DASH-aware Scheduling in Cellular Network

Albert Sunny
Computer Science and Engineering
Indian Institute of Technology Palakkad

joint work with Prof. El-Azouzi, Prof. Altman and Huawei

13 July, 2022

About DASH

- Each video is divided into multiple segments (each containing about 2-10 seconds of video).
- Each segment is then encoded into multiple bitrates/resolutions.
- Based on estimated throughput and media playout buffer occupancy, an adaptation engine within the MPEG-DASH client chooses a video bitrate, on a segment-by- segment basis.

Key QoE Metrics

- *Average bitrate*: sum of the video bitrate of the segments downloaded by the player divided by the total number of downloaded segments.
- *Number of bitrate switching*: number of times the video quality has changed during its playout.
- *Buffering ratio*: fraction of the total session time spent in re-buffering.
- *Re-buffering rate*: number of interruptions observed by a user watching a video.
- *Startup delay*: duration between initiation of a video session and the start of its playout.

Buffer Evolution

Let $l(t)$ and $b(t)$ denote the bit-rate selected and buffer size at time t, respectively.

$$b(t_n) = \left[b(t_{n-1}) + s - \frac{s \cdot l(t_{n-1})}{r} \right]^+ \quad (1)$$

where $s \cdot l(t_{n-1})$ represents the size of $(n-1)^{\text{th}}$ segment in bytes. In this case we have $t_n - t_{n-1} = s \cdot l(t_{n-1})/r$. Assuming that the bit-rate remains constant during segment download, we have

$$\frac{db(t)}{dt} = \begin{cases} \frac{r}{l(t)} - 1 & \text{if } b(t) > 0 \\ 0 & otherwise \end{cases} \quad (2)$$

Stability under Continuous Bit-rate Mapping

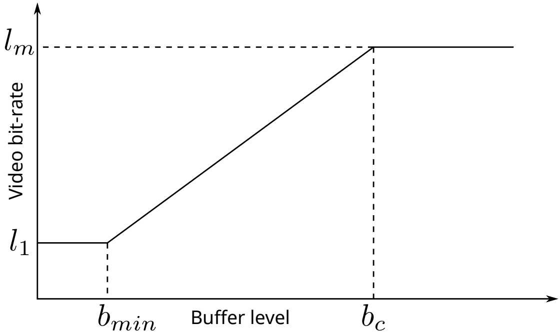


Figure: A continuous bit-rate map.

$$\text{Fixed points are } \begin{cases} [0, b_{min}] & \text{if } r \leq l_1 \\ [b_c, b_{max}] & \text{if } r \geq l_m \\ b_{min} + \frac{(r-l_1)(b_c-b_{min})}{(l_m-l_1)} & \text{otherwise} \end{cases}$$

Stability under Discrete Quality Levels

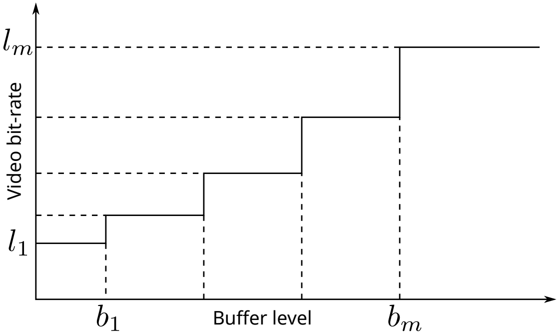


Figure: A mapping of discrete bit-rates.

Fixed points are $[b_{i-1}, b_i)$ when $r \in [l_{i-1}, l_i)$ for all $1 \leq i \leq m$. The jump points will lead to quality switches because quality selection is a discrete time process.

Stability under Discrete Quality Levels

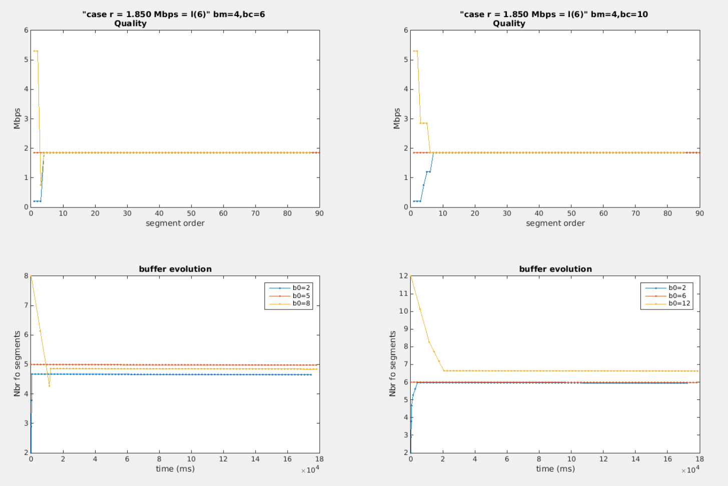


Figure: r is chosen as 1.85 $Mbps$; one of the DASH quality levels.

