Resource-aware Service Prioritization in a Slice-supportive 5G Core Control Plane for Improved Resilience and Sustenance

Supriya Kumari, Shwetha Vittal, Antony Franklin A
Department of Computer Science and Engineering, Indian Institute of Technology Hyderabad
Email: {cs21mtech12002, cs19resch01001, antony.franklin}@iith.ac.in

***Abstract*—While the 5G standards have been evolving continuously, researchers aim for the resilience and sustainability of 5G systems for building cognizant and autonomous Beyond 5G (B5G) and 6G networks. This is quite challenging in the Service Based Architecture of distributed 5G Core (5GC) as multiple Network Functions (NFs) are involved to help serve the various User Service Requests (USRs) arriving in the control plane. In this regard, the continuous monitoring of the availability and performance of individual NFs in a Closed Loop Automation (CLA) is a need of hour to keep up the robust and resilient functioning of the distributed 5GC overall. Any unforeseen situations like the sudden failure, overload, or congestion of the NFs of the 5GC can drop even the critical USRs unnecessarily. This paper proposes the proactive monitoring of the NFs of the 5GC in the control plane and utilizes it to intelligently schedule and serve the frequently arriving USRs and prioritize the mission-critical slice service requests. Specifically, the Ford-Fulkerson algorithm popularly known as the Max-Flow problem solver is leveraged to proactively assess the NFs' performance and availability and use it effectively to serve critical service requests arriving during unexpected situations of failure and overloads. Our experiments based on the 3GPP-compliant 5G testbed show that, with the proposed solution, the native 5GC can serve 20% more predominant USRs, and the slice-supportive 5GC can serve 33% more mMTC slice USRs, and 47% more uRLLC slice USRs while handling the worst case of peak traffic for different services.**

I. Introduction

Current 5G systems are capable of providing a wide range of services to their end users by adopting network slicing on top of Network Function Virtualization (NFV) and Software Defined Networking (SDN) technologies. With control and user plane separation, the 5G Core (5GC) is defined through a Service Based Architecture by 3GPP [1]. Specifically, the control plane here is distributed with various Network Functions (NFs) being virtualized and made to run on either compute-intensive Virtual Network Functions (VNFs) or container-based Cloud-native Network Functions (CNFs) being hosted on remote or edge clouds. Here, different NFs, mainly, the Access and mobility Management Function (AMF), Authentication and Security Function (AuSF), Unified Data Management (UDM), Network Repository Function (NRF), and Session Management Function (SMF) interact on the Service Based Interface (SBI) (see Fig. 1a) to serve a variety of User Service Requests (USRs) arriving in the control plane.

But, slicing is mainly observed on the data (or the user) plane of 5G to provide a variety of services to its users like enhanced Mobile Broad Band (eMBB), ultra Reliable Low-Latency Communications (uRLLC), massive Internet of Things (mIoT), Vehicle to Everything (V2X), and High-performance Machine Type Communications (HMTC) (see Fig. 1b). So, in this regard, the convergence of virtualization and slicing on 5G has been very effective in saving costs for Mobile Network Operators (MNOs) to provide a variety of services on a common and shared physical infrastructure. However, it poses a set of challenges in providing a robust and resilient service.

In this context, this paper proposes a proactive resource utilization mechanism to schedule and serve the prominent and critical USRs arriving during unexpected and unforeseen situations of the 5GC control plane. Specifically, we leverage the Ford-Fulkerson Algorithm (popularly known as Max-Flow problem solver) [2] to realize the current situation instantly, by proactively peeking into the total available resources of the distributed 5GC control plane. We then use its output to effectively schedule the most needed and critical service requests arriving at the 5GC control plane, to ensure resilient and sustainable service to the best of its (5GC control plane) ability. Overall, using the proposed solution, a shared and slice-supportive control plane of 5GC, not only be effective in serving the prominent and frequently arriving USRs but it can prioritize the mission-critical service requests, which may require immediate data service on the corresponding slice. To summarize, the key contributions of this paper are

- Identifying the need for proactive and continuous assessment of the distributed 5GC control plane's total availability and capacity.
- System model with Ford-Fulkerson (Max-Flow) [2] algorithm to facilitate resilience and sustenance in terms of dynamic scheduling and serving the predominant and critical slice service USRs.
- Evaluation of the proposed solution on the 3GPP-compliant 5GC framework to illustrate the improved resilience and sustenance in serving the mission-critical and predominant USRs.

The rest of the paper is organized as follows. Starting with motivation and related work in Section II, we propose the system model with the Ford-Fulkerson algorithm in Section III to assess the sudden unforeseen situations and the current total

