Slicing in 5G networks

November 18, 2020

1 Formulation

We model our system as a time-discrete Markov Chain, which we control with a Markov Decision Process (MDP). Therefore, our model is time-slotted, with time slot duration equal to T.

Table 1: Notations

H_{arrivals}	Arrivals histogram
H_{capacity}	Server capacity histogram
$H^{s'}_{\text{departures}}$	Departures histogram with s' running servers
$m(k)$	Number of jobs in the queue during the k-th time-slot
$s(k)$	Number of allocated servers during the k-th time-slot
$act(k)$	Action "allocate act servers" during the k-th time-slot
$a(k)$	Number of incoming jobs during the time-slot k-1
$l(k)$	Number of lost jobs during the time-slot k
$p(k)$	Number of processed jobs during the time-slot k
(m, s)	State of a slice with m jobs and s servers
I	Number of slices
$\{(m_i, s_i)\}_i$	State of a multi-slice system with m_i jobs and s_i servers

1.1 Incoming Traffic Model

We assume to have the ability to compute a discrete probability distribution from the incoming traffic data, we call it *arrival histogram* H_{arrivals}. This describe the incoming traffic in a slice during a time-slot, i.e., the probability to receive a jobs in a time-slot is $H_{\text{arrivals}}(a)$.

1.1.1 Example: Arrival histogram

The following notation can be read as "there is 10% probability that 0 jobs arrive within a time-slot, 80% probability that 1 job arrive, ..."

$$H_{\text{arrivals}}(i) = \begin{cases} 0.1 & \text{if } i = 0 \\ 0.8 & \text{if } i = 1 \\ 0.1 & \text{if } i = 2 \end{cases} \tag{1}$$

1.2 Server Capacity Model

As we will assume homogeneous servers, we compute a discrete distribution called *server capacity histogram* H_{capacity}. This describe the serving capability of a single server, i.e., the probability for a single server to process x jobs in a time-slot is $H_{\text{capacity}}(x)$.

1.2.1 Example: Server capacity histogram

The following notation can be read as "there is 100% probability that one server process 1 job per time-slot"

$$H_{\text{capacity}}(i) = \begin{cases} 0 & \text{if } i = 0 \\ 1 & \text{if } i = 1 \end{cases} \tag{2}$$

1.2.2 Departures Histogram

The serving capacity of the entire system, which we denote *departures histogram* $H^{s'}_{\text{departures}}$, with s' running servers is the following convolution:

$$H^{s'}_{\text{departure}} \triangleq \underbrace{H_{\text{capacity}} \circledast \cdots \circledast H_{\text{capacity}}}_{s' \text{ times}} \tag{3}$$

This describes the departure distribution of the system, i.e., the probability for the system to process x jobs in a time slot is $H^{s'}_{\text{departure}}(x)$, where s' is the number of active servers. As [10] states, notice that if the number of active servers s' is equal to 0, then the departures histogram will be the Dirac histogram in 0:

$$\Delta_0(i) = \begin{cases} 1 & \text{if } i = 0 \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

1.3 Single-Slice Model

In this section we focus on one slice. The slice has a limited number, denoted with *max servers*, of servers (VMs) which can be allocated and a FIFO queue with limited size, denoted with *qsize*, which contains incoming jobs still to be processed.

