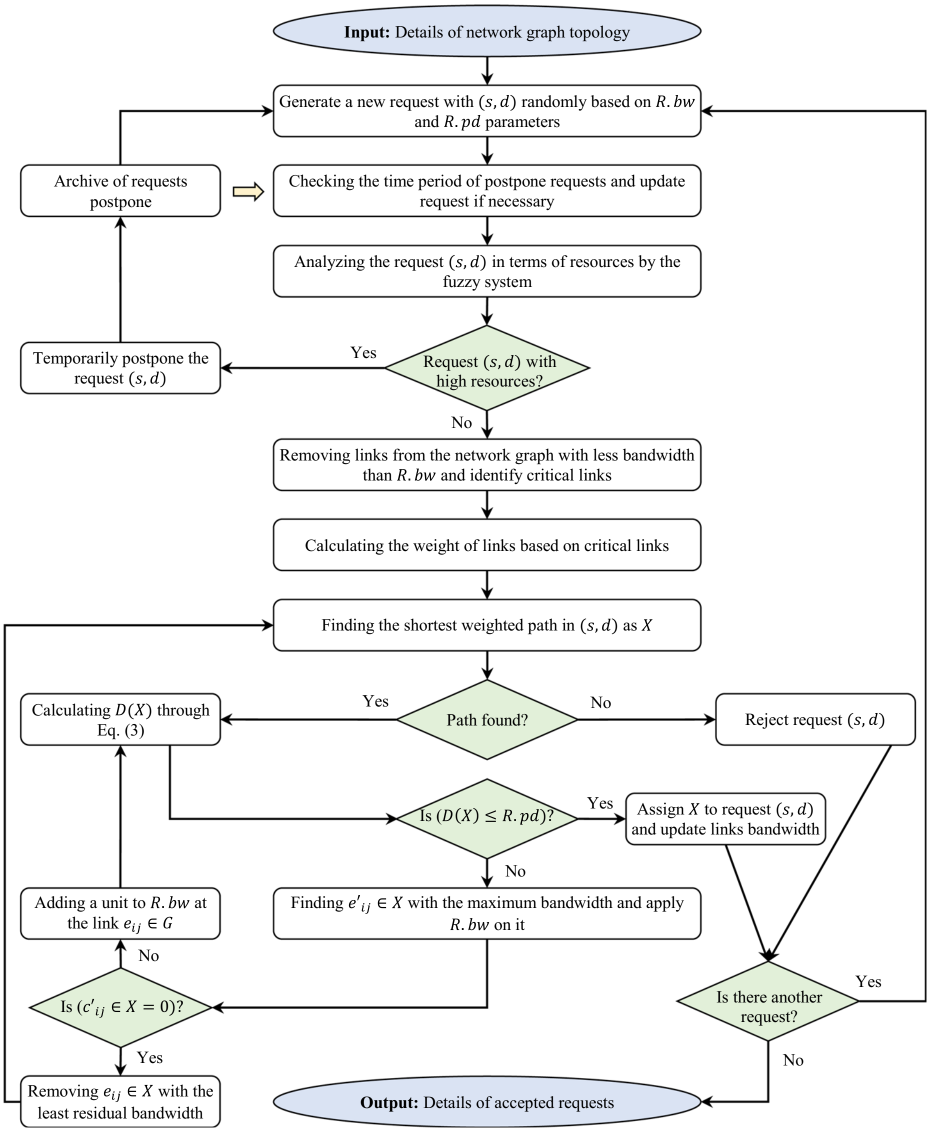

**Input:** Details of network graph topology
Generate a new request with $(s, d)$ randomly based on $R.bw$ and $R.pd$ parameters
Archive of requests postpone
Checking the time period of postpone requests and update request if necessary
Analyzing the request $(s, d)$ in terms of resources by the fuzzy system
Temporarily postpone the request $(s, d)$
Yes
Request $(s, d)$ with high resources?
No
Removing links from the network graph with less bandwidth than $R.bw$ and identify critical links
Calculating the weight of links based on critical links
Finding the shortest weighted path in $(s, d)$ as $X$
Calculating $D(X)$ through Eq. (3)
Yes
Path found?
No
Reject request $(s, d)$
Is $(D(X) \leq R.pd)$?
Yes
Assign $X$ to request $(s, d)$ and update links bandwidth
No
Adding a unit to $R.bw$ at the link $e_{ij} \in G$
Finding $e'_{ij} \in X$ with the maximum bandwidth and apply $R.bw$ on it
No
Is $(c'_{ij} \in X = 0)$?
Yes
Removing $e_{ij} \in X$ with the least residual bandwidth
Is there another request?
Yes
No
**Output:** Details of accepted requests

**Fig. 2** Flowchart of the proposed routing algorithm

$D$ is the propagation delay in $e_{ij}$. In addition, $(s, d) \in P$ is an ingress - egress pair with source $s$ and destination, and the path setup request for a given traffic flow. Requests are entered online and sequentially, where we generate them randomly. Each request has details $(s, d, T, R)$ that must be set up for ingress - egress pairs. Here, $T$ and $R$ include the details of network traffic and QoS, respectively. Let the network traffic details be $(l, r, t, b)$, where $l$ is the maximum packet length, $r$ is the requested rate, $t$ is the maximum rate, and $b$ is the burstiness rate. In addition, QoS details are described as $(bw, pd)$, where $bw$ is the minimum bandwidth required and $pd$ is the maximum propagation delay required.

The proposed algorithm is equipped with a fuzzy system based on rules that temporarily postpone routing requests with high resources. Temporarily postponing high-resource requests by the FDBGR can distribute network workload across all links, as well as conserve network resources for future requests. Postponed requests are archived and restarted within a specified time period. The time period is determined based on the number of specific requests. Here, after completing each $\varphi$ request from the all-existing $K$ request, the routing process is applied to the postponed requests. This can reduce the waiting time for postponed requests.

The routing process in FDBGR is performed by identifying critical links in terms of minimum delay and maximum flow. We identify critical links using a heuristic approach. By assigning weight to network links based on critical links, FDBGR can provide routing for future requests with little interference. Paths are searched by applying the Dijkstra algorithm [35] to the weighted network graph. Let $X$ be the path found for a request, $D(X)$ be the propagation delay on path $X$, $e'_{ij} \in X$ be the link with the largest remaining bandwidth of $X$, and $c'_{ij} \in X$ be the value of the residual bandwidth associated with $e'_{ij}$ of $X$. When $D(X)$ is less than the minimum required maximum delay (i.e., $R.\ pd$), then the request is routed to $X$. Otherwise, the minimum required bandwidth (i.e., $R.\ bw$) is reduced from $c'_{ij} \in X$, as shown in Eq. (6).

$$c_{ij} = c'_{ij} - R.bw \tag{6}$$

After updating the bandwidth and seeing the bandwidth zero in $e'_{ij} \in X$, a unit is added to the bandwidth required by the current request in $e_{ij} \in G$. Accordingly, $D(X)$ can be recalculated to satisfy the propagation delay. However, if $c'_{ij} \in X$ has bandwidth capacity, then the $e_{ij} \in X$ link with the least residual bandwidth is removed from the network (i.e., bottleneck link) and a new path is searched in the remaining subgraph. Here, all links are removed when the bandwidth of several links is the same. This process is repeated until a possible path is reached or the request is rejected. After setting the path for the request, the requested resources are reserved and the status of the links is updated. Based on the proposed routing process, FDBGR guarantees maximum delay in routing in addition to bandwidth.

## 4.1 Fuzzy filtering system

Given the importance of both delay and bandwidth factors in providing video conferencing services, FDBGR proposes a fuzzy delay-bandwidth guaranteed routing algorithm to improve the SDN-based routing process. FDBGR is equipped with a rule-based fuzzy filtering system that is applied with the aim of conserving network resources for future requests. The fuzzy filtering system temporarily postpones high resource routing requests to load balancing network on all links. Postponed requests are archived and re-launched at regular time periods to reduce waiting times. Each request contains two factors for each ingress - egress pair:

maximum delay and required bandwidth. In order to improve QoS, FDBGR temporarily postpones the routing process by a fuzzy filtering system for high-resource requests (i.e., lower maximum delay and higher bandwidth). The fuzzy system can decide whether to postpone the request by analyzing the maximum delay (as $P_d$) and bandwidth (as $P_b$) parameters of each input request. In addition to these parameters, the fuzzy system also considers the number of active requests in the queue for routing (as $P_q$). The proposed fuzzy system structure is presented in Fig. 3.