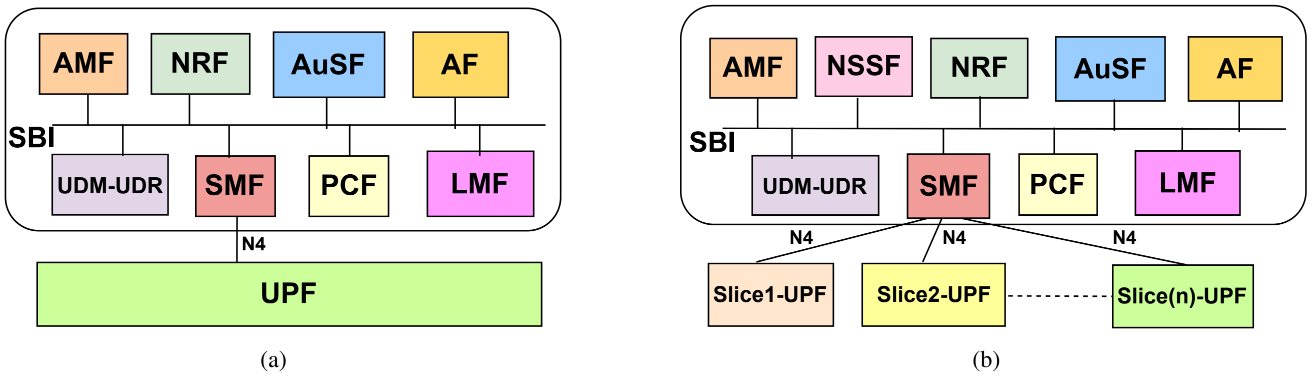


Fig. 1: View of SBA-based control plane with native 5GC in a) and slice-supportive 5GC in b).

capacity of 5GC and thereafter leverage its output efficiently to schedule and serve the critical USRs arriving during such situations. Further, in Section IV, we detail the implemented experimental setup of the proposed work in a 5G system followed by details on data collection and analysis techniques used to evaluate the proposed algorithm's effectiveness in efficient resource utilization for serving the needed USRs, in both native and slice-supportive 5GC. Finally, in Section V we conclude our paper with the future work. Note that, in the rest of the paper, the term 5GC refers to the control plane of 5GC, unless mentioned specifically.

II. Motivation and Related Work

The advent of a variety of devices connecting to 5G has increased the network capacity and service demands, necessitating efficient resource allocation strategies for 5G system deployments. However, managing network resources and meeting the diverse service requirements of the 5GC present complex challenges. Our work is inspired by two such important challenges in the 5GC, which we notice when a slice-supportive 5GC aims to provide a variety of services to a variety of users.

1) First, the rapidly growing control plane traffic (USRs) arriving at the 5GC control plane urges the MNOs to evaluate different proactive mechanisms to make it (the 5GC) cognizant, robust, scalable, yet resilient and sustainable towards various unexpected and unforeseen situations like the sudden failure of NFs, and the sudden arrival of heavy traffic [3].
2) Second, 3GPP [4] defines different procedures for the 5G users to successfully access the data service. Specifically, the 5G users need to first register to the 5GC (via the Radio Access Network (RAN)) and establish a Packet Data Unit (PDU) session on the control plane before accessing the required service on the user plane. As different users and devices keep arriving at the 5GC, the MNO must not only ensure that all the NFs of 5GC are available but also that they should be capable of handling different service requests coming from various users [3, 5].

In this regard, our work aims to provide a resilient and highly available service on the control plane with sustainability even during sudden failures of NF(s) and the sudden arrival of heavy traffic at the 5GC.

The 5G systems have been on continuous research to improve their scalability, resilience, and high availability of service to their end users. In this regard, several works have leveraged Artificial Intelligence (AI)-based proactive methods [6, 7, 8] . Authors in [6], employ Recurrent Neural Networks (RNN) to perform load forecasting and use its output for proactive scaling of the 5GC at AMF. It ensures efficient resource allocation, improved latency, and optimal utilization of the AMF functions based on the forecasted traffic load. In a similar context, authors in [7], propose a deep learning-based resource usage forecasting approach that provides useful insights for decision-making in a CNF scaling environment. Recently, authors in [9] have also demonstrated the per slice-based AMF instance deployment to deal with resilience issues on the control plane. All these techniques are very effective when the forecasted traffic or the load for the next time cycle is quite accurate, but the sudden arrival of heavy USRs and failure of NFs is still a challenge, where the NFs may be incapable of handling emergencies [10] like natural disaster management or mission-critical USRs.

In close relation to our work, authors in [5], have similar intentions of catering to reliability and high availability of the service during unexpected situations on the 5GC control plane. Here [5], the authors propose a greedy approach of scheduling the frequent USRs on the NF instances that are least loaded and running all the rarely arriving USRs on NF instances that are loaded above the average system load. This scheduling approach is very effective per NF-basis. However, the work does not consider the inter-NF dependencies in the distributed 5GC. In this context, work in [11] demonstrates the inter-NF dependencies that could potentially become the bottleneck for resilient service for mission-critical applications. To overcome this problem, the same authors have proposed a Per Procedure 5G system (PP5GS) [12] where Per-Procedure NFs are developed in 5GC, such as an NF for serving UE registration, another NF for serving PDU session establishment, and so on. This is quite good for private 5G deployments. However, a

3GPP defined 5GC [1] is completely distributed by nature with multiple NFs involved each having their role to play. Hence, in this regard, our work being 3GPP-compliant is not only unique in handling the unexpected situations of the sudden arrival of emergency and heavy USRs and failure(s) of NFs but also is an add-on solution to improve resilience on the existing works [5, 12].

