IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received September 28, 2020, accepted October 18, 2020, date of publication October 21, 2020, date of current version November 2, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3032704

Beyond 5G: Hybrid End-to-End Quality of Service Provisioning in Heterogeneous IoT Networks

MUHAMMAD ASAD[1], (Graduate Student Member, IEEE), ABDUL BASIT[1], SAAD QAISAR[1,2], (Senior Member, IEEE), AND MUDASSAR ALI[1,3], (Member, IEEE)

[1]School of Electrical Engineering and Computer Science, National University of Sciences and Technology, Islamabad 44000, Pakistan
[2]Department of Electrical Engineering, University of Jeddah, Jeddah 23218, Saudi Arabia
[3]Department of Telecommunication Engineering, University of Engineering and Technology, Taxila 47050, Pakistan

Corresponding author: Muhammad Asad (14phdmasad@seecs.edu.pk)

ABSTRACT A significant increase is expected in video and multimedia traffic in Beyond 5G networks. The inclusion of a huge number of IoT nodes in Beyond 5G networks further complicates the design of such networks. These futuristic networks are expected to deal with this increased traffic and number of nodes while ensuring that network delays do not exceed a certain threshold. In such networks, Quality of Service (QoS) provisioning has become vital, not only to guarantee certain key performance indicators but also to improve user experience. This paper proposes a hybrid approach for end-to-end QoS provisioning, involving both clients and controllers to address these challenges. Each client tries to satisfy its own access QoS requirements by choosing optimal access device(s) and makes decisions based on locally available view. Controllers are then responsible for finding optimal paths in the core network to satisfy client core QoS requirements. Experimental results show that the proposed approach provides better QoS guarantees than several other access device selection and routing schemes.

INDEX TERMS Beyond 5G networks, quality of service, QoS in beyond 5G networks, QoS system architecture, QoS system model.

I. INTRODUCTION

With the auctioning and roll-out of 5G networks in several countries across Europe, the United States, and Asia, the research community has started to lay out plans for Beyond 5G (B5G) networks. B5G networks, among other requirements, aim to provide higher peak data rates of up to 10 terabits per second, universal connection, low latency, reliability, higher energy efficiency, universal connectivity, ubiquitous intelligence, and native security. B5G networks also aim to integrate several revolutionary technologies, including holographic radio, terahertz communications, large intelligent surface, orbital angular momentum, and visible light communications [1].

Around 37 billion connected devices are forecasted by the year 2025, with 25 billion of them related to the Internet of Things (IoT) [2]. Connected IoT nodes would include sensors, consumer electronics, wearables, machines, and connected cars. In addition to the increased number of devices, a continual increase in demand for mobile data traffic is expected due to web applications, real-time streaming, and IoT applications [3]. According to Cisco Systems, Global IP traffic will increase from 122 exabytes per month in 2017 to 396 exabytes per month by 2022 [4]. Video traffic will make up 82% of the total IP traffic. B5G networks will have to deal with a massive increase in the number of connected IoT devices and manifolds increase in multimedia traffic. To ensure a certain level of network quality for such traffic and guarantee a certain user experience, providing Quality of Service (QoS) guarantees will be very important in next-generation wireless networks, such as higher data rates, seamless mobility, ultra-low latency, high reliability, and energy efficiency [3].

IoT has no universal definition, but it points to the concept of connecting everyday objects we see around us through a network, usually the Internet. With the concepts of device-to-device and machine-to-machine communications being envisioned as core concepts in future networks, it seems likely that objects with capabilities such as identification, sensing, and processing would be networked to communicate with

The associate editor coordinating the review of this manuscript and approving it for publication was Amjad Ali.

other devices. These networked devices would help achieve several collaborative tasks, including sensing data from a remote location, identifying a particular person, and remotely controlling homes, etc.

For modern cellular networks, two approaches for provisioning QoS are proposed; client-centric and network-centric [5]. In client-centric approaches, the client is responsible for Radio Access Technology (RAT) selection. Whereas, in network-centric approaches, the network devices are responsible for providing a certain level of QoS to the client based on its requirements. There are pros and cons to both approaches. Modern end-devices have high processing power and are equipped with multiple RAT receivers. These devices are very sophisticated and can monitor signal strength, data rates, delays, and other radio access parameters available through various RATs. Furthermore, the end-devices also have a better view of the access network, available battery, client-side application preferences, and user behavior [6]. However, clients do not have a complete overview of the network topology and its parameters, making it very difficult for a client-centric approach to efficiently solve QoS provisioning problems. Whereas, in a network-centric approach, the network devices have a complete overview of the topology and also have information about continuously changing conditions of the links. However, these devices do not have an as accurate view of the access layer as the clients have. Further, as the number of IoT nodes increases as expected in beyond 5G networks, the scalability issues would arise [7]. For network devices, in addition to their data routing and forwarding responsibilities, it might become challenging to handle the QoS requirements of each flow arising from such a massive number of devices in IoT.

QoS provisioning in conventional networks has been performed at network core devices, which are already responsible for routing and forwarding the data. If network core devices are also involved in QoS provisioning, it would significantly burden them, increasing the core's data forwarding delays. Because of the size of IoT networks, it would also cause scalability issues for these devices. Several QoS provisioning architectures have been proposed using Software Defined Networking (SDN) instead of network core devices, thus taking the load away from them [8]–[15]. Recently, there has also been profound research interest in 5G and beyond 5G networks for client-based RAT selection [6], [16]–[19].

The rest of the paper is organized as follows. Section II lists the relevant literature review. The proposed QoS provisioning architecture, system model, and algorithms are presented in Section III. Section IV lists the experimentation setup and is followed by results in Section V. Section VI concludes the paper.

II. LITERATURE REVIEW

A massive number of IoT devices would be interconnected in B5G networks. Providing seamless connectivity to such a massive number of IoT devices with QoS guarantees would be challenging. Conventionally used multiple-access techniques in current cellular networks are insufficient to accommodate such massive IoT networks. B5G networks would need to enable massive IoT for accommodating the massive number of IoT nodes expected in such networks [20]. Orthogonal multiple access techniques are used in current networks. However, as the number of IoT nodes increases, such techniques might not support massive access, along with the inability to provide high spectral efficiency and low latency [21]. Non-orthogonal multiple access techniques are researched to improve spectral efficiency, and significantly improve bandwidth efficiency, while introducing some degree of interference at the receiver [22].

Multi-access Edge Computing (MEC) paradigm has been proposed to move storage and computing resources at the network's edge, closer to wireless end devices. MEC's demand has been driven by high bandwidth and low latency applications, thus improving the quality of experience and QoS [23]. However, increasing data-intensive applications may overwhelm the computing and storage resources at MEC nodes. An extension to MEC's concept is device-enhanced MEC, using modern end devices with increasingly powerful central processing units and storage capabilities. This community of wireless devices, referred to as mobile clouds, aggregates their resources to serve individual end devices along with MEC jointly [24]. Device-enhanced MEC improves the scalability of MEC, with further improved network QoS. A Layback architecture [25] is proposed for resource sharing of communication and computational resources among different wireless operators and technologies. Layback architecture decouples the front-haul from the backhaul by placing the coordination points just behind the gateways. A virtualized radio access network approach, named FluidRAN [26], relocates some base station functions to a central unit, thus partially reducing RAN operational costs while satisfying the needs of MEC.

Quality of service has been a highly researched topic in networking. Work has been done for QoS provisioning in IoT networks, including scheduling techniques, monitoring, resource estimation, and cooperative communication for provisioning QoS. A study particularly focuses on which layers of IoT architecture have been researched the most concerning QoS, which quality factors have been considered for measuring performance, and what kind of research has been conducted in this area [27]. An architecture for the virtualization of IoT services is proposed in [28] to satisfy user-requested QoS requirements. A simple and generic model is proposed in [29] deploying QoS aware IoT applications to fog infrastructures. Some researchers have proposed models to support the QoS-aware deployment of IoT applications over fog resources [30]. A probabilistic analysis is performed for QoS provisioning in [31] for IoT in LTE-A based heterogeneous networks with partial spectrum usage. Effective bandwidth concept is leveraged to ensure users are provided probabilistic QoS guarantees. In [32], an approach for efficient network planning with QoS constraints is proposed for IoT networks. The paper provided a low-cost solution by minimizing the

cost of deployed devices and resources in the network while achieving minimal QoS requirements specified. A three-layered QoS architecture for IoT is proposed in [33]. The lower perception layer collects data from the environment. The middle network layer is responsible for data forwarding, whereas the application layer presents data to the users. Different QoS parameters for each layer have been identified; application layer (service time, delay, accuracy, load, priority), network layer (bandwidth, delay, packet loss rate, jitter), and perception layer (coverage, time synchronization, mobility).

The use of SDN in IoT networks is around for some time now. It has been discussed in several papers, including modification to OpenFlow protocol for WSNs [34], integration of arbitrary SDN controllers [35], program injection in WSN sensor nodes [36], and a software-defined sensor node [37], etc. However, we focus on the use of SDN in QoS provisioning for various communication networks. A QoS framework is proposed in [8] using SDN, where high priority traffic takes precedence over low priority traffic. A single controller is introduced in each autonomous system for this purpose. HiQoS technique ensures bandwidth guarantees for different traffic classes by using a queuing mechanism over multiple paths between source and destination [9]. To ensure QoS requirements of bandwidth and jitter, a precise bandwidth allocation scheme over multiple paths is proposed in [10], while minimizing the number of active OpenVSwitches in the network. OpenQoS solves a constrained shortest path problem for dynamic QoS routing [11]. It allows selecting a path with the least cost ensuring specific QoS requirement for path delay. A four-layered QoS provisioning architecture using an SDN controller is proposed in [12]. The user specifies an abstract task description, which is converted into network requirements, resulting in corresponding network resources being reserved using the SDN controller. An approach named LearnQoS is proposed in [13] for QoS provisioning in multimedia video services. SDN is integrated with machine learning and policy-based network management for compliance with QoS requirements. For guaranteed service level agreements in multi-tenant networks where multiple tenants share a common network, an SDN based approach is presented in [14]. Challenges for QoS provisioning in such networks where multiple tenants compete for network resources are being addressed. For 5G networks with multimedia video communications comprising 70 percent of mobile traffic, a statistical delay bounded QoS provisioning approach is presented in [15] to accommodate time-sensitive and bandwidth-intensive applications. SDN is used for dynamic reconfiguration of 5G wireless and radio access resources.

There has been recent research interest in client-based RAT selection in 5G and beyond 5G cellular networks. In [6], an overall utility function maximization approach for all clients in a particular 5G access network is proposed, with certain constraints from each client. A dual-sim User Equipment (UE) based RAT selection approach is presented in [16]. The client is simultaneously connected with two different base stations from two different networks. RAT selection is based on call quality, cost of making a call, power consumption at UE, and handover rate offered by base stations from each RAT. Work has been done for cooperative network prediction by fusing knowledge from the client and network domains [17]. The paper presents a proof-of-concept for 6G networks, arguing that better data rates could be achieved by cooperatively involving UEs and network devices to make machine learning-based network optimization decisions compared to network-centric approaches. In a context-aware radio access technology selection technique, client and network contexts are considered for choosing a RAT [18]. Results show such a context-aware approach outperforms conventional RAT selection approaches like received signal strength, number of handovers, delay, and throughput, etc. A multi-homing approach for 5G heterogeneous networks is presented in [19], where various traffic classes can be transmitted over different RATs by the client. A distributed decision making client-centric algorithm is proposed, making a polling order for allocating network resources for different classes. The literature discussed here for using the SDN controller in QoS provisioning and client-based RAT selection is summarized in Table 1.

A. CONTRIBUTION

As discussed in the literature review and shown in Table 1, only network-centric QoS provisioning or client-centric RAT selection techniques exist in the literature. This paper presents the first hybrid end-to-end QoS provisioning technique to the best of our knowledge, combining client-centric and SDN based network-centric approaches. The proposed architecture achieves QoS provisioning by solving two sub-problems; access-QoS and core-QoS. Major contributions of this paper are as follows:

- Access-QoS: An optimization-based approach is presented for access device selection at the client-side. With access-QoS being the client's responsibility, a client can better select access devices from various RATs.
- Core-QoS: An optimal QoS provisioning approach is presented, which SDN controllers use to reserve network resources in the core network as per client requirements.
- Heterogeneous QoS Parameters: The approach allows to specify different access-QoS and core-QoS parameters, which are more suitable for each network segment than the same QoS parameters for the whole network. Each client also has the ability to use QoS parameters different from other clients.
- Scalable End-to-End QoS Provisioning: As the access-QoS is shifted to the client, the load of last hop QoS provisioning is shifted from the network to the client. This would reduce the amount of signaling information required to inform client-specific requirements to the SDN controller and switching decision information sent from the controller to the clients in such large IoT networks. The reduced signaling improves

TABLE 1. Summary of SDN based QoS provisioning and client based RAT selection techniques.