The reasoning for estimating the input parameters in the proposed fuzzy system is the same and is based on a fuzzy trapezoidal set with low, middle and high terms, as shown in Fig. 4. As illustrated, fuzzification parameters are performed based on threshold values $a$, $b$, $c$ and $d$, where these values are determined by an expert.

According to the defined fuzzy set, the degree of membership of the parameters is determined for each term. Let $P_*$ be an input parameter and $\mu_k(P_*)$ denote the degree of membership of this parameter relative to the term $k$ of the fuzzy set. The reasoning process for $\mu_k(P_*)$ is performed according to the Eq. (7) – Eq. (9).

$$\mu_{low}(P_*) = \begin{cases} 0 & P_* \geq b \\ \frac{b-P_*}{b-a} & a \leq P_* < b \\ 1 & P_* < a \end{cases} \tag{7}$$

$$\mu_{middle}(P_*) = \begin{cases} 0 & (P_* < a) or (P_* \geq d) \\ \frac{P_*-b}{b-a} & a \leq P_* < b \\ \frac{d-P_*}{d-c} & c \leq P_* < d \\ 1 & b \leq P_* < c \end{cases} \tag{8}$$

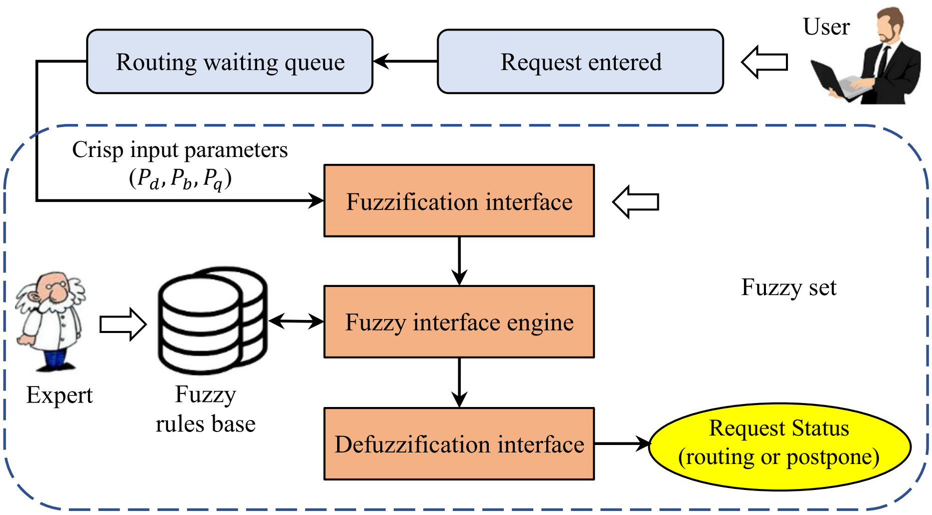
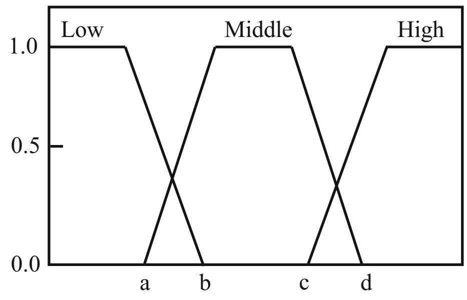


**Fig. 3** The structure of the proposed fuzzy filtering system

**Fig. 4** Fuzzy trapezoidal set for fuzzification



$$\mu_{high}(P_*) = \begin{cases} 0 & P_* \leq c \\ \frac{P_* - c}{d - c} & c \leq P_* < d \\ 1 & P_* > d \end{cases} \tag{9}$$

In addition, the defuzzification of the parameters is performed via the $D_{centroid}$ function, as shown in Eq. (10). Here, $x$ is assumed to be the output parameter.

$$D_{Centroid}(x) = \frac{\int_0^\omega x\ \mu(x)\ dx}{\int_0^\omega \mu(x)\ dx} \tag{10}$$

Fuzzy rules in the proposed system are defined as Eq. (11) in the knowledge base, where we formulate fuzzy rules based on the experience of an expert. The fuzzy system can decide whether to routing the request or postpone the request according to defined rules. Table 2 shows the defined fuzzy rules.

$$If\ (p_d\ \ is\ \ T)\ and\ (p_b\ \ is\ \ T)\ and\ \left(p_q\ \ is\ \ T\right)\quad Then\quad Y\ \ is\ \ G \tag{11}$$

Where, $T$ is a fuzzy term of the fuzzy set, $Y$ is the output of the rule, and $G$ is a fuzzy term of the output fuzzy set.

**Table 2** Defined rules for fuzzy filtering system

| No. | System input | | | System output |
|---|---|---|---|---|
| | $p_d$ | $p_b$ | $p_q$ | |
| 1 | Low | Low v Middle | Low | Postpone |
| 2 | Low | Low | Middle v High | Routing |
| 3 | Low | Middle | Middle v High | Routing |
| 4 | Low | High | Low v Middle | Postpone |
| 5 | Low | High | High | Routing |
| 6 | Middle | Low v Middle | – | Routing |
| 7 | Middle | High | Low | Postpone |
| 8 | Middle | High | Middle v High | Routing |
| 9 | High | Low | – | Routing |
| 10 | High | Middle v High | Low | Postpone |
| 11 | High | Middle | Middle v High | Routing |
| 12 | High | High | Middle v High | Routing |

The inference engine from the knowledge base deduces the winning rule and the output is determined according to this rule. In this paper, Mamdani fuzzy inference system with multiplication operator is used to infer the winning rule. The degree of compatibility of the input parameters $X = \{P_d, P_b, P_q\}$ with the premise of the $k$-th rule is calculated based on Eq. (12). According to the winning rule is selected based on the degree of confidence factor for decision-making.

$$\mu_k(X) = \prod_{T\in\{L,M,H\}} \mu_{T(x)} \quad \forall x\in X \tag{12}$$

Where, $L$, $M$ and $H$ refer to the low, middle and high fuzzy sets, respectively.

## 4.2 Calculating the links weight

FDBGR performs the routing process of requests in a weighted network by searching for the shortest weighted paths using the Dijkstra algorithm. We calculate the weight of network links based on critical links. Critical links are calculated based on minimum delay and maximum flow. Calculating weights based on critical links can assign paths with minimum delay and maximum flow to requests and minimize interference for future requests. Here, the weight of the links is calculated by simultaneously considering the minimum delay from Eq. (4) and the maximum flow from Eq. (5) as a multi-objective optimization problem. Therefore, the weight of the links should be adjusted with the purpose of maximizing Eq. (13).

$$\max\, v_d\left(-\sum_{(s,d)\in LSP} \lambda_{sd} d(s,d)\right) + \max\, \mu_f \sum_{(s,d)\in LSP} \gamma_{sd}\theta(s,d) \tag{13}$$

Where, $v_d$ and $\mu_f$ refer to the delay and flow priority, respectively, and $v_d + \mu_f = 1$. Meanwhile, we are looking to minimize delay and so a negative sign has been added to turn the problem into a maximization problem.

Accordingly, $w_{ij}$ is the weight of the $e_{ij}$ link, as shown in Eq. (14).

$$w_{ij} = \frac{v_d{*}CD + \mu_f{*}CF}{c_{ij}} \tag{14}$$

Where, $c_{ij}$ is the residual bandwidth of the $e_{ij}$ link, $CD$ is the number of critical links in terms of delay, and $CF$ is the number of critical links in terms of flow. In general, $CD$ and $CF$ are calculated for all ingress - egress pairs, as shown in Eq. (15) and Eq. (16).

$$CD = \sum_{e_{ij}\in CD_{(s,d)}} \lambda_{sd}, \forall(s,d)\in LSP \tag{15}$$

$$CF = \sum_{e_{ij}\in CF_{(s,d)}} \gamma_{sd}, \forall(s,d)\in LSP \tag{16}$$

Where, $\lambda_{sd}$ refers to the weight of the ingress - egress pair $(s,\ d)$ for critical links in terms of delay, and $\gamma_{sd}$ refers to the weight of the ingress - egress pair $(s,\ d)$ to the critical links in terms of flow. $LSP$ refers to all ingress - egress pairs. Also, $CD_{(s,\ d)}$ and $CF_{(s,\ d)}$ are sets of critical links in terms of delay and flow based on $LSP$.

### 4.3 Identify critical links

The purpose of identifying critical links is to use the minimum number of critical links when routing. This can conserve the resources needed for future requests and provide load balancing on all network links. In addition, the routing process based on the least use of critical links as much as possible can ensure the routing of future requests with minimal interference. In this paper, the identification of critical links in terms of delay (i.e., $CD_{(s,\ d)}$) and current (i.e., $CF_{(s,\ d)}$) in all ingress - egress pairs are emphasized.

