Multicast–Aware Proactive Caching in Wireless Networks with Deep Reinforcement Learning

Samuel O. Somuyiwa, András György and Deniz Gündüz
Department of Electrical and Electronic Engineering,
Imperial College London, UK

***Abstract*—We consider mobile users randomly requesting contents from a single dynamic content library. A random number of contents are added to the library at every time instant and each content has a lifetime, after which it becomes irrelevant to the users, and a class-specific request probability with which a user may request it. Multiple requests for a single content are served through a common multicast transmission. Contents can also be proactively stored, before they are requested, in finite capacity cache memories at the user equipment. Any time a content is transmitted to some users, a cost, which depends on the number of bits transmitted and the channel states of the receiving users at that time instant, is incurred by the system. The goal is to minimize the long term expected average cost. We model the problem as a Markov decision process and propose a deep reinforcement learning (DRL)-based policy to solve it. The DRL-based policy employs the deep deterministic policy gradient method for training to minimize the long term average cost. We evaluate the performance of the proposed scheme in comparison to traditional *reactive* multicast transmission and other multicast-aware caching schemes, and show that the proposed scheme provides significant performance gains.**

I. Introduction

Caching and multicast transmission are two methods that have been proposed to handle some of the challenges associated with growing video content traffic in wireless networks. Caching enables replication of contents at nearby base stations (BS) or even directly at user equipment (UE), which can reduce network congestion [1], [2], energy consumption [3], as well as latency [4]. On the other hand, multicasting enables the delivery of a content to multiple users in a single transmission, resulting in the efficient utilization of the available bandwidth and energy [5]. Muticast transmission is already included in 3GPP standards as evolved Multimedia Broadcast Multicast Service (eMBMS) [6].

Joint implementation of these two solutions promises to bring gains greater than the sum of the two, as the ability to proactively cache contents can create further multicasting opportunities. This has been studied in [7] for reduction of energy cost, in [8] for joint minimization of delay, power and fetching costs, and in [9] for maximization of transmission probability. However, existing works do not consider dynamic content generation or variations in content popularity, which are typical in real settings, e.g., viral contents in social networks or news videos, where contents are continuously generated and remain popular for a limited time duration [10].

In this paper, we consider multiple mobile users randomly requesting contents (video files) from such a dynamic content library, where contents are generated with random *lifetimes*, which denotes the time duration that they remain relevant to at least one user, and each content has a request probability at which it can be independently requested by any user at any time period. Contents are grouped into a finite number of classes, which determines their request probabilities. We assume that the users request a content from the library only once, as is the case in some social network platforms. Therefore, any content that is requested by a user becomes irrelevant to that user. Contents that become irrelevant to all the users are removed from the library.

Each UE is equipped with a cache memory of finite capacity. A content manager (CM) updates the cache memory of each UE by downloading contents in advance of user requests and storing them in the cache memory, and may also remove contents from the cache memory when necessary. We assume that all the contents are of equal size, and each content occupies a unit space in memory. The state of the channel between each UE and the BS varies with time due to several factors, such as user location, coverage, and channel conditions. In particular, we consider the channel state to be the power at which a content has to be transmitted to the UE. A transmission energy cost is incurred any time the BS transmits contents to the UEs. This cost depends on the number of contents downloaded and on the channel states of the users. Whenever a user requests a content that is not already in its cache, it is downloaded at a cost that depends on the current channel state of the user; if the content is already in the user's cache, it is served from the cache at no additional cost.

We formulate the problem as a Markov decision process with side information (MDP-SI) [11], which is a special Markov decision process (MDP) that consists of a controllable state and an uncontrollable state, which consists of the independently and identically distributed (i.i.d) side information that affects the cost. The size of the state and action spaces are very large, making it infeasible to compute an optimal policy. Therefore, we employ policy approximation and deep reinforcement learning (DRL), in particular deep deterministic policy gradient method (DDPG) [12], to train the CM to minimize the long term expected average cost on the system.

This work was partially supported by the European Research Council (ERC) through project BEACON (No. 725731), the European Union's Horizon 2020 Research and Innovation Programme through project SCAVENGE (No. 675891), and the Petroleum Technology Development Fund (PTDF).