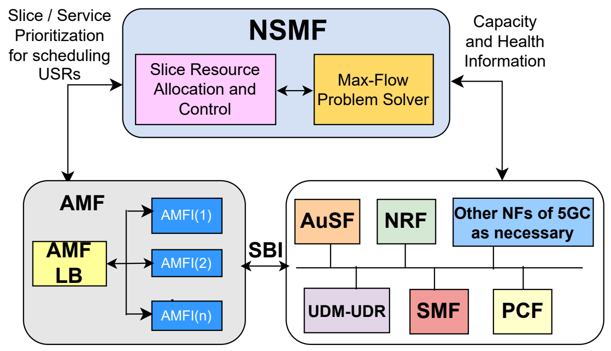

Slice / Service Prioritization for scheduling USRs
NSMF
Slice Resource Allocation and Control
Max-Flow Problem Solver
Capacity and Health Information
AMF
AMF LB
AMFI(1)
AMFI(2)
AMFI(n)
SBI
AuSF
NRF
Other NFs of 5GC as necessary
UDM-UDR
SMF
PCF

Fig. 2: Proposed CLA framework with 5GC SBA and Max-Flow problem solver inside NSMF.

III. Proposed System Model

We aim to build a highly available and resilient 5GC that can continue to serve the USRs to the best of its ability even during sudden unforeseen situations. In this regard, as the SBA of 5GC is distributed by nature consisting of various NFs, it is important to monitor their health and capacity continuously so that any adjustments in resource allocation and reconfiguration can be done proactively. Hence, it requires a Closed Loop Automation (CLA)-based framework. Therefore, in the proposed work, we design two fundamental building blocks working in CLA with three important phases. These building blocks are

1) 5GC with a set of NFs that serve different arriving USRs (see Section I and Section II for description).
2) Network Slice Management Function (NSMF) [13]: NSMF is a slice management entity responsible for configuring, deploying, monitoring, and re-configuring the slices as per the orchestration needs in collaboration with the Management and Orchestration (MANO) entity. In this work, we leverage it to monitor, reconfigure the 5GC, and aid in resource control and reallocation to eventually help in slice service prioritization in terms of serving the USRs.

Fig 2, depicts the working of different blocks mentioned above. We now elaborate on the functionalities of three important phases in which these building blocks work in CLA to aid in building a resilient 5GC overall.

A. Serving of USRs and 5GC Monitoring

As detailed in Section I and Section II, NFs of 5GC serve different types of USRs, that allow the 5G users to establish the PDU session(s) for availing the service from the data plane. Note that, to address the scalability needs, every NF here may consist of a Load Balancer (LB) to schedule the USRs and multiple instances of NF each of which actually serves the USRs [5, 9]. Nevertheless, we consider every NF as a logically centralized entity, as this is sufficient to understand our work. Hence, in Fig. 2, we show the scalable view for AMF only, and other NFs are shown as a centralized entity. However, all these NFs of 5GC (including AMF) report their overall health information to the NSMF (see Fig. 2). The health information here includes their availability and capacity in terms of total serving instances.

B. Resource Allocation and Control of 5GC

As shown in Fig. 2, resource allocation and control are the functionality inside NSMF with the communication between the respective module and the Max-Flow problem solver module. We propose to run the Max-flow problem solver module using the Ford-Fulkerson algorithm [2], popularly known as the max-flow problem solver.

1) ***Overview of Ford-Fulkerson Algorithm****:* The Ford-Fulkerson algorithm is a fundamental method in network flow optimization that solves the problem of estimating the maximum flow through a network with specified edge capacities. The primary goal is to maximize the total flow from a source node to a sink node while keeping the edge capacities in mind. In essence, the Ford-Fulkerson algorithm seeks the best way to distribute a flow, such as data packets, through a network of connected nodes and edges while obeying the available edge capacity. It works in repeated steps, augmenting the flow along paths from the source to the sink incrementally. At each step, it finds a path in the network's residual graph, which is a modified version of the original graph that accounts for the previously admitted flow. Depth First Search (DFS) or Breadth First Search (BFS) is commonly used to find this path. The algorithm's ability to optimize resource distribution while adhering to capacity restrictions makes it a useful tool for tackling a wide range of optimization problems like flow optimization in transportation systems and traffic routing. To justify the usage of this algorithm in this work, we briefly explain the motivational factors next.

2) ***Motivation for Ford-Fulkerson (Max-Flow problem solver)****:* The rationale behind choosing the Ford-Fulkerson algorithm over other alternatives, such as Dynamic Programming (DP), for this study, is rooted in several key factors that contribute to its superior performance in terms of complexity, scalability, and suitability for 5GC's resource allocation challenges. The Ford-Fulkerson algorithm's time complexity is $\mathcal{O}(V \cdot E^2)$, where E represents the number of edges in the network and V is the number of nodes. The above implementation of the Ford-Fulkerson Algorithm is called the Edmonds-Karp Algorithm which has the the proof that the algorithm always terminates and yields maximum flow [2]. While this complexity can be relatively high in worst-case scenarios, its scalability can be efficiently managed through heuristics, optimizations, and dynamic adjustments. This characteristic aligns well with the dynamic nature of 5GC where network conditions and users' demands can change rapidly.

The algorithm's adaptability to varying capacities and dynamic adjustments make it well-suited for real-time decision-making in resource allocation. In contrast, the DP approach [2] often involves higher time and space complexities, particularly when solving optimization problems with multiple variables and constraints. The DP-based solutions may become computationally expensive as the problem scales, leading to potential scalability issues in large-scale 5G networks with numerous instances for different NFs and varying capacities. Hence, we choose the Ford-Fulkerson algorithm to address the resilience needs in a distributed 5GC during the unexpected failures and overload of different NFs, while handling a variety of different USRs. Note that, henceforth, the term max-flow in the rest of the sections represents the Ford-Fulkerson algorithm, unless mentioned specifically.