- *Critical link set for* $CD_{(s,\ d)}$*:* A path delay is dependent on both static (propagation delay and link bandwidth) and dynamic (bandwidth assigned to the path) factors. Therefore, critical links in terms of delay are not based on maximum flow alone, as shown in Eq. (3). Here, the data plane delay and control plane delay are used to identify critical links in terms of delay. We use this model to calculate the weight of network links and then identify critical links in terms of delay based on the shortest weighted paths between each ingress - egress pair. All critical links identified in $(s,\ d)$ are considered as critical links in terms of delay and are added to the $CD_{(s,\ d)}$ set. The $CD_{(s,\ d)}$ set contains all the critical links in terms of delay for all ingress - egress pairs.

The process of identifying critical links in terms of delay for an ingress - egress pair $(s,\ d)$ is as follows: First, the weight of all network links is calculated based on the LR-servers model in Eq. (1). Then, the shortest weighted path for $(s,\ d)$ is found by the Dijkstra algorithm. The link with the minimum unreserved bandwidth in the path found is critical and belongs to the $CD_{(s,\ d)}$ set. Next, the critical link is removed from the path and the shortest weighted path is searched again to identify other critical links. When several links have the same bandwidth, all links are removed. This process is repeated until there are no paths to $(s,\ d)$. Appendix 1 shows MATLAB code to identify critical links in terms of delay. A related example Identifying the critical links in terms of delay for the ingress - egress pair (1, 13) on the graph $G$ is shown in Fig. 5, where $G$ has 15 nodes and 28 links (MIRA topology). Here, the minimum bandwidth is the same for $1 \rightarrow 2$, $1 \rightarrow 3$ and $1 \rightarrow 4$ links, so these links are critical. This process continues by removing these links to identify other critical links belonging to the $CD_{(s,\ d)}$ set.

*Critical link set for* $CF_{(s,\ d)}$*:* All links from an input-output pair $(s,\ d)$ belonging to the set of minimum cut links are considered as critical links in terms of flow and are added to the $CF_{(s,\ d)}$ set. The $CF_{(s,\ d)}$ set contains all the critical links in terms of flow for all ingress - egress pairs. Critical links in terms of flow are identified based on the Max-Flow Min-Cut algorithm [38]. This algorithm according to Eq. (5) explicitly considers the bottlenecks when routing a flow in the network. Appendix 1 shows the MATLAB code to identify critical links in terms of flow, where 'maxflow' function is used. This function can identify critical links based on the graph G (network topology with link bandwidth). A related example Identifying the critical links in terms of flow for the ingress - egress pair (1, 13) on the MIRA topology is shown in Fig. 6. Critical links can be identified by applying the 'maxflow' function to other ingress - egress pairs. Accordingly, it can be shown that the critical links in the MIRA topology include $1 \rightarrow 2$, $1 \rightarrow 3$ and $1 \rightarrow 4$.

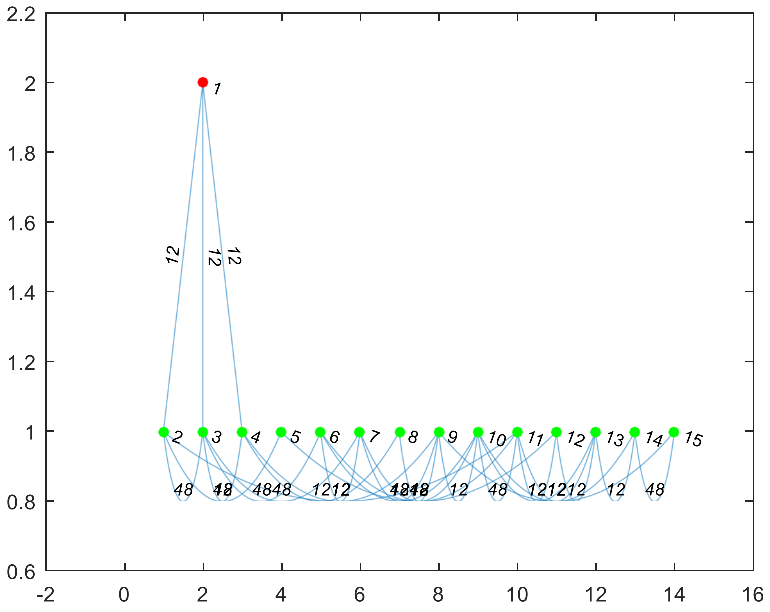


Fig. 5 Critical links in terms of delay for ingress - egress pair (1, 13)

## 5 Experimental results

In this section, the performance of the proposed routing algorithm (i.e., FDBGR) is evaluated in comparison with some SDN-based routing methods such as H-MCOP [17], MH-MCOP [11], QoMRA [2], QROUTE [34] and REDO [1]. Ultimately, we will demonstrate that the

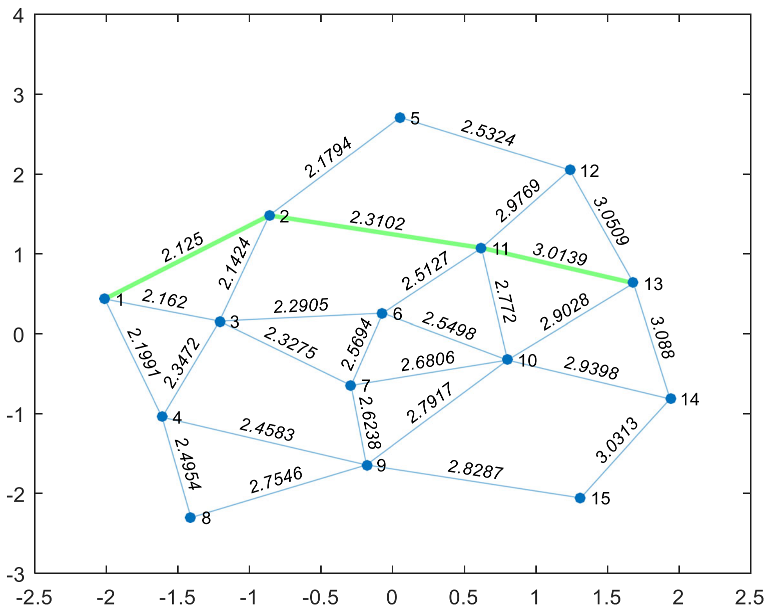


Fig. 6 Critical links in terms of flow for ingress - egress pair (1, 13)

FDBGR algorithm increases the number of accepted requests and satisfy QoS requirements for video conferencing services over SDN.

This section is divided into five subsections. The first subsection discusses the experimental setup. The second subsection shows the network topologies used in the simulation. Evaluation criteria are given in the third subsection. The fourth subsection compares the FDBGR with state-of-the-art methods. The discussion on the time complexity of the FDBGR is presented in the fifth subsection.

## 5.1 Experimental setup

The simulation is carried out on MATLAB 2019a and all comparisons are reported to increase reliability based on an average of 25 distinct runs. All experiments were performed on a PC with Intel Core i7 processor at 3.2GHz, 16GB RAM, and Windows 10 operating system. In order to investigate the effect of different network traffic conditions on the performance of routing algorithms, different scenarios of delay and bandwidth on the SDN have been analyzed. The four scenarios considered in Table 3 are given, where one item is randomly selected for each video conferencing service request with uniform distribution. Here, the delay is the maximum delay in milliseconds and bandwidth in units.

Besides, setting the values of the proposed algorithm parameters is as shown in Table 4. Most of these parameters are adjusted based on research done in [17], and other parameters are determined using the Taguchi method to achieve the best solution [11]. Since the experiments are carried out on algorithms using the same parameters and network topologies, the comparisons are performed in fair conditions.

In order to reflect the real-world conditions better, the following traffic units are considered [2]:

- For directional voice call, the required bandwidth is 90.4 kbps (1 unit).
- For bi-directional voice calls with symmetric flows, the required bandwidth is 180.8 kbps (2unit).
- For video calls, the required bandwidth is 676.8 kbps (7.5 unit).
- For a bi-directional video conference session, the required bandwidth is 857 kbps (9.5 unit).