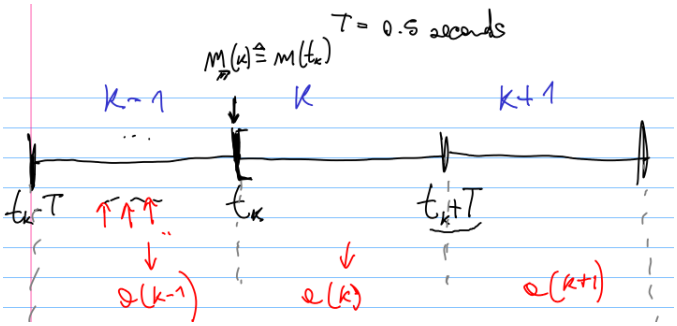

Figure 1: Time slots

1.3.1 State definition and evolution

The underlying system is in reality continuous. We represent its state as a pair $(m(t), s(t))$, where $m(t)$ is the number of jobs and $s(t)$ is the number of servers at instant t. Observe that such state continuously evolve with $t \in [0, +\infty[$. However, the *network operator* is able to measure the system only at the beginning of every time-slot and is not able to do anything (neither measuring nor taking actions) within a time-slot. So, for the point of view of our formulation everything happens only at the starting instant of each time-slot. Refer to the Fig. **??**

More formally, suppose the duration of a time slot is T, for instance $T = 1$ sec. The k-th time slot is $k = [t_k, t_k + T[$, where t_k is the starting instant of the time slot. We associate a state to each time slot corresponding to the sampling of $(m(t), s(t))$ at the beginning of that time slot:

$$m(k) = m(k - \text{th slot}) = m(\text{time slot } [t_k, t_k + T[) \triangleq m(t_k) \tag{5}$$

$$s(k) = s(k - \text{th slot}) = m(\text{time slot } [t_k, t_k + T[) \triangleq s(t_k) \tag{6}$$

The formula above emphasizes the fact that, despite the state continuously evolve, we only sample it in discrete time instants, corresponding to the beginning of each time-slot.

1.3.2 Operations in a time slot

Let us consider a time slot $k = [t_k, t_k + T[$. In the starting instant t_k, the network operator executes the following operations:

1. It observes the current state $(m(k), s(k)) = (m(t_k), s(t_k))$, i.e., the number of jobs in the queue and of allocated servers.

2. It chooses an action $act(k)$ (which consists in enforcing exactly act servers) and executes it instantaneously (meaning that VMs are allocated or deallocated instantaneously at this exact instant). Therefore, the number of servers change from $s(t_k)$ to $s(t_k^+)$

3. Then, we observe the jobs that have arrived within time-slot $k-1 = [t_k - T, t_k[$. We call the number of such arrivals $a(k-1)$. We put such jobs in the queue, up to qsize, all the remaining jobs are lost.

4. Then, we dequeue the jobs and fill the free servers for the processing. The number of processed jobs in time slot $k = [t_k, t_k + T[$ is denoted with $p(k)$.

1.3.3 State

The set of states is:

$$\mathcal{S} \triangleq \{(m, s) | 0 \leq m \leq \text{qsize}, 0 \leq s \leq \max \text{servers}\} \tag{7}$$

where m is the number of jobs in the queue and s is the number of allocated servers.

1.3.4 Actions

The network operator at the beginning of every time-slot takes an action that can change the number of allocated servers for a slice. Since we are dealing with a single slice, the admitted actions are

$$\mathcal{A} \triangleq \{act | act \leq \max \text{server}\} \tag{8}$$

1.3.5 Transition Probability

The transition probability from (m, s) to (m', s') is:

$$Q^{\text{act}}(m, s \to m', s') = \begin{cases} Q(m, s \to m', s') & \text{if } s' = \text{act} \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

where

$$\begin{aligned} Q(m, s \to m', s') &\triangleq \mathbb{P}\left(m \to m' | s \text{ severs currently active}, s' \text{ severs active in the next slot }\right) \\ &= \sum_{a=[m'-m]^+}^{\text{qsize}-m} P(\text{arr} = a) \cdot P(\text{proc} = m + a - m' | a + m) \\ &+ \sum_{a=\text{qsize}-m+1}^{\infty} P(\text{arr} = a) \cdot P(\text{proc} = \text{qsize} - m' | \text{qsize}) \end{aligned} \tag{10}$$

where $P(\text{proc} = x|y)$ is the probability of processing x jobs given that y jobs are found in the queue the instant when the processor starts to pick jobs

from the queue:

$$P(\text{proc} = x|y) = \begin{cases} H^{s'}_{\text{departures}}(x) & \text{if } x < y \\ \sum_{x=y}^{\infty} H^{s'}_{\text{departures}}(x) & if x = y \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

