Bitlis Eren Üniversitesi Fen Bilimleri Dergisi
BİTLİS EREN UNIVERSITY JOURNAL OF SCIENCE
ISSN: 2147-3129/e-ISSN: 2147-3188
VOLUME: 11 NO: 2 PAGE: 553-563 YEAR: 2022
DOI: 10.17798/bitlisfen.1056963

# Mobile user type aware load balancing algorithm in SD-RAN

Müge EREL-ÖZÇEVİK[1*]

[1]*Manisa Celal Bayar University, Department of Software Engineering, TR-45400, Manisa, Turkey (ORCID: 0000-0003-3077-160X)*

**Keywords:** Software defined network, Radio access network, Queuing theory, iOS, Android.

**Abstract**

Under extreme increase on video contents in eMBB applications; the 5G requirements cannot be handled by the conventional self-organizing in 4G infrastructure. While executing load balancing in 5G RAN, mobile user type for eMBB applications should be considered. Nowadays, eMBB has been carried by QUIC and HTTP2.0 protocol for Android and iOS users, respectively. In mobile user aware load balancing, Deep Packet Inspection (DPI) up to application layer for packet routing is required. This can be only handled by Software-Defined Network (SDN) without any hardware expenditure in physical infrastructure. Therefore, this paper proposed Software-Defined Radio Access Network (SD-RAN) with two novel functions: Waiting Time Function (WTF) and Load Balancing Function (LBF). In WTF; the queuing inspired approach is proposed for the low complex implementation of the mobile user aware load balancing in 5G-RAN. Waiting Time parameters for iOS and Android users are analytically defined by M/G/1 and G/G/1 Markov queues. It is also executed by M/M/c/K Markov model SD-RAN topology. In LBF; a novel Mixed Integer Linear Problem is defined for waiting time optimization. To overcome NP hardness, a local search for the eMBB load threshold analysis is performed and determined as 0.79 and 0.94 for UMas and UMis. A low complex load balancing algorithm is proposed in the light of these thresholds. According to performance results; SD-RAN outperforms nearly 40% QoS than the conventional SON according to received packet count. It can serve 40% more users than the conventional one without any extra expenditure on physical infrastructure. As a result, it can handle eMBB flows with an acceptable waiting time under 2 milliseconds level.

## 1. Introduction

In today's Covid process; the main reason for the increase in data traffic is online-video contents in mobile applications, which is newly called as enhanced Mobile BroadBand (eMBB) service of 5G new radio [1,2]. It is estimated that this content in mobile applications is accounted for 74 percent of all data traffic in 2024 [3]. According to International Mobile Communications (IMT-2020), eMBB service should be served in under a few milliseconds. To handle this requirement, it is inevitable that Radio Access Network (RAN) should meet a 100-fold traffic load as compared to the 4G infrastructure [4]. It is believed that the 5G requirements can be only handled by a centralized load balancing instead of the self-organizing in 4G.

EMBB has been carrying by UDP-based QUIC (HTTP3) protocol in the application layer of 5G network stack. It outputs zero latency for connection setup and it overcomes possible packet losses caused by UDP-based connections by multiplying such connection packets. This provides us to serve mobile users with lower latency than conventional TCP-based HTTP2.0; however, it increases traffic load much more than conventional ones. QUIC protocol has been implemented in Android applications; whereas, there has been no library for iOS ones, yet. Therefore; iOS users are still using HTTP2.0 for eMBB applications. However, the legacy network cannot differentiate users as iOS or

*Corresponding author: muge.ozcevik@cbu.edu.tr
*Received: 12.01.2022, Accepted: 01.04.2022*

Android without the aforementioned application layer details. This motivates to consider mobile user type as iOS or Android while executing load balancing in 5G RAN [5,6].

In mobile user aware load balancing, the deep packet inspection (DPI) up to the application layer for packet routing is required [7]. However, it needs extra computing resource in the control plane due to the dynamic changes in the signature of the application. Moreover, it is hard to manage because of its complex implementation in both the control and data planes [8,9]. To handle low complex execution of DPI, Software Defined Network (SDN) has been used in the literature [10,11]. However, DPI is not supported in even the standard OpenFlow library [12,13]. Therefore, the queuing inspired approach is proposed for the low complex implementation of the mobile user aware load balancing in this paper. Thanks to the analytical model of iOS and Android users, the mobile user awareness would extremely decreases the complexity of proposed load balancing algorithm. Moreover, the Waiting Time parameter of eMBB contents varying as iOS and Android would be easily calculated in the control plane without any hardware implementation in the data plane. As a result; the Software-Defined Radio Access Network (SD-RAN) is proposed, which has an easy implementation on 5G protocol stack. It has two new functions in the control plane: Waiting Time Function (WTF) and Load Balancing Function (LBF) where the whole contributions are given below:

- WTF:
  - iOS and Android mobile users are modeled by M/G/1 and G/G/1 Markov models by considering application layer protocols such as HTTP2.0 and QUIC, respectively.
  - A Waiting Time parameter per SD-RAN is modeled by M/M/c/K Markov model and it is extracted by Jackson theorem.
- LBF:
  - A novel Mixed Integer Linear Problem is defined for the Waiting Time optimization.
  - A local search for load threshold analysis is performed to overcome NP hardness of the optimization problem.
  - A novel Load Balancing Algorithm is proposed in the light of load thresholds and optimization constraints.

The rest of paper is organized as follows. In section 2, the proposed network architecture and protocol stack of SD-RAN are given. In section 3, the proposed system architecture models Waiting Time by using Queuing Theory. Here, the implementation details of the Waiting Time optimization problem and the Load Balancing Algorithm which executes it are defined. In section 4, the performance of the proposed SD-RAN is evaluated and the paper is finalized in section 5.