Stability under Discrete Quality Levels

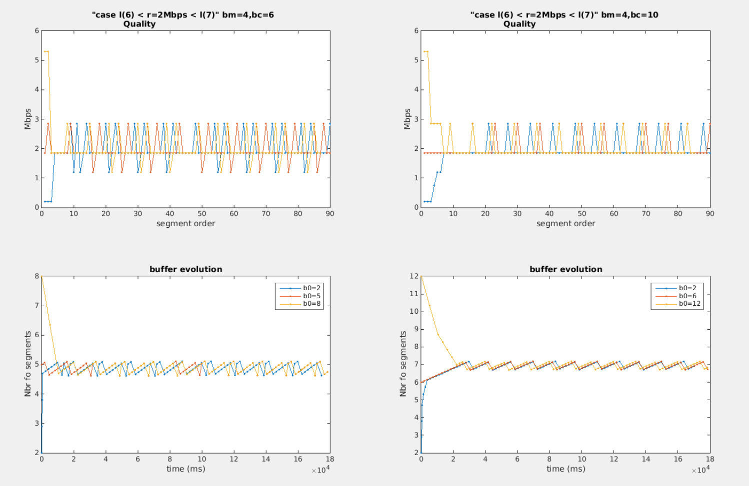


Figure: r is chosen as 2.0 $Mbps$; not one of the DASH quality levels.

Stability under Discrete Quality Levels

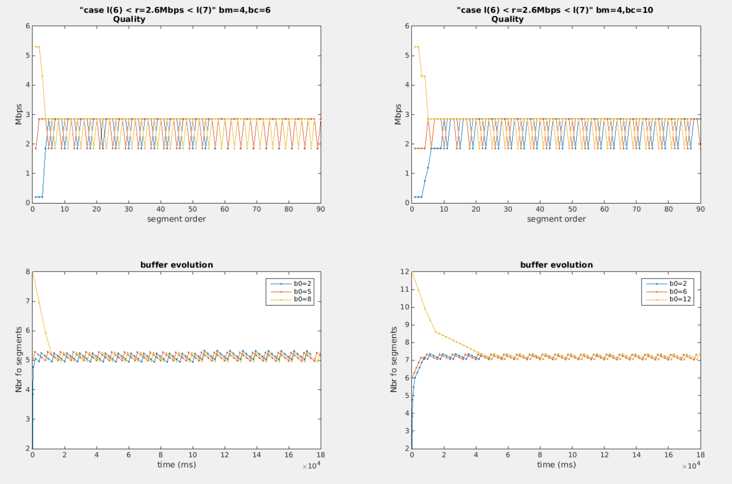


Figure: r is chosen as 2.6 $Mbps$; not one of the DASH quality levels.

