Cost of Network Slice Collaboration: Edge Network Slicing for In-Flight Connectivity

Arled Papa*, H. Murat Gürsu*, Leonardo Goratti†, Tinku Rasheed†, Wolfgang Kellerer*
*Chair of Communication Networks, Technical University of Munich
†Zodiac Inflight Innovations, Weßling Germany
*arled.papa@tum.de, *murat.guersu@tum.de, †leonardo.goratti@zii.aero, †tinku.rasheed@zii.aero, *wolfgang.kellerer@tum.de

Abstract—**Network edge environments like in-flight or in-train communications utilize satellite-terrestrial integrated networks. These networks however suffer from limited backhaul and cache resources, leading to sustainability issues due to increasing traffic demands. The problem becomes more challenging for 5G ecosystems, where applications have distinct requirements, rendering the management and orchestration of conventional satellite-terrestrial networks harder. Therefore, software-defined networking and edge network slicing are envisioned to enhance resource management and increase flexibility of resource allocation. However, the complexity of management and orchestration increases in cases where service providers, allocated to a slice, do not share information about their users with the infrastructure providers, due to privacy or other concerns. To incorporate the aspect of slice collaboration, we define network slices with respect to their willingness of sharing user traffic statistics with the infrastructure provider. Taking in-flight entertainment and connectivity services (IFECS) as an interesting 5G use-case, we introduce a system model mimicking the practical deployment of slicing for aircrafts using satellites. We propose a mixed integer non linear program that aims at maximizing the number of slices served. Utilizing our model we evaluate the deployment cost of slices with respect to cache and backhaul resources. Our results show that uncooperative slices have a lower selection probability. Nonetheless, we demonstrate that if the slice cost is paid by slice owners, uncooperative slices increase their chances of being served by 33%. Overall, cooperative slicing can revolutionize the IFECS system as it accommodates 200% more slices compared to uncooperative slicing.**

Index Terms—**SD-RAN, Edge Network Slicing, 5G, Virtualization, IFECS.**

I. Introduction

Ubiquity is foreseen as one of the main challenges of next generation networks, especially since now users demand to be connected everywhere, while still maintaining high data rates and low delays. In that regard, the integration of satellite and terrestrial networks is becoming crucial to accommodate such requirements for extreme network edge environments such as aircrafts or trains [1], [2], [3]. However, given the distinct nature of applications in 5G and limited backhaul capacity offered by satellite links [4], current satellite-terrestrial network infrastructures urge for more dynamic techniques for the control and management of network resources.

An interesting 5G use-case in a network edge environment is the In-Flight Entertainment and Connectivity Services (IFECS), where the market is expected to reach USD 7.65 billion by 2023 [5]. For IFECS the complexity of resource management grows due to limited backhaul capacity (e.g., a satellite link [4] or an Direct Air to the Ground (DA2G) link [6]) and large round trip delay, (e.g., delay of a Geostationary (GEO) satellite link is $\sim$ 200 ms). Additionally, the possibility to store large data volumes in the aircraft is restricted, leading to reduced cache sizes, while the traffic for capacity demanding and heterogeneous applications on board (e.g., video streaming, Voice over-IP (VoIP) calls and web browsing) is increasing.

On the one hand, to cater for high round trip delays, the enhancement of the Radio Access Network (RAN) with storing and computing capabilities to reduce the usage of the limited backhaul link is suggested [7]. On the other hand, solutions to ease the management and orchestration of the limited resource bottleneck in the aircraft are emerging 5G techniques such as Software-Defined Networking (SDN) and edge network slicing. The former is foreseen as a powerful tool to enable programmability in the RAN by centralizing control over Software-Defined RAN (SD-RAN) controllers [8]. The latter benefits from the multiplexing gain of the stochastic user behavior. It enables the support of multiple heterogeneous networks/applications under the same infrastructure. It moreover offers the ability to co-exist and share wireless connectivity, cache and backhaul resources, yet without interfering with each other and thus enabling network slice isolation [9].

Previous works [10], [11], [12] that considered network slicing for both cache and backhaul links, assume that the user file request profile is shared with the slice manager as a part of the slice request. This is an assumption that may not be possible for some slices especially due to privacy concerns. Service providers may not have access or may not be willing to share such data. We refer to these slice types as **uncooperative** slices. Moreover, some slices may not have access to a user profile but may share the anonymized file user statistics over time. We call these slices **semi-cooperative**. Finally, the slices that are willing to share the real profile of users we refer to as **cooperative**. The network slice multiplexing gain cannot be benefited from in case of many uncooperative slices and the number of slices served is minimized if all slices are uncooperative.