## 5.2 Network topology

Two well-known network topologies including MIRA and ANSNET are used to compare different routing algorithms. The MIRA topology has been used in numerous studies such as [13, 31] and the ANSNET topology has been accepted by studies such as [13, 15]. MIRA topology consists of 15 nodes and 28 links, as shown in Fig. 7. This topology displays the

**Table 3** The defined scenario for performance analysis

| Scenarios | Delay (millisecond) | Bandwidth (unit) |
|---|---|---|
| Scenario 1 | 95–100 | 1–4 |
| Scenario 2 | 60–65 | 1–4 |
| Scenario 3 | 95–100 | 5–9 |
| Scenario 4 | 60–65 | 5–9 |

**Table 4** The FDBGR algorithm parameters

| Parameter | Description | Value |
|---|---|---|
| $M$ | Packet maximum length | 2 KB |
| $r$ | Request rate | 1 |
| $t$ | Maximum rate | 5 |
| $b$ | Invasion rate | 100 |
| $v_d$ | Delay priority coefficient | 0.5 |
| $\mu_f$ | Flow priority Coefficient | 0.5 |
| $\lambda_{sd}$ and $\gamma_{sd}$ | Ingress - egress pair weight | 1 |

connections of an undirected network (bi-directional links). As illustrated, there are two different types of network links: thin links with capacity equals 12 units, and thick links with capacity equals 48 units. In particular, a subset of nodes in the network operates as ingress - egress pairs (i.e., LSPs) which are ($4 \rightarrow 2$, $5 \rightarrow 9$, $1 \rightarrow 13$, $5 \rightarrow 15$) in MIRA. ANSNET topology is a heterogeneous topology with bi-directional links and includes 18 nodes and 30 links, as shown in Fig. 8. The capacity of all links is equal to 20 units, and the nodes ($4 \rightarrow 15$, ($2 \rightarrow 16$, $1 \rightarrow 17$) are considered as the ingress - egress pairs.

This paper has increased all the link capacities of MIRA and ANSNET network topologies hundredfold units, which has enabled us to examine a wide variety of LSPs. Accordingly, the capacity of thin links and thick links in MIRA topology is 1200 and 4800 units, respectively, and the capacity of links in ANSNET topology is 2000 units. The overall number of requests in the MIRA and ANSNET topologies is equal to 3000 and 3500, respectively. The pairs of ingress - egress routers for the request of establishing an LSP are randomly selected. Furthermore, let all routed paths not be broken until the end of the simulation, so all LSPs are long-lived [13].

## 5.3 Evaluation criteria

We have considered several criteria to evaluate the FDBGR performance to compare to other algorithms. These criteria include the number of accepted requests, average path length, energy consumption, load balancing, and average delay. These criteria are briefly discussed below.

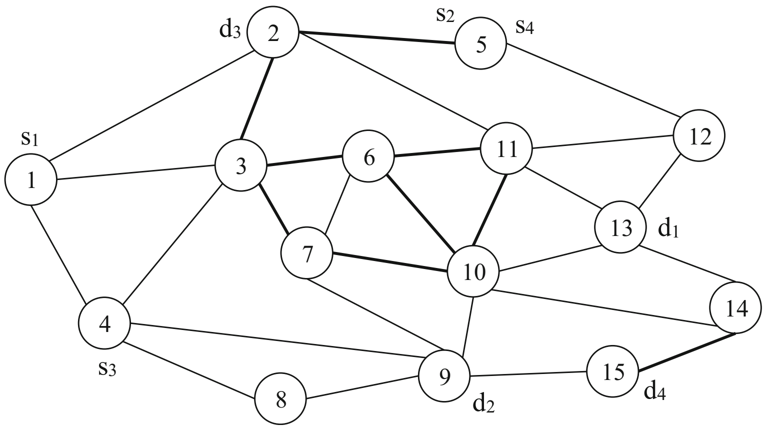


**Fig. 7** MIRA topology structure

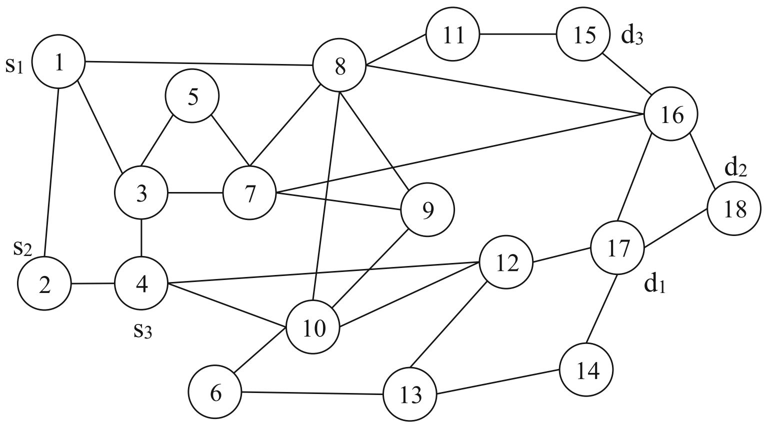


**Fig. 8** ANSNET topology structure

- *Number of accepted requests:* This criterion indicates the number of requests that have been successfully routed. A larger value for this criterion indicates that the routing algorithm performs well in resource management.
- *Average path length:* The path length for each request is calculated according to the number of passed links on the path from source to destination [31]. A lower value of this criterion is more desirable, as longer paths use more links as network resources. This criterion is defined according to Eq. (17).

$$Average\ path\ length = \frac{\sum_{i=1}^{N_{LSP}} length(path_i)}{N_{LSP}} \tag{17}$$

Where, $N_{LSP}$ denotes the number of determined paths and $path_i$ refers to the $i$-th determined path.

- *Energy consumption:* The consumed energy of each path is defined as the total consumed energy of each pair of consecutive nodes. If $(i,\ j)$ is a pair of nodes within the path, the consumed energy is equal to the sum of the energy required for transmitting from the node $i$ and receiving in the node $j$. Processors in the network are recognized as a major component in energy consumption. Energy consumption in nodes is calculated based on a linear relationship between energy consumption and the amount of processor utilization [10, 20, 30]. Thus, the total energy consumed by a node can be calculated based on the integral relationship of the energy consumed over a given time interval (for example $t_0$ to $t_1$), as defined in Eq. (18).

$$Energy = \int_{t_0}^{t_1} P(u(t))dt \tag{18}$$

Where, $u(t)$ refers to processor utilization at time $t$, and $P$ is a function for determining energy consumption based on processor utilization.

- *Load balancing:* This criterion shows how fair the workload is distributed across all network links, and we use the variance of links utilization to calculate it. Its lower value leads to optimizing resource utilization, increasing power, lowering response time, and avoiding overhead. Therefore, less load balancing variance on the links leads to a more uniform distribution of network traffic.
- *Average delay:* This criterion indicates the average path delay of established requests. Its lower value is desirable.

## 5.4 Comparisons and discussions

This section is related to the evaluation of the proposed FDBGR algorithm in comparison with H-MCOP, MH-MCOP, QoMRA, QROUTE and REDO algorithms. In the first experiment, the effectiveness of FDBGR was compared based on the number of accepted requests compared to the algorithms. The results of this comparison are reported in Table 5 with different scenarios for the MIRA topology. This experiment analyzes different algorithms in terms of different workloads for 3000 requests. Likewise, Table 6 shows this comparison in ANSNET topology for 3500 requests. Each row shows the results of different algorithms for a scenario, where the last row is dedicated to the average results. Also, the bold values in Tables 5 and 6 indicate the best results for each algorithm.

Overall, the results of the algorithms in both topologies are almost alike. The results in Scenario 1 show that all algorithms have almost the same performance. However, the number of routed requests in H-MCOP, MH-MCOP and QoMRA algorithms is less, which can lead to increased delay. Although the QoMRA algorithm guarantees the delay, it is unable to satisfy the requested delay based on the available routes. However, the H-MCOP and MH-MCOP algorithms do not guarantee delay and ignore it for video conferencing service requests. Given that the bandwidth of Scenario 2 requests is the same as Scenario 1, the results for these two scenarios are almost the same. Request bandwidth is increased in Scenario 3, so algorithms guaranteeing both delay and bandwidth have fewer requests accepted than in Scenario 1 and 2. The results show that FDBGR, REDO and QROUTE have the best performance, respectively, and offer better results compared to H-MCOP, MH-MCOP and QoMRA. H-MCOP, MH-MCOP and QoMRA algorithms do not guarantee bandwidth and their results are not comparable to FDBGR, REDO and QROUTE. The bandwidth of requests is increased in Scenario 4 while the maximum delay is reduced. Here, FDBGR outperforms all algorithms, where REDO and QROUTE are next. FDBGR outperforms H-MCOP, MH-MCOP and QoMRA algorithms in all scenarios. FDBGR offers better results than QROUTE in all scenarios except scenario 1 of the MIRA topology and scenario 3 of the ANSNET