Stability under Discrete Quality Levels

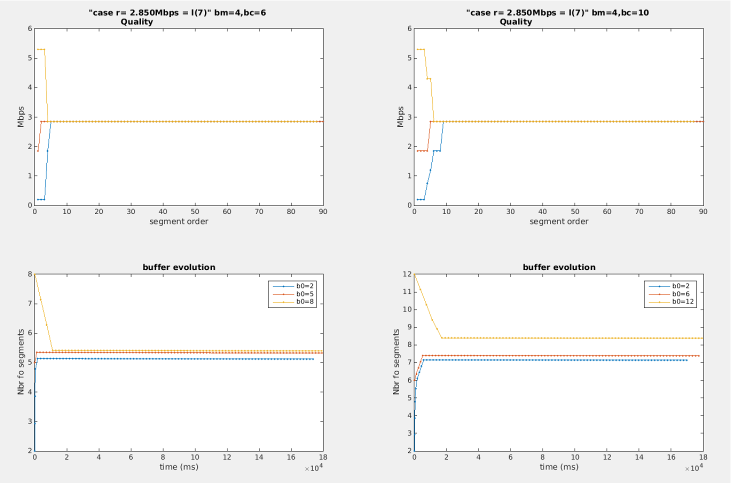
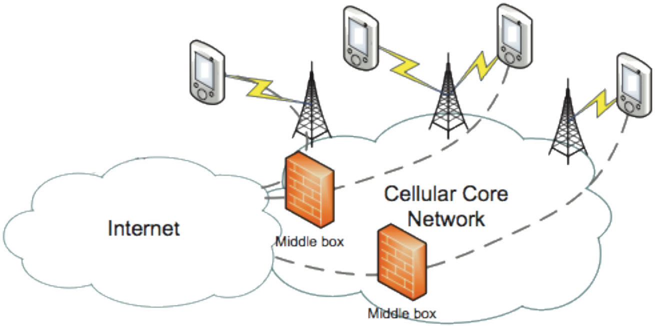


Figure: r is chosen as $2.85\,Mbps$; one of the DASH quality levels.

Scheduling



1

[1] Source: https://hothardware.com/news/cellular-networks-have-issues-says-research

Scheduler Design Objectives

- Aim to provide better throughput while ensuring that the average throughput of the flows matches one of the video bit-rate levels.
- Exhibit fairness.
- Adapt existing scheduling frameworks, with minimal modification, to handle DASH flows.
- Average throughput should stabilize quickly.

Scheduler Design

In any time slot t, we are interested in the allocation that achieves the following

$$\max \sum_{i \in \mathcal{N}} U_i^{'}(\theta_i(t)) \cdot \Gamma_i(t)$$

where $\Gamma_i(t)$ is the instantaneous channel capacity in time-slot t. When only one user can be scheduled at a time, the solution of the above optimization problem is as follows

$$i^*(t) = \arg \max_{1 \leq i \in \leq n} U_i^{'}(\theta_i(t)) \cdot \Gamma_i(t)$$

When $U(\cdot) = \log(\cdot)$,we have

$$i^*(t) = \arg \max_{1 \leq i \leq n} \frac{\Gamma_i(t)}{\theta_i(t)}$$

Average Throughput Based

$$i^*(t) = \arg\max_{1 \le i \le n} \Gamma_i(t)/\gamma_i(t)$$

where $\gamma_i(t)$ is the average throughput of user i at time t, i.e.,
$\gamma_i(t) = \frac{1}{t}\sum_{\tau=1}^{t} \Gamma_i(t) \cdot 1_{\{i^*(t)=i\}}$

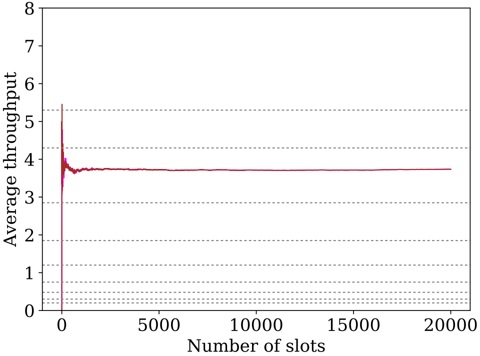


Figure: 2 homogeneous users, *ON-OFF* channel with *ON* capacity 10 *Mbps*, and *ON* probability 0.5.

Average Throughput Based

- User get proportional fairness.

Average Throughput Based

- User get proportional fairness.
- Average throughput may not be equal to one of the video bit-rates; resulting in frequent video quality switches.

Penalty Weighted Average Throughput

Guiding average throughput to a target rate r_i

$$i^*(t) = \arg\max_{1 \le i \le n} \frac{\Gamma_i(t)}{\gamma_i(t) \cdot \phi_i(t)} \tag{3}$$

where $\phi_i(t) = \exp(\beta \eta_i(t))$. In Equation (3), $\eta_i(t)$ is the deviation of the average throughput from the desired value, i.e, $\eta_i(t) = \gamma_i(t) - r_i$.

- β denotes the growth rate of penalty function $\phi_i(t)$.
- $\lim_{x \to \infty} \phi_i(t) = \infty$, i.e., large positive deviations impose a high penalty.
- We have $\lim_{x \to -\infty} \phi_i(t) = 0$, i.e., for large values of β, penalty function assign a penalty very close to zero for negative deviation.
- If $\beta = 0$, then $\phi_i(t) = 0$ and we get *proportional-fair* scheduling.