Referred Study	Decision	Solution Method	Access Network Parameters	Core Network Parameters	Calculation Load	Network/Application Type	SDN Controller
[8]	Network Centric	Routing Protocol: BGP and OSPF	-	Traffic Priority Class	Controller	Multipath Routing	Floodlight
[9]	Network Centric	Path Cost Calculation	-	Delay	Controller	Data Center	Floodlight
[10]	Network Centric	Optimization Theory	-	Bandwidth, Delay	Controller	Multipath Routing	Floodlight
[11]	Network Centric	Optimization Theory	-	Delay	Controller	IoT	Floodlight
[12]	Network Centric	Genetic Algorithm	-	Delay, Jitter, Throughput	Controller	IoT	Custom
[13]	Network Centric	Q-Learning Method	-	Bandwidth, Packet Loss Rate	Controller	Multimedia Video Services	Floodlight
[14]	Network Centric	Optimization Theory	-	Bandwidth	Controller	Multi-tenant Networking	Floodlight
[15]	Network Centric	Optimization Theory	-	Delay	Controller	5G Networks	Custom
[6]	Client Centric	Optimization Theory	Throughput	-	Client	5G Networks	-
[16]	Client Centric	Path Performance Calculation	Call Quality, Financial Cost, Power Consumption, Handover Rate	-	Client	5G Networks	-
[17]	Client Centric	Machine Learning	Throughput	-	Client	6G Networks	-
[18]	Client Centric	Analytical Hierarchical Process	Packet Delivery Ratio, Throughput, Number of Handovers	-	Client	5G Networks	-
[19]	Client Centric	Optimization Theory	Traffic Class	-	Client	5G Networks	-
Proposed Approach	Hybrid	Optimization Theory	Data rate, Delay, SINR and other access network parameters	Throughput, Delay, Jitter, Packet Loss Rate and other core network parameters	Controller and Client	Beyond 5G IoT Networks	Ryu

network scalability. As an SDN-based approach is used for core-QoS provisioning, the load is completely shifted from network core devices to SDN controllers. To mitigate any possible SDN controller scalability issues, multiple controllers could be used [7].

III. QoS PROVISIONING TECHNIQUE

A. ARCHITECTURE

The proposed hybrid QoS provisioning architecture comprises a four-layered model, as shown in Figure 1; end device layer, access layer, controller layer, and core layer. A layered architecture helps to achieve reuse of various resources, better designing of network services and applications.

1) LAYER-1: END DEVICE LAYER

In this layer, end devices or hosts exist, which could connect to different access devices in the access layer. Today, end devices also have the capability to connect to multiple access devices. For example, mobile devices can connect to a cellular network (like 4G or 5G) and a WiFi network simultaneously. A sensor network would contain sensor nodes in this layer, which are responsible for transmitting sensed data. Other examples of devices that could exist in the end device layer are personal computers, laptops, and micro-controllers, etc.

2) LAYER-2: ACCESS LAYER

The access layer considered in this architecture has multiple access networks. Such networks are referred to as Heterogeneous Networks (HetNets). Access devices communicate directly with end devices and are located at a one-hop distance from them. Depending upon the access technology used, access device varies as well. For example, the cellular network access layer may contain a Base Transceiver Station (BTS) or an eNodeB. Similarly, an Access Point (AP) acts as an access device in a WiFi network. A Base Station (BS) acts as an access device for WSNs. These access devices can communicate with devices in both the controller layer (control traffic) and the core layer (data traffic).

3) LAYER-3: CONTROLLER LAYER

The controller layer comprises of SDN controllers. The access layer queries the controller layer only if it does not

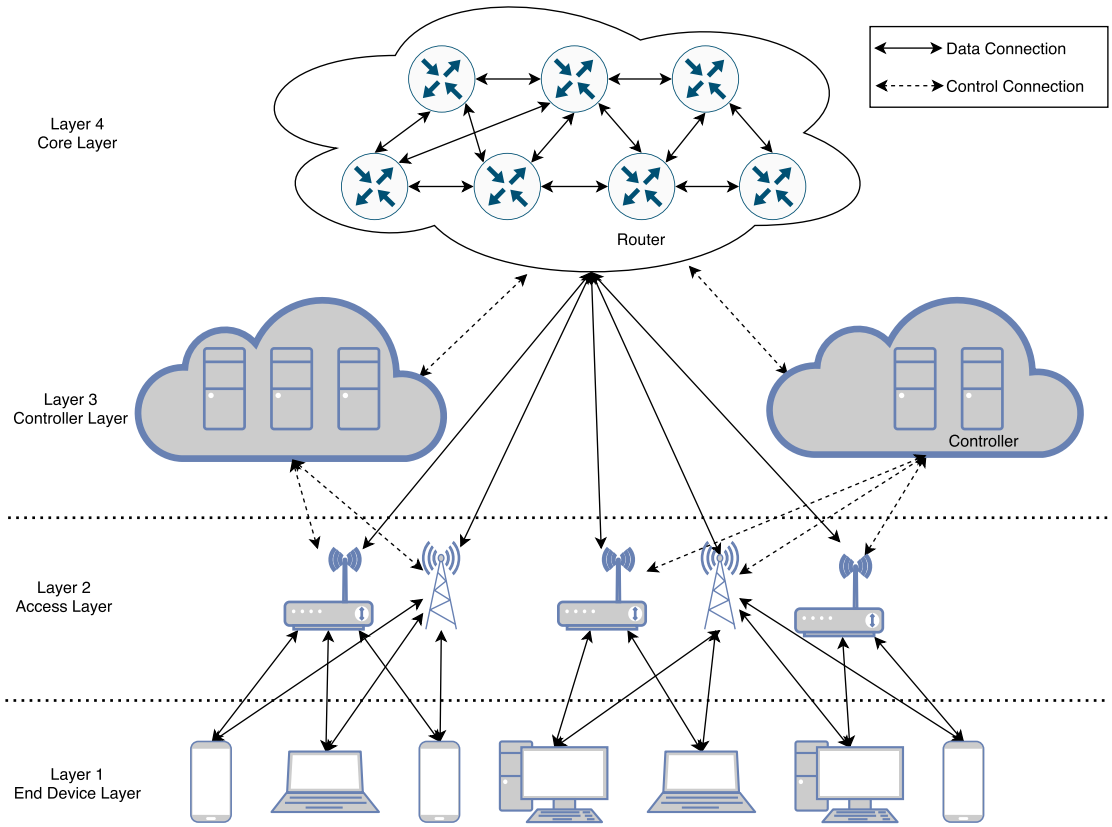


FIGURE 1. Proposed layered architecture for QoS provisioning.

have a rule for forwarding such traffic. Depending on the core-QoS logic implemented in the controller, a controller instructs access layer devices and core layer devices on how to forward the packet and may also install a flow in the flow table of these devices, instructing them on how to deal with similar packets in the future. The controller layer also receives information about complete network topology from the core layer, along with network parameters for each link. This helps it to build network topology of the core layer, which is vital for core-QoS provisioning. Controllers only communicate over control connections, and there is no actual data traffic transmitted to and from controllers to the access layer and the core layer. As end devices and access layer devices are responsible for mobility and handovers, such information does not necessarily need to be relayed to controllers. Controllers can automatically detect the currently active access device from the incoming traffic. In this way, controllers are mainly responsible for forwarding traffic, and the possible latency that could be caused by mobility and handovers in controllers could be avoided.

Multiple controllers can be deployed in this layer for the purpose of scalability. As B5G networks are expected to have billions of IoT devices with each device having multiple flows, packet processing at a single controller may cause processing and queuing delays, as well as overflowing of queues. It would not be possible for a single controller to handle this massive amount of incoming data. Therefore, multiple controllers are usually deployed to handle incoming data. With multiple controllers at this layer, a single point of failure can also be avoided, as with a single controller, the failure of this controller may result in no traffic to flow from the access layer to the core layer. Multiple controllers also add resiliency to the network in case of attacks like Denial of Service (DoS) and Distributed Denial of Service (DDoS).

Controllers do not necessarily need to have connections with the core layer. To ensure QoS provisioning, controllers need to know the core layer's exact topology. In communication networks with changing network conditions, it would be challenging to update this information at controllers manually. Therefore, it is vital to provide a control connection between core and controller layers to automatically propagate the core layer's changing network conditions to controllers.

4) LAYER-4: CORE LAYER

The core layer consists of devices usually responsible for carrying data from one network to another. This layer consists of a massive number of interconnecting devices like routers and Layer-3 switches. With the control plane moved to SDN controllers in Layer-3, the core layer devices are only responsible for performing the functionality of forwarding data from

source to destination through a specific path provided by the controller. The source and destination networks can be part of the same Internet Service Provider (ISP) or may have to traverse through an Internet Exchange Point (IXP).

B. SYSTEM MODEL AND PROBLEM FORMULATION

The proposed model is a hybrid model, implying that both end-devices and controllers participate in improving the quality of service available to the end-user. The optimization problem formulations at both devices are specified in the coming sub-sections. Similar system models are used in literature for both access-network based RAT selection and core-network based QoS provisioning [6], [11].

1) ACCESS-QoS

A distributed optimization approach is proposed, where each client tries to maximize some utility function, subject to certain constraints. We consider a multi-RAT network, with each RAT having one or more access devices in the access layer. Let $\mathcal{V}$ be the set of access devices in the access layer, which are geographically distributed, each being connected to the controller cloud in the controller layer. Let $\mathcal{N}$ be the set of IoT nodes in the end device layer, each using different applications and requiring different QoS levels.

An IoT node $n \in \mathcal{N}$ is connected to multiple access devices. Each access device $v \in \mathcal{V}$ also has multiple IoT nodes connected to it. Let $x_v^n \in \{0,1\}$ be an assignment indicator, where $x_v^n = 1$ means that node n is connected to access device v, otherwise, $x_v^n = 0$. Each access device v is constrained by the maximum number of IoT nodes X_v that can connect to it simultaneously, as well as the maximum workload $W_v = \sum_{\forall n} w_v^n$ from the IoT nodes communicating through it. The workload w_v^n represents the workload from the n-th node to the v-th access device, and is a function of data transmitted in bits from the n-th IoT node to the v-th access device.

The latency l_v^n of a one-hop communication link between the n-th IoT node and the v-th access device is dependent on the workload from the the n-th IoT node w_v^n and the data rate R_v available to the v-th access device:

$$l_v^n = \frac{w_v^n}{R_v} \tag{1}$$

The access device data rate R_v is defined as the sum of data rates of all IoT nodes connected to the v-th access device and is given as:

$$R_v = \sum_{n \in N_v} r_v^n \tag{2}$$

where r_v^n is the data rate of the n-th IoT node provided by the v-th access device and is defined as follows:

$$r_v^n = \beta_v^n log_2(1 + SINR_v^n) \tag{3}$$

where β_v^n is the bandwidth assigned to the n-th IoT node from the v-th access device. $SINR_v^n$ is the signal to interference plus noise ratio at node n for the signal received from access device v and is calculated as follows:

$$SINR_v^n = x_v^n \frac{p_v^n h_v^n}{\sum_{v' \neq v \in \mathcal{V}} p_{v'}^n h_{v'}^n + N_{0_v}^n} \tag{4}$$

Here p_v^n is the power received at IoT node n from the access device v, h_v^n is the channel gain between IoT node n and the access device v, whereas $N_{0_v}^n$ represents the noise signal. The channel gain h_v^n is defined as:

$$h_v^n = \bar{h_v^n} \zeta G_{0_v}^n (\frac{d_{0_v}^n}{d_v^n})^{\alpha} \tag{5}$$

In the above equation, for a link between the n-th node and the v-th access device, $G_{0_v}^n$ is the antenna gain, ζ is a zero-mean Gaussian random variable with σ standard deviation, d_v^n is the distance between IoT note n and access device v, $d_{0_v}^n$ is the reference distance of antenna far-field, path loss exponent is α, and Rayleigh random variable is given as $\bar{h_v^n}$.

