EMITTER International Journal of Engineering Technology
Vol. 11, No. 2, December 2023, pp. 174~191
DOI : 10.24003/emitter.v11i2.772

The Network Slicing and Performance Analysis of 6G Networks using Machine Learning

Mahesh H. B[1], Ali Ahammed G. F[2], Usha S. M[3]

[1]Department of Computer Science & Engineering, PES University, Bengaluru | Visvesvaraya Technological University, Belagavi, India
[2]Department of Computer Science & Engineering, PG Center, Visvesvaraya Technological University, Mysuru, India
[3]Department of Electronics & Communication Engineering, JSS Academy of Technical Education, Bengaluru, India
Corresponding author: ushasm@jssateb.ac.in

Received January 31, 2023; Revised September 23, 2023; Accepted November 19, 2023

Abstract

6G technology is designed to provide users with faster and more reliable data transfer as compared to the current 5G technology. 6G is rapidly evolving and provides a large bandwidth, even in underserved areas. This technology is extremely anticipated and is currently booming for its ability to deliver massive network capacity, low latency, and a highly improved user experience. Its scope is immense, and it's designed to connect everyone and everything in the world. It includes new deployment models and services with extended user capacity. This study proposes a network slicing simulator that uses hardcoded base station coordinates to randomly distribute client locations to help analyse the performance of a particular base station architecture. When a client wants to locate the closest base station, it queries the simulator, which stores base station coordinates in a K-Dimensional tree. Throughout the simulation, the user follows a pattern that continues until the time limit is achieved. It gauges multiple statistics such as client connection ratio, client count per second, Client count per slice, latency, and the new location of the client. The K-D tree handover algorithm proposed here allows the user to connect to the nearest base stations after fulfilling the required criteria. This algorithm ensures the quality requirements and decides among the base stations the user connects to.

Keywords: 6G Technologies, KD Tree, Slicing, Connection ratio, Latency.

1. INTRODUCTION

In terms of capability, fulfilment, and amalgamation of a broad spectrum, 6G networks have recently undergone a transformation. The requirement for a higher data rate was one of the main factors that influenced the development of 6G. Data rate, coverage area, and the number of accessories that could connect to a given mobile network increased significantly with each successive generation that was introduced. Additionally, As the demand for the network increases, the allocation of existing infrastructure to all the subscribers becomes a tedious task. It demands new technology as per the requirements. The authors, C. Yang, W. M. Shen, and X. B. Wang, developed the new technology to cope with the existing infrastructure to support the demand. If the newer technology advanced meets the requirements of the subscribers by sustaining the required quality of performance with time, then it is called the next generation evolution [1]. E. C. Strinati, S. Barbarossa, J. and R. H. Wen, G. Feng, J. H. Tang, T. Q. S. Quek, G. Wang, W. Tan, and S. Qin describe the peak data rate and impairments. Assuming that there are no impairments in the wireless channel, the minimum requirement peak data rate is at least expected to be 20 GB/s. The data rate under ideal conditions is 20 GB/s, which is the maximum possible data rate. It is difficult to meet this requirement in real time [2-3]. J. Mei, X. B. Wang, and K. Zheng analysed the network delay, the capacity of the network, and the requirements of 6G, noting that according to ITU-R, the network should be able to provide service to those devices that are moving at 500 Km/hr, and at least one million devices must be able to be connected to the network in an area of 1 km/hr. meaning that 6G should have the capacity to connect enormous devices at the same time. The latency should not be greater than 1 ms for URLLC and 4 ms for eMBB[4]. T. Taleb, M. Corici, and C. Parada described the data rate experienced by the users. The user located in the same building should be at least able to experience a 1000 Mbps data rate. Table 1 depicts the 5G and 6G technology requirements in a nutshell.

In addition to all the technical prerequisites, there are three primary 6G network usage scenarios [5]. This is briefly described in Figure 1. M. Bagaa, T. Taleb, and A. Laghrissi explained the three layers of 6G networks: the service layer, the network function layer, and the infrastructure layer. The infrastructure layer consists of radio access networks, transport networks, and core networks. Network-related functions and activities are performed through the network function layer. The service layer includes the services provided to the virtual mobile operator and third-party service providers [6]. The authors, P. Caballero, A. Banchs, and G. de Veciana, explained that the 5G scenarios include Enhanced Mobile Broadband, Massive Machine types, and Ultra-reliable and low-latency communications such as data transfer, voice, smart homes and buildings, 3D video, UHD screens, work and play in the cloud, Augmented reality, Industry automation, mission-critical applications, e.g., e-health, and self-driving cars [7]. Faster data rates and a richer user experience are made possible by Enhanced Mobile Broadband (eMBB). The bandwidth and the data relation are given by Nyquist's formula: data rate = 2 * Bandwidth * log2(M) as depicted in Table 1 [6-7].

Table 1. Bandwidth and Data rate supported by 4G, 5G and 6G network

Attribute	**4G**	**5G**	**6G**
Bandwidth	5-20Mhz	1 GHZ	100G
Data rate	Up to 20 Mbps or more	0.1 G/sec	1 G/sec

A typical architecture of a network slice as depicted in Figure 1 would include:

1 The resource layer, which is the lowermost layer consisting of network resources and network functions, is the functional block providing network capabilities.

2 The network slice instance layer, which is the middle layer consisting of the actual slices – the slices can be isolated or can share resources.

3 Service instance layer, the uppermost layer which consumes the slices from the lower layer and provides services to the customers. The network slice instance layer is considered to analyze the latency.