Penalty Weighted Average Throughput

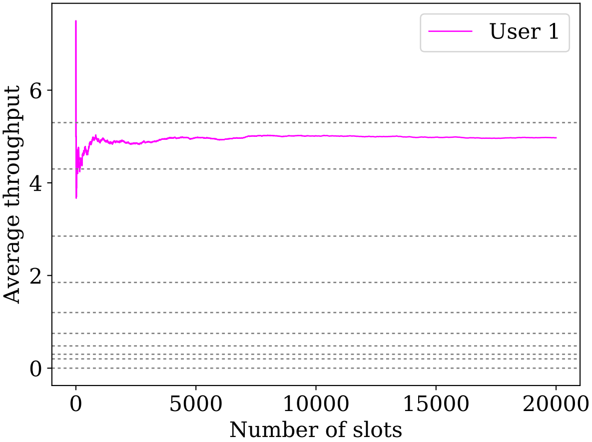


Figure: 1 user, *ON-OFF* channel with *ON* capacity 10 *Mbps*, and *ON* probability 0.5.

Penalty Weighted Average Throughput With Virtual User

Consider a network with n users. To this network, we add a virtual user indexed as $n+1$. Then, the scheduling scheme is as follows

$$i^*(t) = \begin{cases} \arg\max_{1\le i\le n} \frac{\Gamma_i(t)}{\gamma_i(t)\cdot\phi_i(t)} & \text{if } \max_{1\le i\le n} \frac{\Gamma_i(t)}{\gamma_i(t)\cdot\phi_i(t)} \\ & \quad \ge \max_{1\le i\le n} \frac{\Gamma_i(t)}{\gamma_i(t)(1+\epsilon)} \\ n+1 & \text{otherwise} \end{cases} \tag{4}$$

- If $\gamma_i(t) > r_i + \frac{\epsilon}{\beta}$, then $i^*(t) \neq i$, i.e., nodes whose average throughput is higher than the respective target rates are not scheduled.
- If $\gamma_i(t) \le r_i + \frac{\epsilon}{\beta}$ for some $i \in \mathcal{N}$, then $i^*(t) \neq n+1$, i.e., the virtual node always shares its air time with nodes whose average throughput is below their respective target rates.

Penalty Weighted Average Throughput With Virtual User

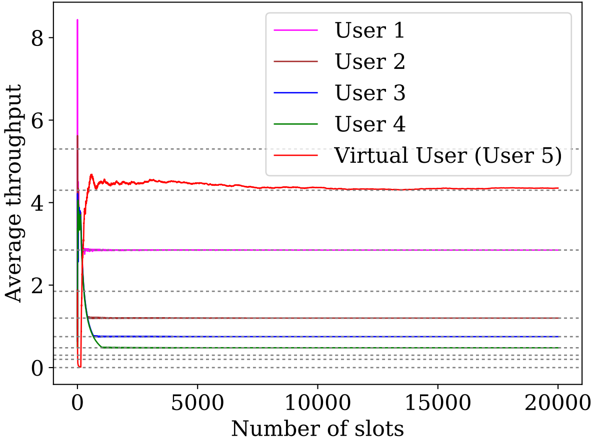


Figure: 4 users, $r_1 = 2.85\ Mbps$, $r_2 = 1.2\ Mbps$, $r_3 = 0.75\ Mbps$, $r_4 = 0.48\ Mbps$, $\beta = 1000$, $\epsilon = 0.001$, each user has an *ON-OFF* channel with *ON* capacity 15 *Mbps*, and *ON* probability 0.5.