Access-QoS can be defined as providing a certain level of guarantees for parameters like data rate, delay, jitter and packet loss, etc. An IoT node can build constraint over any of these parameters and try to maximize or minimize certain parameters simultaneously. In this paper, we formulate a rate optimization problem that distributively runs at each node, with QoS constraints of SINR and delay. Similar optimization problems can be constructed for various other access-QoS parameters. Assuming an IoT node n is connected to a subset $\bar{\mathcal{V}}$ from a set $\mathcal{V}$ of all access devices, i.e. $\bar{\mathcal{V}} \subseteq \mathcal{V}$, the objective function would then be given as:

$$\max_{SINR,l} \quad \sum_{v \in \mathcal{V}} x_v^n r_v^n \quad \forall \ n \in \mathcal{N} \tag{6}$$

subject to:

$$C1: \quad SINR_{\bar{v}}^n \geq S_n \quad \forall \ \bar{v} \in \bar{\mathcal{V}} \tag{7}$$

$$C2: \quad l_{\bar{v}}^n \leq L_n \quad \forall \ \bar{v} \in \bar{\mathcal{V}} \tag{8}$$

$$C3: \quad \sum_{v \in \mathcal{V}} x_v^n \leq M_n \quad \forall \ n \in \mathcal{N} \tag{9}$$

TABLE 2. Access-QoS system model abbreviations with descriptions.

Abbreviation	Description
$\mathcal{V}$	Set of access devices
$\mathcal{N}$	Set of IoT nodes
x_v^n	Assignment indicator of node n with access device v
w_v^n	Workload from node n to access device v
l_v^n	Latency between node n and access device v
r_v^n	Data rate provided to node n from access device v
β_v^n	Bandwidth assigned to node n from access device v
$SINR_v^n$	Signal to interference plus noise ratio between node n and access device v
p_v^n	Power of signal received at node n from access device v
h_v^n	Gain of channel between node n and access device v
$N_{0_v}^n$	Noise of signal received at node n from access device v
S_n	Minimum required SINR at node n
L_n	Maximum allowed latency at node n
M_n	Multi-homing parameter for node n
R_v	Data rate available to access device v
W_v	Maximum workload at access device v
X_v	Maximum allowed nodes at access device v

$$C4: \quad \sum_{n \in \mathcal{N}} x_{\bar{v}}^{n} \leq X_{\bar{v}} \quad \forall \ \bar{v} \in \bar{\mathcal{V}} \tag{10}$$

$$C5: \quad \sum_{n \in \mathcal{N}} w_{\bar{v}}^{n} \leq W_{\bar{v}} \quad \forall \ \bar{v} \in \bar{\mathcal{V}} \tag{11}$$

Here constraints C1 and C2 are the access-QoS provisioning constraints. The constraint C1 ensures that the SINR on the link with an access device is at least greater than or equal to the required SINR threshold S_n. Similarly, C2 ensures that the delay on the link with an access device is less than or equal to the required delay threshold L_n. The constraint C3 limits the maximum number of access devices a particular node n can connect at the same time to M_n, where M_n is the multi-homing parameter. The constraint C4 limits the number of nodes connecting to a particular access-device $\bar{v}$ concurrently not to exceed $X_{\bar{v}}$. The last constraint C5 ensures that the workload from all nodes connected to a specific access device $\bar{v}$ does not exceed the allowed maximum workload threshold of $W_{\bar{v}}$.

2) CORE-QoS

The hybrid architecture leaves the selection of the most suitable access device for access-QoS provisioning to the client. SDN controllers perform path selection for core-QoS provisioning in core networks. This is because the clients are unaware of the network resources available in the core network, which the controllers know. That is why clients have to rely on controllers to do the core-QoS provisioning job.

We assume a core network with e network quality parameters. These network parameters could be classified into four different categories, as listed in Table 3. Let us define a Path Cost matrix $\mathbf{D}$ containing specific values of network parameters that should be considered for path cost calculation (Table 3, Category No. 2 & 3): $\mathbf{D} = [\mathrm{d}_1\ \mathrm{d}_2\ \ldots \mathrm{d}_\mathrm{e}]$, where d_i is the value of the i-th network parameter used for path cost calculation. However, based on the requirement and kind of application running at each client, every client may not request for each network parameter (Table 3, Category No. 1 & 4) to be considered for cost calculation, setting $d_i = 0$ for the i-th such network parameter.

Next, we define QoS matrix $\mathbf{Q}$, which contains the QoS thresholds for network parameters for which QoS is requested by the client (Table 3, Category No. 1 & 3): $\mathbf{Q} = [\mathrm{q}_1\ \mathrm{q}_2\ \ldots \mathrm{q}_\mathrm{e}]$, where q_i is the threshold of the i-th network parameter, which must be satisfied by a specific network path. For unconstrained network parameters (Table 3, Category No. 2 & 4), corresponding values in matrix $\mathbf{Q}$ should be set to ∞.

Next we define two selection variables; path cost selection variable x_i and QoS selection variable y_i, where $x_i \in \{-1, 0, 1\}$ and $y_i \in \{0, 1\}$. The value of x_i identifies if a particular network parameter is considered for path cost calculation (value of -1 or 1) or not (value of 0). For network parameters where a higher value is desired like bandwidth or SINR, x_i equals 1, whereas, for parameters where a lower value is desired, like delay and jitter, x_i equals -1. For the second selection variable y_i, the value is 0 if the i-th network parameter has a lower desired value and 1 if the i-th network parameter has a higher desired value.

TABLE 3. Classification of Core-QoS network parameters.

Category No.	Description
1	Core-QoS parameter which must satisfy a particular threshold
2	Cost parameter considered in the cost calculation of the network path
3	Both core-QoS and cost parameter
4	Not considered

If there are f paths from a particular source to destination, contained in a set $\mathcal{P}$, then each of these f paths will have a particular value for each of e network parameters. The matrix A of actual values of network parameters is then defined as:

$$A = \begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1e} \\ a_{21} & a_{22} & \ldots & a_{2e} \\ \vdots & \vdots & \vdots & \vdots \\ a_{f1} & a_{f2} & \ldots & a_{fe} \end{bmatrix} \tag{12}$$

Here, a_{11} might be the bandwidth, a_{12} might be the delay, and a_{1e} might be the packet loss rate across the first path. Similarly, a_{f1}, a_{f2}, and a_{fe} would represent the actual bandwidth, delay, and packet loss rate respectively across the f-th path.

For core-QoS provisioning, a controller tries to find a path $p \in \mathcal{P}$ from a source to the destination. If $C(p)$ is the path cost along a path p, the minimum path cost can then be calculated as:

$$\{\min(C(p)) \mid 1 \leq p \leq f\} \tag{13}$$

where:

$$C(p) = \sum_{i=1}^{e} x_i \frac{d_i - a_{pi}}{a_{pi}} \qquad a_{pi} \neq 0 \tag{14}$$

subject to:

$$C6: \quad \begin{cases} \forall_i a_{pi} \leq q_i & \text{if } y_i = 0 \\ \forall_i a_{pi} > q_i & \text{if } y_i = 1 \end{cases} \tag{15}$$

Here C6 is the QoS provisioning constraint for the core network. C6 could be a single constraint or multiple constraints depending on the number of network constraints defined (Table 3, Category No. 1 & 3). This constraint makes sure that the actual value of every network parameter along the selected path must be greater than or equal to the required QoS value for parameters with a higher desired value or lower than the QoS value for the parameters with lower desired values.

Example: Consider a network with four parameters; bandwidth, jitter, delay, and packet loss ratio. A particular client requests a path with cost consideration of bandwidth as 1 Mbps and delay as 200 msec. The path cost matrix would then be: $\mathbf{D} = [1\ 0\ 200\ 0]$. For this particular client, the path cost selection variables would be $x_1 = 1$, $x_2 = 0$, $x_3 = -1$, and $x_4 = 0$.

The client also requests for QoS provisioning of delay and packet loss ratio at 300 msec and 0.01, respectively. The

TABLE 4. Core-QoS system model abbreviations with descriptions.

Abbreviation	Description
$\mathbf{D}$	Path cost matrix
$\mathbf{Q}$	QoS matrix
x_i	Path cost selection variable for network parameter i
y_i	QoS selection variable for network parameter i
$\mathcal{P}$	Set of all paths from a source to destination in core network
f	Total number of paths from a source to destination in core network
e	Total number of network quality parameters
$\mathbf{A}$	Matrix of actual values of network parameters in a core network
$C(p)$	Cost of path p

resultant QoS matrix would be: $\mathbf{Q}$: $= [\infty\ \infty\ 300\ 0.01]$. The interesting thing is that the client has considered the delay in both QoS request and path cost calculation. However, the QoS requirement is 300 msec, and the contribution of delay in the path cost calculation is 200 msec. This means that the path will still be considered if the delay is less than 300 msec for a particular path. However, if the delay is greater than 300 msec for a path, then the path will not be considered as it is violating the core-QoS constraint C6, mentioned in equation (15). In this scenario, the QoS selection variables would have values of $y_1 = 1$, $y_2 = 0$, $y_3 = 0$, and $y_4 = 0$.

C. PROPOSED ALGORITHMS

This sub-section presents algorithms for the proposed access-QoS and core-QoS provisioning problem formulations. The access-QoS algorithm makes use of the same abbreviations as defined for access-QoS problem formulation in Table 2. Similarly, the core-QoS algorithm uses the same abbreviations as defined for core-QoS problem formulation in Table 4. However, a few new variables are used in these algorithms, which are defined as used.

Algorithm 1 is a rate-maximization algorithm and selects an access-device in the access-network based on a single parameter. On the other hand, Algorithm 2 is a cost-minimization algorithm that calculates the minimum cost path in the core-network based on multiple QoS parameters, as per the client's requirements. Both algorithms could have very different QoS parameters because of the different nature of the network segments and the different devices running these algorithms. Several utility maximization algorithms similar to Algorithm 1 are proposed in the literature. However, a novel algorithm using core network parameters simultaneously for path cost calculation and QoS constraints is proposed in Algorithm 2.

1) ACCESS-QoS

The access-QoS provisioning algorithm for a specific node n is considered in Algorithm 1. The algorithm takes various radio and device-specific parameters as input. All possible binary combinations for x_v^n are calculated in line 3 and assigned to X. Line 4 and line 5 initializes an empty set to the list of access devices AD and assigns a null value to rate calculation variable R, respectively.

Algorithm 1 Access-QoS Algorithm

1: Input: n, β, $SINR$, S_n, L_n, M_n, R_v, w_v^n
2: Initialize M_n
3: $X \leftarrow$ all possible binary combinations for x_v^n
4: $AD \leftarrow \{\}$
5: $R \leftarrow 0$
6: **for** i in X **do**
7: $\quad R' \leftarrow 0$
8: $\quad AD' \leftarrow \{\}$
9: $\quad$ **for** v in $\mathcal{V}$ **do**
10: $\quad\quad x_v^n \leftarrow i_v$
11: $\quad\quad l_v^n \leftarrow (w_v^n / R_v)$
12: $\quad\quad$ Initialize β_v^n, $SINR_v^n$, S_n and L_n
13: $\quad\quad$ **if** $x_v^n = 1$ AND $SINR_v^n \geq S_n$ AND $l_v^n \leq L_n$ **then**
14: $\quad\quad\quad r_v^n \leftarrow \beta_v^n log_2(1 + SINR_v^n)$
15: $\quad\quad\quad R' \leftarrow R' + (x_v^n \cdot r_v^n)$
16: $\quad\quad\quad AD' \leftarrow AD' \,||\, v$
17: $\quad\quad$ **end if**
18: $\quad$ **end for**
19: $\quad$ **if** $R' > R$ **then**
20: $\quad\quad AD \leftarrow AD'$
21: $\quad\quad R \leftarrow R'$
22: $\quad$ **end if**
23: **end for**

The access-QoS algorithm consists of two nested for-loops. The inner for-loop from lines 9 to 18 uses a single possible combination of x_v^n in each iteration and calculates the rate available R', as well as the set of access devices AD', to which the node n will connect for such a combination of x_v^n. The outer for-loop runs from lines 6 to 23 and iterates over all possible combinations of x_v^n. Each value of x_v^n is input to the inner for-loop. If the inner for-loop computes a new higher rate, the outer for-loop updates the rate R, as well as the set of access devices AD, from which such a rate is achieved.