## 2. Network Architecture

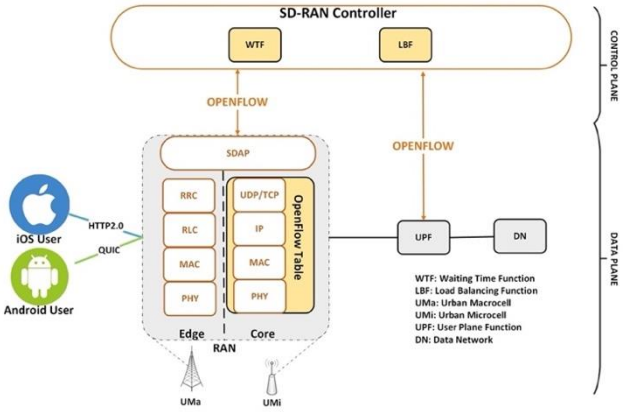



**Figure 1.** Network architecture of SD-RAN.

The proposed network architecture of SD-RAN is shown in Figure 1. It is decoupled as Data and Control planes. In the data plane, there are 5G components named as User Plane Function (UPF) and Data Network (DN) on the core side and there are Software Defined - Radio Access Networks (SD-RAN); i.e. base stations varying in terms of coverage area such as Urban Macrocell and Urban Microcell in the edge side. Here, a traffic flow is routed over SD-RAN, UPF and DN through to Internet Service Providers (ISP). To handle traffic flows in core, SD-RAN has an OpenFlow table where the IP-protocol stack can be managed up to the transport layer, i.e UDP/TCP. To handle traffic flow in edge, it has also a 5G protocol stack where the layers are physical (PHY), medium access control (MAC), radio link control (RLC), radio resource control (RRC). By using protocol stacks of edge side; SD-RAN manages wireless traffic in random access procedure, admission control, resource management etc. By using protocol stack of core side, it dynamically routes a traffic flow through ISP. In the core side, each device are dummy and controlled by SD-RAN controller in Control plane thanks to OpenFlow 1.5.1. protocol [13]. However, deep packet inspections are not supported in legacy and in OpenFlow switches [14]. To handle application awareness in this paper, deep packet inspection is performed by Service Data Adaptation Protocol (SDAP) layer in SD-RAN. Therefore, the proposed SD-RAN can differentiate the mobile user type such as iOS and Android. The mobile application sends this data to SD-RAN via SDAP service as exemplified in Table 1.

**Table 1.** Proposed OpenFlow stack and matching fields of SD-RAN.

| MATCH FIELDS | | | | ACTION | STATISTICS |
|---|---|---|---|---|---|
| IN_ PORT | ETH_ TYPE | IP_ PROTO | USER_ TYPE | Output | TX_ Packets |
| Port1 | 0x800 | 6 (HTTP2) | iOS | Port3 | 15535 |
| Port2 | 0x800 | 17 (QUIC) | Android | Port4 | 17760 |

The OpenFlow stack of the proposed SD-RAN by considering different user types and different protocols of eMBB applications, is given in Table 1. EMBB applications such as Youtube, Spotify etc. in Android device has been newly implemented over UDP based QUIC (HTTP3) protocol. While comparing it with HTTP2.0, it provides low latency in connection setup (0-RTT), secure transmission by QUIC handshake, and easy implementation due to handling in user space instead of core space in a mobile device. However, the iOS operating system has no support for QUIC protocol. Therefore, it still uses HTTP2.0 for eMBB applications in a mobile device. In the proposed architecture, the load caused by the iOS/Android mobile applications in edge network is dynamically balanced by the SD-RAN controller. To meet this contribution, there are two newly defined functions in the control plane. They are named as Waiting Time Function (WTF) and Load Balancing Function (LBF), and they are detailed in the following section.

## 3. The System Architecture of the Proposed SD-RAN

The SD-RAN controller periodically takes statistics from OpenFlow switches in the data plane and dynamically creates forwarding rule to optimize load in UMa and UMi. It calculates analytically defined Waiting Time (W) per mobile user and per SD-RAN by Waiting Time Function (WTF). Then, it runs the Load Balancing Algorithm which dynamically changes the route of iOS/Android traffic flows by Load Balancing Function (LBF). The Markov model of W per user and per SD-RAN are detailed in subsection 3.1., and then the optimization formula and implementation of its in an LBF are given in subsection 3.2.

### 3.1. Waiting Time Function (WTF)

This function in the control plane periodically calculates Waiting Time per mobile user and per UMa and UMi RANs. The analytical definition of the Markov models are given below:

#### 3.1.1. W model for mobile user

**Table 2.** Queuing models of eMBB applications and protocol details.

| User Type | Application Layer protocol | Transport Layer Protocol | Queuing Model |
|---|---|---|---|
| iOS | HTTP2.0 | TCP | M/G/1 |
| ANDROID | QUIC (HTTP3) | UDP | G/G/1 |

In this paper, we consider two mobile operating systems such as iOS and Android. They are also called as the mobile user types in SD-RAN. EMBB applications in iOS and Android systems use different protocols such as HTTP2.0 and QUIC. The proposed queuing models of them and the protocol details are defined as in Table 2. EMBB applications in iOS uses HTTP2.0 as an application layer protocol which is carried on TCP as a transport layer protocol; whereas, such applications in Android newly uses QUIC (HTTP3) over UDP which extremely reduces RTT to meet the eMBB requests in 5G. However, there has been no implementation library of QUIC protocol in iOS yet, then, EMBB applications in iOS are carried on TCP-based HTTP2.0.