Unlike in [11], the complexity of the problem also makes it infeasible to compute a lower-bound to evaluate the performance of the trained CM. Instead, we compare its performance with other multicast-aware caching schemes as well as the traditional *reactive* scheme with multicast transmission.

II. System Model

We consider a slotted time communication system. We denote the set of contents in the library at the beginning of time t by $\mathcal{F}_t$. We assume that a random number of contents, denoted by $M_t \in \{0, \dots, M_{max}\}$, are generated and added to the library at the beginning of each time slot. Each content comes with a lifetime after which it expires and becomes irrelevant to all the users. We denote the lifetime of content m generated at time t by K_t^m, where $1 \leq K_t^m \leq K_{max}$. A content generated at time t with lifetime K_t^m expires by the end of time $t + K_t^m - 1$. Each content in the library belongs to one of Q classes, where a content belonging to class q is requested by any user with probability p_q, independently at each time slot. We assume that the lifetime and class of each content are known to the CM when the content is generated. We also assume that a user requests a content only once. Therefore, any content that has not expired but has been requested by a user becomes irrelevant to that user and can no longer be requested by that user. We also assume that any content that becomes irrelevant to all the users, either because its lifetime expires or because it has already been requested by all the users, is automatically removed from the library. All the remaining contents at time t are called *relevant contents*.

We consider N users in the system, where user $n \in [N] \triangleq \{1, \dots, N\}$ is equipped with a cache memory of capacity B_n where the contents can be stored in a proactive manner. We denote the set of contents in the cache memory of user n at the beginning of time t by $\mathcal{I}_t^n$, where $|\mathcal{I}_t^n| \leq B_n$, and the set of relevant contents that have already been requested by user n by the beginning of time t by $\mathcal{H}_t^n$. To simplify notation, we denote the vector of sets of contents in the cache memory of all the users by $\mathcal{I}_t$, that is, $\mathcal{I}_t = [\mathcal{I}_t^1, \dots, \mathcal{I}_t^N]$, and the vector of sets of requested contents by all the users by $\mathcal{H}_t$, that is, $\mathcal{H}_t = [\mathcal{H}_t^1, \dots, \mathcal{H}_t^N]$. Any content in the library can be uniquely identified by its remaining lifetime, its class, the users that have it in their cache, and the users that have requested it. Let us define a *location flag* for each content as a sequence $\boldsymbol{i}_t \in \{0,1\}^N$, where $\boldsymbol{i}_t(n) = 1, n \in [N]$, implies that user n has the content in its cache at time t, while $\boldsymbol{i}_t(n) = 0$ implies otherwise. Similarly, we define a request vector $\boldsymbol{h}_t \in \{0,1\}^N$, where $\boldsymbol{h}_t(n) = 1$ implies that user n has requested the content and $\boldsymbol{h}_t(n) = 0$ implies otherwise. Given these definitions, any content $f \in \mathcal{F}_t$ that has remaining lifetime L at time t and belongs to class q can be represented by a quadruple $(L, q, \boldsymbol{i}_t, \boldsymbol{h}_t)$.

We denote the set of contents requested by user n at time t by $\mathcal{R}_t^n = (\mathcal{R}_t^{(1,n)}, \mathcal{R}_t^{(2,n)})$, where $\mathcal{R}_t^{(1,n)} \subseteq \mathcal{F}_t \setminus (\mathcal{I}_t^n \cup \mathcal{H}_t^n)$, that is, it is the set of requested contents, which are not in the user's cache memory and have not been requested by the user, and $\mathcal{R}_t^{(2,n)} \subseteq \mathcal{I}_t^n$ is the set of requested contents that are in the user's cache memory. Each user request $\mathcal{R}_t^n$ depends on the class probability p_q of relevant contents and is independent of other user requests. All the contents in $\mathcal{R}_t^{(1,n)}$ are downloaded at time t, for all n, and contents in $\mathcal{R}_t^{(2,n)}$ are moved from the cache memory for the user to consume. We denote the combined request of all the users at time t by $\mathcal{R}_t = [\mathcal{R}^{(1)}, \mathcal{R}^{(2)}]$, where $\mathcal{R}_t^{(1)} = [\mathcal{R}_t^{(1,1)}, \dots, \mathcal{R}_t^{(1,N)}]$ and $\mathcal{R}_t^{(2)} = [\mathcal{R}_t^{(2,1)}, \dots, \mathcal{R}_t^{(2,N)}]$.