2) CORE-QoS

The Core-QoS algorithm is listed as Algorithm 2. This algorithm takes three matrices D, Q, and A as input. A variable SP to store the shortest-cost path's path number is initialized to zero in line 2. The core-QoS algorithm also makes use of two nested for-loops. The inner for-loop from lines 5 to 14 computes the path cost C' by iterating over each network quality parameter. If a specific path does not satisfy the QoS constraint(s), C' is considered ∞ for such a path. The outer for-loop from lines 3 to 19 iterates over each path from a source to a destination. If the inner for-loop discovers a new shortest path-cost, it is updated in the least-cost variable C, and the path number is stored in SP.

D. COMPLEXITY OF ALGORITHMS

In this sub-section, we evaluate the complexity of both access-QoS and core-QoS provisioning algorithms. Complexity is measured by counting the number of flops, where

Algorithm 2 Core-QoS Algorithm

```
1:  Input: D, Q and A
2:  SP ← 0
3:  for p ← 1 to f do
4:      C ← ∞
5:      for i ← 1 to e do
6:          C′ ← 0
7:          Initialize x_i, y_i, d_i, q_i and a_pi
8:          if (y_i = 0 AND a_pi ≤ q_i) OR (y_i = 1 AND a_pi > q_i)
            then
9:              C′ ← C′ + x_i((d_i − a_pi)/a_pi)
10:         else
11:             C′ ← ∞
12:             break
13:         end if
14:     end for
15:     if C′ < C then
16:         C ← C′
17:         SP ← p
18:     end if
19: end for
```

each flop is considered as a real floating-point operation. We consider the following rules for calculating flops [38]:

- One flop for each real addition, subtraction, multiplication, or division operation
- One flop for each assignment operation
- One flop for each logical operation
- One flop for each addition or removal of an element from a set

1) ACCESS-QoS

Algorithm 1 for access-QoS provisioning algorithm takes various radio, and device-specific parameters as input in line 1. M_n is assigned a value from the input in line 2, which will take one flop. All possible binary combinations for x_v^n are calculated in line 3 and assigned to X. This will take $|\mathcal{V}|^2$ flops for computation of combinations, and $|\mathcal{V}|^2$ flops for assignment to X. However, possible combinations of interest for x_v^n are only those in which at least one x_v^n equals 1, and also $x_v^n \leq M_n$. If Z be the number of reduced combinations of interest, then Z can be defined as $Z = \sum_{k=1}^{M_n} |\mathcal{V}|!/(k!(|\mathcal{V}|-k)!)$. Therefore, the flops required by line 3 are reduced from $2|\mathcal{V}|^2$ to $2Z$. Lines 4 and 5 use one flop each to initialize an empty set AD and assign a null value to variable R, respectively.

The inner for-loop from lines 9 to 18 uses three flops in line 9 to assign value to v, incrementing its value, and perform a logical operation to check if v is still in $\mathcal{V}$. Line 10 uses one flop for assigning value to x_v^n, line 11 uses two flops for division operation and value assignment to l_v^n, whereas four flops for initializing input to the corresponding variables are used in line 12. Line 13 uses five flops to perform five logical operations. Five flops are used in line 14 for performing four arithmetic operations and one assignment operation to calculate the value of l_v^n. Similarly, three flops are used by line 15 to perform two arithmetic operations and then assigning the computed value to R'. Line 16 adds an element to the set AD', using one flop. The inner-for loop in total uses 24 flops. As the inner for-loop is repeated $|\mathcal{V}|$ times, the flops used by the inner for-loop would be $24|\mathcal{V}|$.

The outer for-loop from lines 6 to 23 also uses three flops in line 6 for assigning value to i, incrementing the value of i, and to check if i is still in the vector X. Lines 7 and 8 use one flop each to assign a null value to R', and an empty set to AD' respectively. Line 19 uses one flop to perform a logical comparison between R' and R. If true, line 20 copies this set of access devices AD' to AD. In the worst case, the length of AD' would be $|\mathcal{V}|$. For updating AD, we might need to pop at most $|\mathcal{V}|$ elements from AD. Therefore, we consider line 20 takes $2|\mathcal{V}|$ flops. The new higher rate R' is assigned to R in Line 21, utilizing one flop. In total, one iteration of the outer for-loop would take $7+26|\mathcal{V}|$ flops. As the outer-loop iterates Z times, the total number of flops required would then be $(7+26|\mathcal{V}|)Z$. Adding the number of flops used by lines 2 to 5, the overall flop count of Algorithm 1 F_{A1} can be given as:

$$F_{A1} = 3 + (9 + 26|\mathcal{V}|)Z \tag{16}$$

$$F_{A1} \approx (9 + 26|\mathcal{V}|)Z \tag{17}$$

As we have two nested loops from 1 to Z, and from 1 to $|\mathcal{V}|$ respectively, the algorithmic complexity in Big O notation will be quadratic i.e. $O(Z \times |\mathcal{V}|)$.

2) CORE-QoS

The core-QoS algorithm is listed as Algorithm 2, which consists of two nested for-loops. A variable SP to store the shortest-cost path's path number is initialized to zero in line 2, using one flop. The inner for-loop from line 5 to line 14 uses three flops in line 5, one each for assigning value to i, updating the value of i, and logical operation to check if the value of i is still less than or equal to e. Line 6 uses one flop for assigning null value to C', whereas, line 7 uses five flops for initializing five different variables i.e. x_i, y_i, d_i, q_i, and a_{pi}. Line 8 performs seven logical operations, thus using seven flops. Line 9 uses five flops for performing four arithmetic operations and one assignment operation to evaluate the value of C'. The body in else-condition uses two flops, one for line 11 to assign infinite value to C', and one for line 12 to break the inner for-loop. As, if-condition requires five flops, and else-condition requires two flops, the worst-case scenario of five flops is considered for complexity calculation. In total, the inner for-loop from lines 5 to 14 uses twenty-one flops for each iteration. Thus, the inner for-loop will use $21e$ flops in total.

In the outer for-loop, line 3 uses three flops for assignment of value to p, incrementing the value of p, and logical operation to check if the value of p is still less than or equal to f. In line 4, one flop is used to initialize C. For the if-condition from lines 15 to 17, one flop is used per line. If a newer

shortest-cost path is discovered, the path's cost is updated from C' to C, and the path number is stored in SP. In total, the outer for-loop will use $7+21e$ flops for each iteration. As the outer for-loop repeats f times, the number of flops required would then be $(7+21e)f$. Adding one additional flop required by line 2, the overall flop count of Algorithm 2 F_{A2} can thus be given as:

$$F_{A2} = 1 + (7 + 21e)f \tag{18}$$

$$F_{A2} \approx (7 + 21e)f \tag{19}$$

As we have two nested loops from 1 to e, and from 1 to f respectively, the algorithmic complexity in Big O notation will be quadratic i.e. $O(e \times f)$.

E. OPTIMALITY OF ALGORITHMS

Problem formulation or specifically mathematical model of a given problem determines the convexity of the optimization problem. Convex optimization problems offer some important and useful properties. Local minima of the convex optimization problems serve as global minima of the problem guaranteeing the achievable optimal point. However, to determine the convexity, second-order partial derivatives of the objective function are required. If the Hessian matrix, based on these second-order partial derivatives, comes out to be positive definite, then the problem is classified as convex. Mix integer convex problems are considered a subset of linear problems in which constrained variables can take integer or non-integer values. Unlike integer convex problems where the solution must be limited to the integer values, a mixed-integer solution has a bigger search space. Both access-QoS and core-QoS optimization problems are classified as linear, mix integer convex optimization problems with a bounded optimal solution.

A separate or distributed optimization problem is defined for access-QoS. Each IoT node separately runs the optimization algorithm to maximize its objective function. As each node tries to optimize the access device selection based on the local view available to the node, the solution achieved might not be the most optimal solution from the network's point of view. The core-QoS optimization is classified as a multi-constrained optimal shortest-path selection problem. Subject to constraints in equation (15), the optimization problem will achieve an optimal solution by finding a minimal-cost path between a source and a destination.

IV. EXPERIMENTATION SETUP

Evaluation of our proposed approach requires an experimental setup for both access-QoS and core-QoS provisioning.

A. ACCESS-QoS

For the access-QoS provisioning experimentation, an indoor network scenario with two WiFi APs is considered. The experimentation is first performed using a network emulation setup and then using a hardware experimentation setup.

1) NETWORK EMULATION

For network emulation based setup, an indoor network scenario is considered with two WiFi APs, as shown in Figure 2a. The total field size considered for experimentation is 140 × 100 meters, whereas, the range of both APs in each direction is 50 meters. The first AP ap1 is located at position (50,50), whereas, the second AP ap2 is situated at position (90,50). Since the nodes should have access to at least two APs to switch opportunistically between them, the area of interest is the overlapping coverage area of both APs, shown with dark purple color in Figure 2a. Therefore, network topologies where nodes are only placed in such an overlapping region are considered in the experimentation setup. One such topology with a network size of 20 nodes is shown in Figure 2b.

For experimentation, network sizes where 10, 20, and 40 nodes are located in the overlapping region are considered. The nodes are randomly placed in the overlapping region in each instance of the experimentation. The Log-distance path loss model, being simple and widely used in modeling WiFi networks [39], is used as a propagation model. Both APs used are 802.11g based and operate in separate channels, with channel 1 used for AP ap1, and channel 6 used for AP ap2.

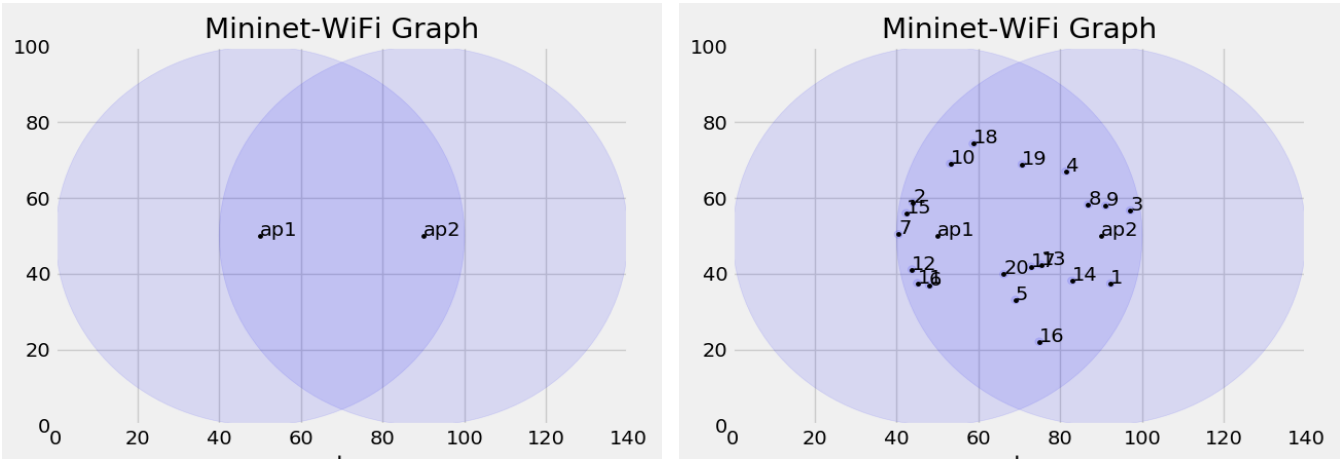
For emulating such an access network, we used Mininet-WiFi [40] network emulator, a fork from earlier released Mininet [41]. Mininet-WiFi allows Python-based code execution without any modification in either kernel or applications and supports the Linux mac80211 framework, allowing testing most of the IEEE 802.11 functionalities.

2) HARDWARE EXPERIMENTATION

For hardware experimentation setup, two WiFi APs setup similar to the network emulation setup is considered. As shown in Figure 3, a WiFi AP is placed each in Room 1 and Room 2 of an office building, with a separation between them of 6 meters along the x-axis and 20 meters along the y-axis. Both APs are connected to the intranet over a Fast Ethernet connection. The end-devices can be placed in the building corridor labeled as node placement area in Figure 3, and is shown with a blue background. This blue background area falls in the coverage area of both WiFi APs.