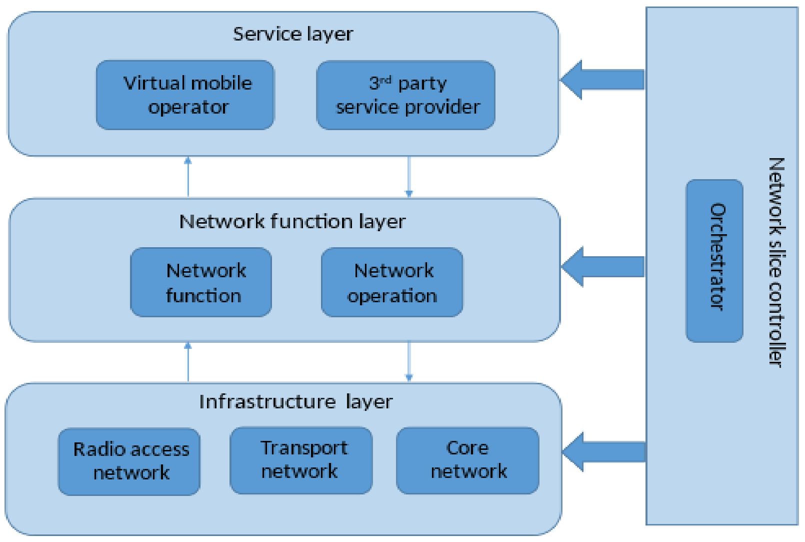


Figure 1. Network Slice Controller

2. RELATED WORKS

L. Lee, J. Loo, T. C. Chuah, and L. C. Wang, and Y. N. Liu, X. B. Wang, and G. Boudreau Massive described the machine-type communications (mMTC): This technology enables high connection density and long battery life for a large number of connected devices. Smart homes and cities are only a few examples of their uses. Certain applications are time-critical, such as surgery done in remote places or industrial automation, to name a few examples. URLLC supports such low-latency applications [8-10]. Network slicing can be implemented at a lower cost. Business growth can be seen through next-generation 5G and 6G wireless techniques

L. Li, W. S. Shi, P. Yang, Q. Ye, X. S. Shen, and M. Zambianco and G. Verticale explained the "hierarchical soft RAN slicing framework for differentiated service provisioning with the use of network slicing", 5G and advanced technology can nimbly offer a variety of services, including voice and video streaming. By building logical end-to-end networks on top of a shared network infrastructure, it makes this possible. Each of these networks is capable of supporting a particular service type with diverse requirements. It's a paradigm that permits the provision of distinct 6G services while using the same infrastructure. The theory behind this is that focusing on a smaller set of criteria rather than a larger number is more efficient because a service only has a small set of requirements that are specific to the use case it supports. A network slice instance [11-12] is an end-to-end logical network pertaining to a specific use case. A network slice subnet instance is a logical network, and one or more subnet instances together form a network slice instance.

The authors, Jie Mei, Xianbin Wang, and Khan Zheng, published a paper entitled "an intelligent self-sustained RAN slicing framework for diverse service provisioning in 5G-beyond and 6G networks". They have noted that in Intelligent and Converged networks, the components of Network slicing, ranging from the types of resources present to orchestration and isolation, are presented, as well as how SDNs andFV technologies are combined to realise the true potential of Network slicing in 5G architecture. For a discussion of key terms related to network slicing, including virtualization, orchestration, NFs, etc. According to the circumstances in the rest of the network, the basic concept of this study [13] is to locally redefine the delay requirements in each network domain. In this paper, the architectures that can satisfy different QoS demands for the clients and the open challenges that are faced while deploying 5G systems in the public, which include network re-construction, are discussed, as well as the proposed 5G models on 5G breakout, mobility management, and virtualized resource allocation. The core cloud, which divides the control plane from the user plane, has evolved from the CN's typical centralised architecture in order to minimise control signalling and data transmission latency [14–18].

The notion of 5G network slicing, its levels and structural framework, as well as the prevention of attacks, threats, and challenges that indicate how network slicing affects the 5G network, are all covered in this article. Along with making a substantial addition to the subject, this research also compares earlier surveys and plots categories to show different machine learning approaches for various application parameters and network functions [19–20]. In this paper, an architectural framework for 5G networks called "network slicing" is designed to support a range of different networks. Due to the rising need for data rates, bandwidth capacity, and low latency, network slicing is becoming more crucial in order to completely satisfy the demands of diverse use cases on the network [21–24]. This article emphasises the advantages and potentials of AI-based methods in the study of NGWNs [25]. In this study, for delay-oriented Internet of Things (IoT) services, a computing job scheduling problem in the space-air-ground integrated network (SAGIN) was provided. When compared to probabilistic configuration approaches, the suggested algorithm can meet the UAV energy capacity restriction while reducing task processing duration by up to 30% [26]. Adiraju, P. R., & Voore Subba Rao. [27], stated the dynamic resource allocation in 5G network using intelligent algorithm, and Saeed, A. B., & Gitaffa, S. A.-H [28] described about artificial neural processor used in wireless sensor network.

3. ORIGINALITY

The system is unique in that it presents a real-world scenario in a clear, useful way. A network operator can reuse the modules as and when needed in accordance with their preferred standards if they want to monitor the operation of 5G network slicing before deployment. Because it adheres to object-oriented programming, the code is maintainable, and its modules can be used in various scenarios. The system is incredibly resourceful because it makes efficient use of the resources that are at its disposal. The locations are randomised for the user.

1. The ratios used to divide the three slices ensure that they all sum up to one.
2. The network service providers can evaluate how well the slices perform.
3. The customer would proceed to the waiting room if the base station did not have enough room to accommodate them. The slices are given a certain distribution in the appropriate ratio after only three slices were taken into consideration for the investigation. The linked devices must always be connected in the sense that a seamless handoff is taking place to prevent connectivity issues. As a result, the connections are active at all times. The system as a whole has been developed with security in mind.

4. SYSTEM DESIGN

The approach chosen here was to take input from the user regarding base station locations and attributes of each base station, which include coordinates, maximum number of clients, slice ratios (fraction of total base station resource that is willing to be allotted for each slice, which totally add up to one), and the coverage area up until the base station is able to service a client. The number of clients that are simulated by this setup of base stations can also be changed and tested. On running the simulation, a set of graphs is returned, and on the command line, a log is generated about the connect and disconnect actions of each client with a base station. The graphs that are returned are a geographical location graph at the end of the simulation showing the final positions of clients with the base stations, a connection ratio graph, and various connected client count graphs. This approach takes inspiration from another project called SliceSim. The previous approach was similar in the sense that it took base station locations and analysed them with random client generation, but the graphs returned were not real-time. The graphs that our simulation gives give a much better picture of how the simulation is working with different slices. Figure 2 represents the flowchart of the proposed methodology.

Phase 1 starts with checking if the client is already connected or not to a base station. In the case that it already is and the client is in the coverage area of the base station it was connected to, it will directly jump to the next phase. In the case that it is not connected, the K-D Tree is queried to find the nearest connectable base station. If it still cannot find a base station, then the connected attribute will just be set to false.