The CM can exploit the cache memory at the UEs by pushing contents into them when the wireless links between the UEs and the BS have good quality, in order to reduce the cost. It may also discard contents from cache memories if necessary. We denote the set of contents downloaded by user n at time t by $\mathcal{D}_t^{(1,n)}$, and the set of contents discarded from the cache memory of user n at time t by $\mathcal{D}_t^{(2,n)}$. Note that $\mathcal{D}_t^{(1,n)} \supseteq \mathcal{R}_t^{(1,n)}$ and $\mathcal{D}_t^{(2,n)} \supseteq \mathcal{R}_t^{(2,n)}$. We denote the combined set of contents downloaded and discarded at time t by $\mathcal{D}_t = [\mathcal{D}_t^{(1)}, \mathcal{D}_t^{(2)}]$, where $\mathcal{D}_t^{(1)} = [\mathcal{D}_t^{(1,1)}, \dots, \mathcal{D}_t^{(1,N)}]$ and $\mathcal{D}_t^{(2)} = [\mathcal{D}_t^{(2,1)}, \dots, \mathcal{D}_t^{(2,N)}]$. The CM can also exploit the capability of the BS to perform multicast transmissions to download a single content to multiple UEs at the same time. Therefore, the CM decides the contents that will be transmitted by the BS as well as the users that will receive the transmitted contents. We denote the set of contents multicast at time t by $\mathcal{D}_t' \triangleq \bigcup_{n=1}^N \mathcal{D}_t^{(1,n)}$. We denote the cost for transmitting content $j \in \mathcal{D}_t'$ at time t by ε_t^j.

The cost at time t depends on the channel states (power at which a content has to be transmitted to the user) of the users at that time. We denote the channel state of user n at time t by C_t^n, and the vector of channel states of all the users at time t by $\mathcal{C}_t = [C_t^1, \dots, C_t^N]$. For all n and t, C_t^n is an independent realization of a random variable $C \in \mathbb{R}^+$ with cumulative distribution function (cdf) $P_C(c)$ and is bounded by C_{max}. Assuming that the BS multicasts content $j \in \mathcal{D}_t'$ at time t to users in set $\mathcal{U}$, $|\mathcal{U}| \leq N$, the associated cost is

$$\varepsilon_t^j = \max_{n \in \mathcal{U}} C_t^n, \tag{1}$$

and the total cost of transmitting all the contents in $\mathcal{D}_t'$ is

$$\mu_t = \sum_{j \in \mathcal{D}_t'} \varepsilon_t^j. \tag{2}$$

We assume that the sequences $\{K_t^m\}$, $\{M_t\}$, and $\{\mathcal{C}_t\}$ are independent. Based on the above description, the problem can be formulated as an MDP as described in the next section.

III. Problem Formulation

The CM aims to exploit proactive caching and mutlicast transmission jointly to reduce the transmission cost incurred by the BS in the long run. Therefore, based on (2), its goal is to minimize the long-term expected average cost defined as

$$\rho \triangleq \limsup_{T \to \infty} \mathbb{E}\left[\frac{1}{T}\sum_{t=1}^{T} \mu_t\right]. \tag{3}$$

We formulate the problem as an MDP-SI, which is characterized by the quintuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mu, P_Z)$, where $\mathcal{S} = (\hat{\mathcal{S}}, \mathcal{Z})$

is the state space, which consists of a controllable state $\hat{S}_t \in \hat{\mathcal{S}}$ and an uncontrollable state $Z_t \in \mathcal{Z}$. In particular, Z_t is a sequence of i.i.d side information with cdf P_Z that is independent of the actions of the agent but affects the cost function. $\mathcal{A}$ is the action space, $\mu : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is the cost function and $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ is the probability kernel. Assuming that the actions of a learning agent are governed by a policy $\pi : \mathcal{S} \rightarrow \mathcal{A}$, and we let Π denote the set of all such policies. The goal in the MDP-SI is to find the optimal policy $\pi \in \Pi$ that minimizes the infinite horizon average cost $\rho = \lim_{T\rightarrow\infty} \mathbb{E}\left[\frac{1}{T}\sum_{t=1}^{T} \mu(S_t, A_t) \middle| S_1\right]$. We denote the infinite horizon average cost when $A(S_t) = \pi(S_t)$ by ρ^π. The differential value function for any state $s \in \mathcal{S}$ is given as

$$V^\pi(s) = \mathbb{E}\left[\sum_{t=1}^{\infty}(\mu(S_t, \pi(S_t)) - \rho^\pi) \middle| S_1 = s\right] . \quad (4)$$