Two Raspberry Pi 4 are used as WiFi APs, using IEEE 802.11ac wireless network interface card, with an operating frequency of 2.4 GHz. Three different end-devices are used, including Galaxy Samsung S10, Google Pixel C Tablet, and Motorola Nexus 6. All the devices use LineageOS custom ROM as an operating system, a customized Android version. An android application is also installed in all the devices, which automatically and opportunistically switches between multiple WiFi APs.

For experimentation, all three devices are placed in the overlapping coverage area of both APs. Each device is placed at multiple positions in the node placement area. On the x-axis, the devices' position is fixed with a distance of 3 meters from both APs. Whereas, on the y-axis, all



(a) APs placement (b) Network with 20 nodes

FIGURE 2. Network emulation based 140 × 100 meters topology for Access-QoS.

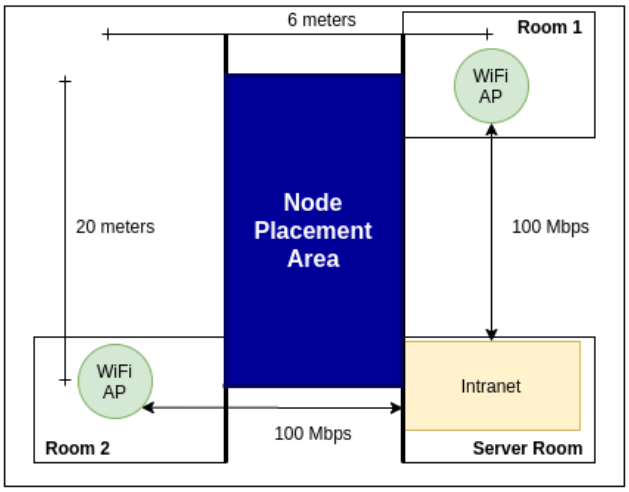


FIGURE 3. Hardware experimentation network architecture for Access-QoS.

devices' position is varied from 1 to 20 meters from both APs, with a step-size of 1 meter.

B. CORE-QoS

We have created a testbed for core-QoS evaluation that conforms with our architecture presented in Figure 1. We have considered the example topology shown in Figure 4 for a simplified network with multiple paths to examine acceptance or rejection of QoS constraints required by the emulated testbed. Discussion related to the Autonomous Systems (ASes) structure of the core network is beyond this paper's scope. However, we assume that the core consists of switches from various Internet eXchange Points (IXPs), inter-connected through transit providers.

A central controller is connected to all access layer devices (S_1, S_4) and core layer devices ($S_2, S_3, S_5.S_6$) and maintains a global view as proposed in [42]. Here we have assumed that as an outcome of access-QoS provisioning, clients H_1

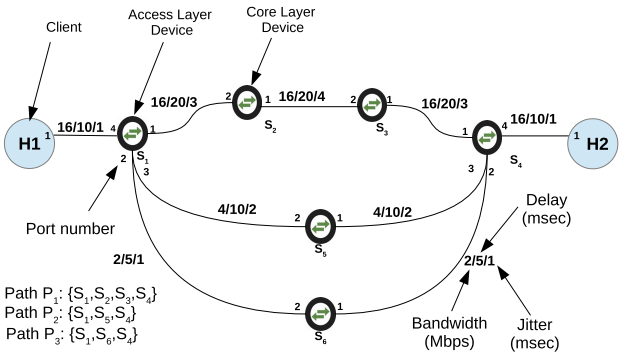


FIGURE 4. Network topology for evaluating core-QoS provisioning.

and H_2 have decided to connect to access devices S_1 and S_4, respectively. For the sake of simplicity, only two clients are shown here. Experimental results are obtained for networks with a larger number of clients.

In this emulation, three network parameters are considered; throughput, delay, and jitter. Three paths with these network parameters can be observed through this topology. Path P_1 forms by connecting $\{S_1, S_2, S_3, S_4\}$. Formation of second path P_2 includes $\{S_1, S_5, S_4\}$. Whereas, third and the last path P_3 is established using connecting edges $\{S_1, S_6, S_4\}$ of the network. For evaluation, we have setup path P_2 with bandwidth twice the path P_3, whereas, path P_1 has exactly twice the bandwidth of path P_2. Concerning delay and jitter, P_3 has lower values than P_2, which has lower values than P_1. Values of delay over paths P_1, P_2, and P_3 are 80, 40, and 30 milliseconds respectively. Jitter over paths P_1, P_2, and P_3 are 12, 6, and 4 milliseconds respectively, as depicted in Figure 4.

We have chosen values of network parameters in this manner to evaluate our proposed approach for two different QoS provisioning problems using the same network topology. For the first QoS problem, we have considered bandwidth as a constraint only. Therefore, we have considered

different bandwidths for each path. The first path P_1 has bandwidth twice of P_2, and path P_2 has twice the bandwidth of path P_3. In the second QoS problem, we have considered a multi-constrained problem considering throughput, delay, and jitter constraints. In order to evaluate the performance of our proposed technique in scenarios where paths other than path P_1 are dynamically preferred, we assigned smaller delay and jitter values to path P_2 and path P_3, than path P_1. The assignment is in reverse order where path P_3 has the smallest values for delay and jitter, whereas path P_2 has a lower delay and jitter values than path P_1. Although transmission delay is inversely proportional to the link bandwidth, and path P_1 would have the smallest transmission delay. However, we can assume that the path P_1 has a longer length, resulting in higher propagation delay and a higher overall delay. Similarly, path P_2 has smaller length than path P_1, but is longer than path P_3. Similar network topologies and link parameters have been used in [9], [11], and [13].

We have emulated this experimental testbed over the Mininet emulation environment [41]. Mininet not only helps to build a virtual space consisting of network nodes and hosts by utilizing Linux kernel API, but also leverages to set various QoS parameters over the emulated network. Virtual hosts and nodes are connected together to emulate this core network. Among available options, we have selected Python-based SDN controller Ryu with proper documentation available and has been developed by observing operating system design principles [43]. Modular architecture, scalability provisions, and agile development style make Ryu a favorable choice for WAN deployments. One of its prominent deployment in the core network is the transformation of Toulouse IXP into Software Define Internet eXchange (SDX) using the Ryu controller framework [44].

V. RESULTS

In this section, we explain the results obtained for both access-QoS and core-QoS experimentation, using the setup described in section IV.

A. ACCESS-QoS

Results from the network emulation-based setup are first discussed, followed by hardware experimentation results.

1) NETWORK EMULATION

For the experimental evaluation of network emulation based setup, we have compared our Proposed Methodology (PM) with conventionally used Received Signal Strength Indicator (RSSI) based AP selection approach in today's end devices (referred here as C-RSSI). The PM distributively runs on each node and tries to maximize the rate as per our proposed Algorithm 1 in Section III. In PM, each node scans for available RSSI after a specific time interval from both APs. As soon as the RSSI for the non-connected AP is detected to be greater than the connected AP, the node automatically handovers to the non-connected AP. Whereas in C-RSSI, a node initially selects an AP based on the maximum available RSSI, but does not continuously scan for other available APs after connection. In C-RSSI, a node does not handover to a non-connected AP (even providing a higher RSSI signal), until disconnected from the connected AP.

Such a C-RSSI approach is energy efficient, but is not suitable in situations where users are mobile. The user could still be connected to a lower RSSI providing AP, while a stronger RSSI signal is available from another AP due to mobility. Such situations result in degraded user throughput where the user enters the coverage area of one AP and gets connected to it and then moves to a place where coverage of the first AP is still available, but a stronger signal is available from a second AP. For example, a user connects to a WiFi AP located at the entrance upon entering a building. Later, the user moves to his office, where he stays most of the time. A second AP near his office has stronger signal strength, but a weak signal from the first AP is still available. In such a situation, the user would remain connected to the first AP with lower signal strength in C-RSSI, whereas, PM would handover the user to the second AP with better RSSI. Similarly, passengers entering an airplane from the front would connect to the AP at the front, but might have a better signal strength from another AP at their seats, while still in the front AP's coverage area. In such a situation, C-RSSI would keep the user connected to the AP at the front, whereas, PM would shift the user to the AP near his seat.

We compared PM with C-RSSI for our experimentation topology shown in Figure 2 for a network of 10, 20, and 40 nodes. The results presented are similar for different network sizes, as the RSSI is independent of the network size. The results shown in Figure 5 are for a network size of 20 nodes. Each experiment is repeated ten times, with the average plotted as a data-point and the standard deviation as an error-bar. We define the percentage of nodes with incorrect initial AP assignment as the number of nodes out of the total number of nodes that receive a better RSSI from a non-connected AP than the connected AP. We compare the performance of PM and C-RSSI by varying this percentage of incorrectly assigned nodes.

It can be seen from Figure 5 that as the percentage of incorrectly assigned nodes increase, the average RSSI per node in C-RSSI decreases as well. This behavior is expected because as more nodes incorrectly connect to an AP with a lower RSSI signal, the average RSSI per node would decrease. However, as each node individually switches to the AP with the highest RSSI signal in the PM, the average RSSI per node remains almost constant. The C-RSSI curve can be approximated as a linearly decreasing line, whereas, a constant line can approximate the PM curve.

Next, we compute the channel capacity using the Shannon-Hartley theorem for the same network size of 20 nodes. As both APs use different channels, we consider no interference between the APs. We also assume no interference between the nodes, and the nodes are, on average, equally distributed among both WiFi APs. The available WiFi bandwidth for each channel is 20 MHz, whereas,

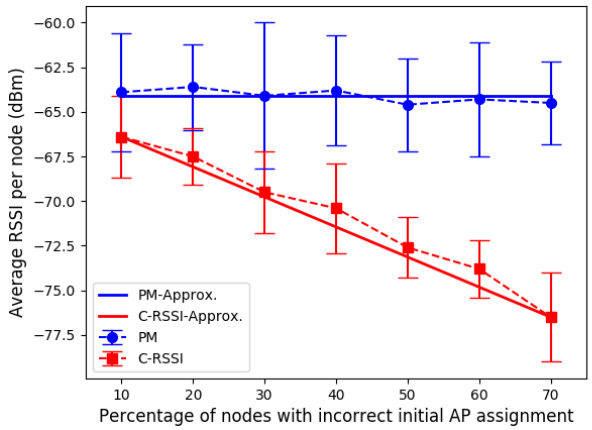


FIGURE 5. Observed average RSSI per node with increasing percentage of nodes assigned initially to incorrect AP.

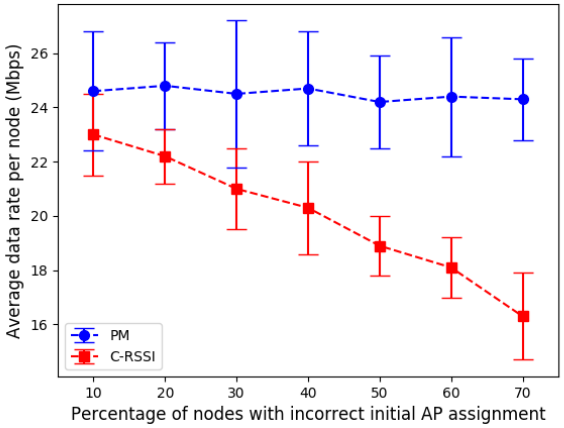


FIGURE 6. Achievable average data rate per node with increasing percentage of nodes assigned initially to incorrect AP.

Power Spectral Density (PSD) of noise considered is -174 dBm/Hz [45]. This approximately corresponds to a noise floor of -100 dBm.

A curve similar to Figure 5 can be seen for the average data rate achieved per node in Figure 6. With the same noise floor's assumption and no interference at all nodes, the RSSI maps without any variation to the data rate using the Shannon-Hartley theorem. With different noise floors and interference levels at each node, the curve might not be the same but would still show a similar trend. It can be seen from Figure 6 that a consistent average data rate of around 24 Mbps per node can be achieved by PM. Whereas for C-RSSI, as the percentage of incorrectly assigned nodes increases to 70%, the data rate reduces from around 24 Mbps to 16 Mbps.