Penalty Weighted Average Throughput With Virtual User

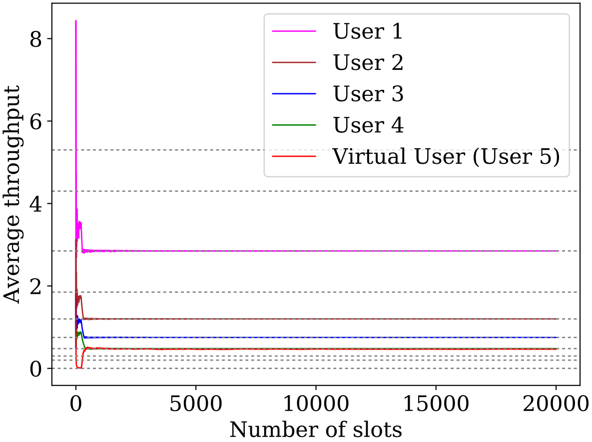


Figure: 4 users, $r_1 = 2.85\, Mbps$, $r_2 = 1.2\, Mbps$, $r_3 = 0.75\, Mbps$, $r_4 = 0.48\, Mbps$, $\beta = 1000$, $\epsilon = 0.001$, user $i \in \{1, 2, 3, 4\}$ has an *ON-OFF* channel with *ON* capacity $\frac{15}{i}\, Mbps$, and *ON* probability 0.5.

Penalty Weighted Average Throughput With Virtual User

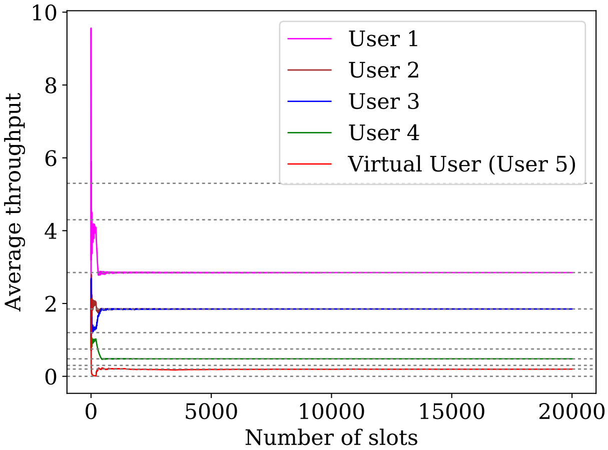


Figure: 4 users, $r_1 = 2.85\ Mbps$, $r_2 = 1.8\ Mbps$, $r_3 = 1.8\ Mbps$, $r_4 = 0.48\ Mbps$, $\beta = 1000$, $\epsilon = 0.001$, user $i \in \{1, 2, 3, 4\}$ has an *ON-OFF* channel with *ON* capacity $\frac{15}{i}\ Mbps$, and *ON* probability 0.5.

Computing the Optimal Target Rates

$$P_1 : \max_{\{a_i(\mathsf{c})\ i\in\mathcal{N}\ \mathsf{c}\in\mathcal{C}, y_{ij}\ i\in\mathcal{N}\ 1\le j\le m\}} \sum_{i\in\mathcal{N}} U_i\left(\sum_{j=1}^{m} b_j \cdot y_{ij}\right)$$

Subject to:

$$\sum_{\mathsf{c}\in\mathcal{C}} \pi(\mathsf{c})\cdot a_i(\mathsf{c})\cdot \mathsf{c}[i] = \sum_{j=1}^{m} b_j y_{ij} \quad \forall i\in\mathcal{N}$$

$$\sum_{i\in\mathcal{N}} a_i(\mathsf{c}) \le 1 \quad \forall \mathsf{c}\in\mathcal{C} \text{ and } a_i(\mathsf{c}) \ge 0 \quad \forall i\in\mathcal{N}, \mathsf{c}\in\mathcal{C}$$

$$\sum_{j=1}^{m} y_{ij} = 1 \quad \forall i\in\mathcal{N}, \mathsf{c}\in\mathcal{C} \text{ and } y_{ij}\in\{0,1\} \quad \forall i\in\mathcal{N}, 1\le j\le m$$

where $\mathsf{c}[i]\in\mathcal{C}_i$ denotes the i^{th} element of vector $\mathsf{c}\in\mathcal{C} = \mathcal{C}_1\times\mathcal{C}_2\times\cdots\times\mathcal{C}_n$, $\pi(c)$ is probability of occurrence of state c and $\{b_1, b_2, \cdots, b_m\}$ denote the set of m video bit-rates.

Computing the Optimal Target Rates

- The optimization problem is a *Mixed Integer Non-linear Program*; a well-know *NP-hard* problem.

Computing the Optimal Target Rates

- The optimization problem is a *Mixed Integer Non-linear Program*; a well-know *NP-hard* problem.
- We may resort to heuristics to solve the problem. However, the number of real-valued variables grows rapidly as the number of user increases; courtesy of the joint channel state space $\mathcal{C} = \mathcal{C}_1 \times \mathcal{C}_2 \times \cdots \times \mathcal{C}_n$.