Phase 2 consists of very simply adding values to the coordinates of the client's location to change them. This will happen to all clients, regardless of their location or connected state.

Phase 3 consists of checking if the new location is still present in the coverage area of the previously connected base station. Above are the main phases of the simulation for a client. During these stages, arrays and dictionaries are maintained as attributes in the base station and client classes for the purpose of stat collection and graphical analysis. Also, delays are introduced between each stage to simulate a real-time usage environment.

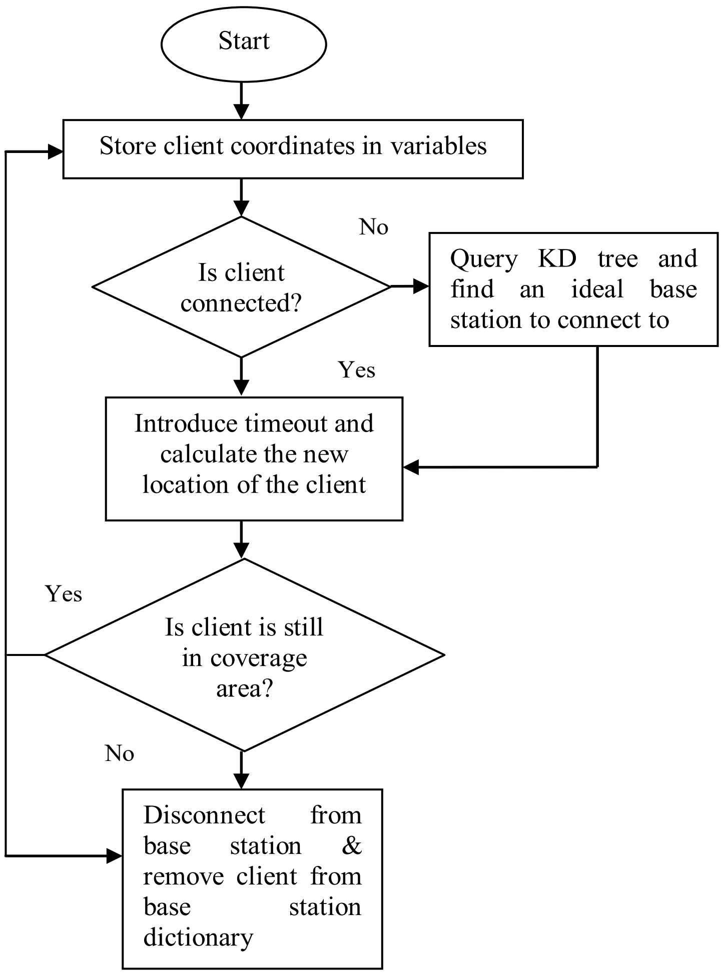

Start
Store client coordinates in variables
Is client connected?
No
Query KD tree and find an ideal base station to connect to
Yes
Introduce timeout and calculate the new location of the client
Yes
Is client is still in coverage area?
No
Disconnect from base station & remove client from base station dictionary

Figure 2. Proposed methodology

A binary search tree, termed a K-D Tree, has data in each node that is a K-Dimensional point in space. In a nutshell, it is a data structure for organising objects in a K-Dimensional sector that uses spatial segmentation. In a K-D tree, a non-leaf node divides the space into two halves, or half-spaces. The left subtree associated with that node represents points to the left of this area, while the right subtree represents values to the right of this space, as depicted in Figure 3, and K-D tree with points plotted on a plane is represented in Figure 4.

1. Create a 2D tree with the points (3, 6), (17, 15), (13, 15), (6, 12), (9, 1), (2, 7), and (10, 19).
2. Make (3, 6) the root node.
3. Compare (17, 15) with the root node [3, 6].
4. Looking at the alignment of the X node, the Y node is aligned.
5. Now compare (13, 15) with (17, 15); the X-value of is greater in the root node (17, 15), so this will be aligned to the right of the subtree of (3, 6).
6. Compare the Y-value of (13, 15) with the Y-value of (17, 15); since these are equal, the point lies on the right portion of the subtree (17, 15).
7. Compare (6, 12) with the root node (3, 6); the X-value 6>3, hence this point will lie on the right of the subtree of (3, 6).
8. Compare (6, 12) with the root node (17, 15); the Y-value is 1215; hence, this point will lie on the left of the subtree of (17, 15).
9. The point (9, 1) will lie at the right point of (6, 12).
10. The point (2, 7) will lie in the left point of (3, 6).
11. The point (10, 19) will lie to the left of the point (13, 15).

Figure 3. Determination of element insertion into the K-D tree

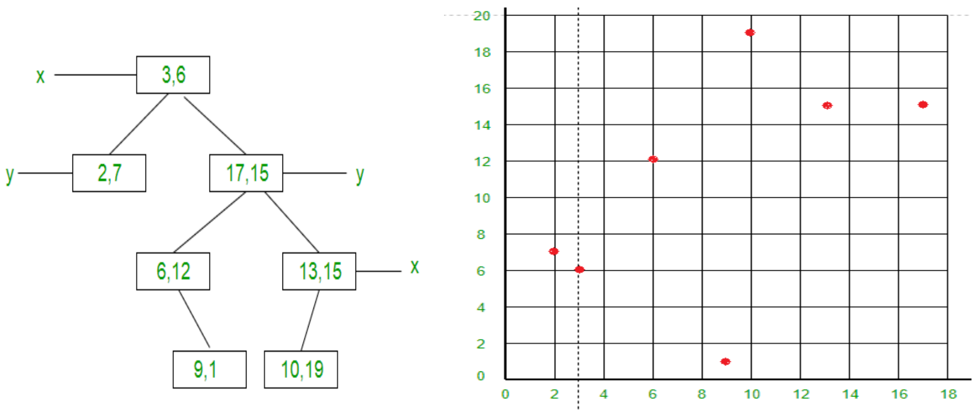


Figure 4. K-D Tree with the points (3, 6), (17, 15), (13, 15), (6, 12), (9, 1), (2, 7), (10, 19) plotted on plane

5. EXPERIMENT AND ANALYSIS

In the implementation phase, various modules were created, as explained below. The modules created for this work are Slice, Base Station, and Client.