Thus, it is of utmost importance to investigate the cost of a slice with respect to the slice type and incorporate that to the network slicing problem. The main contributions of this paper are summarized below:

1. An evaluation of the cost of a network slice with respect to cache and backhaul resources for IFECS is presented, while distinguishing among cooperative, semi-cooperative and uncooperative slices with different slice profiles.
2. A dynamic approach is introduced to investigate the effect of the runtime of slicing algorithms in case an adaptation is required. We pose user profile learning as an exemplary problem that requires adaptation and investigate the effect of learning to the network slice cost.
3. An optimization approach based on Mixed Integer Non Linear Programming (MINLP) is formulated and solved for increasing the number of served slices for IFECS and thus, increasing the profit.

The rest of the paper is structured as follows: We provide an overview on edge network slicing in Section II. Section III introduces the system model and details the problem formulation. The evaluation of the cost of a network slice with respect to backhaul and cache resources as we well as results of the optimization problem are presented in Section IV. Finally, the paper is concluded in Section V, where the main findings of our work are summarized and discussed.

II. Related Work

The problem we consider is mainly investigated under the topic of edge network slicing. In principle edge network slicing consists of Radio Access Network (RAN) slicing, cache or computing slicing and backhaul slicing. Previous works tackle RAN slicing, where an efficient allocation of the radio resources for the network slices is achieved while maintaining isolation and achieving QoS [13], [14]. However, none of the aforementioned works considers content storing or backhaul allocation for the network slicing problem. In our scenario, RAN slicing can be tackled by applying more sophisticated wireless technologies inside the aircraft such as optical fiber or LiFi, that have high re-usability in short range. Although the efficient use of the scarce wireless resources is one of the most important challenges in a wireless scenario, studies in the literature demonstrate that effective caching techniques in the wireless access points can enhance the network performance and reduce the backhaul usage [7], [15]. Recent papers [10], [11], [12], [16], [17] consider also cache storing and task offloading for the network slicing problem. While the proposed schemes are relevant and provide interesting insights on the joint allocation problem, they do not consider a constrained backhaul capacity such as a satellite link and they all assume perfect knowledge about the user profile within a slice.

Content popularity prediction is proven challenging especially because the user profile changes dynamically according to time and it is not known beforehand [18]. Differently from previous works, we assume that the user profile is not always known and an interaction with network slices is required. The interaction concerns building of such a profile dynamically and investigates the impact of time required until correct estimation of the user profile in terms of resource utilization.

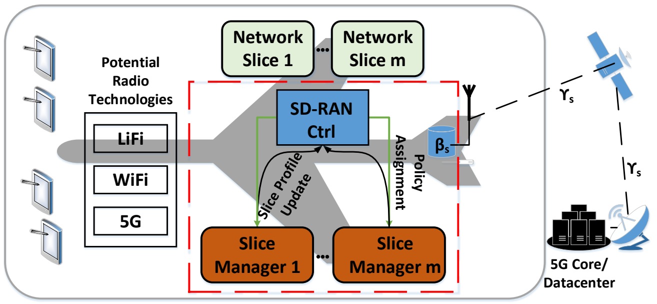


Figure 1: SD-RAN platform architecture. An SD-RAN controller is envisioned for the management of the cache and backhaul resource distribution. Each network slice is identified as a manager and exchanges statistics with the SD-RAN controller for the resource allocation.

Motivated by the user behavior, we investigate the definition of the cost of a network slice with respect to the amount of cache and backhaul resources required to achieve the goal. A similar approach is taken in [19] for identifying the cost of caching among the network and service providers. Alternatively, we also consider the cost of backhaul resources in our model since it is the most critical bottleneck for aircraft communications.

III. System Model

We propose a system inside an aircraft where the components of the IFECS system, wireless connectivity, cache and backhaul link are called the infrastructure, managed by the infrastructure provider through an SD-RAN controller as shown in Fig. 1. The aircraft can utilize a radio technology among WiFi, LiFi or 5G for the wireless connectivity. In the latter the 5G RAN is deployed in the aircraft or satellites [1], whereas the 5G core is deployed on the ground.