**Table 5** Comparison of different algorithms based on the number of accepted requests for MIRA topology

| Scenario | H-MCOP | MH-MCOP | QoMRA | QROUTE | REDO | FDBGR |
|---|---|---|---|---|---|---|
| Scenario 1 | 2419 | 2422 | 2430 | **2453** | 2443 | 2441 |
| Scenario 2 | 2009 | 2042 | 2087 | 2144 | 2167 | **2228** |
| Scenario 3 | 1454 | 1831 | 1794 | 2016 | 2052 | **2096** |
| Scenario 4 | 1604 | 1441 | 1270 | 1787 | 1781 | **1816** |
| Average | 1871 | 1934 | 1895 | 2100 | 2111 | **2145** |

**Table 6** Comparison of different algorithms based on the number of accepted requests for ANSNET topology

| Scenario | H-MCOP | MH-MCOP | QoMRA | QROUTE | REDO | FDBGR |
|---|---|---|---|---|---|---|
| Scenario 1 | 2402 | 2434 | 2441 | 2456 | **2467** | 2462 |
| Scenario 2 | 2375 | 2378 | 2320 | 2382 | 2402 | **2414** |
| Scenario 3 | 1742 | 1803 | 1818 | **1854** | 1813 | 1833 |
| Scenario 4 | 1336 | 1347 | 1356 | 1430 | 1423 | **1437** |
| Average | 1964 | 1990 | 1984 | 2030 | 2026 | **2036** |

topology. In addition, FDBGR reports a higher number of accepted requests than REDO in all scenarios except Scenario 1 of the ANSNET topology. FDBGR, QROUTE, and REDO algorithms guarantee delay and bandwidth, but FDBGR delays high-resource requests to be able to conserve more resources for future requests. This process in FDBGR has led to an increase in the number of accepted requests. On average, for the MIRA topology, FDBGR improved the number of accepted requests by 14.64%, 10.91%, 13.19%, 2.14% and 1.61% compared to H-MCOP, MH-MCOP, QoMRA, QROUTE and REDO algorithms, respectively. Also, the FDBGR superiority for ANSNET topology is 3.67%, 2.31%, 2.63%, 0.29% and 0.49%, respectively.

In the following, we have conducted various experiments in terms of criteria such as average path length, energy consumption, load balancing, and average delay to compare algorithms. The results in all comparisons are reported on average for the four defined scenarios. Here, comparisons for each evaluation criterion are performed on both MIRA and ANSNET topologies. Average path length is one of the important criteria in evaluating the performance of routing algorithms. The results of the proposed FDBGR algorithm in comparison with H-MCOP, MH-MCOP, QoMRA, QROUTE and REDO algorithms for MIRA and ANSNET topologies are given in Figs. 9 and 10, respectively.

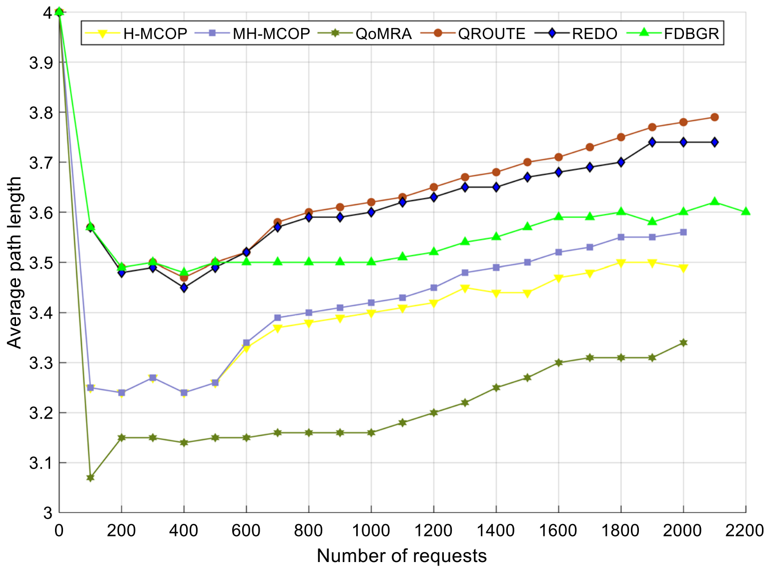


**Fig. 9** Comparison of different algorithms based on the average path length for MIRA topology

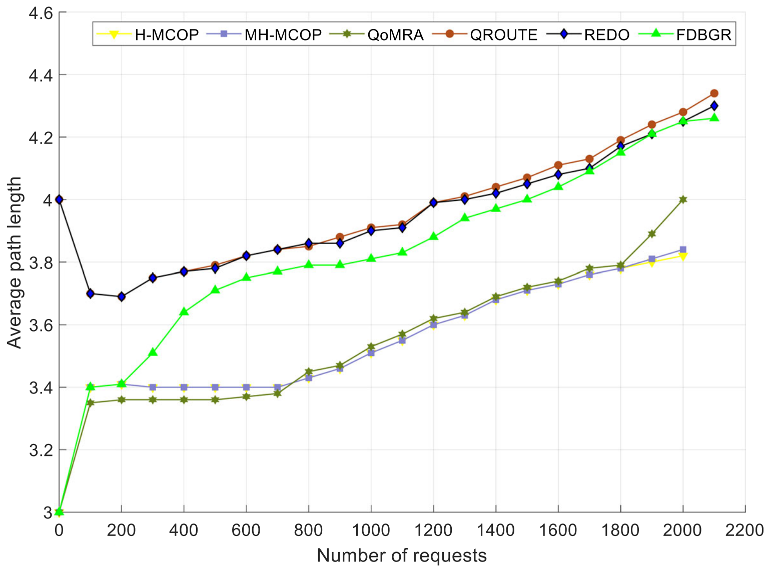


**Fig. 10** Comparison of different algorithms based on the average path length for ANSNET topology

The H-MCOP, MH-MCOP, and QoMRA algorithms use paths with a lower average length in routing, because these algorithms find the paths only to satisfy the bandwidth, which increases network congestion. In this regard, FDBGR, REDO and QROUTE algorithms have a longer average path length, because in addition to bandwidth, they guarantee delay in the network. However, FDBGR reported better performance in both MIRA and ANSNET topologies compared to REDO and QROUTE. The reason for this superiority is the distribution of network workload by allocating weight to the links as well as preserving network resources for future requests by identifying critical links. Therefore, FDBGR would be able to accept more requests. Although, the H-MCOP, MH-MCOP, and QoMRA algorithms offer shorter path lengths than the FDBGR, but these algorithms accept fewer requests. In particular, QoMRA also has a lower average path length than FDBGR, but this algorithm merely guarantees bandwidth in requests. Totally, FDBGR offers a shorter path length than QROUTE and REDO although this amount is very low. On average, for the MIRA topology, FDBGR improved the path length by 2.52% and 1.93% compared to QROUTE and REDO algorithms, respectively. Also, the FDBGR superiority for ANSNET topology is 3.71% and 3.38%, respectively.

Energy management has become one of the most important topics in the last few decades due to the dramatic increase of smart device users. In SDN, resource management and energy consumption are calculated at the level of flows. In general, energy consumption will be reduced by reducing network traffic. Figures 11 and 12 report the comparison results of different routing algorithms based on energy consumption for the MIRA and ANSNET topologies, respectively. The results of this comparison are averaged for all defined scenarios. In general, the energy consumption of all algorithms is almost the same, because the network is analyzed with a relatively small number of requests. However, the proposed algorithm provides less energy consumption. As illustrated, FDBGR and QROUTE algorithms provide lower energy consumption due to the proper load distribution on all network links. In contrast,

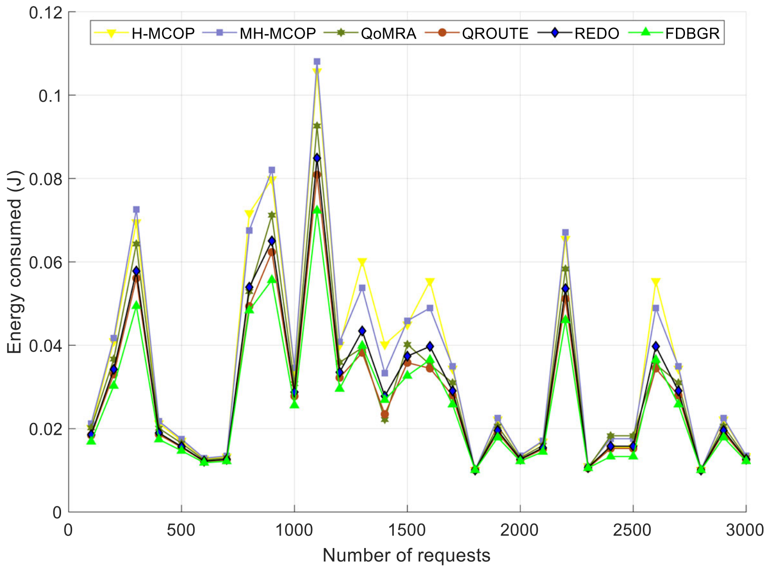


**Fig. 11** Comparison of different algorithms based on the energy consumption for MIRA topology

H-MCOP and MH-MCOP algorithms have the highest energy consumption compared to other algorithms due to the lack of guaranteed delay and the use of critical links. On average, for the MIRA topology, FDBGR improved the energy consumption by 37.71%, 35.73%, 15.44%, 4.36% and 9.92% compared to H-MCOP, MH-MCOP, QoMRA, QROUTE and REDO algorithms, respectively. Also, the FDBGR superiority for ANSNET topology is 37.74%, 25.99%, 14.81%, 5.08% and 9.21%, respectively.