2) HARDWARE EXPERIMENTATION

Like network emulation setup, the PM is compared with C-RSSI in hardware experimentation setup as well. PM is implemented in hardware experimentation setup by running a custom Android application on each end-device, maximizing the rate as per our proposed Algorithm 1 in Section III. The

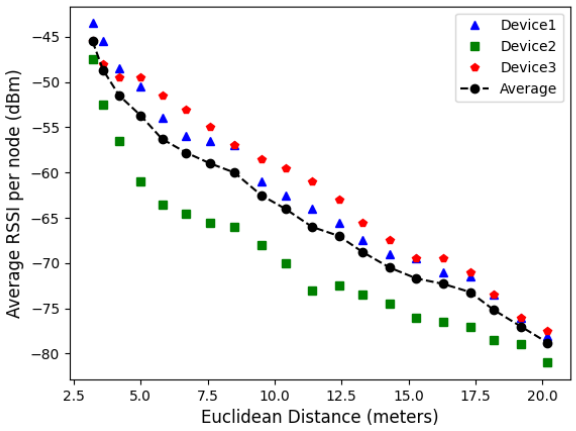


FIGURE 7. Observed average RSSI per node with increasing Euclidean distance.

Android application runs a script that scans for available RSSI from both WiFi APs every ten seconds and switches to the AP providing higher RSSI. In C-RSSI, the same end-devices are used, but without our custom Android application.

Before comparing PM and C-RSSI, the RSSI received from both APs at different positions is evaluated. The devices are placed in the node placement area shown in Figure 3, with an x-axis distance of 3 meters from both APs, whereas the y-axis distance from each AP is varied from 1 meter to 20 meters, in a step-size of 1 meter. As shown in Figure 7, the Euclidean distance from AP is compared with the received RSSI values at Galaxy Samsung S10 (Device1), Google Pixel C Tablet (Device2), and Motorola Nexus 6 (Device3). Each data point shown is the average RSSI received from the first AP and the second AP at a specific device. Whereas the black-line shows the averaged values of all the three devices. As the Euclidean distance from the AP increases, the average RSSI per node decreases.

The next analysis in Figure 8 shows the C-RSSI technique's expected behavior, where a node keeps connected to the same AP, even with increasing Euclidean distance. The blue curve shows that the average RSSI per node decreases with increasing Euclidean distances when the nodes are connected with AP1. The black curve shows a similar behavior when the nodes are connected with AP2. Each data point is the average of three devices placed at the same Euclidean distance from each AP.

Next, the PM is compared with C-RSSI. We considered a 20 nodes setup for evaluation, similar to the number of nodes in the network emulation setup. A node is placed at each of the measured Euclidean distance points shown in Figure 8. A certain percentage of nodes are randomly selected and are initially assigned to the incorrect AP. This percentage is varied from 10% to 70%. For comparison between PM and C-RSSI, the increasing percentage of incorrectly assigned nodes is compared with the average RSSI per node. Each experiment is performed 20 times, with the mean representing the data point, and the standard deviation is shown as the error bar.

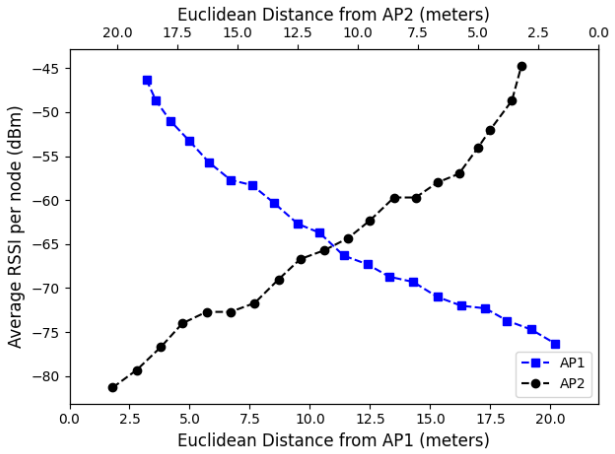


FIGURE 8. Observed average RSSI per node with increasing Euclidean distance from the connected AP.

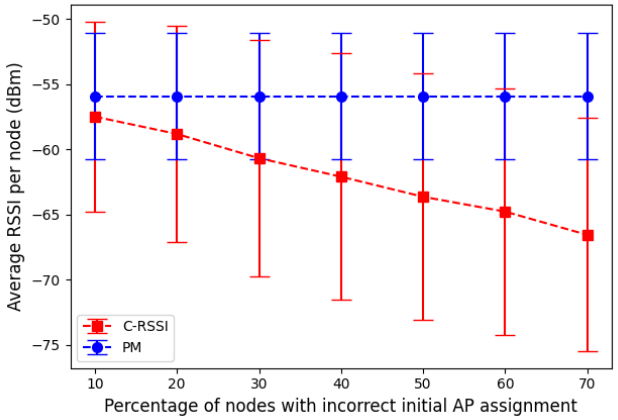


FIGURE 9. Observed average RSSI per node with increasing percentage of nodes assigned initially to incorrect AP.

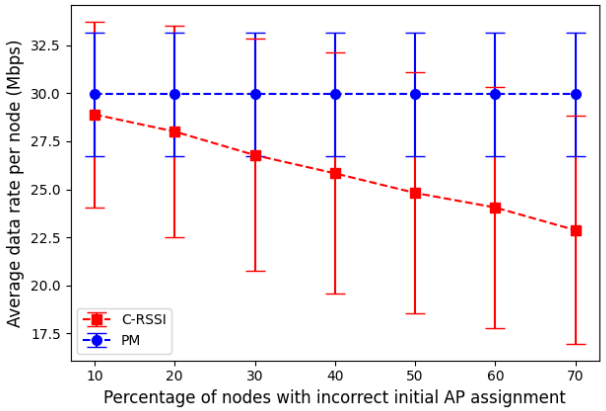


FIGURE 10. Achievable average data rate per node with increasing percentage of nodes assigned initially to incorrect AP.

As shown in Figure 9, as the percentage of incorrectly assigned nodes increase, more nodes incorrectly connect to an AP with a lower RSSI signal in C-RSSI, resulting in a lower average RSSI per node. Whereas in PM, even if the nodes are initially connected to an AP offering a lower RSSI signal, the node is automatically switched to the AP offering a better RSSI signal in the next scan of APs running inside the Android application. This results in the same average RSSI per node for PM, even with an increasing percentage of incorrectly assigned nodes. The behavior observed in the hardware experimentation setup is similar to the network emulation setup. In hardware experimentation setup, the average RSSI per node decreased from -55.9 dBm to -66.6 dBm (16.1% decrease), when the percentage of incorrectly assigned nodes increased from 0% to 70%. In comparison, the average RSSI per node decreased in network emulation setup from -64.1 dBm to -76.5 dBm (16.2% decrease). The almost similar percentage of decrease in average RSSI per node in network emulation and hardware experimentation results validates our network emulation setup.

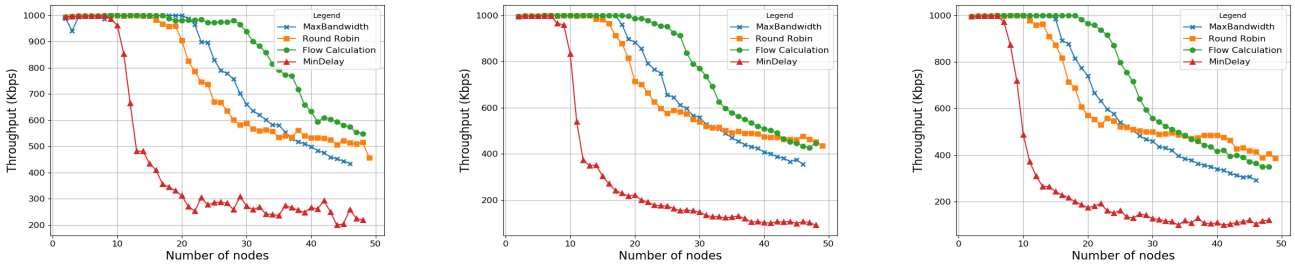
The same method explained in network emulation results to map RSSI to data rate is used for hardware experimentation results. The estimated average data rate per node against an increasing percentage of nodes with an incorrect initial AP assignment is shown in Figure 10. As expected, with the increase in the percentage of incorrectly assigned nodes, the average data rate per node decreases. With the increase of the percentage of incorrectly assigned nodes from 0% to 70%, the average data rate per node decreases from 29.95 Mbps to 22.87 Mbps (23.64% decrease).

B. CORE-QoS

Before explaining the results, it is essential to describe the process of experiment and results collection methodology for core-QoS provisioning. We have passed flows of fixed packet length across the emulated network to evaluate QoS provisioning of the throughput network parameter. Each traffic flow demands a fixed amount of bandwidth (1 Mbps), whereas, each sender node installs only one flow. If we formulate this setup according to the proposed system model in Section III, assuming the whole network requests for only one QoS parameter i.e. throughput in units Mbps, the QoS matrix $\mathbf{Q}$ will contain only one value i.e. $\mathbf{Q} = [1]$. The path cost matrix for the throughput parameter could have any value ≥ 1 Mbps. The higher the value is, the lesser would be its impact on cost minimization. Here, we assume this value to be also 1 Mbps, resulting in the matrix $\mathbf{D} = [1]$. As throughput is a parameter with higher desired values, the selection variables would be $x_1 = 1$ and $y_1 = 1$. From the topology shown in Figure 4, the actual values matrix $\mathbf{A}$ would be:

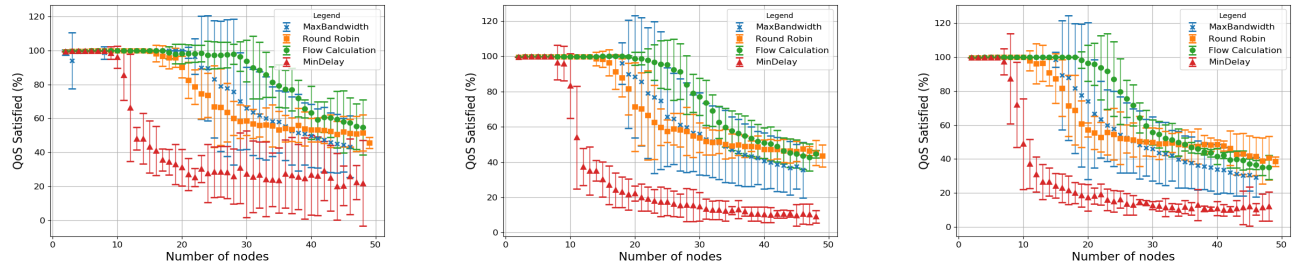
$$A = \begin{bmatrix} 16 \\ 4 \\ 2 \end{bmatrix} \tag{20}$$

We collect transmitted packets at the receiver end and compare the achieved throughput with the demand. For all flows where the constraint C6 in equation (15) is satisfied as per QoS matrix $\mathbf{Q}$, we mark the case as QoS satisfied. As we increase the number of nodes, the total bandwidth demand



(a) Without any background traffic (b) With background traffic 10% of the total path capacity (c) With background traffic 25% of the total path capacity

FIGURE 11. Achieved average throughput per node against number of nodes with various path selection strategies considering only throughput path cost factor.



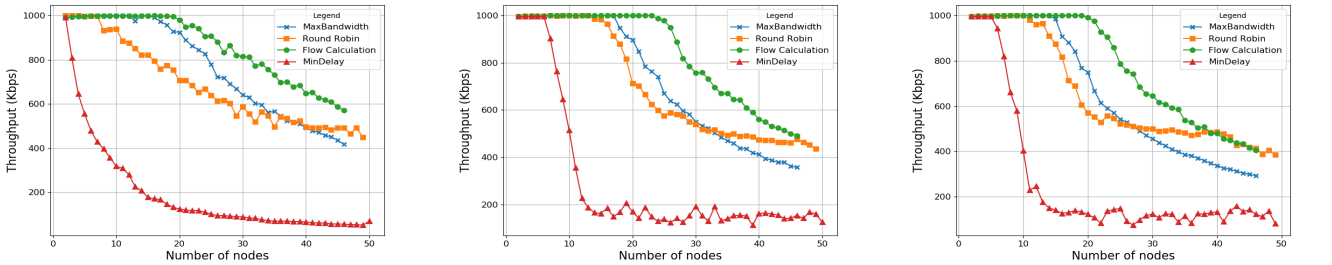
(a) Without any background traffic (b) With background traffic 10% of the total path capacity (c) With background traffic 25% of the total path capacity

FIGURE 12. Percent of QoS satisfied flows against number of nodes with various path selection strategies considering only throughput path cost factor.

of all flows increases proportionally. In case the achieved throughput for a flow is less than the requirement in QoS matrix **Q**, the constraint C6 in equation (15) is not satisfied, and we mark the case as unsatisfied QoS. In situations where QoS is unsatisfied, the traffic will still be forwarded but over the path with the least cost.