Moreover, the infrastructure contains an internal cache of size B files and a backhaul link of capacity Γ Mbps (i.e., a satellite link). The infrastructure is leased to a set of network slices $\mathcal{S}$ of size m slices, that are assigned cache and backhaul resources. These resources are denoted by γ_s and β_s, respectively and specify percentages among total capacity. The entity, SD-RAN controller, is in charge of deploying and assigning these resources to slices. For each accepted network slice, the SD-RAN controller creates a slice manager. The latter manages the users within a slice. It is responsible for accepting the users in the system, exchanging information with the SD-RAN controller about the users and distribute the received resources within the slice's domain. The information regarding the users and their file preference profiles is not always shared between the slice managers and the SD-RAN controller. However, in some cases (i.e., semi-cooperative, cooperative slices) anonymous content statistics are shared to improve the network performance.

Each network slice contains a set of users $\mathcal{U}_s$ that can request files among the file catalog $\mathcal{K}$. The probability of a user $u \in \mathcal{U}_s$ to request file f is expressed with $p_{u,f}$ and follows a

Zipf distribution as assumed in [10]. For each file $f \in \mathcal{K}$ the $p_{u,f}$ given $|\mathcal{K}|$ files, is calculated according to

$$p_{u,f;z,|\mathcal{K}|} = \frac{1/(f^z)}{\sum_{j=1}^{|\mathcal{K}|} 1/j^z}. \tag{1}$$

Each user of each slice has a λ_u mean number of file requests per second obeying Poisson distribution.

This profile is then utilized by the slice managers and the SD-RAN controller to help assigning the cache and backhaul resources, respectively. To assign resources according to the request, the usage of cache and backhaul resources need to be considered. This can be achieved via modeling cache hit and miss rate. In the following section we introduce these models.

A. Caching Model

We adopt the caching technique introduced in [10], where for each file in the slice's catalog the hit probability is defined as follows:

$$h_{s,f} = \frac{\sum_{u=1}^{|\mathcal{U}_s|} \lambda_{u,f}}{\sum_{u=1}^{|\mathcal{U}_s|} \lambda_u} \cdot \beta_s \cdot B \quad \forall\ s \in \mathcal{S}, \tag{2}$$

where $\lambda_{u,f} = p_{u,f} \cdot \lambda_u$, denotes the request rate of each file f for each user u in slice s.

The rate of files that are found in the cache is referred to as hit rate. For each user u, the hit rate is denoted by λ_u^{hit} and expressed as $\lambda_u^{\text{hit}} = \sum_{f=1}^{|\mathcal{K}|} \lambda_{u,f} \cdot h_{s,f}$.

Finally, the caching miss rate for each user is given by

$$\begin{aligned}\lambda_u^{\text{miss}} &= \lambda_u - \lambda_u^{\text{hit}} \\ &= \lambda_u \{1 - \sum_{f=1}^{|\mathcal{K}|} p_{u,f} \cdot \frac{\sum_{u=1}^{|\mathcal{U}_s|} \lambda_{u,f}}{\sum_{u=1}^{|\mathcal{U}_s|} \lambda_u} \cdot \beta_s \cdot B\}.\end{aligned} \tag{3}$$

B. Backhaul Model

In our system, once the file results in a cache hit, then it is downloaded with no delay to the user in the aircraft. In contrast, if the file is missed then the file should be downloaded from the data center on the ground including the round trip delay from the satellite link. In that case, the backhaul link should be used. We model the delay of the backhaul link assuming an M/M/1 queuing system and the backhaul delay is given as:

$$\text{D}[\gamma_s, \beta_s] = \begin{cases} \frac{1}{\mu_s(\gamma_s) - \lambda_s^{\text{miss}}(\beta_s)} + t_{\text{rtt}}, & 0 \leq \lambda_s^{\text{miss}}(\beta_s) < \mu_s(\gamma_s) \\ \infty, & \text{otherwise} \end{cases} \tag{4}$$

where $\mu_s(\gamma_s) = \gamma_s \cdot \frac{\Gamma}{L}$, is the backhaul rate in terms of files/s given L is the size of each file in Mbit and Γ the capacity in Mbps. Finally t_{rtt} is added as the satellite round trip time whereas $\lambda_s^{\text{miss}} = \sum_{u=1}^{|\mathcal{U}_s|} \lambda_u^{\text{miss}}$ the total cache miss rate of the network slice.