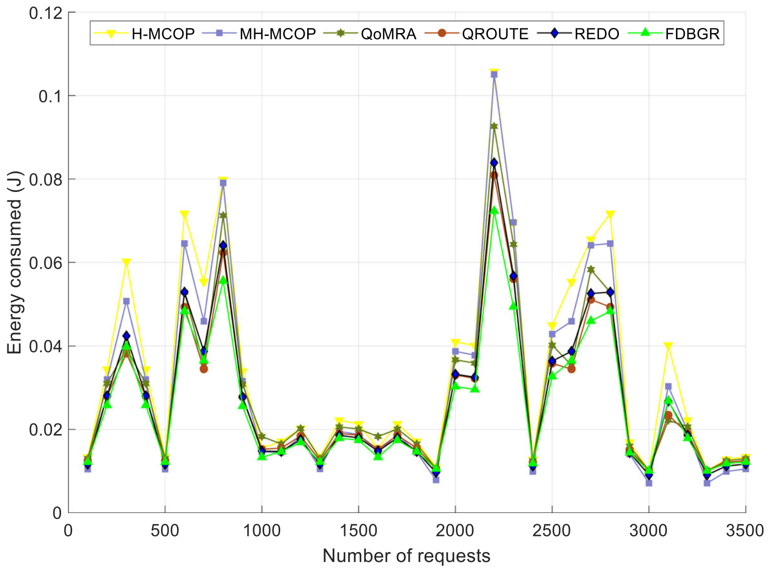


**Fig. 12** Comparison of different algorithms based on the energy consumption for ANSNET topology

Many data centers use special hardware resources to achieve load balancing, however, this equipment requires a lot of investment. While multiple sources are considered for a particular request, load balancing can provide maximum network efficiency and resource balancing criterion. Therefore, considering the network load distribution in the routing process can improve the load balancing criterion. In the next experiment, the network load balancing in routing algorithms is analyzed. We use the load variance of the links to describe the load balancing in the network. The results of this comparison for MIRA and ANSNET topologies are shown in Figs. 13 and 14, respectively. A lower value for load variance in the links used may indicate a better load balancing criterion. As illustrated, the FDBGR, REDO and QROUTE algorithms report better load balancing criterion. Although, the QROUTE algorithm uses longer paths, it offers the least variance and therefore the best load balancing criterion. Overall, the FDBGR has the best performance in terms of load balancing with an average load balancing variance of 0.035 (for MIRA) and 0.047 (for ANSNET) after QROUTE algorithm. Nevertheless, unlike H-MCOP, MH-MCOP and QoMRA guarantees both delay and bandwidth constraints. On average, for the MIRA topology, FDBGR improved the average load balancing variance by 45.46%, 31.4%, 22.54% and 4.49% compared to H-MCOP, MH-MCOP, QoMRA and REDO algorithms, respectively. Also, the FDBGR superiority for ANSNET topology is 37.34%, 37.8%, 27.39% and 6.6%, respectively.

The average delay on paths upon the MIRA and ANSNET topologies is shown in Figs. 15 and 16, respectively. The simulation results show that H-MCOP has the lowest delay, but it instead has a minimal number of accepted requests and does not consider the importance of the critical links for future requests. MH-MCOP, and QoMRA algorithms provide the minimum delay in low requests (less than 2000). This is due to utilizing the shortest path for routing; however, their delay increases by increasing the number of requests.

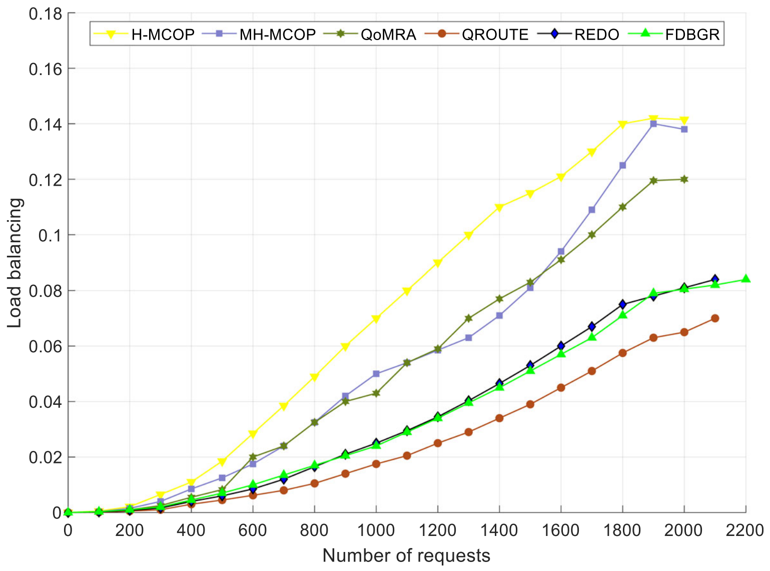


**Fig. 13** Comparison of different algorithms based on the load balancing for MIRA topology

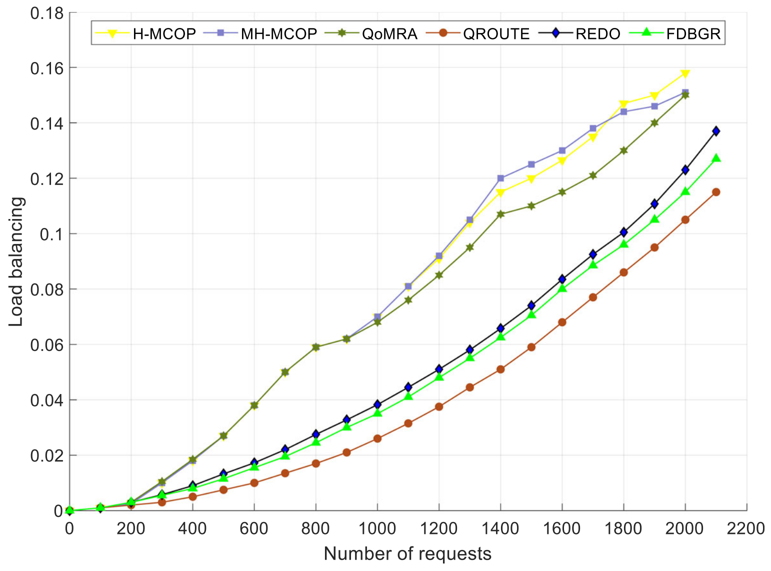


**Fig. 14** Comparison of different algorithms based on the load balancing for ANSNET topology

The QROUTE algorithm reports maximum delay on paths because QROUTE uses the paths with longer lengths, and its purpose is to distribute the load uniformly across the network. Furthermore, the main purpose of QROUTE algorithm is to administer only the maximum flow for accepting subsequent requests. Therefore, as the number of requests increases, the performance of this algorithm decreases in terms of delay. The performance of REDO and FDBGR algorithms is almost equal for all numbers of requests. The FDBGR