5.1 Slice Module

The various slices, including eMBB, mMTC, and URLLC, are started using a "Slice" class. To provide us with real-time data about a given slice, each slice is given attributes like slice id, bw guaranteed, max users, latency, and functions like add user() and check usage().he former gives us an estimate of the number of users present in the slice, while the latter informs us about the guaranteed bandwidth possible with the current users in the slice.

From the slice module:

1. Class Slice is created and initialized with a maximum_users, Latency, slice ID, and a temporary variable.
2. Class slice: adds the user to the network.

Class slice: removes the user from the network.

1. Checks for latency
2. Checks for slice usage.

5.2 Base Station

The base stations' behaviour is managed by the base station class. The base station positions in the simulation are fixed because they are designed to be hardcoded. It has three primary components: a list of all the client objects kept in it, a hashmap showing the number of clients connected to it per slice, and a list of clients in a waiting area. The attributes that are taken by the base station are the coordinates, maximum bandwidth, coverage area, maximum users, and slice ratios, which are meant to be taken as input from the user. The waiting room specified the clients that were trying to connect to that base station, but there was no space left. The client list specifies the clients that are currently connected to that network. The required hardware and software for the work are depicted in Table 2. It contains the attribute names and specifications. The hardware attributes are namely PC with ubuntu software version 18.04, Intel i5 core, 8Gb RAM, 10Gb hard disk are used. The software attributes include cycler 0.10.0, kiwisolver 1.1.0, matplotlib 3.0.3, numpy 1.16.3, pillow 6.2.0, pyparsing 2.4.0 and python-dateutil 2.8.0 were used for the conduction and completion of the work. Ubuntu is more secure and runs without installation and friendly resources. Intel i5 processors have sufficient processing power to handle multiple tasks simultaneously. Kiwisolver is an efficient algorithm and it is lightweight and fast. Matplotlib is a library used to create graphs and charts here. Mathematical operations are performed using NumPy in this work. In this work pillow python is used for image processing. Pyparsing is used to process and manipulate the text or a bigger file into smaller texts. Python-datautil is used to enhance the readability, reusability, and maintainability of the code to perform the required task.

Table 2. Hardware & software requirements

Attribute No.	Attribute Name	Specification
1	PC running Ubuntu	18.04 or above,
2	Intel	i5
3	RAM	8GB
4	Hard Disk space	10GB
5	Cycler	0.10.0
6	Kiwisolver	1.1.0
7	Matplotlib	3.0.3
8	Numpy	1.16.3
9	Pillow	6.2.0
10	Pyparsing	2.4.0
11	python-dateutil	2.8.0

5.3 Client

The primary simulation class that manages the behaviour of the clients is called the client class. It is set up with the slice it is supposed to connect to and its coordinates. The assistance functions are connect(), which connects the client to a base station; distance(), which returns the distance between a client and a base station; and disconnect(), which would disengage the client from the base station. The above just shows the attributes that the client declares. The variables ‘connect_array’, ‘dic’, ‘connect_count’, ‘tree, and ‘bs’ are static variables. These are updated for graphical analysis in the client methods. All clients start with the connected variable set to false. The ‘gnb’ attribute of the client also starts as null and is updated only when a connection happens. As explained previously, the three major phases can be seen above, with timeouts in between to truly simulate the environment accurately. The main connect function sets the connect variable for that client to true. The arrays used for client connection ratio graphs are updated here. The static variable client connect count is also updated here. The disconnect function sets the disconnect variable to false. The static variable of connect client count is also reduced by one when a disconnect happens. The client is also removed from the base station's list of connected clients. The connect array is also updated here for the connection ratios graph. Slices are a dictionary that stores the number of clients for each slice ID. This is to check the number of slices residing currently and if the number is at its limit.

5.4 Results and Discussion

The geographic locations of the base station and the users are represented in Figure 5. Here, the base station coverage area is shown by each oval shape. Each slice is represented by a distinct shape: the eMBB slice is symbolised by a plus sign (+), the URLLC slice is shown by a diamond, and the final slice, the mMTC slice, is symbolised by an x.

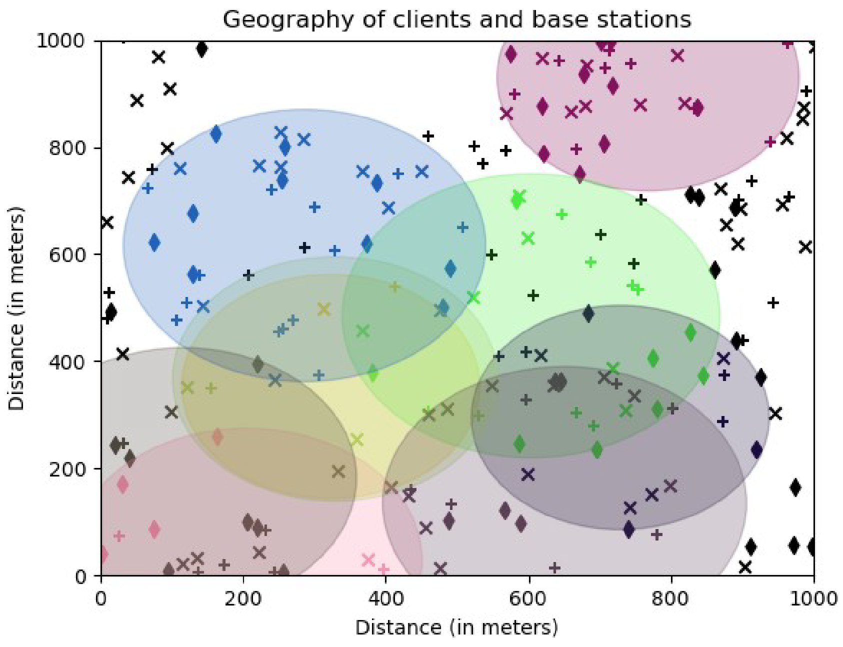


Figure 5. Geography of clients and base stations

Each coverage area is depicted by a different colour on the graph. This gives an idea of the user's connection to a particular base station. The unlinked users, those who are not fit for any of the cell areas, are depicted beyond the cell coverage area. Few users may be connected to more than one coverage area, as seen in the diagram. This is the simulation log output that is generated while the simulation is running. It can be used to check how many clients are able to connect and which base it is trying to connect to. Through this inference, one may examine how slices can be divided up and how a base station reacts to clients connecting to it in a real-world geographic area.

