Facoltà di Ingegneria dell'Informazione,
Informatica e Statistica

Corso di Laurea Magistrale in
Ingegneria delle Comunicazioni

ENERGY EFFICIENT ONLINE PROACTIVE CACHING FOR MOBILE WIRELESS NETWORKS

ADVISORS:
PROF.SSA MARIA-GABRIELLA DI BENEDETTO

DR. DENIZ GUNDUZ

AUTHOR:
CARMINE BRANCATI
CID: 1560592

Anno Accademico 2014-2015

Ai miei genitori, che mi hanno
sempre sostenuto e spinto a dare
il meglio.

ACKNOWLEDGEMENTS

CONTENTS

1. INTRODUCTION

The exponential growth of the Internet, in the last few years, has completely changed the global population's behaviour. Within the last decades, we have witnessed a tremendous evolution in communication's technologies, in which the mobile computing systems and networks have been the real protagonists. Many new communication solutions have been developed, from the first WAP (wireless access protocol) to the most recent wireless communication protocols (like 4G and expected 5G) which guarantee high-speed data transmission for all mobile devices.

The new mobile communication systems, supported by the worldwide increase in smartphone and tablets usage, have opened the possibility of being constantly connected to the Internet. The mobile systems performance enhancement has led to the development of upgraded mobile data platforms and applications. Indeed, through a high data rate leverage, delivered by the last generation networks, it has been possible to offer to users new services and applications such as video streaming and social networking.

According to the latest estimates, in figure 1.1, the traffic data growth seems to be endless, owing to the possibility to have a constant access to news, email, video content and other

personal data, anywhere and anytime. The next 5 years forecast predicts an increase in tenfold for the data traffic and in twofold for mobile networks speed. This incredible phenomenon is strongly supported by the unstoppable diffusion of hand-held devices. Every day new smarter and more sophisticated devices become an integral part of our daily lives. It has been estimated that in 2020 the number of this kind of handset will be greater than the earth's population [1]. The rapid proliferation of smartphones and tablets is also due to the high performance delivered: everyone seems to have the world in his/her hand and it inevitably leads customers to desire more and more. New technologies create new desires and new services. For example, larger and high-definition display opened up to the consumers the chance to take advantages of services like video sharing and video-on-demand.

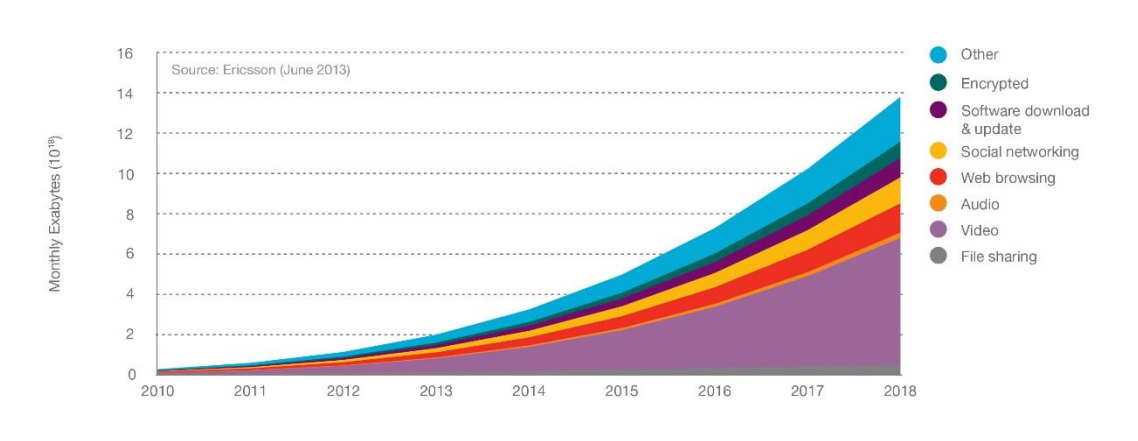

Figura 1-1 Mobile data traffic by application [3]

In fact, the video traffic represents the highest percentage of mobile data and the forecast predicts a further increase over the coming three years.

The advent of smartphones and, in turn, the increasing interest in video contents, yielded to the birth of a huge number of apps, generating the need of good throughput and latency performance; the first has to guarantee no freezing effects during the streaming and the second is extremely important for real-time service. This led to a new communication

degree, which may pledge good performance to the huge number of devices, in terms of latency, data rates, energy consumption and throughput [3].

In order to manage the new data burden, without completely overthrowing the network infrastructure, the mobile network operators (MNOs) are developing new communication systems based on the deployment of small cells, changing the topology of the traditional cellular network. The small cell network is made of different low-power small base stations comparable to many access points.

In order to get along with the growing of data traffic and to meet the ubiquitous user's requirements, especially in crowded areas, the aim of the mobile operators is to deploy cells of different sizes (micro, pico, femto) able to improve coverage through re-usage of the resources, and, at the same time, to offload the macro cells directly connected to the backhaul. The cooperation among macro cell and small cells has led to the generation of heterogeneous network with higher performance, in particular in terms of bit rates, for the final user [6].

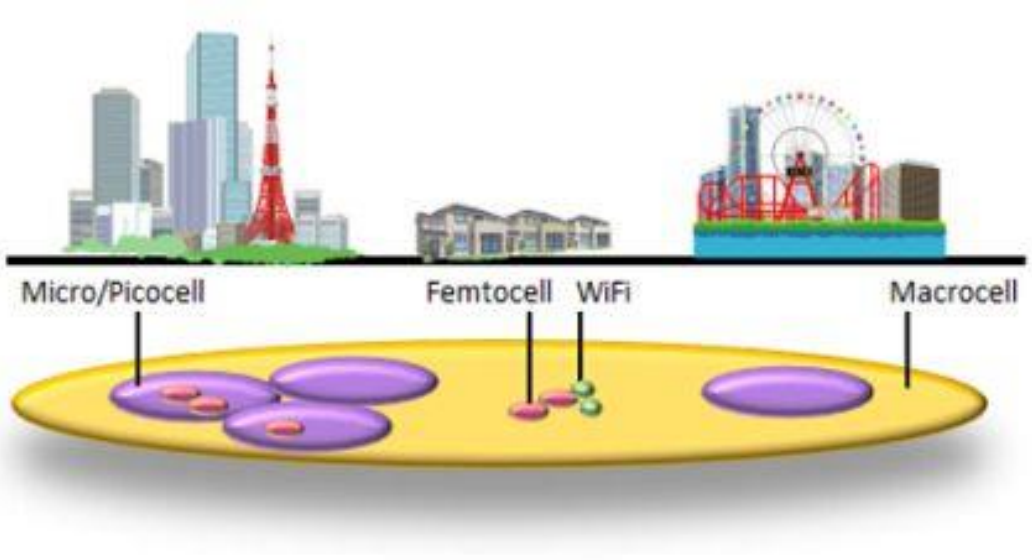

Figura 1-2: Example of Heterogeneous network [http://www.telecomabc.com/h/hetnet.html]

The main objective of heterogeneous networks, is not only to unburden the backhaul but also to get the data closer to the final user. In order to achieve this, every access point has in it a cache memory that is able to store relevant contents. This opportunity results to be very appealing, particularly in an area where an important event is taking place, and a lot of people demand for similar contents. In this case, most of the contents will be already stored in the small cell, avoiding a very high number of requests to the mobile network [4].