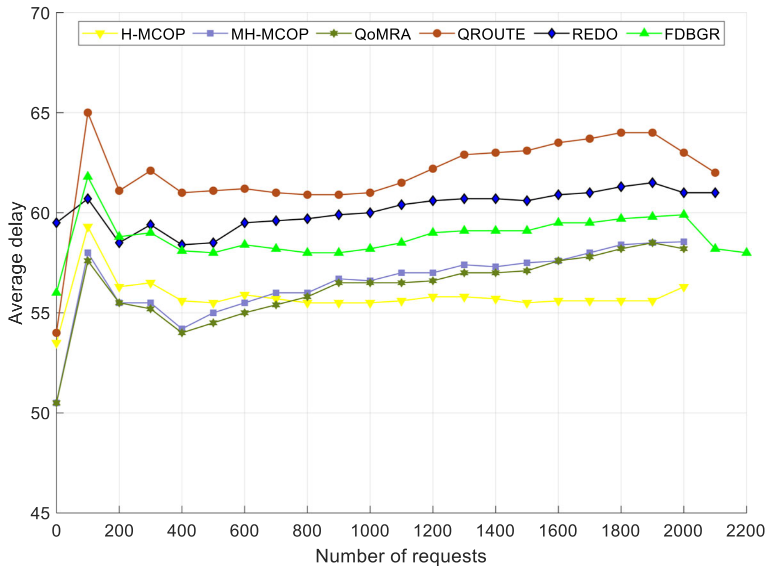


**Fig. 15** Comparison of different algorithms based on the average delay for MIRA topology

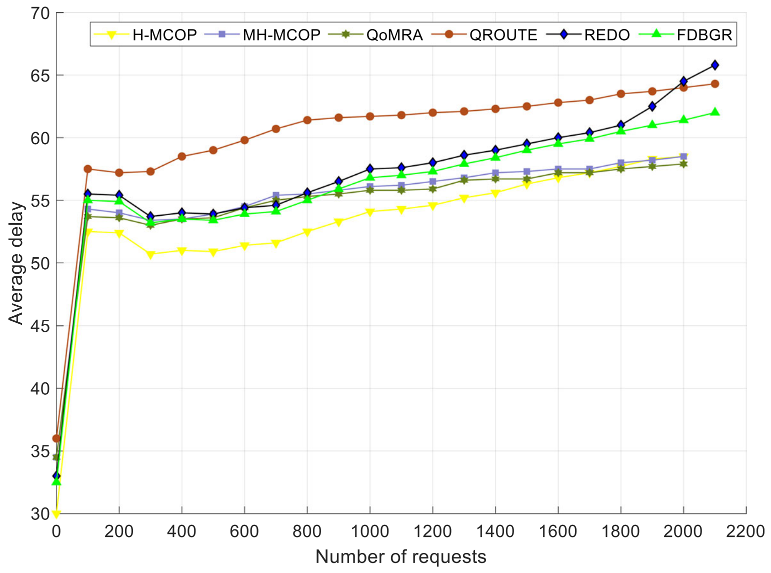


**Fig. 16** Comparison of different algorithms based on the average delay for ANSNET topology

algorithm performs well in the high number of requests because the weight of links in FDBGR is directly related to the minimum delay. Hence, FDBGR maintains delay for subsequent requests. Overall, the average delay for FDBGR during the whole simulation period is relatively unchanged, even if the network is saturated. On average, for the MIRA topology, FDBGR improved the energy consumption by 5.43% and 2.42% compared to QROUTE and REDO algorithms, respectively. Also, the FDBGR superiority for ANSNET topology is 7.35% and 1.53%, respectively.

## 5.5 Time complexity

Computational complexity theory is a branch of mathematics and computer science that examines the difficulty of solving problems using an algorithm. The time complexity of an algorithm refers to the time required to execute it. Let $P$ be the number of ingress - egress pairs, $N$ the number of nodes, and $L$ be the number of network links. Accordingly, the time complexity of the proposed routing algorithm is as follows: The time complexity of the fuzzy system in FDBGR is $O(|M| \times |K|^2)$ according to [13], in which $M$ is the number of rules and $K$ is the number of fuzzy samples. Since the number of rules is limited, the complexity of the fuzzy system can be considered as $O(|K|^2)$. In the rest of the algorithm, $w_{ij}$ is calculated $|L|$ times. Here, all the minimum cutting sets among all the ingress - egress pairs are processed by the Edmonds-Karp [19] algorithm whose complexity is $O(|P| \times |N| \times |L^2|)$. Also, the time complexity of the algorithm main loop is $O(|L|)$. Furthermore, the time complexity of performing the Dijkstra algorithm [35] is $O(|L| \times |N| log_2 |N|)$. Overall, the structure of the algorithm main loop is performed in $O(|L| \times W)$ in which refers to the link of the network with the highest bandwidth. In general, the time complexity of FDBGR can be denoted by

$O(|K|^2 + |P| \times |N| \times |L|^2 \times W \times |N| \times log_2|N|)$ where small execution times are omitted. Here, $W$ and $|L| \geq |N| - 1$ is constant coefficients, and the time complexity can be expressed as $O(|K|^2 + |P| \times |N| \times |L^2|)$.

We compared the time complexity of other algorithms through their runtime in the simulation, as shown in Table 7. The results of this experiment are reported as averages for all scenarios. Also, bold values refer to lower running times for each algorithm. Investigations show that none of the algorithms outperforms others in all cases, which means that the optimal performance of the algorithm depends on the network topology and the number of ingress - egress pairs. In general, the proposed FDBGR algorithm shows the best performance in sparse networks with appropriate ingress - egress pairs. Conversely, other algorithms report the best time complexity in dense networks with a limited number of conditions.

## 6 Conclusion

Providing satisfactory video conferencing services by MCUs is challenging because it requires low delay and high bandwidth. In general, the SDN paradigm for routing network traffic improves flexibility and scalability because it separates the data plane from the control plane. SDN provides the basic infrastructure for video conferencing services and enables delayed and makes it possible to provide QoS-aware guarantee of delay and bandwidth. In this paper, we introduce FDBGR to improve video conferencing services over SDN, which follows a rule-based fuzzy system to guarantee both delay and bandwidth. The main purpose of the FDBGR is to increase the number of conference connection requests faces network congestion by reducing the resources utilization. To do this, FDBGR maximizes delay of routes by increasing the links bandwidth. In addition, FDBGR protects network resources for future requests by identifying critical links. Moreover, FDBGR first performs routing for requests with lower resource demands, which is accomplished by deferring requests with high bandwidth and maximum the least delays. Here, a fuzzy system based on weighted rules has also been introduced that can accurately filter out requests with high resource demands. We evaluated FDBGR performance in the field of video conferencing through comparing with the most related algorithms. The results have confirmed the superiority of FDBGR over other algorithms in terms of maximum flow and delay management. This is a remarkable result because a majority of algorithms considers only one constraint (delay or bandwidth) while FDBGR considers both constraints simultaneously and, of course, provide a fair traffic load balancing. In addition, our future work considers packet loss considerations to make the system fault tolerant and to satisfy QoS requirements.

**Table 7** Comparison of different algorithms based on runtime (s)

| Topology | H-MCOP | MH-MCOP | QoMRA | QROUTE | REDO | FDBGR |
|---|---|---|---|---|---|---|
| MIRA | 139 | **129** | 146 | 139 | 140 | 131 |
| ANSNET | 203 | 187 | 211 | 190 | 221 | **184** |
| Average | 171 | 158 | 178 | 164 | 180 | **157** |

## Appendix 1. MATLAB code to identify critical links in terms of delay

```
function CD = CDsd(E, C, M, R, Mij, P)
    Propij = linspace(0,1, size(E, 1))';
    dw = (M / R) + (Mij ./ C) + Propij;
    CD = [];
    for i = 1 : size(P, 1)
        Ep1 = P(i, 1);
        Ep2 = P(i, 2);
        E = E;
        w = dw;
        tempC = C;
        while true
            s = E(:, 1);
            d = E(:, 2);
            nonLink = max(max(E == Ep1)) +  max(max(E == Ep2));
            if nonLink ~= 2, break; end
            G = graph(s, d, w);
            LP = shortestpath(G, Ep1, Ep2, 'Method', 'positive');
            if isempty(LP), break; end
            LPi = [];
            for j = 2 : length(LP)
                u = ismember(E, [LP(j - 1) LP(j)], 'rows');
                v = ismember(E, [LP(j) LP(j - 1)], 'rows');
                [~, mx] = max(max(u, v));
                LPi = [LPi; mx];
            end
            MinC = min(tempC(LPi));
            f = tempC(LPi) == MinC;
            MinCi = LPi(f);
            CD = [CD; E(MinCi, :)];
            E(MinCi, :) = [];
            w(MinCi) = [];
            tempC(MinCi) = [];
        end
    end
end
```

## Appendix 2. MATLAB code to identify critical links in terms of flow

```
function CF = CFsd(E, w, P)
    s = E(:, 1);
    d = E(:, 2);
    G = graph(s, d, w);
    CF = [];
    for i = 1 : size(P, 1)
        Ep1 = P(i, 1);
        Ep2 = P(i, 2);
        if isempty(find(E == Ep1)) || isempty(find(E == Ep2))
            continue;
        end
        [~, ~, cs, ct] = maxflow(G, Ep1, Ep2);
        [A, B] = meshgrid(cs, ct);
        cl = cat(2, A', B');
        com = reshape(cl, [], 2);
        tmp = [com(:, 2) com(:, 1)];
        com = [com; tmp];
        lc = intersect(E, com, 'rows');
        CF = [CF; lc];
    end
end
```

**Data availability** Data sharing not applicable to this manuscript as no datasets were generated or analyzed during the current study.