C. Learning Model

As already described in the beginning of the section, the SD-RAN controller collects statistics from the slice managers that are willing to share their information with respect to their user profiles. For cooperative slices, we assume the SD-RAN controller possesses instantly all the information about the user profiles from the respective slice managers. Such details can be obtained by the slice managers by utilizing sophisticated estimation techniques before the flight or by requesting user preferences in advance. Alternatively, for semi-cooperative and uncooperative slices, the slice manager estimates the file profile for its users over flight time according to a learning model as detailed in Alg. 1. The initial estimate i.e., $t = 0$ follows a uniform distribution. For uncooperative slices, the estimate does not change over time as no information is shared between the uncooperative slice managers and the SD-RAN controller. Alternatively, for each semi-cooperative slice, the slice manager keeps track of the user preferences for each time instance t and builds a file profile $p_{t,f}$ for that time instance. Furthermore, it updates the estimated file profile $p_{s,f}$ with a learning rate α and sends these information to the SD-RAN controller to perform the resource allocation.

Algorithm 1 Slice File Profile Updates

```
1: Inputs: $|\mathcal{S}|$, $K$, $\alpha$
2: Initialization:
3: $p_{s,f} \leftarrow \frac{1}{|\mathcal{K}|} \quad \forall f \in K, \forall s \in \mathcal{S}$.
4: for t ∈ Learning Rounds do
5:     Slice Manager:
6:     Record the anonymized requested files for each user of
       the slice with respect to $\lambda_u$ and create $p_{t,f}$
7:     Update $p_{s,f} := p_{s,f} + \alpha \cdot p_{t,f} \quad \forall f \in K$
8:     Send $p_{s,f}$ to the SD-RAN controller
9: end for
```

D. Optimization Model

The goal of the SD-RAN controller is to distribute the cache and backhaul resources in the most efficient way. That entails maximizing the number of served slices. In that regard, we model our optimization as a resource maximization problem while satisfying the slice constraints. The developed optimization is denoted by $\mathcal{P}_0$ and is detailed in the latter.

Let a_s be a binary decision variable whose value is 1 if slice $s \in \mathcal{S}$ is served and 0 otherwise. Let γ_s and β_s be continuous decision variables and denote the amount of cache and backhaul resources in % assigned to each slice s accordingly. Moreover, let $\overline{D}_s$ denote each slice's average maximum tolerable delay requirement.

The objective is given in Eq. (5)

$$\mathcal{P}_0 : \max_{\gamma_s, \beta_s} \sum_s a_s \tag{5}$$

The additional constraints related to the maximization problem are as expressed as:

$$\sum_{s \in S} \gamma_s \cdot a_s \leq 1 \tag{6}$$

$$\sum_{s \in S} \beta_s \cdot a_s \leq 1 \tag{7}$$

$$\lambda_s^{\text{miss}} \geq 0 \quad \forall s \in S \tag{8}$$

$$\mu_s - \lambda_s^{\text{miss}} \geq 0 \quad \forall s \in S \tag{9}$$

$$f_s = \begin{cases} 1, & \text{if } \lambda_s^{\text{miss}} > 0 \\ 0, & \text{otherwise} \end{cases} \tag{10}$$

$$\mathrm{D}[\gamma_s, \beta_s] \leq a_s \cdot f_s \cdot \overline{D}_s \quad \forall s \in S. \tag{11}$$

Constraints (6), (7), assure that the number of backhaul, cache resources cannot exceed 100% for the assigned slices.

Let constraint (8) and (9) assure that the cache miss rate can never be smaller than 0, whereas the cache miss rate cannot surpass the serving rate. Furthermore, let f_s be a binary variable that takes the value 1 if $\lambda_s^{\text{miss}} > 0$ and 0 otherwise as detailed in Eq. (10). We will use this binary variable as a helper to solve $\mathcal{P}_0$. At last, constraint (11) guarantees that if the slice s is being assigned any resource, then the average maximum delay $\overline{D}_s$ must be fulfilled. If $f_s = 0$, that means that the delay is 0 due to all the requests resulting in a cache hit, whereas if $f_s = 1$ the backhaul link must be utilized.