1.3.6 Reward

We map the QoS of the system and the energy cost of the underlying hardware into the reward. The cost to be in a state during a time-slot is calculated as:

$$C \triangleq \alpha \cdot C_m \cdot m + \beta \cdot C_l \cdot \mathbb{E}(l) + \gamma \cdot C_s \cdot s \tag{12}$$

Where C_m is the unitary cost to have a waiting job in the queue for one time-slot; C_l is the unitary cost to have a lost job in the queue for one time-slot, $\mathbb{E}(l)$ is the expected value of lost jobs; C_s is the unitary cost to have a running server during one time-slot. α, β and γ are normalized in order to avoid that one part of the equation take over all the rest and to weight differently the different components of the cost. For instance, one might choose to give more weight to the scarcest resource, e.g., the energy or the space in the queue. The reward we take into account is just

$$R \triangleq -C \tag{13}$$

We convert all the components in a single unity of measurement,, which is money metric.

We will fill this table with values from the literature.

Table 2: Money metric (values from [10])

Server cost	300$ per year
Waiting job cost	$6.2 * 10^-6$$
Lost job cost	$6.2 * 10^-6$$

1.3.7 Policy

as: add this section?

1.3.8 Evolution Equations

We said that the number of allocated servers changes only at the beginning of the time-slot. In particular during the time-slot k the number of active

server after the action $act_x(k)$ is:

$$s(t_k^+) = x \tag{14}$$

The number of waiting jobs in the time-slot $k+1$ can be calculated as follows:[1]

$$\underbrace{m(k+1)}_{m(t_k+T)} = \min(\text{qsize}, \underbrace{m(k)}_{m(t_k)} - p(k) + a(k)) \tag{15}$$

The number of lost jobs in the time-slot $k+1$ can be calculated as follows:

$$l(k+1) = [m(k) + a(k) - \text{qsize}]^+ \tag{16}$$

Where $a(k)$ is the number of incoming jobs in the time-slot $k-1$; $p(k)$ is the amount of processed jobs during the time-slot k.

1.4 Multiple slices

In this section we extend the formulation to multiple slices. We assume I slices, which do not change with the time. Every slice i has a limited queue of size qsize_i and shares the amount M of servers with the other slices.

We assume that slices are isolated, so each virtual machine belongs to one and only one slice. We tackle a scenario where slices have heterogeneous requirements, in terms of computation or latency, to model this behaviour each slice have a different server capacity histogram $H_{capacity}$. Therefore, it is very important to prioritize the slices in order to benefit those with more stringent requirements.

1.4.1 Priority Decision Sequence

We assume there is maximum number M of servers. We give the highest priority to the slice the most stringent requirements, which we call slice 0. To do so, we let such a slice decide the number s_0^{min} of servers to allocate in order to meet its stringent requirements, with a maximum of $M_0 \leq M$. This leaves a certain number $\tilde{M}_1 = M - s_0^{min}$ of available servers for the remaining slices. Then, we let the second-priority slice, slice number 1 decide the number s_1 of servers, which in any case cannot exceed neither a certain limit M_1 nor $\tilde{M}_1$, i.e. $s_1 \leq \min(M_1, \tilde{M}_1)$. We then let the other slices make their choices, in the order of priority.

[1] This formula corresponds to the one in Sec. 1.3.1 of [10].

1.4.2 Example: Priority Decision Sequence

Let's suppose to have three slices and so three priority levels. The maximum number of allocable servers in the system M is 10. Since we have three slices we have the following values:

$$M \leq M_0 \leq M_1 \leq M_2 \tag{17}$$

For simplicity let's suppose $M_0 = M_1 = M_2 = M$, but this can be different in order to limit the maximum server cap for each slice.