Dynamic Rate Throttling

Dynamically computing the target rate r_i from the PF scheduler

$$I_{ij}(t) = \begin{cases} 1 & \text{if } j = \arg\max\{b_k | b_k \leq \gamma_i(t), 1 \leq k \leq m\} \\ 0 & \text{otherwise} \end{cases}$$

i.e., $I_{ij}(t) \in \{0, 1\}$ is binary valued variable that takes the value 1 if b_j is the largest bit-rate less than the average throughput $\gamma_i(t)$. We note that $\sum_{j=1}^{m} I_{ij}(t) = 1$. For user i, we define a random time T_i as follows

$$T_i = \min\left\{ \min\left\{ t \middle| \max_{1 \leq j \leq m} \frac{1}{t} \sum_{\tau=1}^{t} I_{ij}(t) \geq \zeta,\ t \geq t_{min} \right\}, t_{max} \right\} \quad (5)$$

where $\zeta \in [0, 1]$. $T_i \in [t_{min}, t_{min} + 1, \cdots, t_{max} - 1, t_{max}]$ is the first time at which the quantity $\max_{1 \leq j \leq m} \frac{1}{t} \sum_{\tau=1}^{t} I_{ij}(t)$ exceeds the threshold ζ

Dynamic Rate Throttling

- We are looking for a time when there exists a bit-rate which was the largest bit-rate less than the average throughput for at least ζ fraction of slots.

Dynamic Rate Throttling

- We are looking for a time when there exists a bit-rate which was the largest bit-rate less than the average throughput for at least ζ fraction of slots.
- There is a possibility of this never happening, in which case the value of T_i is capped at t_{max}.

Dynamic Rate Throttling

- We are looking for a time when there exists a bit-rate which was the largest bit-rate less than the average throughput for at least ζ fraction of slots.
- There is a possibility of this never happening, in which case the value of T_i is capped at t_{max}.
- The first t_{min} slot are discarded to suppress transients of the *PF scheduler*.

Dynamic Rate Throttling

- We are looking for a time when there exists a bit-rate which was the largest bit-rate less than the average throughput for at least ζ fraction of slots.
- There is a possibility of this never happening, in which case the value of T_i is capped at t_{max}.
- The first t_{min} slot are discarded to suppress transients of the *PF scheduler*.
- Given T_i, the target rate r_i of user i is chosen as follows

$$r_i = b_{j^*(i)} \quad \text{where} \quad j^*(i) = \arg \max_{1 \le j \le m} \frac{1}{T_i} \sum_{\tau=1}^{T_i} I_{ij}(T_i)$$

Dynamic Rate Throttling

- We can switch on the penalty function for this user so that the average throughput is guided to this value.

Dynamic Rate Throttling

- We can switch on the penalty function for this user so that the average throughput is guided to this value.
- Achieved by manipulating the growth rate of the penalty function.

Dynamic Rate Throttling

- We can switch on the penalty function for this user so that the average throughput is guided to this value.
- Achieved by manipulating the growth rate of the penalty function.
- Let $T = \max_{1 \leq i \leq n} T_i$, i.e., T is the time when all users have chosen their target rates. Since the scheduler operates in two different settings, we propose the following dynamic growth rate of the penalty function $\phi_i(t)$

$$\beta_i(t) = \begin{cases} 0 & \text{if } t \leq T \\ \beta & \text{otherwise} \end{cases}$$

Dynamic Rate Throttling

- We can switch on the penalty function for this user so that the average throughput is guided to this value.

- Achieved by manipulating the growth rate of the penalty function.

- Let $T = \max_{1 \le i \le n} T_i$, i.e., T is the time when all users have chosen their target rates. Since the scheduler operates in two different settings, we propose the following dynamic growth rate of the penalty function $\phi_i(t)$

$$\beta_i(t) = \begin{cases} 0 & \text{if } t \le T \\ \beta & \text{otherwise} \end{cases}$$

- The above scheme switches on penalty only after all the users have acquired their respective target rates. This scheme may result in under utilization of the resources. However, it ensures fairness.