For a single user, we show in [11] that under certain conditions, the optimal policy $\pi^*(s)$, for any $s \in \mathcal{S}$, which minimizes ρ^π and satisfies

$$V^{\pi^*}(s) = \min_{a\in\mathcal{A}} \left\{ \mu(s, a) - \rho^{\pi*} + \sum_{s'\in\mathcal{S}} P(s'|s, a)V^{\pi^*}(s') \right\}, \quad (5)$$

with $a = \pi^*(s)$ minimizing the expression on the right hand side, exists and that the policy $\pi^*(s)$ is a piecewise constant function.

For this problem, the controllable state $\hat{s} \in \hat{\mathcal{S}}$ at time t is $\hat{S}_t = (\mathcal{I}_t, \mathcal{H}_t, \mathcal{R}_t)$ consisting of the combined set of contents in users' caches, the combined set of relevant contents already requested by users, and the combined set of instantaneous user requests. The uncontrollable state $z \in \mathcal{Z}$ at time t is $Z_t = \mathcal{C}_t$, that is, it is the vector of channel states, which are the i.i.d side information that affect the cost. The action $a \in \mathcal{A}$ at time t is $A_t = \mathcal{D}_t$, that is, it is the combined sets of downloaded and discarded contents.

The controllable state transitions are

$$\mathcal{I}_{t+1}^n = \left(\mathcal{I}_t^n \setminus \mathcal{D}_t^{2,n}\right) \cup \left(\mathcal{D}_t^{1,n} \setminus \mathcal{R}_t^{1,n}\right), \quad (6a)$$

$$\mathcal{H}_{t+1}^n = \mathcal{H}_t^n \cup \mathcal{R}_t^{1,n} \cup \mathcal{R}_t^{2,n}, \quad (6b)$$

with $\mathcal{R}_{t+1}^{(1,n)}$ and $\mathcal{R}_{t+1}^{(2,n)}$ transitioning with respect to request probabilities $\{p_q\}_{q=1}^{Q}$ of contents in $\mathcal{F}_{t+1}$, for all $n = 1, \ldots, N$. The uncontrollable state transitions are governed by the cdf $P_C(c)$. The cost function $\mu(S_t, A_t)$ is given in (2).

Computing the optimal policy for this problem is not feasible due to the complexity of the problem and the curse of dimensionality. Therefore, we employ policy gradient method with function approximation [13], which uses a parameterized function to approximate and represent the policy, and allows a learning agent to directly search the parameter space.

IV. Policy Gradient Reinforcement Learning

We denote a policy parameterized with $\boldsymbol{\theta}$ by $\pi_{\boldsymbol{\theta}}$. The objective is to find the optimal parameter vector $\boldsymbol{\theta}^*$ that minimizes the expected average cost $\rho^{\pi_{\boldsymbol{\theta}}}$ along the infinite trajectory $\tau_{\boldsymbol{\theta}} = (S_1, A_1, \mu_1, S_2, A_2, \mu_2, \ldots,)$ drawn from the trajectory distribution $P_{\boldsymbol{\theta}}$ following policy $\pi_{\boldsymbol{\theta}}$. Let $\boldsymbol{\theta}_i$ denote the estimated vector at the end of step i. We employ stochastic gradient descent, and iteratively update the estimated parameter vector in the direction of the policy gradient $\nabla_{\boldsymbol{\theta}}\rho^{\pi_{\boldsymbol{\theta}}}$, as follows: $\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i - \lambda\nabla_{\boldsymbol{\theta}}\rho^{\pi_{\boldsymbol{\theta}}}$, where $\lambda > 0$ is the step size.