1.1 Caching

By leveraging on the evolution of the memory and storage technologies achieved in the last decades, *caching* may represent a promising solution to support the increasing data demand in mobile wireless networks. The use of the memory cache, which is mainly used in computers technology, has the main purpose to support the functions of the CPU, by prefetching instructions that could be quickly and frequently accessed through the application software. This initial idea has been extended to the browser cache, that stores the information related to the most frequently accessed web platforms, providing a quicker access afterwards.

Beside to browser caching, a copy of the requested files is stored in a cache memory on the client device in order to speed up the page downloading. Caching can also significantly reduce communication costs also in terms of energy, thereby improving the efficiency of the accesses to the networks. This concept of caching, applied to the small-cell world, could represent the solution to speed up network access to data files.

A cache memory installed in a small-cell base station functions more efficiently if provided with an intelligent system that is capable of analysing the traffic flow and predicting future requests.

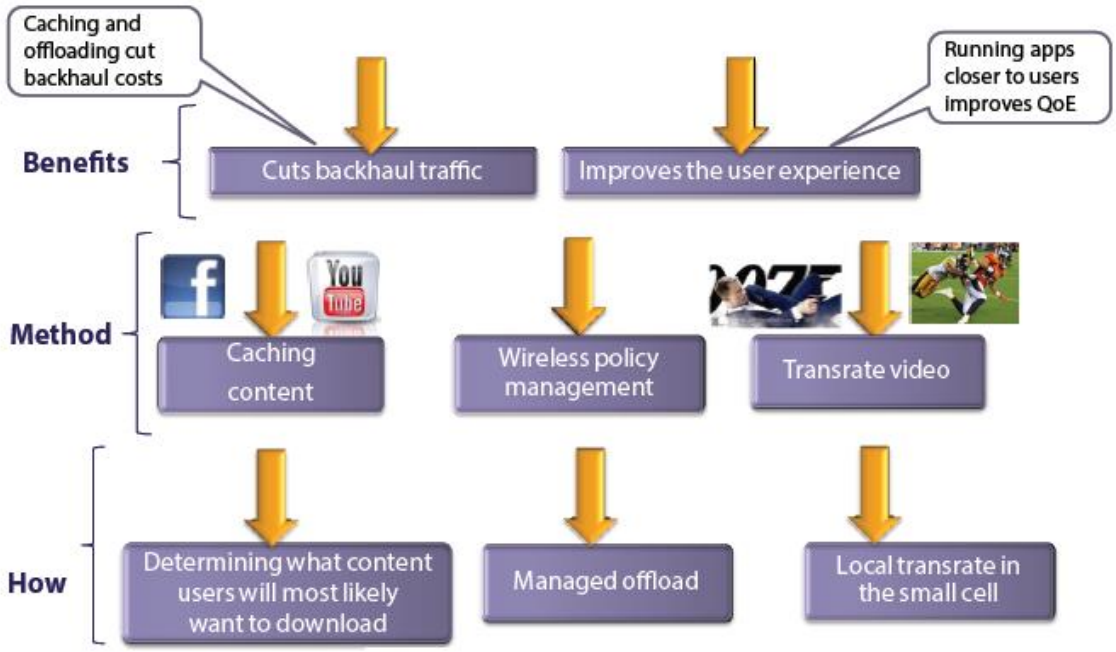

Figura 1-3: Intelligence in a small cell [http://www.jdsu.com/ProductLiterature/smallcellhetnet_wp_nsd_tm_ae.pdf]

In this new environment, the old *reactive* paradigm, which immediately served the impatient users, is replaced with the new strategy of *proactive caching*, whose objective is to store the data *in advance*.

As a demand arrives, the application simply fetches the information from the internal memory, avoiding the access to the wireless network.

Afterwards, through analysing the data flow, a base station is able to create a traffic pattern towards the terminal users, and store the appropriate contents before the user demand comes.

1.2 OSNs' Impact on Data Traffic

A big portion of mobile data is represented by the OSNs, according to the latest data about the increase in data traffic [1]. The dissemination of information over platforms, like Facebook, Twitter etc, has achieved a very high degree. As soon as the social networks came to light, each user began sharing and spreading a large amount of data.

Furthermore, thanks to the astonishing technological evolution of latest devices, furnished with several communications technology (NFC, GPS, etc), it has been possible to track the activity of the user, not only in terms of possible future demands, but also in terms of mobility. This has made the human behaviour to be more predictable [8]. Online social network are an integral part of our lives. This is cconfirmed by the latest statistics, which declare that the 31% of the Facebook users use a smartphone or a tablet to enter their personal profile [9].

These new hand-held devices offer high performance and a lot of features but they have to deal with a limited battery life. The issue of energy consumption is very important for the success of handheld devices. For example, during video streaming, the user requires good network performance and good channel quality in order to avoid loss of packet and consequently the freezing effect. It is not always feasible to keep a high standard in data flow, anywhere and anytime, and sometimes a significantly higher energy consumption is required.

Decreasing the energy consumption of smartphones and tablets is one of the most important challenges. The need for ubiquitous connection has to deal with two main factors: the inevitable variations in network conditions and the demand for high performance during peak hours. Especially in crowded places, demand for high data rate content leads not only to a decay of the data flow, but also the consumption of more power.

In this work, the aforementioned problem of the energy consumption is tackled with a similar caching approach developed for the small-cell-networks, but directly at the user's devices. The high predictability of the users' action plays a key role, because it gives the possibility to anticipate his/her the future requests. Apparently, the knowledge of the future demands in terms of contents seems not to be enough, due to the variability of the mobile network.

Finally, it is essential also to associate a user's mobility pattern to his predictable information, as it would be helpful to draw an accurate profile of the channel conditions. In a mobile environment, the wireless channel condition plays an important role because a communication over a low quality channel could lead to a greater consumption of energy by the user's device.

1.3 Energy Efficient Proactive Caching for mobile device

To wholly exploit the predictability of the final user and minimize the energy consumption, the *Proactive Caching* for mobile wireless networks will been analysed in this work.

The idea of *proactive caching* is not merely related to the network and its resources' management, but is applied directly to the user's device with the objective to minimize the energy consumption.

As aforementioned, the battery represents the biggest obstacle to mobile innovation. Even with all the work that has been put into battery development, the basic problem remains that we are trying to make the linear growth of batteries serve the exponentially growing demands for power.

Due to the large amount of energy consumed, mobile devices have to be continually charged. Being often wired, the smartphones are losing their *mobile characteristics.*

For this reason, we need to device methods that use the battery as efficiently as possible.

In order to do this, assuming that the channel statistics of the user is known, based on user's mobility predictability, *the main objective is to select the right moment to download a content in order to "pay the lowest cost" in terms of energy depletion.* This idea finds its actual implementation in an OSN environment.

Every user frequently accesses his profile to watch the latest news about his friends/groups of interest and, in turn, it will be easy to predict which contents the user is interested in. The reason to anticipate the fetching is not related to the content itself, but to the channel condition.

Hence, the aim of *proactive caching* for mobile device is taking as much advantage as possible since the channel condition shows a good condition. It allows to download a content that will be definitely asked in the future by the user, leveraging on the knowledge of the stochastic distribution of the channel and the future user's demand.