## Declarations

**Conflict of interest** We certify that there is no actual or potential conflict of interest in relation to this manuscript.

## References

1. Al-Jawad A, Comşa IS, Shah P, Gemikonakli O, Trestian R (2021) REDO: a reinforcement learning-based dynamic routing algorithm selection method for SDN. In 2021 IEEE conference on network function virtualization and software defined networks (NFV-SDN) (pp. 54-59). IEEE
2. Barakabitze AA, Sun L, Mkwawa I. H., Ifeachor, E. (2018) A novel QoE-centric SDN-based multipath routing approach for multimedia services over 5G networks. In 2018 IEEE international conference on communications (ICC) (pp. 1-7). IEEE.
3. Bensalah F, El Kamoun N (2019) Novel software-defined network approach of flexible network adaptive for VPN MPLS traffic engineering. Int J Adv Comput Sci Appl 10(4):280–284
4. Li P, Yang M, Wu Q (2021) Confidence interval based distributionally robust real-time economic dispatch approach considering wind power accommodation risk. IEEE Trans Sustain Energy 12:58–69
5. Si Z, Yang M, Yu Y, Ding T (2021) Photovoltaic power forecast based on satellite images considering effects of solar position. Appl Energy 302:117514
6. Binsahaq A, Sheltami TR, Salah K (2019) A survey on autonomic provisioning and management of QoS in SDN networks. IEEE Access 7:73384–73435
7. Blanchy F, Mélon L, Leduc G (2003) A preemption-aware on-line routing algorithm for MPLS networks. Telecommun Syst 24(2):187–206
8. Casas-Velasco DM, Rendon OMC, da Fonseca NL (2021) DRSIR: a deep reinforcement learning approach for routing in software-defined networking. IEEE Trans Netw Serv Manag
9. Das T, Gurusamy M (2021) Multi-objective control plane dimensioning in hybrid SDN/legacy networks. IEEE Trans Netw Serv Manag 18(3):2929–2942
10. Etemadi M, Ghobaei-Arani M, Shahidinejad A (2020) Resource provisioning for IoT services in the fog computing environment: an autonomic approach. Comput Commun 161:109–131
11. Feng G, Makki K, Pissinou N, Douligeris C (2002) Heuristic and exact algorithms for QoS routing with multiple constraints. IEICE Trans Commun 85(12):2838–2850
12. Cheng F, Wang H, Zhang L, Ahmad AM, Xu N (2022) Decentralized adaptive neural two-bit-triggered control for nonstrict-feedback nonlinear systems with actuator failures. Neurocomputing 500:856–867
13. Heydarian M (2012) A high performance optimal dynamic routing algorithm with unicast multichannel QoS guarantee in communication systems. J Supercomput 62(1):315–344
14. Irawan A, Mat Jusoh MF, Abas MF (2018) Label-QoS switching protocol for quality of service assurance in dynamic swarm robot local network. Int J Commun Syst 31(10):e3565
15. Kodialam M, Lakshman TV (2000) Minimum interference routing with applications to MPLS traffic engineering. In proceedings IEEE INFOCOM 2000. Conference on computer communications. Nineteenth annual joint conference of the IEEE computer and communications societies (cat. No. 00CH37064) (Vol. 2, pp. 884-893). IEEE
16. Korkmaz T, Guntaka J (2004) State-path decoupled QoS-based routing framework. In IEEE global telecommunications conference, 2004. GLOBECOM'04. (Vol. 3, pp. 1515-1519). IEEE
17. Korkmaz T, Krunz M (2001) Multi-constrained optimal path selection. In proceedings IEEE INFOCOM 2001. Conference on computer communications. Twentieth annual joint conference of the IEEE computer and communications society (cat. No. 01CH37213) (Vol. 2, pp. 834-843). IEEE
18. Kucminski A, Al-Jawad A, Shah P, Trestian R (2017) QoS-based routing over software defined networks. In 2017 IEEE international symposium on broadband multimedia systems and broadcasting (BMSB) (pp. 1-6). IEEE
19. Lammich P, Sefidgar SR (2016) Formalizing the Edmonds-karp algorithm. In international conference on interactive theorem proving (pp. 219-234). Springer, Cham
20. Liu C, Wang J, Zhou L, Rezaeipanah A (2022) Solving the multi-objective problem of IoT service placement in fog computing using cuckoo search algorithm. Neural Process Lett 54:1–32

21. Mahmoudi M, Avokh A, Barekatain B (2022) SDN-DVFS: an enhanced QoS-aware load-balancing method in software defined networks. Cluster computing, 1-26.
22. Majdoub M, Kamel AE, Youssef H (2019). An efficient MPLS-based approach for QoS providing in SDN. In international conference on intelligent systems design and applications (pp. 497-508). Springer, Cham
23. Majdoub M, Kamel AE, Youssef H (2019) An efficient MPLS-based approach for QoS providing in SDN. In international conference on intelligent systems design and applications (pp. 497-508). Springer, Cham
24. Mesbahi N, Dahmouni H (2015) An efficient algorithm for traffic flow optimization in MPLS networks. In 2015 international conference on protocol engineering (ICPE) and international conference on new Technologies of Distributed Systems (NTDS) (pp. 1-6). IEEE
25. Ogul M, Akçam N, Erkan N (2014) Measurement of OSPF-MPLS-TE-FRR line transitions and data losses. Balkan J Elect Comp Engin 2(2):46–50
26. Pfaff B, Lantz B, Heller B (2012) Openflow switch specification, version 1.3. 0. Open Networking Foundation, 39-46
27. Qayyum AA, Zulfiqar M, Abrar M (2020) Quality of service performance analysis of voice over IP in converged MPLS networks. In 2020 international youth conference on radio electronics, electrical and power engineering (REEPE) (pp. 1-4). IEEE
28. Rezaeipanah A, Amiri P, Nazari H, Mojarad M, Parvin H (2021) An energy-aware hybrid approach for wireless sensor networks using re-clustering-based multi-hop routing. Wirel Pers Commun 120(4):3293–3314
29. Rischke J, Sossalla P, Salah H, Fitzek FH, Reisslein M (2020) QR-SDN: towards reinforcement learning states, actions, and rewards for direct flow routing in software-defined networks. IEEE Access 8:174773–174791
30. Shahidinejad A, Ghobaei-Arani M, Esmaeili L (2020) An elastic controller using colored petri nets in cloud computing environment. Clust Comput 23(2):1045–1071
31. Soorki MN, Rostami H (2014) Label switched protocol routing with guaranteed bandwidth and end to end path delay in MPLS networks. J Netw Comput Appl 42:21–38
32. Stiliadis D, Varma A (1998) Latency-rate servers: a general model for analysis of traffic scheduling algorithms. IEEE/ACM Trans Networking 6(5):611–624
33. Tejaswini V, Sayeekumar M, Karthik GM (2021) Fast bandwidth-delay routing methods for software-defined network (SDN). In artificial intelligence techniques for advanced computing applications (pp. 323–334). Springer, Singapore
34. Varyani N, Zhang ZL, Dai D (2020) QROUTE: an efficient quality of service (QoS) routing scheme for software-defined overlay networks. IEEE Access 8:104109–104126
35. Wang H, Yu Y, Yuan Q (2011) Application of Dijkstra algorithm in robot path-planning. In 2011 second international conference on mechanic automation and control engineering (pp. 1067-1069). IEEE
36. Yang R, Yang C, Peng X, Rezaeipanah A A novel similarity measure of link prediction in multi-layer social networks based on reliable paths. Concurrency and Computation: Practice and Experience, e6829
37. Zhao Y, Li Y, Zhang X, Geng G, Zhang W, Sun Y (2019) A survey of networking applications applying the software defined networking concept based on machine learning. IEEE Access 7:95397–95417
38. Zhou L, Lan Y, Xia Y, Gong S (2020) Extended phase unwrapping max-flow/min-cut algorithm for multibaseline SAR interferograms using a two-stage programming approach. Sensors 20(2):375