1. The slice 0 run his MDP with available servers equal to M_0. The result of the MDP run is an allocation policy, this one have a minimum s_0^{min} and a maximum value s_0^{max}.

2. The slice 1 could have a policy with allocations up to M_1, but this is not possible because the system servers number is limited and the highest priority slice already occupied some computational resources, this bring the slice 1 to be able to compute an MDP up to $\tilde{M}_1 = M_1 - s_0^{min}$ servers. Basically the allocation policy of the slice 1 have to dynamically change based on how much servers are left from the highest priority slice; this information is not something that we know at policy computation time, so we have to compute several single-slice MDP with different value of max servers. To be precise, the slice 1 have to compute an MDP for each value of max server in $[M_1 - s_0^{max}, \tilde{M}_1]$.

3. Returning to our numerical example, let's suppose $s_0^{min} = 1$ and $s_0^{max} = 5$, this means that the policy of slice 0 allocate a number of servers in the interval $[1, 5]$. Logically the slice 1 can allocate up to $\tilde{M} = 9$ when the slice 0 is served from one server. Since the max servers in the system are 10, when the slice 0 increase his allocated servers it can be possible that the slice 1 have to decrease his own servers because of the system constraints. In order to implement this preemption mechanism in the Network Operator with multi-slice control capability, seems appropriate that the slice 1 have to compute several MDP policies as in the previous point.

4. At this point the slice 2, the lowest priority slice in the system, will calculate his $\tilde{M}_2 = M_2 - s_1^{min}$ and will run multiple MDP within the interval of servers left.

1.4.3 State

The state space is

$$\mathcal{S} \triangleq \left\{ ((m_1, s_1), \ldots, (m_I, s_I)) \,|m_i \leq \text{qsize}_i, i = 1, \ldots, I; \sum_{i=1}^{I} s_i \leq \text{max servers}; s_i \leq \tilde{M}_i \right\} \tag{18}$$

where $\tilde{M}_i$ is the maximum amount of servers allowed to slice i. For the sake of compactness, we may write $\{(m_i, s_i)\}_i$ in lieu of $((m_1, s_1), \ldots, (m_I, s_I))$.

1.4.4 Actions

The action space is:

$$\mathcal{A} \triangleq \{(\text{act}_1, \ldots, \text{act}_I) | \text{act}_1 + \cdots + \text{act}_I \leq \max \text{server}; \text{act}_i \leq \tilde{M}_i\} \tag{19}$$

1.4.5 Transition Probability

Since the slices are independent, the transition probability is the product of the transition probabilities in each single slice. Therefore, the transition probabilities from $\{(m_i, s_i)\}_i$ to $\{(m'_i, s'_i)\}_i$ is:

$$Q^{\{\text{act}_i\}_i}\left(\{(m,s)\}_i \to \{(m',s')\}_i\right) = \begin{cases} Q\left(\{(m_i,s_i)\}_i \to \{(m'_i,s'_i)\}_i\right) & \text{if } \{s'_i\}_i = \{\text{act}_i\}_i \\ 0 & \text{otherwise} \end{cases} \tag{20}$$

where

$$\begin{aligned} & Q\left(\{(m_i,s_i)\}_i \to \{(m'_i,s'_i)\}_i\right) \\ \triangleq\ & \mathbb{P}\left(m_i \to m'_i, \forall \text{slice} i | s_i \text{ severs currently active}, s'_i \text{ severs active in the next slot }\right) \\ =\ & \prod_{i=1}^{I} \mathbb{P}\left(m_i \to m'_i | s_i \text{ severs currently active}, s'_i \text{ severs active in the next slot }\right) \\ =\ & \prod_{i=1}^{I} Q(m_i, s_i \to m'_i, s'_i) \end{aligned}$$

where $Q(m_i, s_i \to m'_i, s'_i)$ can be calculated as in (10).

2 Modeling issues

2.1 Time-Slot Dimension

Arrivals should be i.i.d, which means that the arrival histogram in all the time-slots is identical.