The proactivity's benefit is a better efficiency in energy consumption. This will be possible only if the content is cached over the best channel condition: when a transmission requires less energy.

As pointed out in the section caching, the user requests of contents already stored in his cache, can be served directly from the local memory.

1.4 Online Approach

To completely benefit from all the information related to the user's demand and channel state, two different methods proffer possible solutions to the problem [11]:

(i) *Offline algorithm,* in which it is assumed that all the evolution of the process is known *a priori*

(ii) *Online algorithm,* in which the evolution of the process is known only partially or in stochastic distribution terms.

In this work, the online method has been analysed, because the assumption is that only the stochastic model of the processes involved is given. Considering a slotted time, each time the algorithm checks if the user has accessed, checks the quality condition of the channel and, unaware of what is going to happen in the next time slot, makes a decision: *cache or not*. Obviously the choice of an action entails a cost, which in this study represents the energy consumed. The final objective is to choose every time the best action that minimizes the expected cost. In order to minimize the cost, it is important to make the right decision on the basis of the current state of the process and the incurred cost.

As mentioned beforehand, the problem of the proactive caching may be model as a sequential decision making problem, where the system evolve over time, going from one

stage to another. This evolution, from a state to the next, occurs stochastically depending on the algorithm, which choose the right action to take in each stage and at each time, where to each action there is a corresponding immediate cost (or reward).

The proactive caching process is structured this way:

- a set of *states*, which is represented by the different channel condition;
- the *actions*, that the algorithm may choose, are: *{download now, wait};*
- for each action there's a *cost* that depends on the channel condition;
- the transition probability, the random part of the model, which contains for each state the probability to go to another state of the set in the next time slot.

For every time slot, the algorithm can make a decision, only checking the channel and the user's demand in that precise moment. What has been done in the past is irrelevant for the current decision[14].

For this reason, the mathematical model used to treat the proactive caching, is the *Markov Decision Process (MDP).*

These kind of processes are usually handled with *Dynamic Programming (DP),* whose solving approach is iterative, thus making it very suitable for sequential problems like the MDP[14].

Assuming that the user is interested in some contents, the cost, in terms of power, is paid only when the user demands. The objective of this proposal is to find the right moment to cache in order to spend less energy. Every time that the user does not require a file, it is possible to apply proactive caching. This justifies the aforementioned set of actions, in fact the proactive strategy has to choose, in each time slot, whether the current cost is sufficiently low to download or it is better to wait, in anticipation that the user may access in the next time slot at a higher or lower cost as the case may be.

In order to pursue this aim, that is to minimize the cost, the proactive caching process has been approached with a particular case of dynamic programming: the optimal stopping problem. In particular in this work we will show that the optimal stopping problem can be solved through *an optimal threshold policy.*

This means that the algorithm, at each time, decides to cache only if the cost is under a certain threshold, otherwise is better to wait. Obviously the threshold is evaluated dynamically, based on the distribution of the random processes of the system.

1.5 Report Overview

After this short overview about the motivation and the description of the objective, which have led to this work, the literature review will present the most common approaches to the proactive caching and energy-efficiency problems.

The third chapter will provide more information about the Markov Decision Processes and Dynamic Programming.

The forth chapter will present the details about the energy efficient online proactive caching for mobile devices, showing all the analysed approaches and the results obtained.

In the further chapter we will show the experimental results and then the conclusion.

2. LITERATURE REVIEW

In the last years, the wireless traffic has shown a massive growth with an increasing rate of 69 percent within the last year [1]. This tremendous increase in the data traffic has been possible thanks to the simultaneous technological developments in the design of mobile devices, like tablets and smartphones, which permitted the access to Internet contents any time and anywhere enabling people to be constantly connected to the network.

In this mobile environment, the online social networks (OSNs) have found breeding ground for the spread of contents. According to the latest estimates, the percentage of traffic sourced by social networks is second only to the video traffic[1].

There is consensus among technology experts that the demand for more content and data, as well as diffusion of smart devices is going to increase within the next years.

This continuous growth of mobile data has led many researchers to develop new communications techniques to guarantee a wide coverage, better allocation of bandwidth, with high rate and low latency, and above all, high energy efficiency.

These simultaneous and conflicting goals constitute the core challenges in the design of the fifth generation mobile wireless networks [5], which will be a turning point in the communications' world. [5][24].

In order to increase network coverage, and spectral and energy efficiency in the most effective way, the most promising approach is to create small cell employing small base stations (SBS) characterized by self-configuration, rapid deployment and low-cost. To increase the efficiency of content delivery through SBSs, an idea that has gained popularity in recent years is to bring the content as close as possible to the end user[5][24]. This requires caching not only within the core network, but also at the network edge, that is, at the SBSs. In this thesis, we go one step further and consider content caching at the user devices.

Obviously, for content caching at user devices, we need to change completely the networking approach, replacing the *reactive* paradigm with a new way to use the network resources smartly.

Therefore, the objective has been to introduce a *proactive caching* approach, which, knowing in advance the user's future requests, allows the network to manage user resources in the most efficient manner.

For example, the network can push a content to a wireless device in advance when the the channel conditions are better, and when the request for that data actually comes, the information is fetched from the memory instead of accessing the network. This reduces not only the energy cost of downloading content, but also the outage probability and latency.

In [25] the attention is focused on the small cell networks, consisting of SBSs equipped with high storage capacity and limited backhaul link. In this environment by predicting

the user's behaviour, it has been shown that a better performance for the cache algorithm can be obtained through proactive caching when the amount of requests increases.

The idea to store contents at the edge of networks to guarantee a better connectivity has been analysed in [27]-[28]. In [28] *distributed caching helpers*, with a large storage memory, have been used to cache the high predictable data, such as traffic video, which requiring high transmission rates, can be easily managed. While the backhaul has to replace the helper's content with a lower rate, according with the temporal evolution of the user's demand. The goal has been to minimize the expected download time, finding the optimal policy to serve the helpers.

Hence, caching data at the edge of a network generates a decrease in the backhaul load, particularly tangible when special events, like a football match, occur and multiple users wants to download the same content. In [26] the locally caching contents is faced with a *transfer learning approach.* The T-L paradigm exploit the user's features, taken from the source domain, to minimize the backhaul load.

In addition to the SBS, the users can also share the content thanks to the device to device (D2D) communications, by which close mobile users have the possibility to communicate directly without resorting to the cellular infrastructure [5]-[24]. Hence, it is possible to offload some of the network traffic by exploiting the D2D connectivity between users that are in the same area [29]-[30]. This paradigm finds ever more applications with the spreading of the OSNs. OSNs (like Facebook, Twitter etc.), nowadays, have achieved a leading position for the information distribution, due to the continuous sharing of data among friends and people in the same group. At the same time, smartphones and other mobile devices provide regular access to service, increasing the mobile traffic. Thus, in [29] an algorithm has been developed, which proactively *seeds* the content to the final

user anticipating his requests, leveraging the mobile-to-mobile link and sensitive information about the members available on the social network. Furthermore, in [29] they have analysed two different cases: (i) *offline case* where the information diffusion on OSN is given; (ii) *online case* where the knowledge of the information cascade is stochastic.