5.5 Slice Analysis-connection ratio

Figure 6 provides an overview of the ratio of the number of connected clients to the total number of clients present and the number of clients connected per slice. Each slice—eMBB (Slice ID-1), URLLC (Slice ID-2), and mMTC (Slice ID-3)—has a particular ratio associated with it. This graph simply represents how many clients are connected to each slice. Initially, the connection ratio increases with the number of clients, but after a certain point, it saturates. This is because once the base station limit is reached and no more clients can be connected, the same number of clients can exist in that range, depending on how and when they move out of the coverage area.

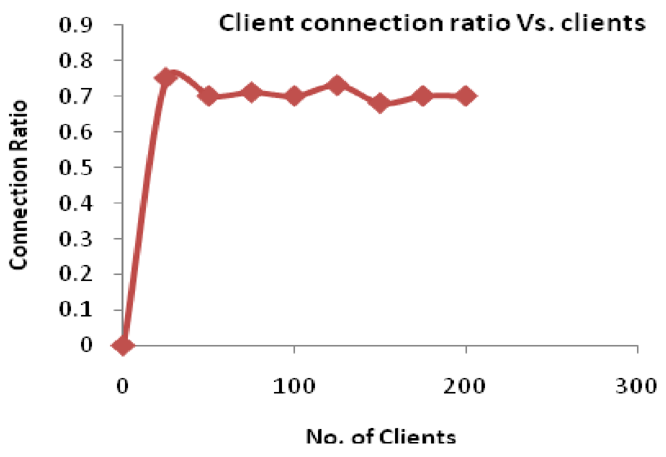


Figure 6. Overall client connection ratio

5.6 Slice analysis-clients connected per slice.

Figure 7 represents the number of clients connected per slice. Each slice-eMBB (Slice ID-1), URLLC (Slice ID-2), and mMTC (Slice ID-3)-has a particular ratio associated with it. This graph simply represents how many clients are connected to each slice. Slice ID-1 is connected to 49 clients; Slice ID-2 supports 49 clients again; and Slice ID-3 provides support to 39 clients.

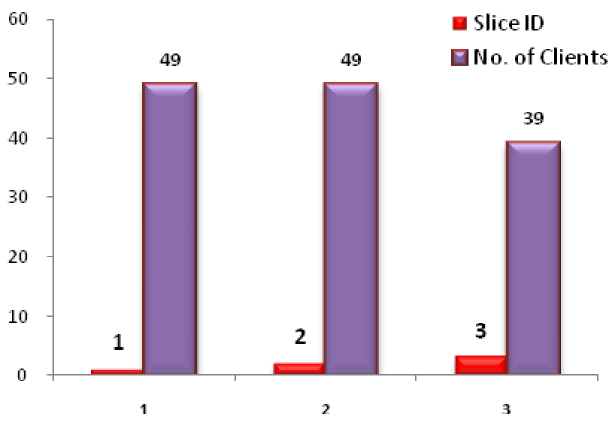


Figure 7. Client connection count per slice

5.7 Slice Analysis-Clients counts per second.

The slope of the graph occasionally drops because the client may leave the base station's service area or otherwise move outside of it. The slope then starts to rise once again. The clients join the base station service area based on the traffic density in that area. As shown below in Figure 8, around 238 clients were connected to the particular network slice, and after 10 seconds, the number of clients using the network increased to 246. Again, at 40 seconds, around 250 clients joined the network, and around 50 seconds later, the user count was reduced to 242.

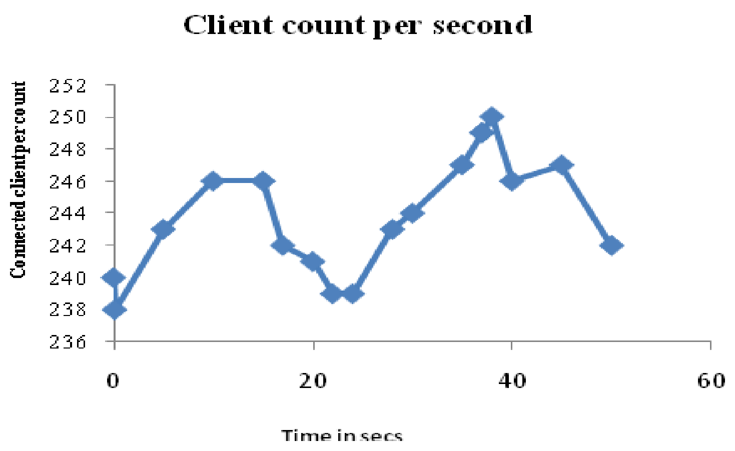


Figure 8. Overall client count wrt Time

5.8 Slice analysis-clients counts per uRLLC, Embb and mMTC Slice

The client count per uRLLC Slice over time is shown in Figure 9. The behaviour of the number of customers connected to a slice and how the connections vary over time can therefore be inferred from a real-world scenario. Around 150 clients were connected to the uRLLc slice at 20 seconds, and at 35 seconds, 158 were connected to the network. Based on the busy hours, the traffic density varies, and the number of clients connected to the network also varies

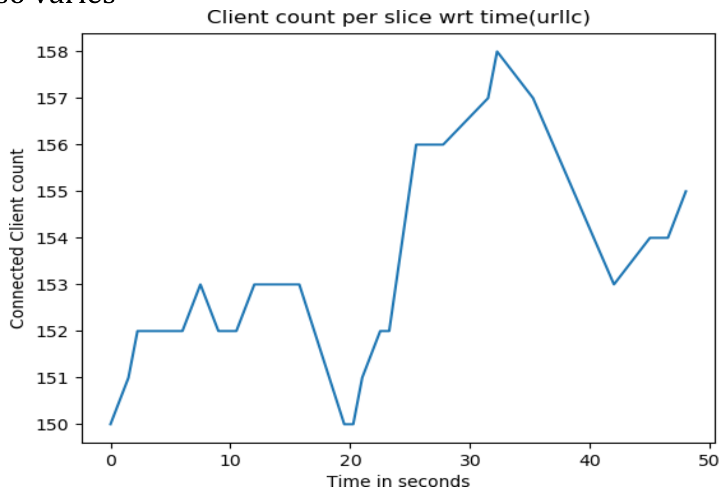


Figure 9. URLLC client count wrt time

In order to mitigate these dropouts, concepts like superpositioning URLLC and emBB packets for multiplexing [7] help to overlap packets from several slices together. In that circumstance, it is possible to simultaneously increase the connected count of customers across several slices. As depicted below in Figure 10, around 158 clients were connected to the Embb network, which remains regulated between 10 seconds and 20 seconds and 35 seconds to 45 seconds.