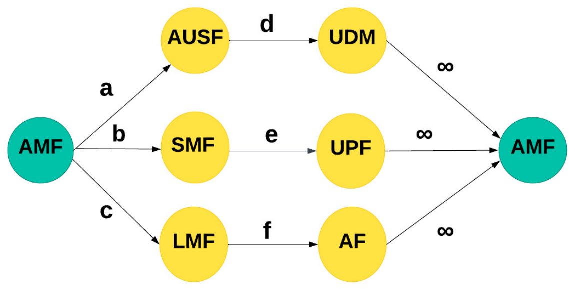


Fig. 3: Ford-Fulkerson (Max-Flow problem solver) representation for control plane functioning of 5GC SBA.

*3) **Max-flow for Resource Allocation and Control in 5GC**:* For the execution of the max-flow problem solver, we first represent the complete 5GC (having required NFs), with a directed graph ($G = (V, E)$) shown in Fig. 3, having a set of vertices and edges between designated source and sink nodes. Here, vertices represent the different NFs involved in handling different USRs, like AMF, AuSF, SMF, UDM, Location Management Function (LMF), Application Function (AF), and so on. An edge between two NFs represents the communication dependency between the respective NFs for processing a USR. For example, UE registration (a USR) is processed by AMF, AuSF, and UDM, and hence, there is an edge between AMF and AuSF and an edge between AuSF and UDM. Similarly, there is an edge between AMF and SMF, and an edge between SMF and UPF for processing the PDU session establishment (another USR). Note that, here, AMF is designated for both the source and sink nodes in the representation (see Fig. 3). This is because, as per 3GPP [1], the AMF is the only point of contact for all control plane communications towards the user (User Equipment (UE)) [1]) and RAN, and hence, both the entry and exit points in the graph. It is to be noted that for the max-flow problem solver, the graph is formed with the required NFs which are responsible for completing the respective control plane functioning of the desired data plane services. For example, in Fig 3, AMF, AuSF, UDM, SMF, UPF, LMF, and AF are shown as these NFs are required to

Algorithm 1: Max-flow problem solver for resource monitoring, allocation, and service prioritization

```
Input : graph, source, sink
1  Global variables: maxFlow, numUsersNotServed
2  interval = 0 (seconds);
3  while true do
4      maxFlow = FORD_FULKERSON(graph, 0, n);
5      if numUsersNotServed > 0 then
6          min = arr[0];
7          ind = 0;
8          for $i \leftarrow 1$ to $n$ do
9              if arr[i] < min then
10                 min = arr[i];
11                 ind = i;
12         arr[ind] = min + 1;
13         update(graph);
14     else
15         if numUsersNotServed == 0 then
16             max = arr[0];
17             ind = 0;
18             for $i \leftarrow 1$ to $n$ do
19                 if max < arr[i] then
20                     max = arr[i];
21                     ind = i;
22             arr[ind] = max - 1;
23             update(graph);
24     sleep interval;
```

complete the control plane processing for UE (the user) [14] to avail the location services on the control plane and further in the data plane too.

The working of the max-flow problem solver for resource monitoring, allocation, and control is shown in Algorithm 1. Here, the solver is invoked at regular intervals with a required/desired periodicity, which is set by the variable *interval*. It takes into consideration the scale-in and scale-out time requirements. The solver updates the capacity of the graph and stores the value in the variable *maxFlow*. Its purpose is to determine the maximum flow across the 5GC (as per the formulated graph, see Fig 3), based on the capacities of different NFs involved in processing the respective USRs and the prevailing demand. It also keeps track of the total number of USRs drooped in the variable *numUsersNotServed*. If the number of dropped USRs is greater than 0, it checks the bottleneck node/NF and triggers scale-out (see lines [5-13]). Similarly, if no USRs are being dropped, the solver finds the node/NF that has excess resources allotted and triggers scale-in for that (see lines [15-23]). The final available capacity and the node that needs to be scaled in or out is then output to the resource allocation and control module, which triggers the scale-out or scale-in.

C. Service Prioritization with Resource Awareness

Service prioritization with resource awareness is the final phase of CLA (see Fig. 2) in the proposed work. In this phase, first, the output from the max-flow problem solver is given to the resource allocation and control module inside NSMF. Using the output, this module computes the maximum number of USRs that could be served and notifies the AMF LB, so that the AMF LB can make decisions as to which USRs shall be prioritized (AMF is the entry point for all the uplink control plane communication to serve different USRs).

1) **Service Prioritization for Frequently arriving USRs:** From works in [5] and [15], we observe that significant portion of USRs is made up of 'N2 release' and 'Service request' or the 'PDU session establishment' procedures arriving at the native 5GC (see Fig. 1a) versus the user registration and handover procedures [4]. Hence, we too consider these significant portions of USRs, as frequent USRs and the rest as rare USRs. So, in situations of sudden failure of one or more NFs' serving instances or overloaded situations, i.e., when enough NF instances are not available or not capable of serving all the arriving USRs, using the output given by the resource allocation and control module (see Section III-B), the AMF LB prioritizes the frequent USRs against the rare ones, thereby, at least the best number of frequent USRs can be served successfully from the available serving instances of corresponding NF(s) in the maximum flow path.
2) **Slice Service Prioritization:** As shown in Fig. 1b, an MNO deploys a slice-supportive 5GC when there are one or more slices (like eMBB, uRLLC, mIoT, and so on) available to serve in the data plane. But, the control plane here is common and shared (see Fig. 1b). So, based on the deployment need, the 5GC here shall be capable of serving different USRs arriving in the control plane asking for different slice services. However, depending on the slice type and the time of the day, the amount of these USRs may vary in number. For example, mIoT-related USRs typically peak during working hours of the day [16, 17], eMBB USRs peak during night hours [17, 18], and uRLLC USRs can arrive on an emergency basis [17, 19] (like e-health, remote surgery, and ambulance services). So, using the output from the resource allocation and control module (see Section III-B), the AMF LB prioritizes the USRs of slice-type which requires attention or has peak traffic against the other slice types, thereby, at least the emergency slice service requests can be served successfully to the best possible level from the available serving instances of corresponding NF(s) in the maximum flow path.