A study about offline policy has been carried out in [31]. Assuming that the user's demands and the channel characteristics, the optimal transmission policy is characterized in order to minimize the energy depletion. The user's requests are satisfied in advance respect to the demand extending the transmission time and downloading with a good channel state.

In [22] an offloading algorithm learns the most popular contents in an online way and store them in order to manage efficiently the cache at an access point when multiple users are served.

In [20] they faced the problem of power allocation for the single queue of a time-varying channel in order to minimize the queueing delay. Under the constraints on power, they defined the optimal power allocation policy as a threshold-policy, which transmits at peak power given a channel state according with the queue length.

In [21] the authors provide an optimal policy to allocate power and transmission rate in order to minimize the mean buffer delay, considering a single user that transmits to a base station over a fading channel. The problem was formulated as a Markov decision process.

The predictable behaviour of the mobile users, in terms of both interests and mobility, led the authors of the paper [18] to present a proactive scheduling strategy for the services providers. The predictable nature of the requests is exploited to serve the user's demands during the light-loaded hours resulting in a significant cost reduction.

Moreover, the human behaviour is not only predictable in terms of contents and requests, but also in terms of mobility. Nowadays users are more mobile than ever. According with this mobility, the quality of the connectivity changes. Thus, in [19] the authors developed a model that track the user's movements and for each position saves the related network conditions. Based on these information, it is possible to predict the quality of the connection, given the current user position. Forecasting mobile connectivity let some applications manage the future user's requests in a smart way, improving the performance or reducing the power depletion of the terminal device [19].

The growing of data traffic has led the developing of techniques to offload the traffic from cellular networks to Wi-Fi. In [23], assuming to know the user's mobility distribution and the related coverage, the authors present an offloading scheme, which minimize the total data usage payment, taking in account the quality of service.

As described in this section, proactive caching is utilized to minimize the load of the backhaul, to improve network performance, to minimize latency and to optimize power allocation for an access point.

In this work, we exploit proactive caching in order to minimize the energy consumption of a mobile device. The main idea of our online approach is to download data when the channel conditions require the least energy consumption.

3. MARKOV DECISION PROCESSES

In this chapter, we present an overview of the Markov Decision Processes and some the most common methods of solving them.

3.1 Markov decision processes (MDPs)

The Markov Decision Processes represents a mathematical definition for the sequential decision making problems when the evolution of the problem is not known *a priori* [14].

Indeed, the outcomes are systematically evaluated on the observation of the system and its random future evolution. For this reason, the MDPs are also known as *stochastic control problems or stochastic dynamic programming* [14].

The MDP theory was studied by Bellman for the first time in 1950. Over the last decades, these mathematical concepts has been developed and now provide the best theoretical model for a wide spectrum of problems.

Nowadays, the MDPs find practical applications in diverse fields as economics, engineering, medicine and biology.

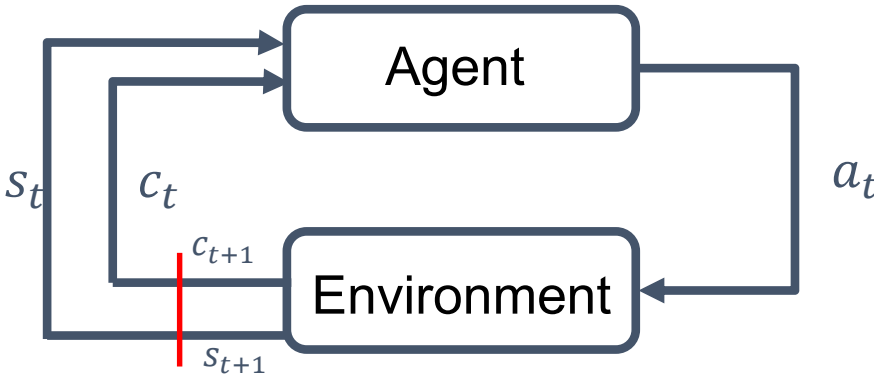

Figura 3-1: MDP scheme

In figure 3.1 we provide a symbolic representation of an MDP. At a certain time t, the agent, which represents the decision maker, takes an action based on the observation of the environment's state s_t. For each action chosen, the agent incurs an immediate cost c_t (or gains an immediate reward), while the environment reaches a new state s_{t+1} according to the probability distribution of the states.

Hence, an MDP consists of the following characteristics:

i. A set of decision time-slots – The decisions are made over a sequence of discrete time-slots $t = \{0,1,2 \dots T\}$. If $T < \infty$ the process is defined as a *finite horizon* problem, otherwise if $T = \infty$ it is defined an *infinite horizon* problem.

ii. A set of system's state – At each slot-time t, the system is in a *state* s_t. S represents the set of all the system's states.

iii. A set of actions – At each slot-time t, the decision maker observes the state s_t and chooses an action a_s from the actions' set A_s, which contains all the actions admissible for the state s_t.

iv. A set of immediate rewards (or costs) – Assuming the system's state s_t and the chosen action a_s, the decision maker will receive an immediate reward $r_t(s,a)$ $[c_t(s,a)]$. Hence, the immediate reward at time t depends only on the current state s_t and the action a_s taken, but not on the previous states and actions.

v. The transition probability matrix – It represents the probability $p_t(s'|s,a)$ to go from the state s_t to the state s'_{t+1}, making the decision a_s.

Hence, a MDP can be described as $\big(S, A, R(r|s,a), P(s'|s,a)\big)$.

For a Markov Decision Process, the action taken, the related rewards (costs) and the transitions probabilities depend only on the present state and decision. Hence, given the current state the past and the future are independent.

$$\boldsymbol{P\big(S_{t+1}=s'|S_t=s_t, A_t=a_t, S_{t-1}=s_{t-1}, A_{t-1}=a_{t-1}, \ldots S_0=s_0\big) = P(S_{t+1}=s'|S_t=s_t,\ A_t=a_t)}$$

In a given system, the *objective* of the decision maker is ***to optimize the system's performance*** by choosing the right *decision rule* d_t for each state at every time step. The sequence of decision rules determines the policy π. Implementing a policy implies a sequence of rewards. Given the current state, the agent's aim is to find the policy that maximize the sequence of expected rewards. This expectation represents the value function $V^{\pi}(s)$ that measures the performance of the system, starting from a certain state s following the policy π [14].

$$V^{\pi}(s) = E_{\pi}\{\, R(r|s)\} = E_{\pi}\left\{\sum_{t=1}^{N-1} r_t(s,a)\right\}$$

3.2 Solving Markov Decision Process-Finite horizon

The dynamic programming (DP) methods represent the best way to solve a Markov decision problem. (In this section the description of the mathematical technique will be done considering the minimization of the cost function.)

The DP provides iterative solving algorithms to find the best actions at each stages. The optimal action, for the current state and time, is the one that minimizes the sum of current cost and total expected cost, as result of the action taken by the agent.