Dynamic Rate Throttling

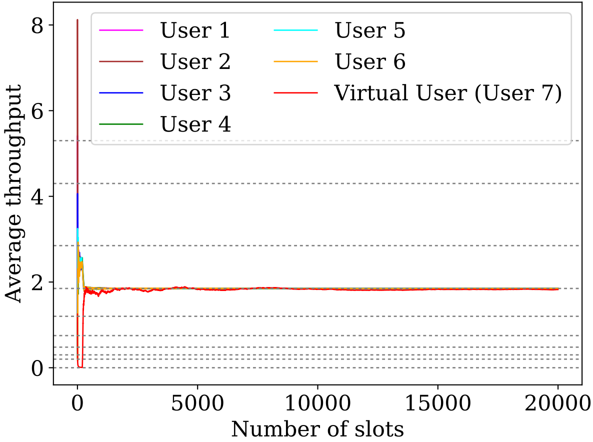


Figure: 6 homogeneous users, $t_{min} = 100$, $t_{max} = 1000$, $\zeta = 0.9$, $\beta = 1000$, $\epsilon = 0.001$, each user has an *ON-OFF* channel with *ON* capacity 15 *Mbps*, and *ON* probability 0.5.

Dynamic Rate Throttling

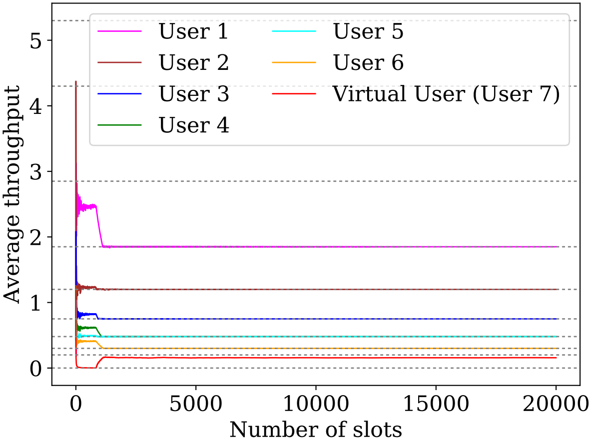


Figure: 6 heterogeneous users, $t_{min} = 100$, $t_{max} = 1000$, $\zeta = 0.9$, $\beta = 1000$, $\epsilon = 0.001$, user $i \in \{1, 2, 3, 4, 5, 6\}$ has an *ON-OFF* channel with *ON* capacity $\frac{15}{i}$ *Mbps*, and *ON* probability 0.5.

Dynamic Rate Throttling

- Alternatively, we could turn on the penalty for user i immediately after its target rate is acquired, i.e., at time T_i. Then, the growth rate of the penalty function $\phi_i(t)$ is given by

$$\beta_i(t) = \begin{cases} 0 & \text{if } t \le T_i \\ \beta & \text{otherwise} \end{cases}$$

Dynamic Rate Throttling

- Alternatively, we could turn on the penalty for user i immediately after its target rate is acquired, i.e., at time T_i. Then, the growth rate of the penalty function $\phi_i(t)$ is given by

$$\beta_i(t) = \begin{cases} 0 & \text{if } t \leq T_i \\ \beta & \text{otherwise} \end{cases}$$

- Even in this scheduler, the virtual user is enabled only after all the users have acquired their target rates, i.e., at time $T = \max_{1 \leq i \leq n} T_i$.

Dynamic Rate Throttling

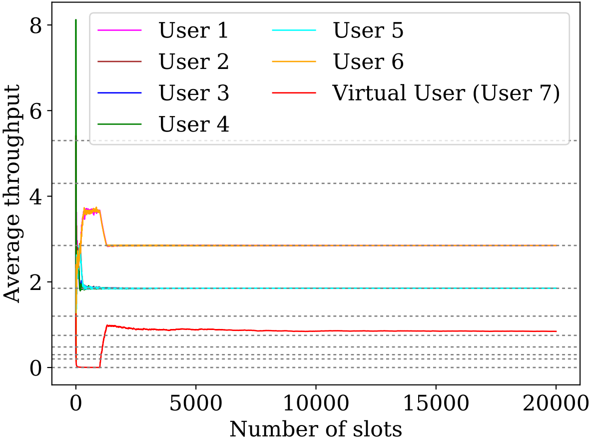


Figure: 6 homogeneous users, $t_{min} = 100$, $t_{max} = 1000$, $\zeta = 0.9$, $\beta = 1000$, $\epsilon = 0.001$, each user has an *ON-OFF* channel with *ON* capacity 15 *Mbps*, and *ON* probability 0.5.