We let $J(\tau) = \limsup_{T\rightarrow\infty} \frac{1}{T}\sum_{t=1}^{T} \mu(S_t, A_t)$. Then the gradient can be estimated as

$$\nabla_{\boldsymbol{\theta}}\rho^{\pi_{\boldsymbol{\theta}}} \approx \int \nabla_{\boldsymbol{\theta}} P_{\boldsymbol{\theta}}(\tau) J_{\pi_{\boldsymbol{\theta}}}(\tau) d\tau,$$
$$\approx \mathbb{E}[\nabla_{\boldsymbol{\theta}} \log P_{\boldsymbol{\theta}}(\tau) J(\tau)],$$

where the expectation with respect to $P_{\boldsymbol{\theta}}$ can be approximated over sampled trajectories of finite length without the knowledge of $P_{\boldsymbol{\theta}}$. Assuming that $\pi_{\boldsymbol{\theta}}$ is a stochastic policy, the gradient can be estimated as

$$\nabla_{\boldsymbol{\theta}}\rho^{\pi_{\boldsymbol{\theta}}} \approx \mathbb{E}\left[\sum_{t=1}^{T} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(A_t|S_t) J(\tau)\right].$$

This gradient can be written as [13]

$$\nabla_{\boldsymbol{\theta}}\rho^{\pi_{\boldsymbol{\theta}}} = \mathbb{E}\left[\sum_{t=1}^{T} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(A_t|S_t) Q^{\pi_{\boldsymbol{\theta}}}(S_t, A_t)\right], \quad (7)$$

where $Q^{\pi_{\boldsymbol{\theta}}}(s, a)$ is the state-action value function, which is given as

$$Q^{\pi_{\boldsymbol{\theta}}}(s, a) = \mathbb{E}\left[\sum_{t=1}^{\infty}(\mu(S_t, A_t) - \rho^{\pi_{\boldsymbol{\theta}}}) \middle| S_1 = s, A_1 = a\right]. \quad (8)$$

It is shown in [14] that a deterministic policy gradient is the limiting case of the stochastic policy gradient in (7) as the variance of the policy tends to zero. If $\pi_{\boldsymbol{\theta}}$ is a deterministic policy, the gradient in (7) can be written as

$$\nabla_{\boldsymbol{\theta}}\rho^{\pi_{\boldsymbol{\theta}}} = \mathbb{E}\left[\sum_{t=1}^{T} \nabla_{\boldsymbol{\theta}}\pi_{\boldsymbol{\theta}}(S_t)\nabla_a Q^{\pi_{\boldsymbol{\theta}}}(S_t, a)\big|_{a=\pi_{\boldsymbol{\theta}}(S_t)}\right]. \quad (9)$$

The state-action value function $Q^{\pi_{\boldsymbol{\theta}}}(s, a)$ can be estimated by using temporal difference learning methods, e.g., actor-critic [15], which can be implemented using a function approximator $Q_{\mathbf{w}}(s, a)$ with parameter $\mathbf{w}$.

Due to the large size of the state and action spaces, we will employ neural networks as function approximators, and the DDPG method [12] to train the policy. DDPG uses off-policy actor-critic temporal difference learning to train the actor $\pi_{\boldsymbol{\theta}}(s)$. It learns the critic $Q_{\mathbf{w}}(s, a)$ through *experience-replay* by storing and sampling mini-batches of the tuple $(S_i, A_i, \mu_i, S_{i+1})$, obtained in step i, from a *replay memory*, and optimising parameter $\mathbf{w}$ by minimising the mean square error loss between a *target value* $\mu_i + \gamma Q_{\mathbf{w}}(S_{i+1}, \pi_{\boldsymbol{\theta}}(S_{i+1}))$, where $\gamma \in (0, 1]$ is a discount factor, and the value $Q_{\mathbf{w}}(S_i, A_i)$ in a supervised learning approach. To improve the stability of learning the actor and the critic networks, separate networks $Q'_{\mathbf{w}'}(s, a)$ and $\pi'_{\boldsymbol{\theta}'}(s)$ are used for the target values. In an initial step, the weights of the actor and critic networks are assigned to the *target networks*, then in the subsequent steps when the actor and critic networks are updated, the weights of the target networks are slowly updated as follows

$$\begin{aligned} \boldsymbol{\theta}' &\leftarrow \beta\boldsymbol{\theta} + (1-\beta)\boldsymbol{\theta}', \\ \mathbf{w}' &\leftarrow \beta\mathbf{w} + (1-\beta)\mathbf{w}', \end{aligned} \tag{10}$$

where $\beta \ll 1$ is a hyperparameter. Action exploration is performed independently from the learning algorithm by sampling noise from a random process and adding it to action $a = \pi_{\boldsymbol{\theta}}(s)$. We will use a discount factor close to 1 to approximate the average cost performance. We describe the learning algorithm in the next section.