Overall, if the USRs for the prioritized service or the slice are dropped, the proposed scheme (see Algorithm 1) identifies the bottleneck NF. In response, a new serving instance of NF is dynamically introduced (scale-out) to cater to the increased demand. Conversely, when all USRs are effectively processed without any drops and all service demands are satisfactorily met, instances are dynamically removed (scale-in) from NFs that have an excess number of these instances. Hence, this dynamic scaling process ensures optimal resource utilization while operating smoothly even in the presence of failures, thus maintaining a responsive and resilient environment.

IV. Performance Evaluation

A. Experimental Setup

To evaluate network resilience, first, we set up the testbed environment as per the framework shown in Fig. 2. Here, we capture the ground truth capacity of serving the USRs from the in-house built 3GPP-compliant 5GC testbed that is leveraged in our previous works [5, 9]. This testbed is built as a C++ library with HTTP/2 leveraged for service-based interactions among the control plane NFs of the 5GC. Several of these NFs are developed, including the AMF, AuSF, UDM, NRF, SMF, NSSF, and LMF on the control plane as necessary. The RAN+UE emulator is also developed as part of this testbed. It generates the USRs in the form of Non Access Stratum (NAS) messages [20] towards the AMF (see Fig. 2). Note that the radio side time duration is not considered here for end-to-end time measurement, as the RAN+UE emulator does not have the radio stack.

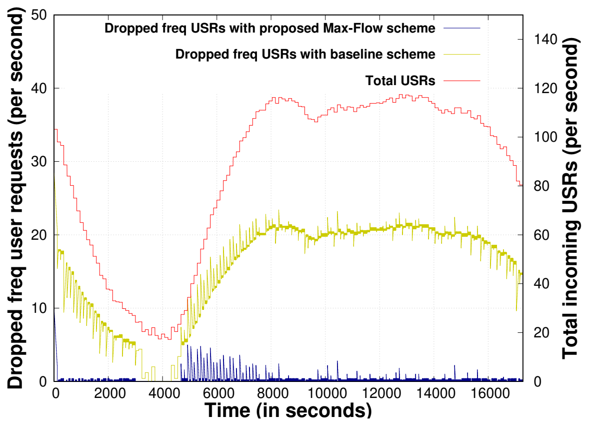
TABLE I: Test setup parameters for performance evaluation.

Parameter	**Value**
Number of USRs	Real dataset from [17] with a maximum of 117 USRs/second. See Section IV-B) for a detailed description of the dataset.
Processing capacity per NF instance	25 USRs/second (see description in Section IV-A).
Variation in the capacity of processing the USRs due to failure of different NFs' instances.	Uniform distribution with random numbers between 80 to 100 USRs/second.
Periodicity of monitoring the NFs and max-flow problem solver invocation	5 seconds

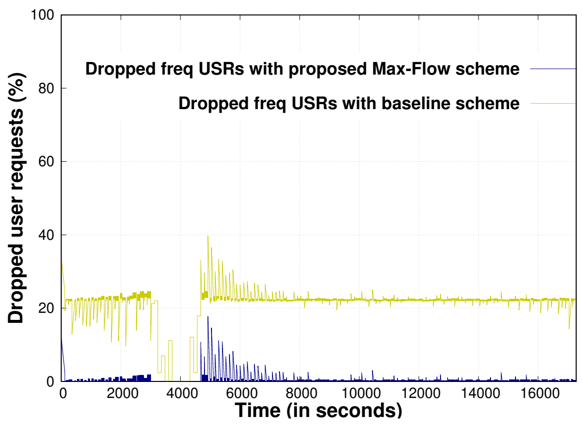
For USRs, we mainly considered UE registration and PDU session establishment procedures, as this is sufficient to evaluate our proposed solution. During the testing, the UE registration consumed more time than the PDU session establishment. i.e., the time taken to complete a single UE registration procedure is '40 ms' and the PDU procedure is '35 ms'. Therefore, using the '40 ms' value, we considered the ground truth capacity of processing 25 USRs per second for the evaluation of the proposed work. Accordingly, we set the initial number of serving instances for every NF to 4 resulting in a total capacity of 100 requests per second (25 req/sec per instance). Other parameters considered for testing are tabulated in Table I.

B. Data Set for User Service Requests

To evaluate the end-to-end resilience benefit from our proposed work, we used the real dataset [17] captured from Milan city, the Big Data Challenge organized by Telecom Italia. The dataset has various Call Detail Records (CDRs) collected in intervals of 10 minutes for two months. So, similar to the



(a)



(b)

Fig. 4: Comparison of dropped frequent USRs in a) and percentage of dropped frequent USRs in b) when some NFs' instances are not available.