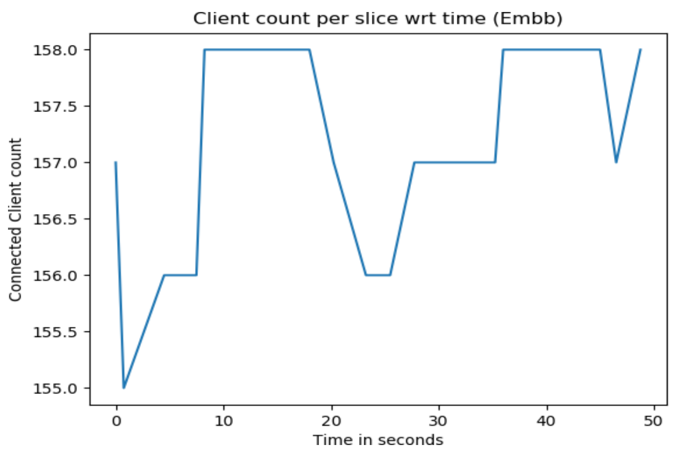


Figure 10. Embb Client count wrt Time

The client count with respect to the mMTC slice is depicted below in Figure 11. Around 129 clients are connected to the network initially, and the number increases to 131 at 5 seconds and remains constant until 10 seconds. At 12 seconds, the user count decreases to 130. The client's ratio increases and again falls back to 127 at 22 seconds. At 35 seconds, the client's ratio rises to 132 and gradually drops back to 129 at 50 seconds.

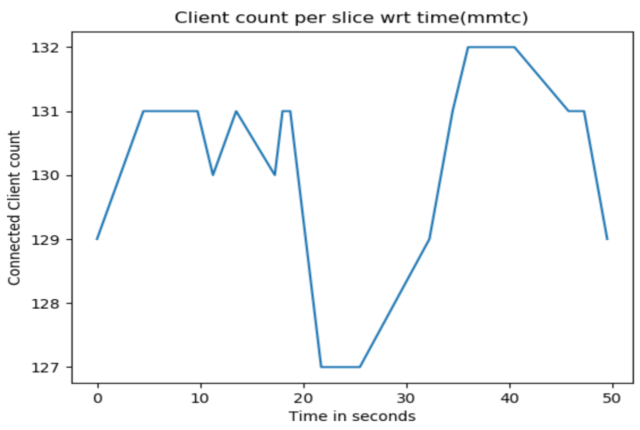
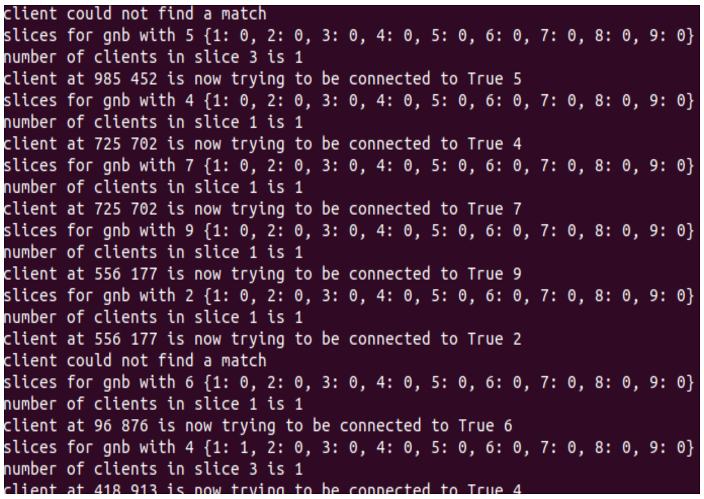


Figure 11. MMTC client count wrt time

Slice Analysis-Geography is represented below in Figure 12. The number of clients in slice 3 and slice 1 is analysed here. The graph that indicates the number of clients tied to the particular network is depicted below.



```
client could not find a match
slices for gnb with 5 {1: 0, 2: 0, 3: 0, 4: 0, 5: 0, 6: 0, 7: 0, 8: 0, 9: 0}
number of clients in slice 3 is 1
client at 985 452 is now trying to be connected to True 5
slices for gnb with 4 {1: 0, 2: 0, 3: 0, 4: 0, 5: 0, 6: 0, 7: 0, 8: 0, 9: 0}
number of clients in slice 1 is 1
client at 725 702 is now trying to be connected to True 4
slices for gnb with 7 {1: 0, 2: 0, 3: 0, 4: 0, 5: 0, 6: 0, 7: 0, 8: 0, 9: 0}
number of clients in slice 1 is 1
client at 725 702 is now trying to be connected to True 7
slices for gnb with 9 {1: 0, 2: 0, 3: 0, 4: 0, 5: 0, 6: 0, 7: 0, 8: 0, 9: 0}
number of clients in slice 1 is 1
client at 556 177 is now trying to be connected to True 9
slices for gnb with 2 {1: 0, 2: 0, 3: 0, 4: 0, 5: 0, 6: 0, 7: 0, 8: 0, 9: 0}
number of clients in slice 1 is 1
client at 556 177 is now trying to be connected to True 2
client could not find a match
slices for gnb with 6 {1: 0, 2: 0, 3: 0, 4: 0, 5: 0, 6: 0, 7: 0, 8: 0, 9: 0}
number of clients in slice 1 is 1
client at 96 876 is now trying to be connected to True 6
slices for gnb with 4 {1: 1, 2: 0, 3: 0, 4: 0, 5: 0, 6: 0, 7: 0, 8: 0, 9: 0}
number of clients in slice 3 is 1
client at 418 913 is now trying to be connected to True 4
```

Figure 12. Execution of slicing simulator results

5.9 Performance analysis

This section delves deeper into the performance of the three main slices: eMBB, URLLC, and mMTC.