While identifying the number of served slices, the minimum required amount of resources to achieve the constraints can be retrieved. We refer to these amount of resources as slice cost. In order to be fair among the slices and increase their chances of being selected, even for slices with high cost, we propose an alternative maximization problem similar to $\mathcal{P}_0$. The slice cost denoted by w_s is added as a price that has to be paid by the slice owner to increase its chances of being selected and name this problem $\mathcal{P}_1$ as follows:

$$\mathcal{P}_1 : \max_{\gamma_s, \beta_s} \sum_s a_s \cdot w_s. \tag{12}$$

IV. Performance Evaluation

In the performance evaluation we investigate the effect of slice cooperation in the overall system performance and demonstrate the results of our proposed optimization approach $\mathcal{P}_0$ stated in III-D. We utilize ApOpt solver of Gekko [20] in Python to solve our optimization. Initially we categorize slices into types i.e., cooperative (C), semi-cooperative (sC) and uncooperative (uC). Accordingly, we evaluate their cost depending on system parameters such as cache size B, backhaul capacity Γ, slice delay requirement $\overline{D}_s$ and slice arrival rate λ_s. Finally, we make use of the defined slice cost and propose a pricing method via setting w_s formulated in $\mathcal{P}_1$ for network slices in order to increase their chances of being served. Unless stated otherwise, we consider video streaming and web browsing as applications, $|\mathcal{K}|$ = 5000 files per slice, and the file size L = 100 KB.

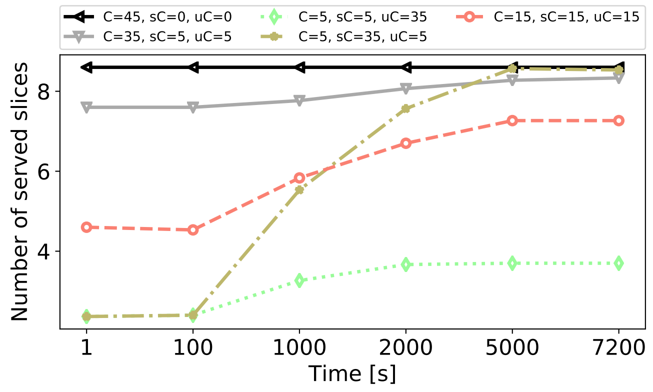


Figure 2: Number of maximum served slices with respect to time for various slice type combinations. Increasing time entails higher accuracy of the user profile for semi-cooperative slices.

A. Maximum number of served slices

In this subsection we investigate the impact of combinations of slice types in terms of the maximum number of served slices. For the given scenario we assume 45 slices each with a distinct file profile following a Zipf distribution with a different z value ranging from 0 to 1. Moreover, we select a cache size of $B = 2000$ files [10], backhaul capacity of $\Gamma = 112$ Mbps (i.e., reflecting a real aircraft setting [4]), whereas for the slices we consider an average delay requirement of $\overline{D}_s = 1$ s and a slice arrival $\lambda_s = 200$ files/s constituted of 20 users each requesting 10 files/s. For our evaluation we select 5 different configurations for the slice type ratios. The first is the one where all the slices are cooperative. Furthermore we consider 3 additional configurations, where alternatively one of the slice types is the majority and finally a configuration of average number of slice types. For each of the configurations we run the optimization 100 times generating different user file profiles each run. The results are illustrated in Fig. 2.

The case where all the slices are cooperative admits the most number of slices. The algorithm minimizes resources allocated per slice which is achieved easier if the user profile is known. At time 1s, the configurations with most cooperative slices have high number of slices served. However, with time the number of slices served increases also for configurations with semi-cooperative slices. At time 7200s, i.e., when a good estimate of the user profile is obtained, the outcome of the configuration with semi-cooperative and cooperative slices is almost the same. Finally, the configurations where the uncooperative slices are the majority demonstrates the least number of admitted slices. Thus, it is concluded that different slice types, even though all their parameters are the same, incur different resource cost. In this example, 9 slices are served with cooperative slices compared to 3 with uncooperative slices. The effect of cooperation is 200% more slices served.