According to these protocol details of the iOS and Android systems, the Waiting Time $W_{ij}$ of $user_i$ on SD-$RAN_j$ are modeled by M/G/1 and G/G/1, respectively. For G/G/1, the Waiting Time is defined as follows:

$$W_{\{ij\}}(t) = \frac{C_A^2 + C_B^2}{2} \cdot \frac{\rho_j(t)}{1-\rho_j(t)} \cdot \frac{1}{\mu_j} \quad (1)$$

where as $\rho_j$ is load of SD-$RAN_j$ and calculated as $\lambda_j/\mu_j$. Here, $\lambda_j$ is the average arrival rate and defined as $\sum \lambda_i / M_j$ and $\mu_j$ is serving rate of SD-$RAN_j$ which varies according to Urban Macrocell or Urban Microcell. They are modeled by General distribution. Therefore, $C_A^2$ and $C_B^2$ are coefficients of arrival rate as follows:

$$\boldsymbol{C_A^2 = \frac{Var\,[T]}{E^2[T]} = \lambda_i^2(t)\sigma_A^2} \quad (2)$$

$$\boldsymbol{C_B^2 = \frac{Var\,[S]}{E^2[S]} = \mu_i^2(t)\sigma_B^2} \quad (3)$$

By combining eqs.1,2, and 3, the Waiting Time of $user_i$ in SD-$RAN_j$ is calculated as follows:

$$\boldsymbol{W_{\{ij\}}(t) = \frac{\lambda_i^2(t)\sigma_A^2 + \mu_i^2(t)\sigma_B^2}{2} \cdot \frac{\rho_j(t)}{1-\rho_j(t)} \cdot \frac{1}{\mu_j}} \quad (4)$$

For iOS users, due to handing eMBB application over TCP, the arrival rate is modeled by Poisson Distribution. Therefore, the queuing model of

iOS users is M/G/1. Because of arrival rate characteristics of this protocol, $\lambda_i^2(t)\sigma_A^2$ equals to 1. The final version of waiting time can be found below:

$$W_{\{ij\}}(t) = \frac{\rho_j(t)}{1-\rho_j(t)}.\frac{1}{2\mu_j}.\begin{cases} \frac{1+\mu_j(t)\sigma_B^2}{2} & ,i \in iOS \\ \frac{\lambda_i^2(t)\sigma_A^2+\mu_i^2(t)\sigma_B^2}{2} & ,i \in Android \end{cases} \quad (5)$$

As shown in eq. 5, the load of SD-RANs directly affects waiting time, and therefore, to handle 5G requirements for eMBB applications the load of UMa and UMi should be dynamically balanced. Here, Waiting Time for SD-RAN should be also modeled which is detailed in following sub-section.

### 3.2.1. W model for SD-RANs

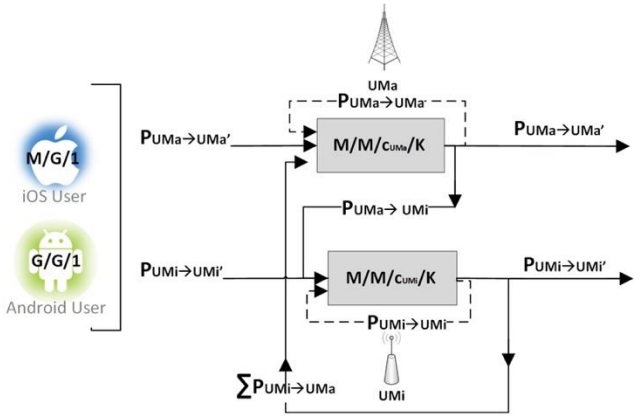



**Figure 2.** Proposed queuing model of SD-RAN data plane.

In SD-RAN, there are Urban Macrocells (UMa) and Urban Microcells (UMi) as the outdoor service types of SD-RANs. Therefore, the data plane of SD-RAN is analytically modelled by using Jackson's network as shown in Figure 2. Here, there are multiple definitions for the transition probabilities where the general version of the probability is defined as $P_{x\to y}$. For example; if a traffic flow of user$_i$ is transferred from UMa to UMi, it is shown by $P_{UMa\to UMi}$. For the case (x=y), it defines the remaining probability in the same SD-RAN. For the case $(x \to x')$, it shows the handover probability for UMa or UMi within itself. These probabilities dynamically changes the load per SD-RAN $(\rho_j)$.Therefore, loads per UMa and UMi are calculated by using Jackson's network as follows:

$$\rho_j = \frac{1}{c_j\mu_j}.[\lambda_{\{j'\}}P_{\{j\to j'\}} + \lambda_j P_{\{j\to j\}} + \begin{cases} \sum\lambda_{\{UMi\}}P_{\{UMi\to j\}}\,], j \in UMa \\ \lambda_{\{UMa\}}P_{\{UMa\to j\}}\,] \;, j \in UMi \end{cases} \quad (6)$$

Moreover, $W_j$ per SD-RAN is modelled by using M/M/c/K Markov queue. The reason to prefer M/M/c/K instead of the M/M/c queuing model is related to computational implementation. In M/M/c/K, the arrival rate is in both Poisson and Geometric series for different cases. These are finite series in M/M/c/K model; and therefore, there is no constraint such as $\rho < 1$ [15]. Then, the computation can be easily performed for overload cases $\rho \geq 1$. Thanks to that, the waiting time for also overload cases can be also analytically calculated. While working on Dense Urban topology of 5G, this case should be also considered. Therefore; each SD-RAN which serves as UMa or UMi is modeled by M/M/c/K queuing model. The details are given in Appendix A and the full formula is given below:

$$W_{\{ij\}}(t) = \frac{\rho_j c_j\left(1-\frac{(\rho_j c_j)^K}{c_j^{K-c_j}c_j!}P_0\right)+\frac{P_0(\rho_j c_j)^{c_j}\rho_j}{c_j!(1-\rho_j)^2}.\left[1-\rho_j^{K-c_j+1}-(1-\rho_j).(K-c_j+1)\left(\rho_j^{K-c_j}\right)\right]}{\lambda_j\left(1-\frac{(\rho_j c_j)^K}{c_j^{K-c_j}c_j!}P_0\right)} \quad (7)$$

### 3.2. Load Balancing Function (LBF)

This function in the control plane dynamically changes the route of iOS/Android traffic flow to balance the load between UMa and UMi. Therefore; the proposed optimization problem executed in LBF is defined as follows:

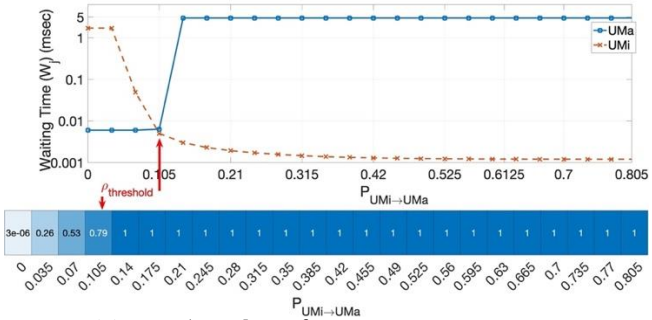
$$\min\left\{\sum_{i\in iOS} W_{ij}, \sum_{i\in Android} W_{ij}\right\}\ s.t.$$
$$\begin{cases} P_{UMa\to UMi} = P_{UMi} | \rho_{UMa} > \rho_{UMa}^t \;, under\ coverage \\ P_{UMi\to UMa} = P_{UMa} | \rho_{UMi} > \rho_{UMi}^t \;, under\ coverage \\ P_{UMa\to UMa} + (\sum P_{UMa\to UMi}) + P_{UMa\to UMa'} = 1 \\ P_{UMi\to UMi} + P_{UMi\to UMa} + P_{UMi\to UMi'} = 1 \\ \sum_i \lambda_{ij} \leq \psi_j \\ \sum_{i\in iOS} W_{ij} < \phi_{iOS} \quad , j \in \{UMa, UMi\} \\ \sum_{i\in Android} W_{ij} < \phi_{Android} \end{cases} \quad (8)$$

where the objective function is minimizing both totals waiting time for iOS and for Android traffic flows. This problem has seven constraints in total. The first two constraints define the transition between UMa and UMi when the SD-RAN is overloaded by considering load thresholds $\rho_{UMa}^t, \rho_{UMi}^t$ . The third and fourth constraints are related to the general theorem such that the total probability of incoming and outcoming flows in a node should be equal to 1. The fifth constraint defines that the resource capacity $\psi_j$ should meet the total arrival of users. Therefore,

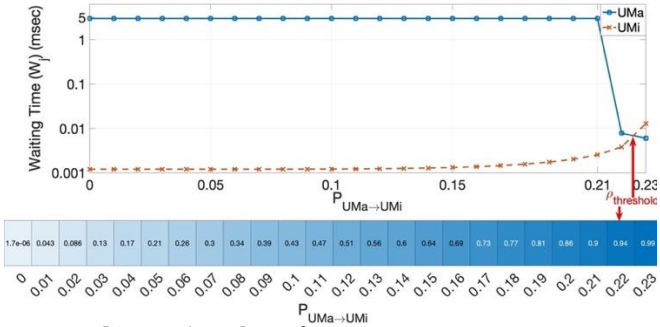
the first five constraints are directly related to load balancing for an optimization problem. Because of including both discrete and continuous variables in these constraints, it is performed by a Mixed Integer Linear optimization problem, and the optimal solution is normally executed by using the Branch and Bound algorithm. However, such a solution results in NP-hard complexity which is not acceptable for 5G requirements. Therefore, the constraints seven and eight are also added to this optimization problem. They defines the acceptable 5G requirements $\phi$ for iOS and Android users. To handle it, this problem is solved by a greedy algorithm. The expected result is negligible serving time in the SD-RAN control plane with a greedy algorithm in low complexity. The threshold analysis is detailed in subsection 3.2.1 and the proposed algorithm is given in subsection 3.2.2.

### 3.2.1. Local Search for ρ Thresholds

In this subsection, the load thresholds are analyzed by using a local search of transition probabilities between UMa and UMi [16]. These thresholds enable to focus of much more small search space on an optimization problem, and therefore, it extremely decreases the complexity of the proposed load balancing algorithm. Thanks to this contribution, the control plane meet one of the 5G requirement defined as giving response to the data plane in 1 second.




**(a)** $\rho_{UMa}{}^{t}$ analysis for UMa to UMi transition.




**(b)** $\rho_{UMi}{}^{t}$ analysis for UMa to UMi transition.

**Figure 3.** Threshold analysis for each transitions.

In Figure 3, there are two subfigures for $\rho_{UMa}^{t}, \rho_{UMi}^{t}$ analysis, respectively. In each subfigure, there is a line-graph for Waiting Time and there is a colormap for the load per SD-RAN. Each graphs are executed according to x-axis which shows the increased transition probabilities for UMa and UMi. The analyses depend the parameters given in a Figure 4.

- $\rho_{UMa}^{t}$ analysis: In figure 3a, x-axis shows the transition probability between UMi to UMa where there is no assigned user in UMa initially. As the probability increases; i.e. as the load is balanced by transfering from UMi to UMa layers, the Waiting Time in UMa increases, whereas it decreases in UMi according to eqs. 6 and 7. There is one optimum point and the local search defines the optimum threshold as 0.79 when the transition probability from UMi to UMa is 0.105.
- $\rho_{UMi}^{t}$ analysis: In figure 3b, x-axis shows the transition probability between UMa to UMi where there is no assigned user in UMis initially. As the probability increases; i.e. as the load is balanced by transfering from UMa to UMis, the optimum threshold is found as 0.94 when the transition probability from UMa and UMi is 0.22.

### 3.2.2. Proposed Load Balancing Algorithm

**Algorithm 1** Load Balancing Algorithm

**Require:** $P_{UMa \rightarrow UMa'}$ ,$P_{UMi \rightarrow UMi'}$ , OpenFlow statistics
**Ensure:** $P_{UMa \rightarrow UMi}$, $P_{UMi \rightarrow UMa}$
1: **for all** UMa in a topology do in parallel **do**
2: **if** $\rho_{UMa} \geq \rho^{t}_{UMa}$**(0.79) then**
3: Call Transition(UMa,UMi)
4: **end if**
5: **end for**
6: **for all** UMi in a topology do in parallel **do**
7: **if** $\rho_{UMi} \geq \rho^{t}_{UMi}$ **(0.95) then**
8: Call Transition(UMi,UMa)
9: **end if**
10: **end for**
11: **function** TRANSITION(x,y)
12: **while** $W_{ix}$ for iOS is not acceptable || $W_{ix}$ for Android is not acceptable **do**
13: **if** There is an Android flow in OpenFlow table of x **then**
14: Find an Android flow that would be transmitted to y
15: Check y is available
16: **if** $\rho_y < \rho^{t}_{y}$ **then**
17: Embed OpenFlow rules to x and y to meet this transition
18: Increase $P_{x \rightarrow y}$
19: **end if**
20: **else**There is an iOS flow in OpenFlow table of x
21: Find an iOS flow that would be transmitted to y
22: Check y is available
23: **if** $\rho_y < \rho^{t}_{y}$ **then**
24: Embed OpenFlow rules to x and y to meet this transition
25: Increase $P_{x \rightarrow y}$
26: **end if**
27: **end if**
28: Calculate new $W_{iy}$ for iOS and Android
29: Calculate new $\rho_x$ and $\rho_y$
30: **end while**
31: **return** $P_{x \rightarrow y}$
32: **end function**