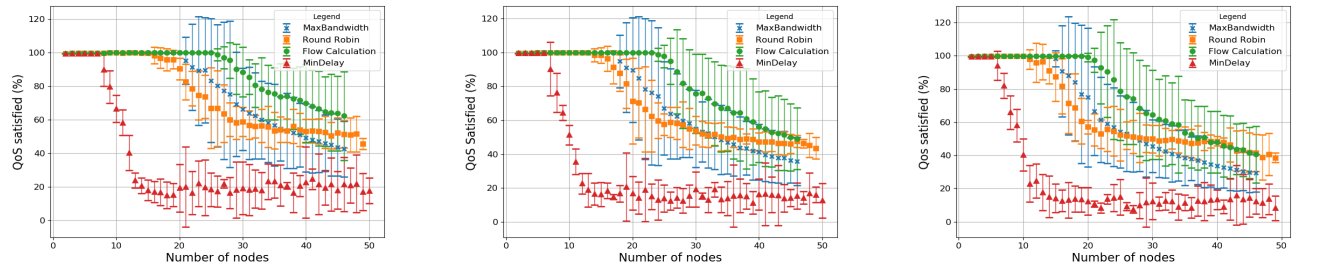
We compared our proposed strategy, named here as Flow Calculation, with three other routing strategies. The Flow Calculation strategy works according to Algorithm 2 and maintains a history of the previously admitted flows. For example, if we have already admitted a flow of 1 Mbps on a path with throughput 4 Mbps, the corresponding value in matrix **A** would be updated to 3 Mbps for admitting future flows. We have compared our flow calculation strategy with three other conventionally used simple routing strategies in communication networks; Maximum bandwidth (MaxBandwidth), Round Robin, and Minimum Delay (MinDelay). MaxBandwidth routing strategy at the SDN controller selects the path with maximum bandwidth. Round Robin shifts the incoming traffic to available paths in a circular fashion. On arrival of first flow, it chooses the path P_1, whereas, for second flow, it selects the next path P_2 without keeping in view the capacities of paths. In the MinDelay routing strategy, the SDN controller always selects the path with the minimum delay.

Results in Figure 11 show the average throughput achieved per node as the number of communicating node pairs increase, where each sender node transmits a single flow of exactly 1 Mbps. The results are shown with no background traffic in Figure 11a. The MinDelay has achieved the least throughput because all nodes select the same path P_3, which has a minimum delay of 30 msec and has a minimum bandwidth of 2 Mbps. As soon as the number of concurrently transmitting nodes increases beyond two, the network would become congested, and the throughput at each node would decrease. The MaxBandwith could accommodate up to 16 nodes in parallel without any congestion, as it always selects the path with the maximum bandwidth, which is P_1 in this case with a bandwidth of 16 Mbps. The Round Robin technique would circularly select all three paths one-by-one. The paths with lower bandwidths would become congested sooner, but the Round Robin technique will still keep circularly installing flows, even if the path is already congested. This technique might install one flow on a non-congested path and the next on a congested path. Our Flow Calculation technique outperforms all other techniques because it always tries to find the network path to satisfy our constraint of throughput $\geq$ 1 Mbps. It will install the first 12 flows on path P_1, next 4 flows in round robin fashion between paths P_1 and P_2, and then the next 6 flows again in round robin fashion between paths P_1, P_2, and P_3. From here onwards, none of the paths satisfy the throughput QoS constraint, and the path costs of all the paths are also the same. Flow Calculation uses the same path P_1 for transmitting any further flows.



(a) Without any background traffic (b) With background traffic 10% of the total path capacity (c) With background traffic 25% of the total path capacity

FIGURE 13. Achieved average throughput per node against number of nodes with various path selection strategies considering throughput, delay and jitter path cost factors.



(a) Without any background traffic (b) With background traffic 10% of the total path capacity (c) With background traffic 25% of the total path capacity

FIGURE 14. Percent of QoS satisfied flows against number of nodes with various path selection strategies considering throughput, delay and jitter path cost factors.

Congestion can start to occur when 23 nodes start to transmit concurrently. For all four techniques, the congestion occurs almost at the points discussed above and can be seen in Figure 11a. This can be verified by looking at the average per-client throughput falling below the required throughput of 1 Mbps per client. In Figure 11b, and Figure 11c, the same results are obtained but with background traffic, which is 10% and 25% of path capacity respectively. We can observe similar trends except for the shift of congestion points to the left for all four techniques in both these results.

Results for the percentage of QoS satisfied with all techniques against the increasing number of communicating nodes are shown in Figure 12. As shown in Figure 12a, the results show a pattern similar to that of throughput without any background traffic. As the congestion occurs most in MinDelay and MaxBandwidth, they also have less percentage of flows with QoS satisfied. Due to alternating path selection, Round Robin performs better than these two techniques. Whereas, our proposed Flow Calculation techniques outperforms all others, with the highest percentage of QoS satisfied flows. With background traffic, similar trends can be seen in Figure 12b, and Figure 12c. However, as the background traffic also consumes the path capacity, the percentage of flows with QoS satisfied decreases for all four techniques. It can be seen that with 10% background traffic and more than 40 nodes, and with 25% background traffic and more than 35 nodes, the percentage of QoS satisfied flows is more for Round Robin than the Flow Calculation technique. With the increase in background traffic and the number of nodes beyond a certain point, the network congestion leaves no room for further QoS provisioning, and the percentage of QoS satisfied curves start to converge for all the techniques. Therefore, this trend should not be seen as if Round Robin starts to outperform Flow Calculation at a certain point.

Another set of results is obtained where we consider network parameters of throughput (in Mbps), delay (in msec), and jitter (in msec). The required throughput by each flow is 1 Mbps. Whereas, there are no constraints on delay and jitter, resulting in a QoS matrix $\mathbf{Q} = [1\ \infty\ \infty]$. For simplicity, we have used the same value of throughput for path cost matrix $\mathbf{D}$. However, for path cost calculation, we considered additional parameters of delay and jitter with values of 40 msec and 6 msec respectively. The resultant path cost calculation matrix would be $\mathbf{D} = [1\ 40\ 6]$. The values of selection variables would be $x_1 = 1$, $x_2 = -1$, $x_3 = -1$, $y_1 = 1$, $y_2 = 0$ and $y_3 = 0$. The actual values matrix $\mathbf{A}$ from Figure 4 for this multiple path cost parameters scenario would then be:

$$A = \begin{bmatrix} 16 & 80 & 12 \\ 4 & 40 & 6 \\ 2 & 30 & 4 \end{bmatrix} \tag{21}$$

The other techniques will still perform in a similar manner. MinDelay will always choose path P_3, MaxBandwidth

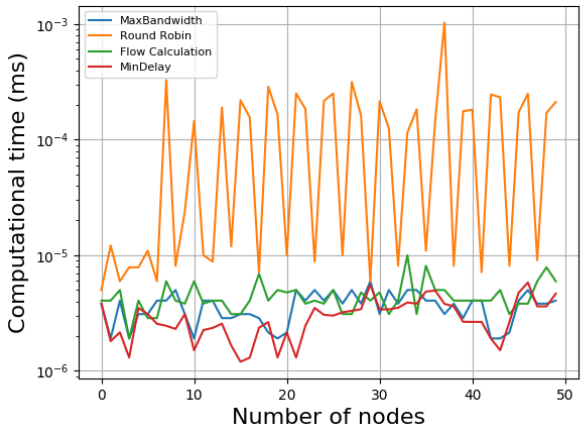


FIGURE 15. Computational time for various strategies.

will always choose path P_1, and Round Robin will alternate between all the three paths. However, the proposed Flow Calculation technique's path cost selection would be a little complex than the previous experiment. The QoS constraint is the same throughput requirement of 1 Mbps. However, path cost calculation is now based on three different parameters i.e. throughput, delay, and jitter. In this case, our optimization problem will find the shortest cost path, which satisfies our throughput constraint as per equations (13), (14), and (15). During experimentation, unlike the first experiment, it has been found that at times paths that have lower path cost but not the maximum available bandwidth have been chosen. However, this does not significantly impact the throughput achieved, as shown in Figure 13. This is because of the fact that although other paths with lower path costs are chosen, they still fulfill the throughput constraint. Like the first experiment, when none of the three paths could satisfy the QoS constraint, the lowest-cost path is chosen. When the path cost also becomes the same for all the three paths, path P_1 is used. We achieved almost similar bandwidth for all four techniques as in the first experiment, with no background traffic (Figure 13a), with 10% background traffic (Figure 13b) and with 25% background traffic (Figure 13c).

As almost similar throughput is achieved for the second experiment compared with the first experiment, the percentage of flows for which QoS satisfied is also similar, as shown in Figure 14. We can conclude that our proposed approach, even though considering multiple factors in path cost calculation, provided similar QoS guarantees without background traffic (Figure 14a), with 10% background traffic (Figure 14b) and with 25% background traffic (Figure 14c).

These experiments help realize that the proposed Flow Calculation technique predicts a more appropriate path selection strategy over the futuristic network core while providing better QoS provisioning. We have also calculated the computational time for all four techniques, as shown in Figure 15. The results are obtained using the tic-toc method on a machine with specifications: Intel(R) Core(TM) i5-2430M processor, dual-core CPU 2.4 GHz, and RAM 6 GB. Load balancing over available paths using a Round Robin technique takes more computational time than the rest of the methods, including the proposed Flow Calculation strategy. From these results, we can deduce that although the proposed Flow Calculation technique does not have the lowest computational time, it outperforms in path selection while considering QoS parameters of throughput, delay, and jitter.

VI. CONCLUSION

Beyond 5G networks face challenges of increased traffic and number of IoT nodes, with delays expected not to increase a certain threshold. To address these challenges, a hybrid QoS provisioning approach is proposed to ensure a certain service level. We have discussed the effectiveness of our proposed completely hybrid end-to-end QoS provisioning technique, involving both clients and SDN controllers. Due to the hybrid nature of our technique, clients can specify access specific QoS parameters. They can also make better decisions to select between access devices due to the local view available to them. This reduces the client's dependency on the controller for access device selection, thus reducing the signaling between both. The reduced signaling also improves network scalability. In the case of access-network, we have shown with the help of Mininet-WiFi emulator as well as with hardware-based experimentation, that our proposed methodology outperforms the conventionally used AP selection approach. For the core-network, Mininet based experimentation results have shown that our proposed approach outperforms several conventionally used routing techniques while considering complex scenarios with multiple network parameters for path cost calculation and constraints.

REFERENCES

[1] Y. Yuan, Y. Zhao, B. Zong, and S. Parolari, "Potential key technologies for 6G mobile communications," *Sci. China Inf. Sci.*, vol. 63, no. 8, pp. 1–19, Aug. 2020.
[2] A. Mourad, R. Yang, P. H. Lehne, and A. De La Oliva, "A baseline roadmap for advanced wireless research beyond 5G," *Electronics*, vol. 9, no. 2, p. 351, Feb. 2020.
[3] V. W. Wong, *Key Technologies for 5G Wireless Systems*. Cambridge, U.K.: Cambridge Univ. Press, 2017.
[4] G. Forecast, "Cisco visual networking index: Global mobile data traffic forecast update, 2017–2022," *Update*, vol. 2017, p. 2022, Feb. 2019.
[5] Q. Zhang, W. Zhu, and Y.-Q. Zhang, "End-to-end QoS for video delivery over wireless Internet," *Proc. IEEE*, vol. 93, no. 1, pp. 123–134, Jan. 2005.
[6] M. Wang, J. Chen, E. Aryafar, and M. Chiang, "A survey of client-controlled HetNets for 5G," *IEEE Access*, vol. 5, pp. 2842–2854, 2017.
[7] S. H. Yeganeh, A. Tootoonchian, and Y. Ganjali, "On scalability of software-defined networking," *IEEE Commun. Mag.*, vol. 51, no. 2, pp. 136–141, Feb. 2013.
[8] S. Sharma, D. Staessens, D. Colle, D. Palma, J. Goncalves, R. Figueiredo, D. Morris, M. Pickavet, and P. Demeester, "Implementing quality of service for the software defined networking enabled future Internet," in *Proc. 3rd Eur. Workshop Softw. Defined Netw.*, Sep. 2014, pp. 49–54.
[9] J. Yan, H. Zhang, Q. Shuai, B. Liu, and X. Guo, "HiQoS: An SDN-based multipath QoS solution," *China Commun.*, vol. 12, no. 5, pp. 123–133, May 2015.
[10] D. L. C. Dutra, M. Bagaa, T. Taleb, and K. Samdanis, "Ensuring end-to-end QoS based on multi-paths routing using SDN technology," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2017, pp. 1–6.