Dynamic Rate Throttling

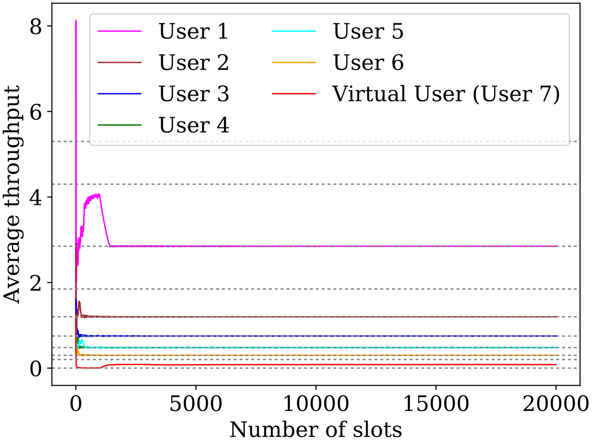


Figure: 6 heterogeneous users, $t_{min} = 100$, $t_{max} = 1000$, $\zeta = 0.9$, $\beta = 1000$, $\epsilon = 0.001$, user $i \in \{1, 2, 3, 4, 5, 6\}$ has an *ON-OFF* channel with *ON* capacity $\frac{15}{i}$ *Mbps*, and *ON* probability 0.5.

Allocating Multiple Resource Blocks

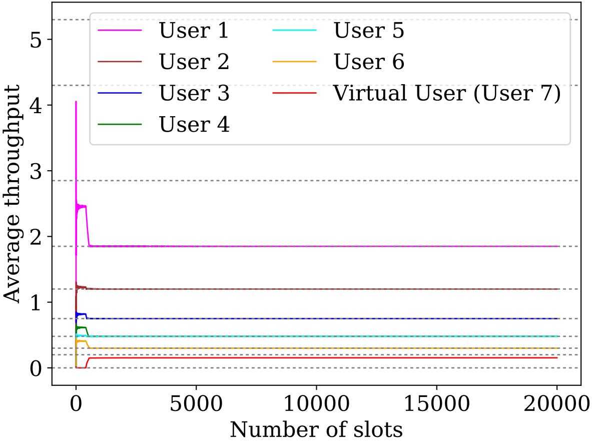


Figure: 6 heterogeneous users, $t_{min} = 100$, $t_{max} = 1000$, $\zeta = 0.9$, $\beta = 1000$, $\epsilon = 0.001$, 100 resource block, user $i \in \{1, 2, 3, 4, 5, 6\}$ has an *ON-OFF* channel with *ON* capacity $\frac{0.15}{i}$ *Mbps*, and *ON* probability 0.5 in each each resource block.

Allocating Multiple Resource Blocks

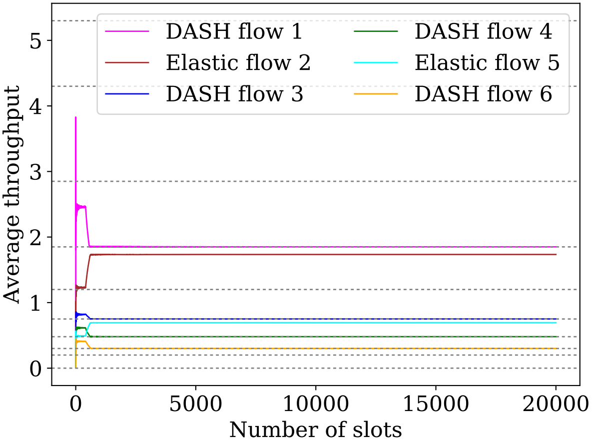


Figure: 4 heterogeneous users with DASH flows, 2 heterogeneous users with elastic flows, user $i \in \{1, 2, 3, 4, 5, 6\}$ has an *ON-OFF* channel with *ON* capacity $\frac{0.15}{i}$ *Mbps*, and *ON* probability 0.5 in each resource block.

Thank You

Albert Sunny et al., "Enforcing Bitrate-Stability for Adaptive Streaming Traffic in Cellular Networks," in *IEEE Transactions on Network and Service Management*, vol. 16, no. 4, pp. 1812-1825, Dec. 2019.