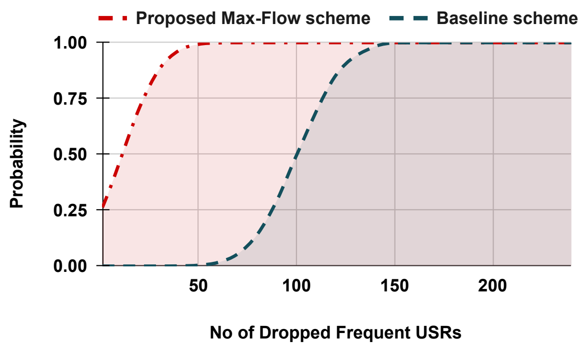


Fig. 5: CDF of the total number of dropped frequent USRs.

works in [6], we consider that at least 10% of the traffic is control plane traffic (USRs) and use it for our testing.

As described in Section III-B, the 5GC in this paper is capable of working both in native mode (i.e., without slice-support) and slice-supportive mode. Hence, for evaluation, we set up the experimental testbed and captured the resilience metrics individually for native mode and slice-supportive mode. Additionally, to establish a baseline for comparison with the proposed CLA-based max-flow scheme, we implemented a scheme without it. Unlike the proposed scheme, in the baseline scheme, the USRs are randomly processed, by neither giving priority to frequent USRs in the native core nor to the specific slice USRs in the slice-supportive core. The rest of the setup parameters are the same as that used for the proposed scheme. Note that, for consistency, we fail the same number of NF instances in both the methods (the proposed max-flow problem solver and the one without it) used for comparison.

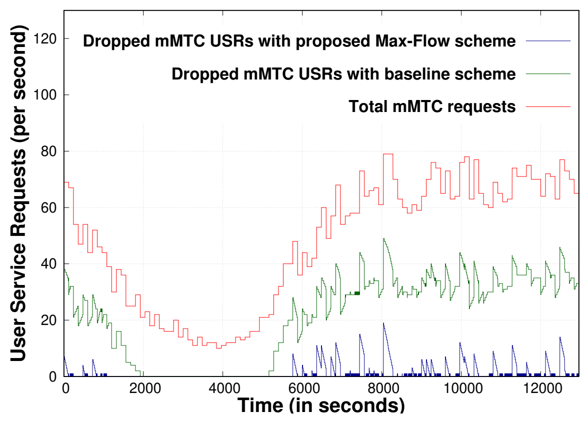
C. Resilience of Native 5GC with Predominant Requests Prioritization:

As per [3] and [21], to measure the resilience of 5GC in this work, we capture the number of impacted USRs during the sudden failures of different NF(s) of 5GC. For testing this scenario of predominant requests (frequent USRs) prioritization, we consider the USRs from the dataset to consist of both frequent USRs and rare USRs, with 82% (like 'N2 release' and 'Service Request') for frequent and the rest for the rare USRs like 'UE Registration' and 'Handover' (see Section III-C for an explanation of this). This exact number of 82% is taken from another real dataset [22] of bike-sharing systems.

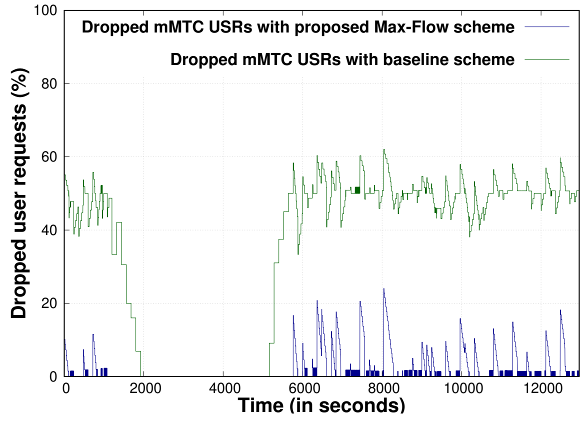
In Fig. 4a, we observe a significant difference in the total number of dropped frequent USRs depending on whether the proposed max-flow problem solver was employed to get to know the current situation of 5GC. When the max-flow problem solver was not used, the total number of such dropped frequent USRs was observed to be higher. Accordingly, Fig. 4b illustrates this difference by presenting the percentage of dropped requests in both schemes. These findings highlight the importance of leveraging the max-flow problem solver to improve the handling of frequent USRs which are predominant in nature requiring more attention and thereby reducing the number of such dropped requests. Fig. 5 shows the Cumulative Distribution Function (CDF) of the total number of dropped frequent USRs in the proposed max-flow-based scheme against the scheme without using it. We can observe that the proposed scheme shows a higher probability of dropping less frequent USRs than the one without it.