Given $V^{\pi}(s)$ defined above, it is possible to introduce the state-action function that evaluate the expected cost starting from a state s_t, taking action a_t, following the policy π onwards [16].

$$Q^{\pi}(s,a) = E_{\pi}\left\{\sum_{t=1}^{N-1} r_t(s,a) \mid s_t = s,\ \ a_t = a\right\}$$

For any policy π and state s the value-function can be defined through the Bellman equation (1). This states that the expected value of the state s is given by the current cost and the values of the further states weighted by the transition probabilities[16]:

$$V^{\pi}(s) = \sum_{s'} P(s'|s,\pi(s))[C(s'|s,a) + V^{\pi}(s')] \qquad (1)$$

The MDP's main goal is to find the optimal policy π^* which lead to the minimum expected cost. Such as:

$$V^{\pi^*}(s) \leq V^{\pi}(s) \qquad \forall\, s \in S$$

$$V^*(s) = \min_{a \in A} \sum_{s'} P(s,a,s')[C(s,a,s') + V^*(s')] \qquad (2)$$

The expression (2) represents the Bellman optimality equation. Given the optimal policy, the cost of a state is the return of the best action chosen in that state [16] [14].

Hence, computing the optimal policy π^* it is possible to solve Markov decision processes. Assuming the full observability of the problem, the Bellman equations provide a method to find the optimal policy and compute the value function.

The main iterative Dynamic Programming solving algorithms are: *policy iteration* and *value iteration*.

3.2.1 POLICY ITERATION

The policy iteration is performed in two phases: *policy evaluation phase* and *policy improvement phase*. In the first step, a generic policy π is chosen and $V^{\pi}$is computed[16].

$$V^{\pi}(s) = \sum_{s'} P(s'|s, \pi(s))[C(s'|s, \pi(s)) + V^{\pi}(s')]$$

Based on this information, the policy improvement look for actions that improve the value $V^{\pi}(s)$ for the generic state s with the following function:

$$Q^{\pi}(s, a) = \sum_{s'} P(s'|s, a))[C(s'|s, a) + V^{\pi}(s')]$$

If the some actions will present improvements, they will replace the previous ones and a better policy π' will be the input of the next evaluation phase.

$$\pi'(s) = arg \min_{a} Q^{\pi}(s, a) = arg \min_{a} \sum_{s'} P(s'|s, \pi(s))[C(s'|s, \pi(s)) + V^{\pi}(s')]$$

When this phase does not produce improvements, it means that the current policy already satisfies the optimal Bellman equations. Hence, for a finite MDP the policy iterations converges after a finite number of steps [16]-[17].

{Policy Evaluation}

repeat

$\epsilon = 0$

for each $s \in S$ **do**

v=$V^{\pi}(s)$

$V(s) = \sum_{s'} P(s'|s, \pi(s))[C(s'|s, \pi(s)) + V^{\pi}(s')]$

$\epsilon = \max(\epsilon, |v - V(s)|)$

until $\epsilon < \sigma$

{Policy Evaluation}

Policy-stable=true

for each $s \in S$ **do**

$b = \pi(s)$

$$\pi(s) = \min_{a} \sum_{s'} P(s'|s, \pi(s))[C(s'|s, \pi(s)) + V^{\pi}(s')]$$

if $b \neq \pi(s)$ **then** policy-stable=false

if policy-stable **then** stop; else go to Policy Evaluation

Algoritmo 3-1: Policy Iteration

3.2.2 VALUE ITERATION

The value iteration is a dynamic programming algorithm that computes the optimal policy starting from the evaluation of the value function. In this case, the algorithm simply iteratively updates the Bellman equations:

$$V_t(s) = \min_{a} \sum_{s'} P(s, a, s')[C(s, a, s') + V_{t+1}(s')] \qquad t = N-1, N-2, \dots 0$$

Starting from the last stage, with V_T known for each state $s \in S$, the algorithm updates the value for each state s, at any time t.

Hence, the optimal value is updated step by step with successive approximations until convergence, as shown in the Algorithm 2.

random initialization of V

repeat

$\epsilon = 0$

for each $s \in S$ **do**

v=$V(s)$

for each $a \in A(s)$ **do**

$V(s) = \min_a \sum_{s'} P(s'|s,a)[C(s'|s,a) + V(s')]$

$\epsilon = \max(\epsilon, |v - V(s)|)$

until $\epsilon < \sigma$

Algoritmo 3-2: Policy Iteration

4. System Model and Optimization Problem

Before defining the details of the mathematical model, a real application for proactive caching will be presented in this chapter in order to introduce and motivate our approach.

4.1 Motivation

Nowadays, online social networks are an integral part of our lives. This is stated by the latest statistics which declare that more and more users use a mobile device to access social network pages, in particular, 31% of Facebook users access their personal accounts only through smartphones or tablets [9].

Number of accesses to Facebook is estimated to be around 14 times per day, and at each access one user downloads posts, photos, videos and other contents published by groups and friends.

Given that the latest statistics presented by Facebook have underlined a significant increase in video views, the amount of energy depleted by mobile devices due to this kind of high data rate applications is getting bigger and bigger [12].

A growing number of people depend on Facebook to get general information; this has pushed the social network's developers to create algorithms which filter the type of contents interesting for every single user. The News Feed, main page of Facebook platform, shows a personalized profile for each account thanks to a learning algorithm which selects and updates merely business pages, public figures, sports and best friends which the user is connected to.

The real objective of the News Feed is, not only to select the right content, but also to choose the right moment to show these contents. Hence, the algorithm updates every time the News Feed, showing in the top of the page the most relevant and the most recent contents [13]. Therefore, as soon as the user *pushes* the app icon on his device, he immediately will see the most *fresh* posts, according to the number of friends and pages of interest, the upper part of the page will be constantly updated while the stale posts will drop over time.

Following the prediction concept mentioned above, *proactive caching* is applicable for mobile access to social network content, because the contents that will be of interest to the user are selected by the filters, and can be delivered to the user in advance.

Assuming a content with a certain *freshness time (period of time that the post remains high-up on the News-Feed page, before it becomes stale)* and taking into account both user's access probability and channel state distribution, the problem is *to choose the best moment to download the content in order to minimize the device's power consumption*. Therefore, the goal of this work is to identify the most suitable strategy aimed to solve

this issue. Once the file has been cached in the internal memory, the future demands by the user will retrieve that content from the memory avoiding to pay the cost to access the network. Note that proactive caching not only reduce the energy consumption, but may also reduce the latency of content accesses.

4.2 System Model

The model is based on a discretization of time in slots $n \in \{1,2,\dots,N\}\ with\ N < \infty$, all slots have the same duration. At any time-slot n , k new contents, generated by the

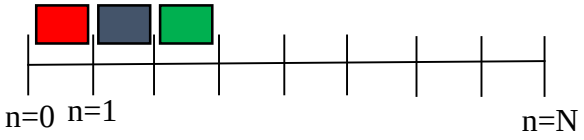

Figure 4-1: Generation of k files per time slot

most relevant friends, appear on the News Feed page and each of them has *freshness time of N time slots.* Hence, the News-Feed algorithm chooses k files as the most important for the user, and these files remain relevant (fresh) for the next N time-steps.

At a certain time n, the user accesses the NewsFeed page, and requests the files downloading them on his device. As a consequence, over the whole time frame N, the number of contents cached will be *N*k.* For each access, the user incurs cost of downloading contents.

This cost, in our model, is defined in terms of power depleted and depends on the channel condition during the data transmission.