The pseudo code of the proposed Load Balancing algorithm is given in Algorithm 1. It requires transition probabilities within UMa and UMis and OpenFlow statistics from the data plane. The main part is defined between lines 1-10. Lines between 1 and 5 check the threshold of UMa which is theoretically defined threshold 0.79 in the previous subsection. Lines between 6 and 10 also check the threshold of UMi which is theoretically defined threshold 0.95. If the SD-RANs are overloaded according to these thresholds, the transition functions are executed for load balancing from x to y. For lines 1-5, x equals to UMa and y equals to UMi; whereas for lines 6-10, they are in the opposite cases. Between lines 11-32, the transition function from x to y is defined. It checks the constraints of the proposed optimization formula in eq.8. Between lines 12-30, there is a loop that runs until all waiting times of iOS and Android users are acceptable. The algorithm firstly tries to find an Android flow to change destination SD-RAN by checking OpenFlow statistics. The reason of it, the Android applications newly use UDP-based QUIC protocol which does not need the whole connection setup as iOS applications need. This characteristic of QUIC protocol makes the route of a flow changeable without any packet loss. While balancing the load in an SD-RAN, firstly the Android flows are tried to be transmitted through other SD-RANs. If there are no suitable Android flows, in order to balance load the route of iOS flows is taken into consideration. If there is neither Android nor iOS flows in topology, this proposed algorithm does not perform load balancing.

## 4. Performance Evaluation

As given in Figure 5, the MATLAB R2019b is used for the simulation environment of SD-RAN. The Simulink builds a topology where there are two UMas and four UMis in total. It is divided into the data and control planes. In the control plane, there is SD-RAN controller where the proposed load balancing algorithm as given in Algorithm 1 is implemented by Stateflow library of MATLAB. In the data plane, there are two user types. One has eMBB flow from remote server in core to Android user in the edge that is carried by QUIC protocol; the other one has also eMBB flow of which the destination is the iOS user and it is carried by HTTP2.0 protocol. These users are shown in blue and red blocks respectively. They are generated at the same rate such as 50% Android and 50% iOS users in a topology. These flows are routed over either UMas or UMis where they are colored by gray and yellow blocks, respectively. The other elements in this environment are for measuring the elapsed time, and generating arrivals of M/G/1 and G/G/1 flows. The arrival rate of eMBB flow varies between 33333 and 8333333 packets/sec in a whole topology where the eMBB load per UMa is in range [0,1.5]. The total number of users is increased up to 36300 while performing an evaluation. The physical layer parameters for UMa and UMi are also found in Table 3. Uma has a 4GHz spectrum with 200MHz bandwidth usage; whereas UMi has 30 GHz spectrum

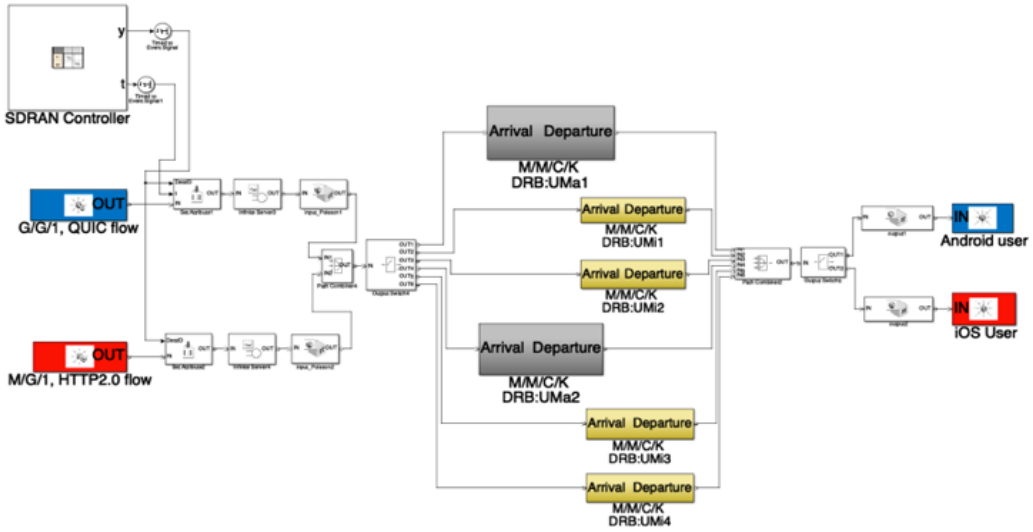



**Figure 4.** Performance evaluation scenario.

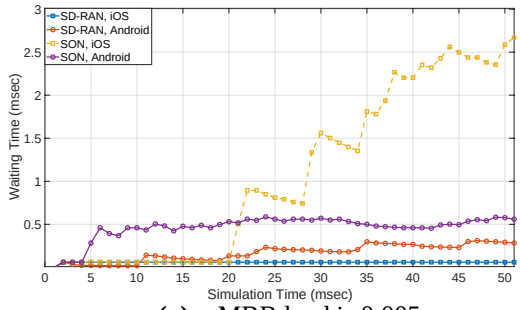
with 1000MHz bandwith usage. The serving rates are taken as 6e-5 and 1.2e-5 in Matlab with 10000 queue size.