We again stress that the time axis in Fig. 2 can also be considered as the user profile estimation accuracy. In the initial

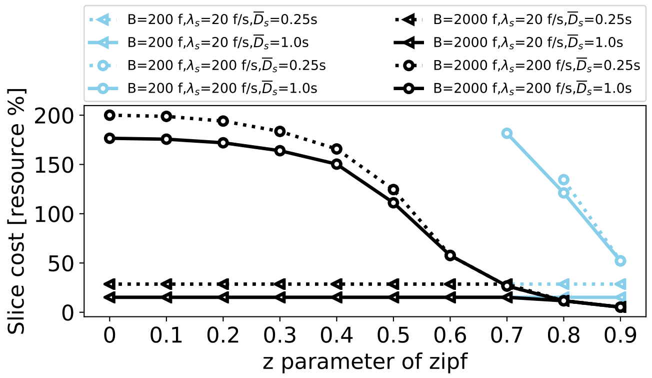


Figure 3: Cost of cooperative slices with respect to various delay, slice arrival and cache combinations for different Zipf distribution shape parameters z.

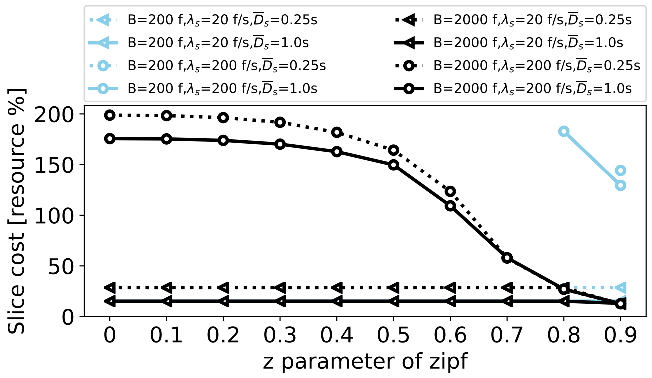


Figure 4: Cost of semi-cooperative slices with respect to various delay, slice arrival and cache combinations for different Zipf distribution shape parameters z at t=1000s.

state (1s), the slice managers possess an inaccurate estimate about the user profile, whereas in the last state (7200s) an accurate estimate is obtained by the slice managers.

B. Slice type cost

In this subsection we investigate the cost of a network slice for various slice types. The cost of a network slice is defined as the % of cache and backhaul resource usage and it is evaluated while varying the z parameter of the Zipf distribution. 8 different configurations are investigated with a high and low cache size $B \in \{200, 2000\}$ files, a high and low slice arrival rate $\lambda_s \in \{20, 200\}$ file requests per second and a high and low average delay requirement $\overline{D}_s \in \{0.25, 1\}$ seconds. The configuration $[B = 200, \lambda_s = 200, \overline{D}_s = 0.25]$ is expected to have the highest cost while the configuration $[B = 2000, \lambda_s = 20, \overline{D}_s = 1]$ is expected to have the lowest.

Fig. 3 illustrates the analysis for the cooperative slices, where the SD-RAN controller knows the file distribution perfectly from the beginning. The y-axis depicts the cost and x-axis depicts the z parameter. The configurations that request more than the available network resources (i.e., 200%) are considered infeasible and thus are not presented in the figure.

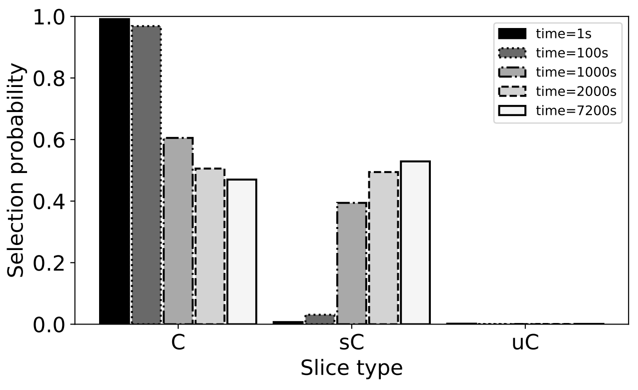
In general, the slice cost increases with increasing slice arrival rate λ_s and decreasing average maximum tolerable delay $\overline{D}_s$. Moreover, the slice cost is inverse proportional to the cache size B. According to the Zipf distribution, increasing z parameter depicts a more skewed distribution. Therefore, the slice cost decreases with increasing z. However, in low slice arrival rate scenario (i.e., $\lambda_s = 20$ files/s) presented by the triangular shape, using the cache is more costly than using the backhaul resources. Therefore, backhaul resources are always booked for that slice and as such, irrespective of the z and cache size value, the cost is the same and the black and blue lines coincide. Nonetheless, in a high cache size scenario depicted by black lines in Fig. 3, for more skewed file profiles (i.e., $z \geq 0.8$) the cache resources are less costly and the slice cost decreases overall. Imposing tight delay constraints increases the resource cost only slightly as demonstrated with the differences between dotted and straight lines.