Indeed, during each epoch n, the channel power gain, defined as $|h_n|^2$, remains constants; h_n is distributed according to a generic *random process*.

Hence, given a fixed transmission rate R, the amount of power depleted for the transmission of data, at time n depends only on the channel condition $|h_n|^2$.

According to the Shannon Capacity Function:

$$C = W \log_2(1 + P_n|h_n|^2)$$

C represents the maximum transmission rate achievable with a small error-probability, depending on the modulation scheme, constellation and medium access control scheme used. All these variables impact on the value of C. But assuming, somehow, that the schemes used approach this upper bound, the behaviour of the model could be:

$$R = W \log_2(1 + P_n|h_n|^2)$$

Hence, assuming to transfer anytime, with a fixed rate R and a fixed bandwidth W, the data-packet the function becomes:

$$P|h|^2 = 2^R - 1$$

which states that the amount of power depleted for each transmission depends on the channel condition.

In particular for each time-step n, defining the ***channel cost*** $c_n \in C \in \Re^+$in terms of power consumed for the condition of the channel as $c_n = \frac{1}{|h_n|^2}$, then:

$$\frac{P}{cost} = 2^R - 1$$

That is, if the channel gets better, the cost for the transmission is lower, hence it is possible to transmit at the same rate with a low power, vice versa when the channel condition gives a higher cost, the power consumption increases.

In order to access his own Facebook account, each user can access only ones per time slot, for this reason the *user access random process* $u_n \in \{A, N\}$, *assumed to be a two-state Markov chain*, where $u_n = A$ means user accesses while $u_n = N$ means user accesses *does not access* his profile in that time slot.

The ***user's demand*** $u_n \in \{A, N\}$ is given by the following matrix:

$$P = \begin{bmatrix} p_{AA} & p_{AN} \\ p_{NA} & p_{NN} \end{bmatrix}$$

where each element of the matrix represent the probability

$$p_{ij} = \Pr\{u_n = j | u_{n-1} = i\}, \qquad i, j \in \{A, N\}, \qquad p_{ij} \in [0,1]$$

Hence, the whole analysis is structured in a *sequential manner* over a *finite horizon* of N epochs. Every epoch, the user may access his page with a certain probability and the cost to download the content changes according to the model described above.

Therefore, given the distribution of the user requests and cost, it is possible to define the ***state of system*** x_n for each time-slot n, as the combination of the cost c_n and the user access u_n: $x_n(u_n, c_n)$.

The objective of this work is to find out a strategy that is able to minimize the average power consumption over the life-time of a content, based on the channel state and the user access probability during the next N time slots.

In order to pursue this goal we proactively push the contents to the mobile device memory, exploiting the stochastic knowledge of the future channel cost and user's accesses, which represent the system's evolution. Hence, when the user will access the application, the contents will be retrieved from the internal memory avoiding to pay the channel cost.

The strategy, that we want to define in this work, has the main role to decide, at each time slot, whether to download or not according to the random processes involved

In order to do that, the algorithm has to choose among two possible ***actions***:

$a_n \in \{download, wait\}$.

Note that, in our model, there are exactly *N*k* contents that are of interest to the user at each time, and we assume that in the mobile device there is enough space to store all those contents. Hence, memory space does not constitute a bottleneck in terms of storage. Therefore, without loss of optimality, it is possible to focus the attention on just one single content per time-slot in order to minimize the power consumed during download.

The goal is to minimize the average cost per content:

$$\text{minimize} \quad \mathrm{E}_{u,c}\left\{c_N(x_n) + \sum_{n=0}^{N-1} c_n(x_n(u_n, c_n), a_n)\right\} \qquad \forall\, n = 0,1,..N-1$$

Once a file has been downloaded, no more actions can be taken on it therefore the process terminates.

Hence, for each content, the system evolves in this manner:

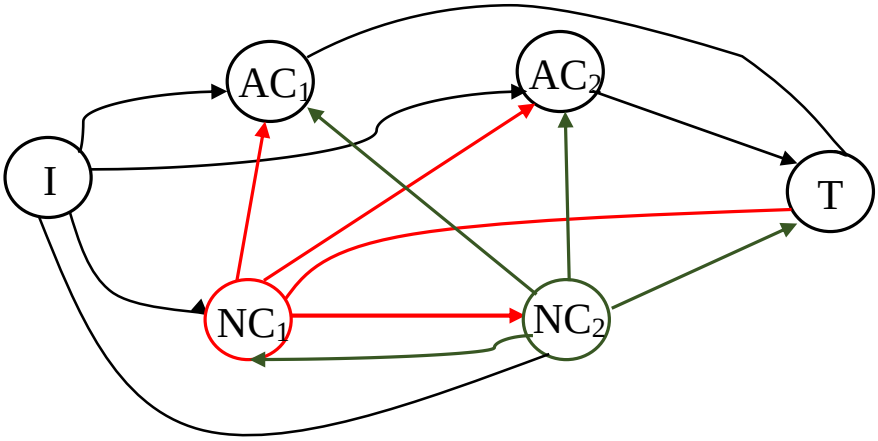

Figure 4-2: System states for a two-state channel

In figure 4.3 the system states for a two-states channel is shown. Given an initial state I for the process, which could represent the previous user's state (Access or Not Access), the system evolves according to the probability distributions of user access and cost.

In this example, if the initial state is that the user accessed, the system reaches:

the state $x(A, c_1)$ with probability $p_{AA} * p_{c1}$; the state $x(A, c_2)$ with probability $p_{AA} * p_{c2}$; the state $x(N, c_1)$ with probability $p_{AN} * p_{c1}$; the state $x(N, c_2)$ with probability $p_{AN} * p_{c2}$. If the system goes in an access state the system pay the cost and terminates.

If the system goes in a state $x(N, c)$, the system could evolve in the next state to a not access state or to an access state according to the user's demand transition matrix.

To sum up, a new content appears on the News Feed page with a specific life time of N epochs. Per each of those, the channel condition, which determines the cost of downloading content, varies following on an *independent and identically-distributed random process.* Another external random process affects the evolution of the system: *the user's demand random process*. This distribution rules all the process as it describes how likely the user is to access his page. As soon as the demand comes, the expense to download the content will be the channel cost related to that specific time, otherwise if the content was already stored the user simply watches it.

As soon as the user watches the files cached, the internal memory will be cleared because the user is usually more interested in the most recent contents. Hence an old content has no reason to stay in the memory.

When the cache is full, the update rule erase from the memory the least-recently saved.

Hence, it is possible describe the evolution of the system over a long period of time with the following scheme:

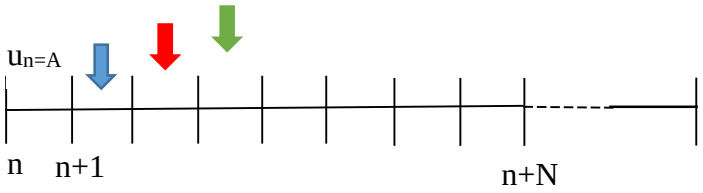

Figure 4-3: Content caching per time slot

Observing the figure 4.3, the arrows represent the generation of the single content. Assuming that at time n, the user accessed, erasing the internal cache, the access state represents the initial state for the next content generated at time n+1. If the user is less likely to access after an access state, for the content generated at time n+1 it is possible to do proactive caching. Hence, if the user does not download at time n+1, for the file generated at time n+2, (red arrow in the figure) the probability that the user accesses is higher and then it is less likely to do proactive caching.