1. **Latency**

   The three aforementioned slices latency performance is shown in Figure 13. Unsurprisingly, URLLC has the lowest latency. Different data traffic devices have different latency needs. The latency will, however, rise in tandem with the number of customers connecting to the slice. As depicted in Figure 13, mMTC has the largest delay, followed by eMBB.

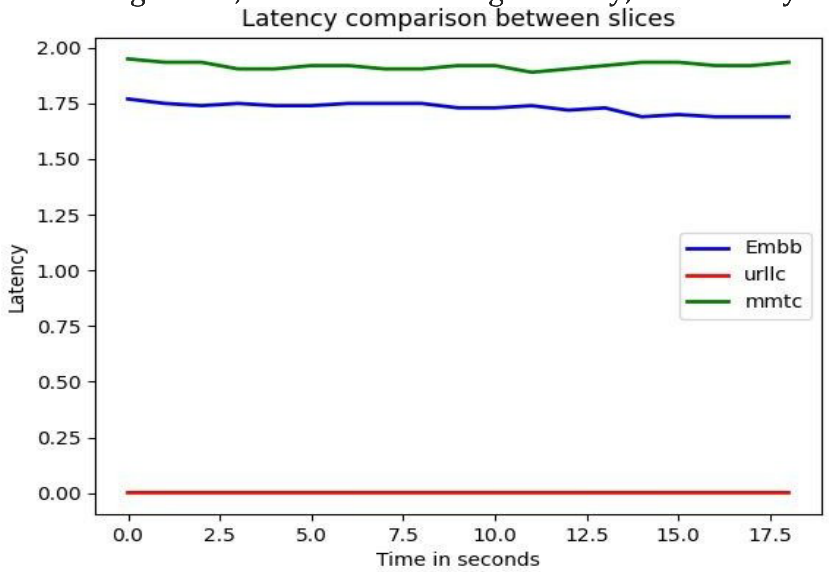


Figure 13. Latency comparison between slices

2. **Usage in different slices**

   The consumption in various slices, or the number of clients connected to that slice at any one time is explored in Figure 14. There is a cost to latency. The utilization rate will rise in response to increased latency. The utilizations, however, reduces in accordance with low latency, making lower usage achievable. eMBB utilizes most, as can be observed, whereas mMTC has the lowest.

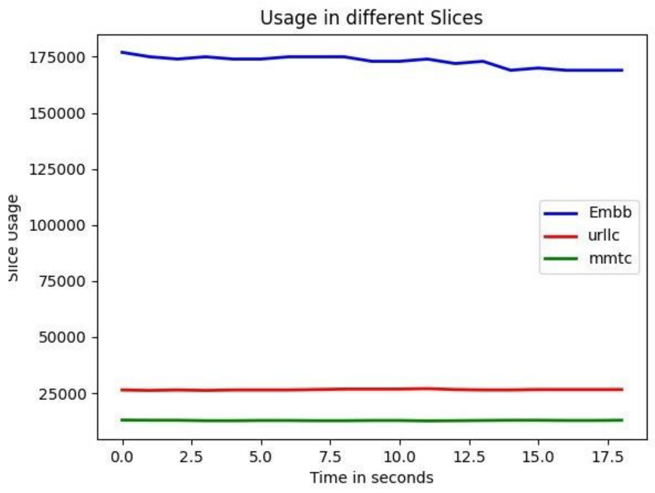


Figure 14. Slice Usage with respect to time

6. CONCLUSION

In this work, the different parameters analysed are client connection ratio, number of clients connected per slice, and overall client count with respect to time. This analysis is carried out with the Embb Client, the uRLLC client, and the mMTC client. With respect to the Latency comparison, the clients of uRLLC are exposed to the lowest latency compared to the clients of mMTC and Embb. As per the slice usage analyses, minimum slice usage is noticed with mMTC and uRLLC clients, and maximum usage is depicted with Embb clients. The outcomes have demonstrated that even in the case of abrupt changes in the system's user base, consistent latency may be offered. The results have shown that consistent latency may be provided even when the system's user base experiences sudden changes. The slicing module can be enhanced further to produce even more beneficial insights. The live simulation's visual feed will aid in the immediate detection of any issues. It is possible to accelerate and optimise the module by making software performance changes. More test cases generated for the same input configurations combined with machine learning modules may produce results that are closer to those of the actual world. In the future, dynamic slicing can be used to better manage the load on the base stations. It is possible to improve the slicing module even more to generate even more useful insights. The visual feed of the live-running simulation will help in the quick identification of any problems. By altering the software's performance, the module can be speeded up and optimised. Machine learning modules in combination with more test cases generated for the same input configurations may result in outcomes that are more representative of the real world. The load on the base stations can be better managed in the future with the help of dynamic slicing.

REFERENCES

[1] C. Yang, W. M. Shen, and X. B. Wang, **The internet of things in manufacturing: Key issues and potential applications**, IEEE Syst. Man Cybern. Mag., vol. 4, no. 1, pp. 6–15, 2018.

[2] E. C. Strinati, S. Barbarossa, J. L. Gonzalez-Jimenez, D. Ktenas, N. Cassiau, L. Maret, and C. Dehos, 6G: The next frontier: **From holographic messaging to artificial intelligence using sub-terahertz and visible light communication**, IEEE Vehicular Technol. Mag., vol. 14, no. 3, pp. 42–50, 2019.

[3] R. H. Wen, G. Feng, J. H. Tang, T. Q. S. Quek, G. Wang, W. Tan, and S. Qin, **On robustness of network slicing for next-generation mobile networks,** IEEE Trans. Commun., vol. 67, no. 1, pp. 430–444, 2019.

[4] J.Mei, X. B. Wang, and K. Zheng, Intelligent network slicing for V2X services toward 5G, IEEE Netw., vol. 33, no. 6, pp. 196–204, 2019.

[5] T. Taleb, M. Corici, C. Parada, A. Jamakovic, S. Ruffino, G. Karagiannis; and T. Magedanz, EASE: EPC as a service to ease mobile core network deployment over cloud, IEEE Netw, vol. 29, no. 2, pp. 78–88, 2015.