This property is crucial in order to consider the probability independence of arrivals in a slice, but in order to obtain this we must sample the traffic data trace with a period for which the arrival jobs can be considered i.i.d [10](par 1.3.3). This could lead to choose a time-slot in the order of seconds, but this is not optimal because we work with a continuous-time system (the real slice) in a discrete-time point of view (our MDP formulation and network operator execution): a time-slot too big could bring our control

system to be slow to handle incoming jobs. In the other hand we also work with servers allocation and this kind of operation need more time than the time that arrivals need.

as: in this section will be the work related the delayed mdp

3 Related Work

The slice admission control problem is studied in [2]. Their model is however unrealistic, as they assume that all slices of a certain type have the same resource requirements. This is generally false, as two slices of different type, say vehicular, have different amount of users and thus need completely different amount of resources. Moreover, we add the decisions of the amount of resources to assign to each tenant.

Admission control of slices has also been studied in [1]. This work has the same limitation as the previous: it assumes there exist a set of slice classes and that all tenants of the same class request exactly the same resources.

In [3], multiple tenants share Physical Resource Blocks under overall capacity constraints. An admission control policy is defined.

CloudScale [4].

To read:

- "Multiclass Multiserver Threshold-Based Systems: A Study of Noninstantaneous Server Activation"
- "Multi-Resource Allocation for Network Slicing"
- Sl-EDGE: Network Slicing at the Edge
- Energy Efficient Communication and Computation Resource Slicing for eMBB and URLLC Coexistence in 5G and Beyond
- Network Slicing: Recent Advances, Taxonomy, Requirements, and Open Research Challenges

4 Extension

- Multi-scale MDP
- Constrained MDP to satisfy minimum requirements

5 Conferences to consider

- IFIP networking, class A, 05/01/2021
- CCNGrid, class A, 15 December

References

[1] Bega, D., Gramaglia, M., Banchs, A., Sciancalepore, V., Samdanis, K., & Costa-Perez, X. (2017). Optimising 5G infrastructure markets: The business of network slicing. Proceedings - IEEE INFOCOM. https://doi.org/10.1109/INFOCOM.2017.8057045

[2] Han, B., Feng, D., Schotten, H. D. (2018). A Markov Model of Slice Admission Control. IEEE Networking Letters, 1(1), 2–5. https://doi.org/10.1109/lnet.2018.2873978

[3] Vila, I., Sallent, O., Umbert, A., & Perez-Romero, J. (2018). Guaranteed Bit Rate Traffic Prioritisation and Isolation in Multi-tenant Radio Access Networks. IEEE International Workshop on Computer Aided Modeling and Design of Communication Links and Networks, CAMAD, 2018-September. https://doi.org/10.1109/CAMAD.2018.8515006

[4] Shen, Zhiming, et al. "Cloudscale: elastic resource scaling for multi-tenant cloud systems." Proceedings of the 2nd ACM Symposium on Cloud Computing. 2011.

[5] Tournaire, T., Castel-Taleb, H., Hyon, E., & Hoche, T. (2019). Generating Optimal Thresholds in a Hysteresis Queue: Application to a Cloud Model. In IEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS) (pp. 283–294). IEEE. https://doi.org/10.1109/mascots.2019.00040

[6] Cesa-Bianchi, N., Dekel, O., & Shamir, O. (2013). Online learning with switching costs and other adaptive adversaries. Advances in Neural Information Processing Systems, 1–9.

[7] Ortner, R. (2010). Online regret bounds for Markov decision processes with deterministic transitions. Theoretical Computer Science, 411(29–30), 2684–2695. https://doi.org/10.1016/j.tcs.2010.04.005

[8] Katsikopoulos, K. V., & Engelbrecht, S. E. (2003). Markov decision processes with delays and asynchronous cost collection. IEEE transactions on automatic control, 48(4), 568-574

[9] Boxma, O. J. (2008). Stochastic Performance Modelling.

[10] Bayati (2019). Data Centers Energy Optimization.