**Table 3.** Experimental setup details.

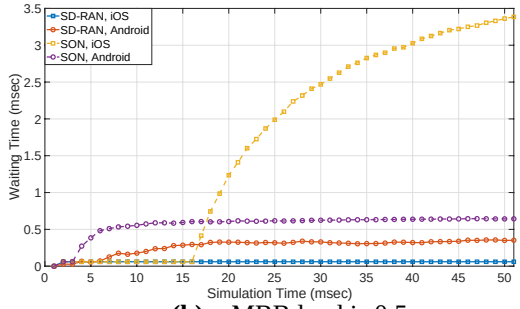
| | Spectrum, Bandwitdh | Cell range | Channel | Serving Rate | Queue size |
|---|---|---|---|---|---|
| UMa | 4GHz, 200MHz | 1000m | 20 | 6e-5 secs/packet | 10000 |
| UMi | 30GHz, 1000MHz | 400m | 7 | 1.2e-5 secs/packet | 10000 |

In Figure 5, the waiting time analysis is given for the proposed SD-RAN and the conventional SON for different user types such as iOS and Android via their protocols named HTTP2 and QUIC. There are four different eMBB loads as the arrival rate for UMa in the topology. As eMBB load is increased from 0.005 to 0.5, it can be served under an acceptable level (4 milliseconds) in both SD-RAN and the conventional SON. However; as it is increased from 1 to 1.5, the utilization of UMas $(\rho_{UMa})$ in the conventional SON goes up to 1, which means the queue becomes full and the packet drops start. In other words, the received packets by iOS or Android users is less than generated in the eMBB server. This extremely damages the QoS of both iOS and Android users because of not having centralized load balancing and user type awareness in the conventional SON. On the other hand; in the proposed SD-RAN, this eMBB load can be dynamically distributed through suitable UMis according to predetermined thresholds in sub-section 3.2.1. When eMBB load is increased up to 1.5, SD-RAN can serve all eMBB flows under 2 milliseconds level; unfortunately, the waiting time extremely exceeds 4 milliseconds level in the conventional SON. Moreover; the utilization of UMas and UMis are balanced in SD-RAN, and therefore, the received packets are nearly the same as generated in the eMBB server during simulation time. It is important to emphasize that the proposed Algorithm 1 firstly transfers QUIC based eMBB flows (Android users) through other RANs not to cause an unnecessary interruption in TCP based HTTP2 flows of iOS users. Because; switching TCP based HTTP2 flows unnecessarily, can make the QoS worse than an unbalanced case in the topology when it tries to balance. Therefore, the waiting time of HTTP based eMBB flows is higher than QUIC ones in the proposed SD-RAN, but all are under the acceptable level where there is no quality damage visible to the user.

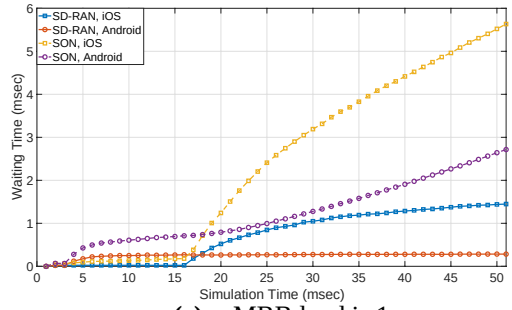
The analysis for the received packet count is also shown in Figure 6. When the eMBB load is 0.005 and 0.5, there is no packet drop in the proposed SD-RAN for iOS users; whereas, the QoS in other cases are over 90% level according to the received packet count. When the eMBB load is 1, the received packet count is nearly 19500 for both iOS and Android users in SD-RAN; whereas, it outputs as 17125 for Android and 15825 for iOS users. Namely; the proposed SD-RAN serves the mobile users with 97.5% QoS; whereas, the QoS decreases 80% level with the conventional SON. Here, SD-RAN outperforms 17.5% QoS than the conventional one. When the load is 1.5, the received packet count is nearly 30000 in the proposed SD-RAN by 85.7% QoS; whereas, the conventional one receives only 17760 and 15535 packets for Android and iOS users respectively, which outputs 44% QoS. Here, SD-RAN outperforms nearly 40% QoS than the conventional SON according to the received packet count.



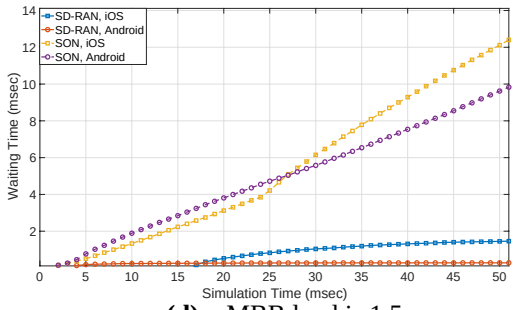

**(a)** eMBB load is 0.005.




**(b)** eMBB load is 0.5.

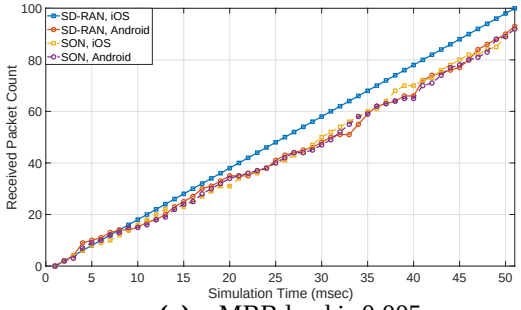



**(c)** eMBB load is 1.

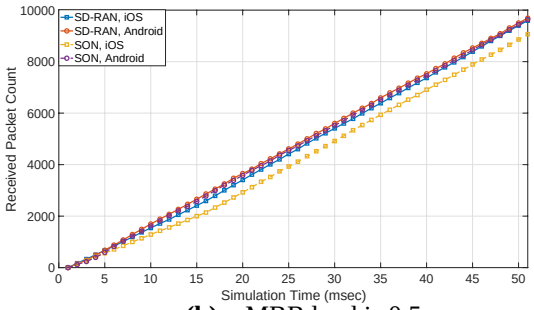



**(d)** eMBB load is 1.5.

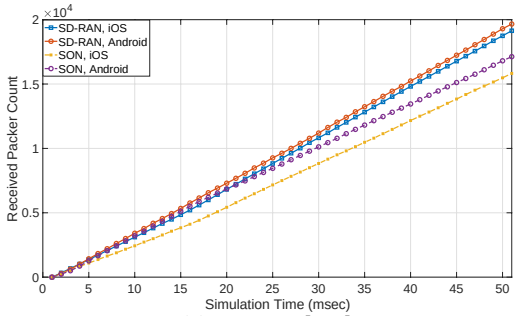
**Figure 5.** Waiting Time analysis of the proposed SD-RAN and the conventional SON for different user types such as iOS and Android.