In a similar fashion we demonstrate the cost of semi-cooperative slices in Fig. 4, where the SD-RAN controller has a semi-accurate file distribution of the slice. The semi-cooperative slice cost at time t=1000s is shown in Fig. 4. A similar trend with the cost of cooperative slices is observed. However, differently for the same configurations at time t=1000s, the cost for semi-cooperative slices is higher. Nonetheless, the semi-cooperative slice cost at time t=5000s is almost the same as that of cooperative slices due to increasing file profile accuracy, thus the results are omitted. Furthermore, the slice cost at time t=1s is the same as that of the uncooperative slices as elaborated next.

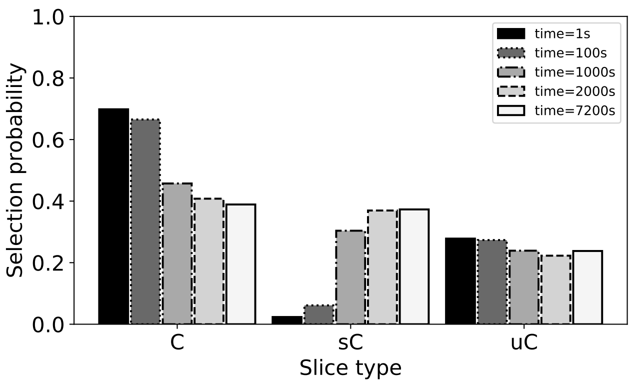
The cost of the uncooperative slices can be obtained from Fig. 3, while observing the results for $z = 0$, which denotes a uniform distribution. The delay requirement makes a slight difference in terms of cost. 10 folds increase in arrival rate almost quadruples the cost. Cache size is the most dominant factor and the slice cannot be served with low cache size. It can be concluded that the highest cost is noted for uncooperative slices since no information is shared with the SD-RAN controller to improve the estimation.

C. Probability of slice selection

In this subsection, we investigate the slice discrimination with respect to the slice type. We achieve this through investigating the slice selection probability for various scenarios. We investigate the case of 45 network slices while randomly selecting the slice types and performing 100 optimization rounds. We want to emphasize that the resources are scarce, i.e., not sufficient to serve all slices. The z parameter is randomly selected for each slice at each round. This problem is inputted in $\mathcal{P}_0$ and the average number of served slices is taken over all cases and the selection probability is reflected in Fig. 5a. The x-axis represents the slice type, whereas the y-axis shows the selection probability. The results reflect that in a resource scarce scenario the uncooperative slices are almost never served and semi-cooperative slices are served only after an accurate estimation of the user profile.



(a) No weights



(b) Weights for all slices

Figure 5: Probability of selection for multiple time instances and slice types:cooperative (C), semi-cooperative (sC) and uncooperative(uC). If weights are introduced into the objective function, slices have almost equal chances of being served.

In order to overcome such starvation for different slice types, we envision that the SD-RAN controller should use weights w_s as in $\mathcal{P}_1$ for the resource allocation problem. Furthermore, we propose to set these weights as the slice costs introduced in the previous section. We have illustrated the selection probabilities with $\mathcal{P}_1$ in Fig. 5b. In the optimal case we would observe $\approx 33\%$ selection probability for all the slice types, which is almost achieved by the proposed weights. These weights are considered as the price to pay for providing each slice fairness without sacrificing network resources.

V. Conclusion

In this paper we investigate the network slicing problem for an IFECS based system, where both cache and backhaul resources are considered as bottleneck. Following a realistic scenario we do not assume a perfect knowledge of the user profiles and distinguish among cooperative, semi-cooperative and uncooperative slices. We propose a MINLP approach that incorporates all slice requirements and aims at maximizing the number of served slices in the network. Moreover, for each slice type we define the deployment cost, and investigate the dependency on time, system and slice parameters. In order to increase the chances of uncooperative slices being served, we propose a pricing system added to the original maximization problem. Our approach increases the selection probability of uncooperative slices by 33% if the slice cost is paid by the slice owners. Overall, we show that cooperative slicing can revolutionize the IFECS system as it accommodates 200% more services compared to uncooperative slicing.

References

[1] G. Giambene, S. Kota, and P. Pillai, "Satellite-5G integration: A network perspective," *IEEE Network*, vol. 32, no. 5, pp. 25–31, 2018.