[6] M. Bagaa, T. Taleb, A. Laghrissi, A. Ksentini, and H. Flinck, **Coalitional game for the creation of efficient virtual core network slices in 5G mobile systems**, IEEE J. Sel. Areas Commun., vol. 36, no. 3, pp. 469–484, 2018.

[7] P. Caballero, A. Banchs, G. de Veciana, X. Costa-P´erez, and A. Azcorra, **Network slicing for guaranteed rate services: Admission control and resource allocation games, IEEE Trans. Wirel. Commun., vol. 17, no. 10, pp. 6419–6432, 2018.**

[8] Y. L. Lee, J. Loo, T. C. Chuah, and L. C. Wang, **Dynamic network slicing for multitenant heterogeneous cloud radio access networks**, IEEE Trans. Wirel. Commun., vol. 17, no 4, pp. 2146–2161, 2018.

[9] Y. N. Liu, X. B. Wang, G. Boudreau, A. B. Sediq, and H. Abou-Zeid, **Deep learning based hotspot prediction and beam management for adaptive virtual small cell in 5G Networks**, IEEE Trans. Emerg. Topics Comput. Intell., vol. 4, no. 1, pp. 83–94, 2020.

[10] Y. L. Lee, J. Loo, T. C. Chuah, and L. C. Wang, Dynamic network slicing for multitenant heterogeneous cloud radio access networks, IEEE Trans. Wirel. Commun., vol. 17, no. 4, pp. 2146–2161, 2018.

[11] J. L. Li, W. S. Shi, P. Yang, Q. Ye, X. S. Shen, X. Li, and J. Rao, **A hierarchical soft RAN slicing framework for differentiated service provisioning**, IEEE Wirel. Commun., doi: 10.1109/MWC.001.2000010.

[12] M. Zambianco and G. Verticale, **Interference minimization in 5G physical-layer network slicing**, IEEE Trans. Commun., vol. 68, no. 7, pp. 4554–4564, 2020.

[13] Jie Mei, Xianbin Wang, and khan zheng, **An intelligent self-sustained RAN slicing framework for diverse service provisioning in 5G-beyond and 6G networks, Intelligent and Converged Networks**, 2020, 1(3): 281–294, ISSN 2708-6240, DOI: 10.23919/ICN.2020.0019

[14] N. Zhang, S. Zhang, P. Yang, O. Alhussein, W. Zhuang, and X. Shen, "**Software defined space-air-ground integrated vehicular networks: Challenges and solutions**," IEEE Commun. Mag., vol. 55, no. 7, pp. 101–109, 2017.

[15] X. Shen, J. Gao, W. Wu, M. Li, C. Zhou, and W. Zhuang, "**Holistic network virtualization and pervasive network intelligence for 6G**," submitted to IEEE Commun. Surveys Tuts., 2021.

[16] R. Minerva, G. M. Lee, and N. Crespi, "**Digital twin in the IoT context:A survey on technical features, scenarios, and architectural models**," Proc. IEEE, vol. 108, no. 10, pp. 1785–1824, Oct. 2020.

[17] X. Shen, J. Gao, W. Wu, K. Lyu, M. Li, W. Zhuang, X. Li, and J. Rao, "**AI-assisted network-slicing based next-generation wireless networks**," IEEE Open J. Veh. Technol., vol. 1, no. 1, pp. 45–66, 2020.

[18] W. Zhuang, Q. Ye, F. Lyu, N. Cheng, and J. Ren, "**SDN/NFV-empowered future IoV with enhanced communication, computing, and caching**," Proc. IEEE, vol. 108, no. 2, pp. 274–291, 2020.

[19] X. You et al., "**Towards 6G wireless communication networks: Vision, enabling technologies, and new paradigm shifts**," Sci. China Inf. Sci., vol. 64, no. 1, pp. 1–74, 2021.

[20] A. Kaloxylos, "**A survey and an analysis of network slicing in 5G networks**," IEEE Communications Standards Magazine, vol. 2, no. 1, pp. 60–65, 2018.

[21] R. A. Addad, T. Taleb, M. Bagaa, D. L. C. Dutra, and H. Flinck, "**Towards modeling cross-domain network slices for 5G**," in 2018 IEEE global communications conference (GLOBECOM).

[22] Afolabi, T. Taleb, K. Samdanis, A. Ksentini, and H. Flinck, "**Network slicing and softwarization: a survey on principles, enabling technologies, and solutions**," IEEE Communications Surveys & Tutorials, vol. 20, no. 3, pp. 2429–2453, 2018.

[23] A. Khare, R. Gupta, and P. K. Shukla, "**Improving the protectioof wireless sensor network using a black hole optimization algorithm (BHOA) on best feasible node capture attack**," in IoT and Analytics for Sensor Networks, P. Nayak, S. Pal, and S. L. Peng, Eds., vol. 244 of Lecture Notes in Networks and Systems, Springer, Singapore, 2022.

[24] C. Ssengonzi, O. P. Kogeda, and T. O. Olwal, "**A survey of deep reinforcement learning application in 5G and beyond network slicing and virtualization**," Array, vol. 14, p. 100142, 2022.

[25] W. Wu, N. Chen, C. Zhou, M. Li, X. Shen, W. Zhuang, and X. Li, "**Dynamic RAN slicing for service-oriented vehicular networks via constrained learning**," IEEE J. Sel. Areas Commun., vol. 39, no. 7, pp. 2076–2089, 2021.

[26] C. Zhou, W. Wu, H. He, P. Yang, F. Lyu, N. Cheng, and X. Shen, “**Deep reinforcement learning for delay-oriented IoT task scheduling in spaceair-ground integrated network**,” IEEE Trans. Wireless Commun., vol. 20, no. 2, pp. 911–925, 2021.

[27] Adiraju, P. R., & Voore Subba Rao. (2022). **Dynamically Energy-Efficient Resource Allocation in 5G CRAN Using Intelligence Algorithm.** *EMITTER International Journal of Engineering Technology*, *10*(1),217-230. https://doi.org/10.24003/emitter.v10i1.661

[28] Saeed, A. B., & Gitaffa, S. A.-H. (2019). **FPGA Based Design of Artificial Neural Processor Used for Wireless Sensor Network.** *EMITTER International Journal of Engineering Technology*, *7*(1), 200-222. https://doi.org/10.24003/emitter.v7i1.346