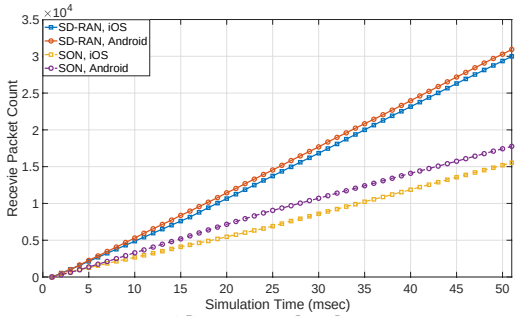

**(a)** eMBB load is 0.005.




**(b)** eMBB load is 0.5.

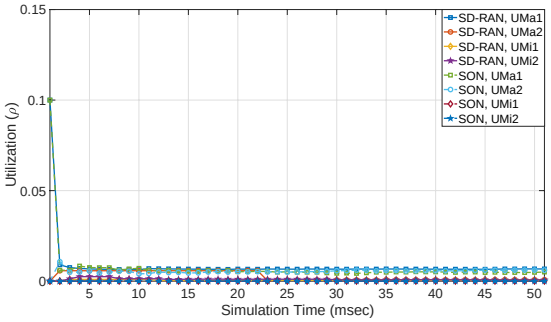



**(c)** eMBB load is 1.

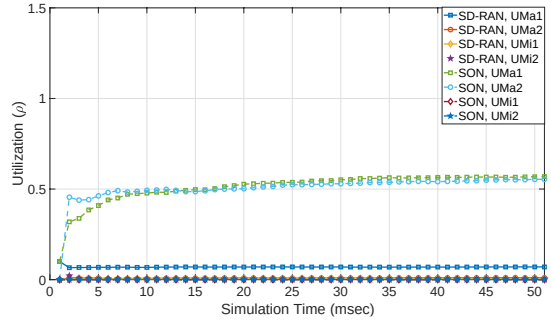



**(d)** eMBB load is 1.5.

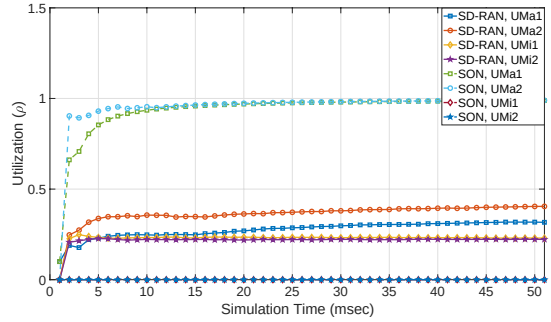
**Figure 6.** Received Packet Count analysis of the proposed SD-RAN and the conventional SON for different user types such as iOS and Android.



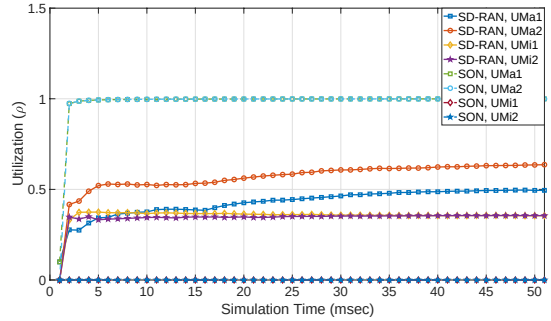

**(a)** eMBB load is 0.005.




**(b)** eMBB load is 0.5.




**(c)** eMBB load is 1.




**(d)** eMBB load is 1.5.

**Figure 7.** Utilization analysis of UMas and UMis for the proposed SD-RAN and the conventional SON.

The analysis for the utilization $(\rho_{UMa}, \rho_{UMi})$ is also shown in Figure 7. When the eMBB load is less than 0.5, the utilization of UMas and UMis are also under 0.5 for both the conventional and proposed schemes which also means each can serve eMBB flows under the acceptable level (4 milliseconds) with at most 0.5 utilization ratio. However; when the eMBB load is 1 and 1.5; the utilization of UMas in the conventional SON quickly becomes 1 during the simulation. Here, the utilization of UMis is at too low level. Due to not having a global view, it cannot balance load in optimal on the topology. On the other hand, the utilization of UMas and UMis are in nearly 0.6 and 0.4 levels in the proposed SD-RAN, which clearly results in the optimal load balancing on the whole topology. While doing this, it also considers the user type as iOS and Android. As a result, it can serve 40% more users than the conventional one without any extra expenditure on physical infrastructure.

## 5. Conclusion

In this paper, a novel mobile user type aware load balancing in SD-RAN is proposed. It has two new functions: Waiting Time Function (WTF) and Load Balancing Function (LBF). In WTF; by considering mobile user type, it differentiates users such as iOS and Android which commonly use HTTP2.0 and QUIC protocol for eMBB flows. Thanks to a novel analytical model of waiting time parameter (M/M/c/K) for iOS (M/G/1) and Android (G/G/1) users, the proposed load balancing algorithm is easily performed in the control plane without any hardware implementation in the data plane. In LBF; to balance eMBB load by considering the user types, DPI in SDN is also handled by queuing inspired approach without any expenditure in OpenFlow standard library. A novel Mixed Integer Linear Problem is defined for the waiting time optimization. To overcome NP hardness of this problem, the load thresholds for UMa and UMis are preliminary analyzed and determined as 0.79 and 0.94, respectively. According to performance results; SD-RAN outperforms nearly 40% QoS than the conventional SON according to received packet count and it can serve 40% more users than the conventional one without any extra expenditure on physical infrastructure. When eMBB load is extremely high, SD-RAN can serve mobile users under a 2 milliseconds level.