These different initial states will affect the evaluation of the average cost. For this reason, we will consider two different types of content: one type generated after an access state $x(A, c)$ and the other type generated after a not access state $x(N, c)$.

This consideration will be analysed in detail in the next solution section.

4.3 Markov Decision Process Formulation

As described in the previous chapter, a *Markov decision process* defines a mathematical structure for the sequential decision making problems when the evolution of the problem is stochastically known.

An *Markov cost process* is characterized by the following elements:

- A finite set of decision slot times t;
- A set S of states s;
- The transition probability matrix P, which defines the probability to go from a state s to s', $P_{s\,s'} = \mathbb{P}[s_{t+1} = s' | s_t = s]$
- C is the cost function, $C = \mathbb{E}[C_{t+1} | s_t = s]$;
- A set of actions A_s that the decision maker can take for a precise state s.

The main property of a MDP is:

$$\mathbb{P}[s_{t+1} | s_t] = \mathbb{P}[s_{t+1} = s' | s_1, s_2, s_3 \dots s_t]$$

The current state s_t capture all the information from the past, then the current state represents a sufficient statistic for the future.

The model described above concerns a sequential discrete-time decision making. This system is not deterministic, because of the uncertainty introduced by the stochastic processes involved. Hence, for a single post, at each epoch the state of the system is given by the user's demand process and channel condition, based on the observation of them, the decision maker, will select an action, which will impact on the system's state at the next time slot.

Hence, the proactive caching problem can be formulated as a finite-horizon sequential decision problem. The file to download has a life time of N time slots, during which the user could access or not and the algorithm can decide to download or not.

The finite ***decision time-slots horizon*****:**

$$n \in \{1,2, \dots, N\} \qquad N \leq \infty$$

As aforementioned, each time slot the condition of the channel vary randomly and so also the cost for the transmission, the ***channel cost*** is $c_n \in C \in \Re^+$, with $c_n = |h_n|^{-2}$; h_n takes values according to a certain random distribution.

The access of the mobile user (MU) follows a Markovian model $u_n \in \{A, N\}$. The access at time n can be derived on the past access pattern of the mobile user. The user download data unconscious of the channel cost.

Given the distribution of the user requests and the cost, it is possible to define the *state space* X which contains all the possible ***system's states*** x. Each state is given by the combination of the random processes involved: $x(U, C) \in X$.

As described in the system model, once the file is downloaded, no more action could be taken about it, hence the system reaches a termination state.

Further, concerning the state space, a *termination state T* has to be added.

$$x(U, C) \in X \cup T$$

The proactive caching, in this case, involves decision making for each content each time slot of its life-time N.

Indeed, for each state x the decision maker, the algorithm in this case, can choose in a set A_x of only two ***actions*** a_x.

$$A_x = \begin{cases} \{download, wait\} & x \neq T \\ \{wait\} & x = T \end{cases}$$

When the system is in the terminal state the only action available is wait, when the user accesses he force the system to download.

Per each action a, taken in a certain state s at time n, corresponds a ***cost*** c_n**,** such that:

$$c_n(x,a) = \begin{cases} 0 & x = T \ or \ \ x = x(N,C), a = \ wait \\ c_n(x) & x \neq T \ and \ a = download \\ \infty & x = x(A,C), \ \ a = wait \end{cases} \qquad \forall n \in N$$

A non-zero cost occurs only when moving from $x_n \ to \ T$, and this cost is equal to the cost of the channel at time n. The cost for the action wait is infinite, since the only action available for the state $x(A,C)$ is download.

The system evolves according with all these parameters descripted above. In particular:

$$x_{n+1} = f_n(x_n, u_n, c_n, a_n)$$

The random evolution of the system in a ***transition probability matrix*** is given by $p_n(x'|x,a)$, which is the probability that the system will go into state $x'(u',c')$ in the next time slot if the action a is taken at state $x(U,C)$.

$$p_n(x'|x,a) = \begin{cases} p_n(x'|x) & x = x(N,C), \ \ x' \in X, \ \ a = \ wait \\ 1 & x = X, \ \ x' = T, \ \ a = download \ or \ x = x' = T \\ 0 & otherwise \end{cases}$$

Since the user access is independent of the channel condition and action a,

$$p_n(x'|x,a) = p[(u',c')|(u,c),a] = p(u'|u)p(c'|(u,c),a)$$

All these objects described above

$$\{N, X, A_s, p_n(\cdot\,|x,a), c_n(\cdot\,|x,a) : n \in N, s \in S, a \in A_s\}$$

are properly the same characteristics which define a Markov decision problem.

Moreover, at each time step, the action to choose are two: {*download now*, *wait*}.

If the action chosen is *download*, the post is cached and the mobile device incurs the cost in terms of power, related to the channel state. In this case, when the download occurs,

the system reach a *termination state*, in which no more decision about that post could be taken anymore.

If the action *wait* is taken, the system evolves to a new state with a certain probability incurring no cost.

Hence, every time the action taken and the eventual cost incurred depend only on the current state. Being at a certain time step in a precise system's state, the decision made is independent of the past because, if the post is cached, no other actions could be taken and, if the channel was good or bad before, it does not impact on the current decision.

Hence, given all the properties that describe our model, we can state that the *efficient energy proactive caching is a Markov decision process.*

Being a MDP, the objective of this work is to find the set *decision rules, i.e. the policy* π which minimizes the *average cost per content*. The sequential character of this problem leads to find the best trade-off between a current low cost and a low future cost.

$$\underset{\pi}{\text{minimize}} \quad \mathbb{E}_{u,c}^{\pi}[c_n(x_n(u_n, c_n), a_n)]$$

This problem could be seen as a particular example of MDP: *Optimal stopping problem*. The objective of an optimal stopping problem is to find the stopping rule that minimize the cost. This objective is pursued observing the system's state each time slot and based on this the algorithm has to define if stop observing or continue. When the process is stopped a cost is paid. Such problem involves the presence of an action which lead to a *termination state* of the system. The aim of this strategy is to find the right moment to terminate the process incurring the lowest cost.

This kind of MDP describes at best our problem. In our optimal stopping problem, the objective is to minimize the cost to download a content, choosing the right epoch over a

finite planning horizon N. In each time slot, given a channel cost, the algorithm has to decide whether to download and stop the system or to wait for the next time-slot. The cost at any time is a random variable whose distribution is known.

4.4 Dynamic Programming solution

The *dynamic programming (DP)* offers the best solution to the Markov decision problem, since it solves complex problem in a recursive manner exploiting the *Bellman "Principle of Optimality".* At each step, the decision is made evaluating the current cost and the expected one in the next steps. The DP technique evaluate the optimal decision rule proceeding step by step, from the smallest sub-problem to the complete problem.

The problem presented in the previous section could be generically solved through the dynamic programming algorithm that solve the Bellman's equation with a value iteration:

$$V(x_n) = \min_{a_n} \left\{ C_n(x,a,j) + \sum_{j \in S} p_n(j|x,a)[V_n(j)] \right\} \qquad (1)$$

$V_n(x_n)$ is the optimal cost function starting in the state x at time n, having taken action a acting optimally [14].