V. Learning Algorithm

In addition to the fact that the state and action spaces are prohibitively large, the size of the content library also varies over time. This implies that the sizes of the state and action spaces vary over time. However, a deep neural network (DNN) must be designed to have a fixed number of neurons/units in each layer. To handle this, we extract a fixed number of features from the contents in the library at every time instant to represent the state (input layer of the DNN) of the actor and critic networks. The action $A_t = \mathcal{D}_t$ in time t consists of contents downloaded to the UEs in the wireless network and contents discarded from the cache memory of users. All user requests at time t, that is, $\mathcal{R}_t$, must be delivered at that time, and are inevitably included in action A_t. Therefore, the action of the CM governed by a policy involves selection of contents to cache, contents to discard from caches, and the users whose caches will be updated with these contents.

To make its caching decisions, the CM assigns *power level* to each content in the library, such that only the users whose channel states are below the assigned power level of a transmitted content can receive it. That is, assigning zero power level implies that the content is not transmitted to any user. To handle this, the output layer of the actor network, which is also the second input layer (action input) of the critic network, is a k-dimensional *weight* vector $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_k]$. The CM extracts a k-dimensional content-specific feature vector $\boldsymbol{x}^f = [x_1, \ldots, x_k]$ from each content $f \in \mathcal{F}$, using the content identifier $(L, q, \boldsymbol{i}, \boldsymbol{h})$, and computes the power level for each content as an inner product of the two vectors. That is, for a content $f \in \mathcal{F}_t$ at time t, its power level is

$$r_t^f = \boldsymbol{\alpha}_t^{\intercal} \boldsymbol{x}_t^f. \tag{11}$$

Note that estimation of the power levels is done *globally*, that is, all the users and contents are considered, whereas cache update is done *locally*, using the assigned power levels, for each user considering the contents relevant to the user. We describe the content ordering and caching strategy of the CM for any user $n = 1, \ldots, N$ and time t in Algorithm 1. We assume that an empty space in the cache memory of user n contains a fictitious content with assigned power level 0, such that discarding the content means discarding no content at all.

Algorithm 1 Cache Update Strategy

```
Input: $\mathcal{G}_1 \leftarrow \mathcal{F}_t \setminus (\mathcal{H}_t^n \cup \mathcal{I}_t^n \cup \mathcal{R}_t^{(1,n)})$ and $\mathcal{G}_2 \leftarrow \mathcal{I}_t^n \setminus \mathcal{R}_t^{(2,n)}$
Obtain $b \leftarrow \min\{|\mathcal{G}_1|, B_n\}$ and initialize $b' \leftarrow 1$
Obtain ordered sets $\{j_1, \ldots, j_b\} \subseteq \mathcal{G}_1$ w.r.t $r_t^{j_1} \geq \ldots \geq r_t^{j_b}$,
and $\{k_1, \ldots, k_b\} \subseteq \mathcal{G}_2$ w.r.t $r_t^{k_1} \leq \ldots \leq r_t^{k_b}$
while $b' \leq b$ do
    if $r_t^{j_{b'}} - r_t^{k_{b'}} \geq C_t^n$ then
        Cache content $j_{b'}$ and discard content $k_{b'}$
        $b' \leftarrow b' + 1$
    else
        $b' \leftarrow b + 1$
    end if
end while
```

VI. Numerical Simulations

We evaluate the performance of the proposed DRL-based multicast transmission with proactive caching (MT-PC) scheme in comparison with three baselines described below.