[2] L. Boero, R. Bruschi, F. Davoli, M. Marchese, and F. Patrone, "Satellite networking integration in the 5G ecosystem: Research trends and open challenges," *IEEE Network*, vol. 32, no. 5, pp. 9–15, 2018.

[3] A. Varasteh, S. Hofmann, N. Deric, M. He, D. Schupke, W. Kellerer, and C. M. Machuca, "Mobility-aware joint service placement and routing in space-air-ground integrated networks," in *IEEE International Conference on Communications (ICC)*, 2019, pp. 1–7.

[4] Gogoair. How do you get loads of bandwidth to a plane. [Online]. Available: https://www.gogoair.com/learning-center/get-loads-bandwidth-plane/

[5] In-flight entertainment and connectivity market. [Online]. Available: https://www.marketsandmarkets.com/Market-Reports/in-flight-entertainment-communications-market-860.html

[6] Inmarsat. The european aviation network. [Online]. Available: https://www.telekom.com/static/-/288318/2/150921-product-sheet-si

[7] X. Wang, M. Chen, T. Taleb, A. Ksentini, and V. C. Leung, "Cache in the air: Exploiting content caching and delivery techniques for 5G systems," *IEEE Communications Magazine*, vol. 52, no. 2, pp. 131–139, 2014.

[8] X. Foukas, N. Nikaein, M. M. Kassem, M. K. Marina, and K. Kontovasilis, "FlexRAN: A flexible and programmable platform for software-defined radio access networks," in *Proc. of the 12th International Conference on emerging Networking EXperiments and Technologies*, 2016, pp. 427–441.

[9] N. Alliance, "Description of network slicing concept," *NGMN 5G P*, vol. 1, p. 1, 2016.

[10] P. L. Vo, M. N. Nguyen, T. A. Le, and N. H. Tran, "Slicing the edge: Resource allocation for RAN network slicing," *IEEE Wireless Communications Letters*, vol. 7, no. 6, pp. 970–973, 2018.

[11] Y. Wang, Y. Gu, and X. Tao, "Edge Network Slicing With Statistical QoS Provisioning," *IEEE Wireless Communications Letters*, vol. 8, no. 5, pp. 1464–1467, 2019.

[12] T. D. Tran and L. B. Le, "Joint wireless access-backhaul network slicing and content caching optimization," in *IEEE International Conference on Communications Workshops (ICC Workshops)*, 2018, pp. 1–6.

[13] A. Papa, M. Klugel, L. Goratti, T. Rasheed, and W. Kellerer, "Optimizing dynamic RAN slicing in programmable 5G networks," in *IEEE International Conference on Communications (ICC)*, 2019, pp. 1–7.

[14] S. Mandelli, M. Andrews, S. Borst, and S. Klein, "Satisfying network slicing constraints via 5G MAC scheduling," in *Proc. IEEE Conference on Computer Communications (INFOCOM)*, 2019, pp. 2332–2340.

[15] E. Bastug, M. Bennis, and M. Debbah, "Living on the edge: The role of proactive caching in 5G wireless networks," *IEEE Communications Magazine*, vol. 52, no. 8, pp. 82–89, 2014.

[16] J. Xu, L. Chen, and P. Zhou, "Joint service caching and task offloading for mobile edge computing in dense networks," in *Proc. IEEE Conference on Computer Communications (INFOCOM)*, 2018, pp. 207–215.

[17] S. D'Oro, L. Bonati, F. Restuccia, M. Polese, M. Zorzi, and T. Melodia, "Sl-EDGE: Network Slicing at the Edge," *arXiv preprint arXiv:2005.00886*, 2020.

[18] P. Yang, N. Zhang, S. Zhang, L. Yu, J. Zhang, and X. S. Shen, "Content popularity prediction towards location-aware mobile edge caching," *IEEE Transactions on Multimedia*, vol. 21, no. 4, pp. 915–929, 2018.

[19] J. Krolikowski, A. Giovanidis, and M. Di Renzo, "Optimal cache leasing from a mobile network operator to a content provider," in *Proc. IEEE Conference on Computer Communications (INFOCOM)*, 2018.

[20] L. Beal, D. Hill, R. Martin, and J. Hedengren, "GEKKO Optimization Suite," *Processes*, vol. 6, no. 8, p. 106, 2018.