D. Resilience of Slice-supportive 5GC with Critical-service Prioritization

In the slice-supportive setup, we first analyze the dataset described in Section IV-B, by taking into account user behaviors and demands during business hours and night hours. During the night hours (12 pm to 6 am), there is an increased allocation for uRLLC and eMBB requests, which are related to real-time collaboration tools, remote desktop applications, video conferencing, and other high-bandwidth applications. During business hours, the allocation for mMTC increases, catering

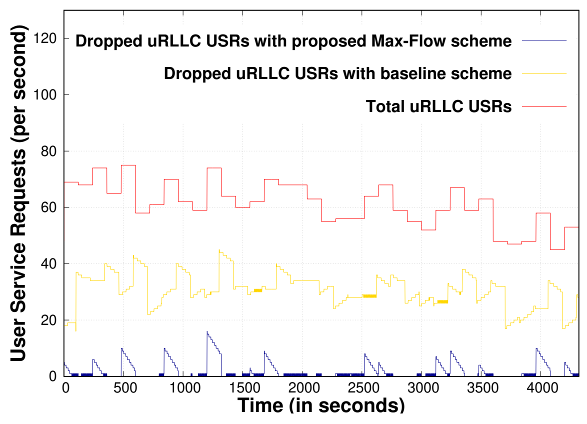


(a)

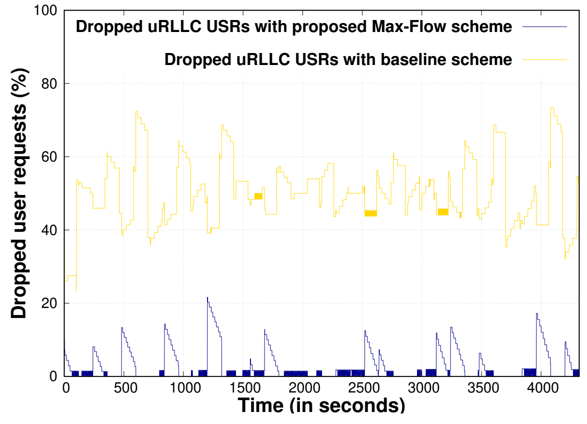


(b)

Fig. 6: Comparison of dropped mMTC USRs in a) and percentage of dropped mMTC USRs in b) when required NFs' all the serving instances are not available while handling peak traffic.



(a)



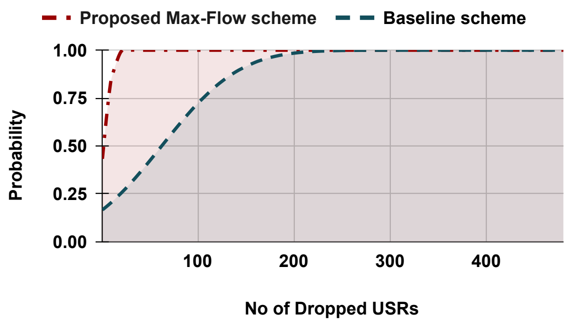
(b)

Fig. 7: Comparison of dropped uRLLC USRs in a) and percentage of dropped uRLLC USRs in b) when required NFs' all the serving instances are not available while handling peak traffic.

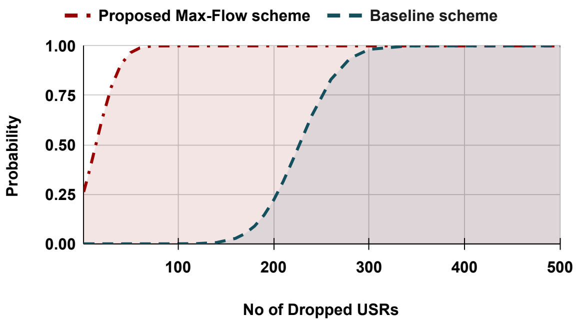
to IoT devices and automated systems performing continuous tasks such as data monitoring and telemetry. So, by taking the dataset analysis observations, the proposed max-flow problem solver prioritizes the mMTC requests over other slice types during working hours of the day. In contrast, the baseline scheme treated all request types equally without considering the time-based patterns. The results, as shown in Fig. 6a and Fig. 6b, indicate that the total number of dropped mMTC requests is higher in the base case compared to the proposed max-flow-based scheme. A similar observation was made for eMBB/uRLLC slices as shown in Fig. 7a and Fig. 7b, which experience sudden peak traffic during night hours. So, by intelligently prioritizing a slice service that experiences sudden higher demand over the others experiencing less demand at any time of the day, MNOs can sustain the situations of sudden unknowns without losing on the high availability of service. Finally, using these experiments, Fig. 8a and Fig. 8b illustrate the CDFs of the total number of dropped USRs with and without the use of the max-flow problem solver for mMTC and eMBB/uRLLC slice types respectively. Note that, in these experiments, we have prioritized one slice only over the other at a time. However, based on the scaling strategies and the availability of serving instances of different NFs involved, multiple slice types that require immediate attention can also be prioritized. Overall, these findings highlight the efficacy of the max-flow problem solver in prioritizing specific slice types requiring immediate attention, thereby reducing dropped requests in accordance with temporal patterns. Hence, by considering these usages patterns and applying the proposed system model, the 5GC can effectively prioritize critical service requests, ensure network resilience, and deliver optimal quality of service to users even during sudden unforeseen situations of failures and overloads.

V. Conclusion and Future Work

Building a resilient 5GC that can sustain various unexpected and unforeseen situations has become a bare necessity as the current 5G is advancing to 6G networks, especially when 5GC



(a)



(b)

Fig. 8: CDFs of the total number of dropped USRs in a) mMTC slice and b) uRLLC slice.

converges with different network slice service demands. This paper has presented a comprehensive exploration of the Ford-Fulkerson algorithm, a max-flow problem solver, for efficient resource utilization and service prioritization in the control plane of 5GC, to tackle the sudden situations of failure of NFs or overloads. By proactive execution of the Max-Flow problem solver and leveraging its capabilities, the proposed solution is able to serve the best possible maximum USRs. Through extensive experiments on the 5G testbed, we have demonstrated the effectiveness of the proposed approach in maximizing the served requests while allowing for prioritization based on specific requirements or network policies on the required slices. We hope that our work provides valuable insights into the 5GC's capacity, aiding MNOs to make informed decisions on resource allocation and ensure resilient service provisioning while optimal utilization of the resources. In the future, we plan to extend this solution to the data plane as well, where a variety of QoS traffic flows arrive on different slices seeking slice service prioritization.