- *Multicast transmissions without caching (MT)*: This is the traditional content delivery scheme in wireless networks. A content requested by multiple users is transmitted in a single multicast stream to the users requesting it.
- *Multicast transmissions with reactive caching (MT-RC)*: This is same as the MT scheme, but with the additional capacity of caching transmitted contents into empty spaces in the cache memories of users who have not requested the contents but whose channel states allow them to recieve the content anyway.
- *Multicast transmissions with priority-based reactive caching (MT-PRC)*: Modified MT-RC scheme, where contents are cached and removed from caches, based on a priority ordering. Denoting the *value* of a content with remaining lifetime L and class-request probability p_q as $\mathcal{V}_L^q$, we estimate the *value* of each content as $\mathcal{V}_L^q = p_q \cdot \mathbb{E}\left[C\right] + (1-p_q)\mathbb{E}\left[\min\{C, \mathcal{V}_{L-1}^q\}\right]$ and keep contents with the highest *values* in the cache.

For simulations, M_t is drawn uniformly at random from the set $\{1, 2, \ldots, 8\}$, and the lifetime K_t^m of each content $m \in \{1, 2, \ldots M_t\}$ is drawn uniformly at random from the set $\{5, 10, 15\}$. We consider $Q = 3$ classes of contents, with request probabilities $[p_1, p_2, p_3] = [0.06, 0.12, 0.18]$, and each generated content is allocated one of the classes uniformly at random. We obtain C_t using Shannon's capacity formula, and we select parameters consistent with the LTE network model and 3GPP channel model [16]. We normalise C_t so that $C_{max} = 1$.

We use 2 hidden layers with 400 and 200 units, respectively, for the actor and critic networks, and set the target update parameter $\beta = 10^{-5}$, discount factor $\gamma = 0.99$, and learning rates 10^{-6} and 10^{-5} for the actor and critic networks, respectively. We train with mini-batch size of 64 and *replay memory* of size 10^4. For action exploration, we sample exploration noise from a Gaussian distribution with mean 0 and variance 0.001, and in each training episode h, we add exploration noise to the *weight* vector $\boldsymbol{\alpha}_i$, obtained in step i, with probability δ_h.

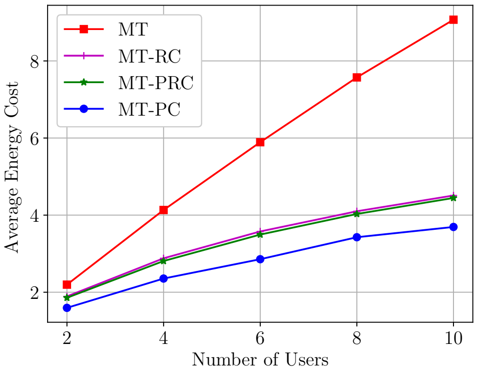

Fig. 1: Average energy cost vs. number of users with $B_n = 50$ for any user n.

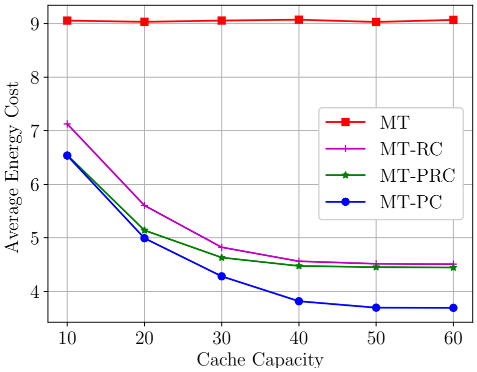

Fig. 2: Average energy cost vs. cache capacity with $N = 10$.

We update the probability as $\delta_{h+1} = \max\{0, \delta_h - 0.1\}$. The number of extracted state features is $3QK_{max} + N + 1$, and the number of content features is 4.

In Figure 1 we plot the average energy cost with respect to the number of users N. As expected, the average energy cost increases with N; however, the proposed MT-PC scheme has the lowest energy cost among all the schemes considered. Moreover, the gap between MT-PC and MT-PRC also increases with N. MT-PC has a performance gain of 14% over MT-PRC and 27% over MT with $N = 2$ users, increasing to 18% and 59%, respectively, with $N = 10$ users. In Figure 2 we plot the average energy cost with respect to the cache capacity. As expected, the performance of the MT scheme does not depend on the cache capacity as it does not utilize the cache memories, while the other schemes improve with increasing cache capacity before saturation. MT-PRC performs much better than MT-RC when the cache capacity is low (less than 40) because it stores contents that are more likely to be requested in the cache. The proposed MT-PC scheme performs close to MT-PRC when the cache capacity is low (below 10), but outperforms it as cache capacity increases, with a performance gain that increases as cache capacity increases, showing that DRL allows us to learn to cache the contents that are likely to be decoded while creating and exploiting the multicasting opportunities as much as possible.