[11] H. E. Egilmez, S. T. Dane, K. T. Bagci, and A. M. Tekalp, "OpenQoS: An OpenFlow controller design for multimedia delivery with end-to-end quality of service over software-defined networks," in *Proc. Signal Inf. Process. Assoc. Summit Conf.*, Dec. 2012, pp. 1–8.

[12] Z. Qin, G. Denker, C. Giannelli, P. Bellavista, and N. Venkatasubramanian, "A software defined networking architecture for the Internet-of-Things," in *Proc. IEEE Netw. Oper. Manage. Symp. (NOMS)*, May 2014, pp. 1–9.

[13] A. Al-Jawad, P. Shah, O. Gemikonakli, and R. Trestian, "LearnQoS: A learning approach for optimizing QoS over multimedia-based SDNs," in *Proc. IEEE Int. Symp. Broadband Multimedia Syst. Broadcast. (BMSB)*, Jun. 2018, pp. 1–6.

[14] G. Li, J. Wu, J. Li, Z. Zhou, and L. Guo, "SLA-aware fine-grained QoS provisioning for multi-tenant software-defined networks," *IEEE Access*, vol. 6, pp. 159–170, 2018.

[15] X. Zhang and Q. Zhu, "Information-centric virtualization for software-defined statistical QoS provisioning over 5G multimedia big data wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 8, pp. 1721–1738, Aug. 2019.

[16] J.-P. Huang, C.-C. Huang-Fu, and P.-Y. Cheng, "Dual SIM dual standby user equipment rat selection," U.S. Patent 10 681 604, Jun. 9, 2020.

[17] B. Sliwa, R. Falkenberg, and C. Wietfeld, "Towards cooperative data rate prediction for future mobile and vehicular 6G networks," in *Proc. 2nd 6G Wireless Summit (6G SUMMIT)*, Mar. 2020, pp. 1–5.

[18] A. Habbal, S. I. Goudar, and S. Hassan, "A context-aware radio access technology selection mechanism in 5G mobile network for smart city applications," *J. Netw. Comput. Appl.*, vol. 135, pp. 97–107, Jun. 2019.

[19] V. Passas, V. Miliotis, N. Makris, and T. Korakis, "Dynamic RAT selection and pricing for efficient traffic allocation in 5G HetNets," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2019, pp. 1–6.

[20] X. Chen, D. Wing Kwan Ng, W. Yu, E. G. Larsson, N. Al-Dhahir, and R. Schober, "Massive access for 5G and beyond," 2020, *arXiv:2002.03491*. [Online]. Available: http://arxiv.org/abs/2002.03491

[21] M. Aldababsa, M. Toka, S. Gökçeli, G. K. Kurt, and O. Kucur, "A tutorial on nonorthogonal multiple access for 5G and beyond," *Wireless Commun. Mobile Comput.*, vol. 2018, pp. 1–24, Jun. 2018.

[22] Y. Liu, Z. Qin, M. Elkashlan, Z. Ding, A. Nallanathan, and L. Hanzo, "Non-orthogonal multiple access for 5G and beyond," *Proc. IEEE*, vol. 105, no. 12, pp. 2347–2381, Dec. 2017.

[23] Q.-V. Pham, F. Fang, V. N. Ha, M. J. Piran, M. Le, L. B. Le, W.-J. Hwang, and Z. Ding, "A survey of multi-access edge computing in 5G and beyond: Fundamentals, technology integration, and state-of-the-art," *IEEE Access*, vol. 8, pp. 116974–117017, 2020.

[24] M. Mehrabi, D. You, V. Latzko, H. Salah, M. Reisslein, and F. H. P. Fitzek, "Device-enhanced MEC: Multi-access edge computing (MEC) aided by end device computation and caching: A survey," *IEEE Access*, vol. 7, pp. 166079–166108, 2019.

[25] P. Shantharama, A. S. Thyagaturu, N. Karakoc, L. Ferrari, M. Reisslein, and A. Scaglione, "LayBack: SDN management of multi-access edge computing (MEC) for network access services and radio resource sharing," *IEEE Access*, vol. 6, pp. 57545–57561, 2018.

[26] A. Garcia-Saavedra, X. Costa-Perez, D. J. Leith, and G. Iosifidis, "FluidRAN: Optimized vRAN/MEC orchestration," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2018, pp. 2366–2374.

[27] G. White, V. Nallur, and S. Clarke, "Quality of service approaches in IoT: A systematic mapping," *J. Syst. Softw.*, vol. 132, pp. 186–203, Oct. 2017.

[28] S. Tomovic, W. Cerroni, F. Callegati, R. Verdone, I. Radusinovic, M. Pejanovic, and C. Buratti, "An architecture for QoS-aware service deployment in software-defined IoT networks," in *Proc. 20th Int. Symp. Wireless Pers. Multimedia Commun. (WPMC)*, Dec. 2017, pp. 561–567.

[29] A. Brogi and S. Forti, "QoS-aware deployment of IoT applications through the fog," *IEEE Internet Things J.*, vol. 4, no. 5, pp. 1185–1192, Oct. 2017.

[30] O. Skarlat, M. Nardelli, S. Schulte, and S. Dustdar, "Towards QoS-aware fog service placement," in *Proc. IEEE 1st Int. Conf. Fog Edge Comput. (ICFEC)*, May 2017, pp. 89–96.

[31] R. Zhang, M. Wang, X. Shen, and L.-L. Xie, "Probabilistic analysis on QoS provisioning for Internet of Things in LTE—A heterogeneous networks with partial spectrum usage," *IEEE Internet Things J.*, vol. 3, no. 3, pp. 354–365, Jun. 2016.

[32] I. Gravalos, P. Makris, K. Christodoulopoulos, and E. A. Varvarigos, "Efficient network planning for Internet of Things with QoS constraints," *IEEE Internet Things J.*, vol. 5, no. 5, pp. 3823–3836, Oct. 2018.

[33] R. Duan, X. Chen, and T. Xing, "A QoS architecture for IOT," in *Proc. Int. Conf. Internet Things 4th Int. Conf. Cyber, Phys. Social Comput.*, Oct. 2011, pp. 717–720.

[34] T. Luo, H.-P. Tan, and T. Q. S. Quek, "Sensor OpenFlow: Enabling software-defined wireless sensor networks," *IEEE Commun. Lett.*, vol. 16, no. 11, pp. 1896–1899, Nov. 2012.

[35] L. Galluccio, S. Milardo, G. Morabito, and S. Palazzo, "SDN-WISE: Design, prototyping and experimentation of a stateful SDN solution for WIreless SEnsor networks," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2015, pp. 513–521.

[36] D. Zeng, T. Miyazaki, S. Guo, T. Tsukahara, J. Kitamichi, and T. Hayashi, "Evolution of software-defined sensor networks," in *Proc. IEEE 9th Int. Conf. Mobile Ad-hoc Sensor Netw.*, Dec. 2013, pp. 410–413.

[37] T. Miyazaki, S. Yamaguchi, K. Kobayashi, J. Kitamichi, S. Guo, T. Tsukahara, and T. Hayashi, "A software defined wireless sensor network," in *Proc. IEEE Int. Conf. Comput., Netw. Commun. (ICNC)*, Feb. 2014, pp. 847–852.

[38] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd ed. Baltimore, MD, USA: The Johns Hopkins Univ. Press, 1996.

[39] A. Bose and C. Heng Foh, "A practical path loss model for indoor WiFi positioning enhancement," in *Proc. 6th Int. Conf. Inf., Commun. Signal Process.*, 2007, pp. 1–5.

[40] R. R. Fontes, S. Afzal, S. H. B. Brito, M. A. S. Santos, and C. E. Rothenberg, "Mininet-WiFi: Emulating software-defined wireless networks," in *Proc. 11th Int. Conf. Netw. Service Manage. (CNSM)*, Nov. 2015, pp. 384–389.

[41] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: Rapid prototyping for software-defined networks," in *Proc. 9th ACM SIGCOMM Workshop Hot Topics Netw. (Hotnets)*, 2010, p. 19.

[42] A. Basit, S. Qaisar, S. H. Rasool, and M. Ali, "SDN orchestration for next generation inter-networking: A multipath forwarding approach," *IEEE Access*, vol. 5, pp. 13077–13089, 2017.

[43] M. Monaco, O. Michel, and E. Keller, "Applying operating system principles to SDN controller design," in *Proc. 12th ACM Workshop Hot Topics Netw. (HotNets)*, 2013, p. 2.

[44] Q. Shafi, A. Basit, S. Qaisar, A. Koay, and I. Welch, "Fog-assisted SDN controlled framework for enduring anomaly detection in an IoT network," *IEEE Access*, vol. 6, pp. 73713–73723, 2018.

[45] X. Wu, M. Safari, and H. Haas, "Access point selection for hybrid li-fi and Wi-Fi networks," *IEEE Trans. Commun.*, vol. 65, no. 12, pp. 5375–5385, Dec. 2017.

MUHAMMAD ASAD (Graduate Student Member, IEEE) received the bachelor's degree in telecommunication engineering from the University of Engineering and Technology Taxila, Pakistan, in 2011, and the master's degree in communications engineering from the Technical University of Munich, Germany, in 2013. He is currently pursuing the Ph.D. degree with the School of Electrical Engineering and Computer Science, National University of Sciences and Technology (NUST). His research interests include software-defined networking, quality of service provisioning, and the IoT networks.

ABDUL BASIT received the bachelor's degree in computer science from the University of Management and Technology, Lahore, in 2003, the M.S. degree in software engineering from the National University of Sciences and Technology, in 2005, and the Ph.D. degree with the Department of Computing, School of Electrical Engineering and Computer Sciences. He has served in various positions and organizations in academia and industry for more than seven years. His research interests include programmable networking, network modeling and optimization, and industrial applications of geographical information systems.

SAAD QAISAR (Senior Member, IEEE) received the master's and Ph.D. degrees in electrical engineering from Michigan State University, East Lansing, MI, USA, in 2005 and 2009, respectively. He is currently serving as an Assistant Professor with the School of Electrical Engineering Computer Science (SEECS), National University of Sciences and Technology (NUST), Pakistan. He is also the Lead Researcher and the Founding Director of the CoNNekT Lab: Research Laboratory of Communications, Networks and Multimedia, National University of Sciences Technology (NUST), Pakistan. Since September 2011, he has been the Principal Investigator or a Joint Principal Investigator of seven research projects spanning cyber-physical systems, applications of wireless sensor networks, network virtualization, communication and network protocol design, wireless and video communication, Internet measurements analysis, multimedia coding, and communication. He has published more than 80 articles at reputed international venues with a vast amount of work in the pipeline.

MUDASSAR ALI (Member, IEEE) received the B.S. degree in computer engineering and the M.S. degree in telecom engineering from the University of Engineering and Technology, Taxila, Pakistan, in 2006 and 2010, respectively, with a major in wireless communication, and the Ph.D. degree from the School of Electrical Engineering and Computer Science (SEECS), National University of Sciences and Technology (NUST), Pakistan, in 2017. From 2006 to 2007, he worked as a Network Performance Engineer with Mobilink (An Orascom Telecom Company). From 2008 to 2012, he worked as a Senior Engineer Radio Access Network Optimization with Zong (A China Mobile Company). Since 2012, he has been an Assistant Professor with the Telecom Engineering Department, University of Engineering and Technology. His research interests include 5G wireless systems, heterogeneous networks, interference coordination, and energy efficiency in 5G green heterogeneous networks.