Acknowledgment

This work was supported by the Department Of Science & Technology, India as part of the "Next Generation Wireless Research and Standardization on 5G and Beyond" project.

References

[1] 3GPP, "System Architecture for the 5G System", Tech. Rep. TS 23.501, 2023.

[2] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein, *Introduction to Algorithms*, The MIT Press, 2nd edition, 2001.

[3] Rui Li, Bertrand Decocq, Anne Barros, Yi-Ping Fang, and Zhiguo Zeng, "Estimating 5g network service resilience against short timescale traffic variation", *IEEE Transactions on Network and Service Management*, pp. 1–1, 2023.

[4] 3GPP, "Procedures for the 5G System (Release 18.2)", Tech. Rep. TS 23.502, 2023.

[5] Shwetha Vittal and A. Antony Franklin, "Harness: High availability supportive self reliant network slicing in 5g networks", *IEEE Transactions on Network and Service Management*, vol. 19, no. 3, pp. 1951–1964, 2022.

[6] A. Ksentini P. Bertin C. Viho I. Alawe, Y. Hadjadj-Aoul and D. Darche, "Smart Scaling of the 5G Core Network: An RNN-Based Approach", , no. GLOCOM.2018.8647590.], 2018.

[7] Menuka Perera Jayasuriya Kuranage, Loutfi Nuaymi, Ahmed Bouabdallah, Thomas Ferrandiz, and Philippe Bertin, "Deep learning based resource forecasting for 5g core network scaling in kubernetes environment", in *2022 IEEE 8th International Conference on Network Softwarization (NetSoft)*, 2022, pp. 139–144.

[8] V. Walunj A. Thantharate, R. Paropkari and C. Beard, "A Deep Learning Approach towards Efficient and Reliable Network Slicing in 5G Networks", , no. UEMCON47517.2019.8993066.], 2019.

[9] Bhavishya Sharma, Shwetha Vittal, and Antony A Franklin, "Flex5gcore: Leveraging xdp-sctp for scalable and resilient network slice service in future 5g core", in *Proceedings of the Symposium on SDN Research*, Hong Kong, China, 2023, APNET '23, p. 83–95, Association for Computing Machinery.

[10] EENA, "5G Technology in Emergency Services", 2019.

[11] Endri Goshi, Michael Jarschel, Rastin Pries, Mu He, and Wolfgang Kellerer, "Investigating inter-nf dependencies in cloud-native 5g core networks", in *2021 17th International Conference on Network and Service Management (CNSM)*, 2021, pp. 370–374.

[12] Endri Goshi, Raffael Stahl, Hasanin Harkous, Mu He, Rastin Pries, and Wolfgang Kellerer, "Pp5gs -an efficient procedure-based and stateless architecture for next generation core networks", *IEEE Transactions on Network and Service Management*, pp. 1–1, 2022.

[13] 3GPP, "Study on Management and Orchestration of Network Slicing for Next Generation Network", Tech. Rep. TR 28.801, 2018.

[14] Supriya Dilip Tambe, Shwetha Vittal, Pratik Abhijeet Bendre, Supriya Kumari, and A Antony Franklin, "Demonstration of 5g-mec assisted location services for mission critical applications", in *2022 IEEE 8th International Conference on Network Softwarization (NetSoft)*, 2022, pp. 272–274.

[15] Rinku Shah, Vikas Kumar, Mythili Vutukuru, and Purushottam Kulkarni, "Turboepc: Leveraging dataplane programmability to accelerate the mobile packet core", in *Proceedings of the Symposium on SDN Research*, New York, NY, USA, 2020, SOSR '20, p. 83–95, Association for Computing Machinery.

[16] M. Z. Shafiq, L. Ji, A. X. Liu, J. Pang, and J. Wang, "Large-Scale Measurement and Characterization of Cellular Machine-to-Machine Traffic", *IEEE/ACM Transactions on Networking*, vol. 21, no. 6, pp. 1960–1973, 2013.

[17] Telecom Italia, "Telecommunications - SMS, Call, Internet - MI", 2015.

[18] Z. Luo, C. Wu, Z. Li, and W. Zhou, "Scaling Geo-Distributed Network Function Chains: A Prediction and Learning Framework", *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 8, pp. 1838–1850, 2019.

[19] Tejas Subramanya, Davit Harutyunyan, and Roberto Riggio, "Machine learning-driven service function chain placement and scaling in MEC-enabled 5G networks", *Computer Networks*, vol. 166, pp. 106980, 2020.

[20] 3GPP, "Non-Access-Stratum (NAS) protocol for 5G System (5GS); stage 3", Tech. Rep. TS 24.501, 2023.

[21] ENISA, "Resilience Metrics and Measurements: Technical Report", https://www.enisa.europa.eu/publications/metrics-tech-report.

[22] Hadi Fanaee-T and João Gama, "Event labeling combining ensemble detectors and background knowledge", *Progress in Artificial Intelligence*, vol. 2, pp. 113–127, 06 2014.