VII. Conclusions

We considered the problem of multicast transmission together with proactive caching of contents into limited capacity cache memories of multiple mobile users. We consider users randomly requesting contents from a dynamic content library, which are served by a BS over a time-varying wireless channel. In this model, proactive caching provides energy savings due to two separate potential gains: It allows transmission of contents at better channel conditions, and also allows multicasting to multiple users, benefiting from the broadcast advantage. In order to minimize the long term average energy cost, a CM must decide which contents to transmit to which users at each time slot, and which content to keep in each user's cache. We proposed a solution based on DRL to tackle the complexity of the problem. We showed that the proposed scheme outperforms baseline multicast transmission and caching schemes.

References

[1] P. Blasco and D. Gündüz, "Multi-armed bandit optimization of cache content in wireless infostation networks," in *2014 IEEE International Symposium on Information Theory*, June 2014, pp. 51–55.

[2] S. Muller, O. Atan, M. van der Schaar, and A. Klein, "Context-aware proactive content caching with service differentiation in wireless networks," *IEEE Trans. on Wireless Comms.*, vol. 16, no. 2, Feb 2017.

[3] M. Gregori, J. Gomez-Vilardebo, J. Matamoros, and D. Gündüz, "Wireless content caching for small cell and D2D networks," *IEEE Journal on Sel. Areas in Comms.*, vol. 34, no. 5, pp. 1222–1234, May 2016.

[4] M. S. ElBamby, M. Bennis, W. Saad, and M. Latva-aho, "Content-aware user clustering and caching in wireless small cell networks," in *Int'l Symp. on Wireless Comms. Systems (ISWCS)*, Aug 2014, pp. 945–949.

[5] C. Huang, J. Zhang, H. V. Poor, and S. Cui, "Delay-energy tradeoff in multicast scheduling for green cellular systems," *IEEE Journal on Sel. Areas in Comms.*, vol. 34, no. 5, pp. 1235–1249, May 2016.

[6] 3rd Generation Partnership Project (3GPP). (2016). [Online]. Available: http://www.3gpp.org/specifications/releases/71-release-9

[7] K. Poularakis, G. Iosifidis, V. Sourlas, and L. Tassiulas, "Exploiting caching and multicast for 5G wireless networks," *IEEE Trans. on Wireless Comms.*, vol. 15, no. 4, pp. 2995–3007, April 2016.

[8] B. Zhou, Y. Cui, and M. Tao, "Optimal dynamic multicast scheduling for cache-enabled content-centric wireless networks," *IEEE Transactions on Communications*, vol. 65, no. 7, pp. 2956–2970, July 2017.

[9] Y. Cui and D. Jiang, "Analysis and optimization of caching and multicasting in large-scale cache-enabled heterogeneous wireless networks," *IEEE Trans. on Wireless Comms.*, vol. 16, no. 1, pp. 250–264, Jan 2017.

[10] A. Lobzhanidze, W. Zeng, P. Gentry, and A. Taylor, "Mainstream media vs. social media for trending topic prediction - an experimental study," in *IEEE Consumer Comms. and Netw. Conf.*, Jan 2013, pp. 729–732.

[11] S. O. Somuyiwa, A. György, and D. Gündüz, "A reinforcement-learning approach to proactive caching in wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 7, pp. 1–14, July 2018.

[12] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *CoRR*, vol. abs/1509.02971, 2015.

[13] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Advances in Neural Info. Proc. Sys. 13 (NIPS)*, 2000, p. 1057–1063.

[14] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *Proceedings of the 31st Int'l Conf. on Machine Learning (ICML)*, 2014, p. 387–395.

[15] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 1998.

[16] T36.814 V9.0.0, "Further advancements for E-UTRA physical layer aspects (release 9)," *3GPP*, Mar. 2010.