## Appendix A. M/M/c/K model

In M/M/c/K queuing model, the probability density function of arrival rate has different distributions for $0 \leq n < c$ and $c \leq n < K$ cases. It is in Poisson and Geometric distributions for these cases, respectively. The probability density function $P_n$ is defined as follows:

$$P_n = \begin{cases} \frac{r^n}{n!} P_0 & , \ 0 \le n < c \\ \frac{r^n}{c^{n-c} c!} P_0 & , \ c \le n \le K \end{cases} \quad \text{(A.1)}$$

where $r = \lambda \backslash \mu$, and c is the number of server and K-c is the length of queue. By executing $\sum P_n = 1$ theorem, $P_0$ is calculated below:

$$P_0 = \begin{cases} \frac{1}{\frac{r^c}{c!}\left(\frac{1-\rho^{K-c+1}}{1-\rho}\right)+\sum_{n=0}^{c-1}\frac{r^n}{n!}} & , \ \rho \neq 1 \\ \frac{1}{\frac{r^c}{c!}(K-c+1)+\sum_{n=0}^{c-1}\frac{r^n}{n!}} & , \ \rho = 1 \end{cases} \quad \text{(A.2)}$$

By executing L'Hospital rule on $\sum nP_n$, the number of packets waiting in a queue (Lq) is calculated. Then, the number of packets in the whole system (L) is found as follows:

$$L = L_q + r(1 - P_k) \quad \text{(A.3)}$$

where $P_k$ is the blocking probability. As a result, the total waiting time (W) for M/M/c/K model is calculated by using eqs.A.1,A.2, and A.3 as follows[15]:

$$W = \frac{L}{\lambda(1-P_k)} \quad \text{(A.4)}$$

**Statement of Research and Publication Ethics**

The authors declare that this study complies with Research and Publication Ethics.

## References


[1] Ericsson, “Update 2020 Ericsson Mobility Report”, *Ericsson Technical Report*, Uen, Stockholm, Sweden, EAB-20:006745, 2020.

[2] S. E. Elayoubi, S. B. Jemaa, Z. Altman and A. Galindo-Serrano, "5G RAN Slicing for Verticals: Enablers and Challenges," *IEEE Communications Magazine*, vol. 57, no. 1, pp. 28-34, January 2019, doi: 10.1109/MCOM.2018.1701319.

[3] Ericsson, “Ericsson Mobility Report”, *Ericsson Technical Report*, Uen, Revision B, Stockholm, Sweden, EAB-18:012366, 2018.

[4] E. Hossain and M. Hasan, "5G cellular: key enabling technologies and research challenges," *IEEE Instrumentation & Measurement Magazine*, vol. 18, no. 3, pp. 11-21, June 2015, doi: 10.1109/MIM.2015.7108393.

[5] Apple, “Apple os deployement quide for the enterprise”, *Apple*, 2022. Available: https://support.apple.com/en-us/HT202944. [Accessed: 01.2022].

[6] W3Techs, “Web technology surveys”, *Q-success*, 2022. Available: https://w3techs.com [Access Date: 01.2022].

[7] C. Xu, S. Chen, J. Su, S. M. Yiu and L. C. K. Hui, "A Survey on Regular Expression Matching for Deep Packet Inspection: Applications, Algorithms, and Hardware Platforms," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 4, pp. 2991-3029, Fourth quarter 2016, doi: 10.1109/COMST.2016.2566669.

[8] M. A. Ashraf, H. Jamal, S. A. Khan, Z. Ahmed and M. I. Baig, "A Heterogeneous Service-Oriented Deep Packet Inspection and Analysis Framework for Traffic-Aware Network Management and Security Systems," *IEEE Access*, vol. 4, pp. 5918-5936, 2016, doi: 10.1109/ACCESS.2016.2609398.

[9] P. Orosz, T. Tóthfalusi and P. Varga, "FPGA-Assisted DPI Systems: 100 Gbit/s and Beyond," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 2, pp. 2015-2040, Second quarter 2019, doi: 10.1109/COMST.2018.2876196.

[10] J. H. Cox et al., "Advancing Software-Defined Networks: A Survey," *IEEE Access*, vol. 5, pp. 25487-25526, 2017, doi: 10.1109/ACCESS.2017.2762291.

[11] S. Sun, M. Kadoch, L. Gong and B. Rong, "Integrating network function virtualization with SDR and SDN for 4G/5G networks," *IEEE Network*, vol. 29, no. 3, pp. 54-59, May-June 2015, doi: 10.1109/MNET.2015.7113226.

[12] M. Jarschel, F. Wamser, T. Hohn, T. Zinner and P. Tran-Gia, "SDN-Based Application-Aware Networking on the Example of YouTube Video Streaming," *2013 Second European Workshop on Software Defined Networks*, 2013, pp. 87-92, doi: 10.1109/EWSDN.2013.21.

[13] Calvin Hue, Yu-Jia Chen and Li-Chun Wang, "Traffic-aware networking for video streaming service using SDN," *2015 IEEE 34th International Performance Computing and Communications Conference (IPCCC)*, 2015, pp. 1-5, doi: 10.1109/PCCC.2015.7410288.

[14] R. Udechukwu and R. Dutta, "Extending Openflow for Service Insertion and Payload Inspection," *2014 IEEE 22nd International Conference on Network Protocols*, 2014, pp. 589-595, doi: 10.1109/ICNP.2014.94.

[15] D. Gross, J.F. Shortle, J.M. Thompson, C.M. Harris, *Fundamentals of Queuing Theory*, Wiley-Interscience, 4th edition, New York, NY, USA, 2008.

[16] F. Hillier, G. Lieberman, *Introduction to Operations Research*, McGraw-Hill International Editions, 2001.