The optimal equation (1) can also be written:

$$V(x_n) = \min_{a_n} \{Q_n(x,a)\}$$

$$Q_n = C_n(x,a,j) + \sum_{j \in S} p_n(j|x,a)[V_n(j)]$$

Where the function Q_n calculate the consequence of the action a in the state x.

To solve this kind of problem, it is possible to implement a backward algorithm in time, which calculate, starting from the last stage N, step by step the expected cost with respect to the transitions probability matrix. The algorithm used is the *value iteration method (described in the previous chapter)* which evaluate the cost function sequentially at any time for each state.

As any Mdp, this problem can be solved numerically with dynamic programming. In this way only the result is given without any other information about the solution.

As aforementioned, this problem could also be described as an *optimal stopping problem* whose objective is to choose at which states to stop in order to minimize the expected cost over the finite horizon N, given by:

$$\mathrm{E}_{u_n c_n}\left\{c_N(x_n) + \sum_{n=0}^{N-1} c_n(x_n(u_n, c_n), a_n)\right\} \qquad \forall\, n = 0,1,..N-1$$

Hence the optimal cost function will be:

$$J_n(x_n) = \min\left[c_n,\ \ E\left[J_{n+1}\left(x_{n+1}(u,c)\right)\right]\right] \quad (2)$$

In this case, c_n is the cost resulting from the action *download* when the system is in state x_n, the $E\left[J_{n+1}\left(x_{n+1}(u,c)\right)\right]$ is the expected cost corresponding to the action *wait.*

The dynamic programming represents the main technique to find the best solution for an optimal control problem [14].

The previous formula could be better explained, considering that the expectation is a sum weighted by the state transition probabilities $p_{ij}(a) = P(x_{n+1} = j | x_n = i, a_n = a)$

$$J_n(i) = \min\left[c_n,\ \sum_{j \in S} p_{ij} J_{n+1}(j)\right] \qquad (3)$$

Furthermore, there is an optimal policy which defines state by state the best action to take in order to minimize the cost.

In order to solve the optimal stopping problem and evaluate the optimal policy π, the optimal cost $J_n(x_n)$ has to be calculated. Assuming to know the cost at last time slot N

$$J_N(x_N) = c_N(x_N)$$

Given:

$$Q(i) = \sum_{j \in S} p_{ij} J_{n+1}(j)$$

The equation (3) becomes:

$$J_n(i) = \min[c_n,\ \ Q(i)]$$

Hence the optimal stopping policy, for a certain state x, can be defined:

$$\mu(x) = \begin{cases} download, & if\ \ c_n \le Q(x) \\ wait, & otherwise. \end{cases}$$

Excluding the termination state, the time N is reachable only if the user has never accessed in the previous slots, i.e. for all the states $x_n(NA, C)\ \ \forall\, n = 0,1 \ldots N-1$.

Further, assuming that in the last stage the life time of the post is terminating, the only possible cost may be due to the user access. Thus:

$$J_N(x_N) = p_{ij} c_N(x_N)$$

With $p_{ij} = P(j = x_N(A,C) | i = x_{N-1}(NA,C))$

Applying this result in (1), and evaluating for the stage N-1, the result is the following:

$$J_{N-1}(x) = p_{ij} c_{N-1} + p_{ii} \min\left[c_{N-1},\ E[J_N(x_N)]\right] \geq J_N(x_N)$$

Continuing in the same way, it is possible to see

$$J_n(x) \geq J_{n+1}(x)$$

Then:

$$Q_n \geq Q_{n+1}$$

This means that the optimal policy is a threshold-policy.

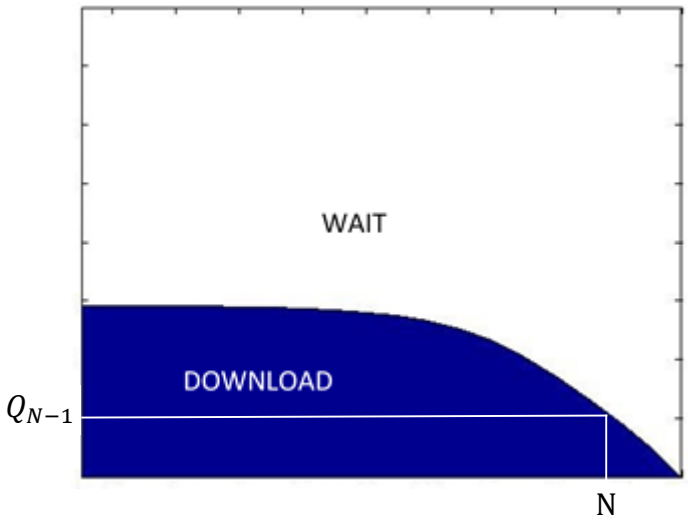

Figure 4-4: Threshold policy example

In this way, an optimal strategy has been found that tells when it is possible download and when not. So for each stage, it is enough to find the optimal threshold in terms of channel state and compare it with the current state. As shown in figure 4.4, if the cost is in the blue zone the optimal action is download, otherwise it is better wait. This solution represents an alternative to the numerical solution offered by dynamic programming.

As described in the system model section, we can define two types of contents:

(i) one type is generated after an access state $x(A, C)$; (ii) the other is generated after a not access state $x(N, C)$.

Assuming that the user's states follow the Markov chain:

$$P = \begin{bmatrix} p_{AA} & p_{AN} \\ p_{NA} & p_{NN} \end{bmatrix}$$

Looking at a long period of time, it can be interesting to find the steady state distribution of the chain $\pi = P\pi$. The vector $\pi = [\pi_A \;\; \pi_N]$ contains the probability that the user access π_A and the user does not access π_N.

Therefore, a solution, to evaluate the *average cost per content*, is to weight the cost obtained from the initial state $x(A, C)$ with π_A and the cost obtained from the initial state $x(N, C)$ with π_N .

In this way, for example, assuming a steady state $\pi = [0.1 \;\; 0.9]$, the 10% of the files will be generated starting from an access-state, the 90% from a not-access state.

The average total cost will be:

$$J = \pi_A * J\big(x(A, C)\big) + \pi_N * J(x(N, C))$$

This will be showed numerically in the next chapter with the experimental results.

5. EXPERIMENTAL RESULTS

In this chapter, some numerical results are presented in order to evaluate the performance of the optimal algorithm described in details in the previous chapter.

The main parameters of the system that might change from user to user are:

1. The life-time of the contents which depends on a lot of variables analysed by the NewsFeed algorithm on Facebook;
2. The number of channel states and the distribution of the cost;
3. The distribution of the user's accesses.

As pointed out in [15], a post on Facebook is considered to be *fresh* for a period of 2 hours on average. Hence, considering time-slots of 10 minutes for the first simulation, the life-time will be $N = 12$.

As mentioned in the previous section, *System model*, the channel cost function at time n is $1/|h_n|^2$. For all the results shown in this section, h_n will be distributed according to an *i.i.d. Rayleigh function*, since it best approximates the characteristic of the wireless signal.

$$f(h) = \frac{h}{\sigma^2} e^{-\frac{h^2}{2\sigma^2}} \qquad h \geq 0$$

The *Rayleigh* distribution, with the probability dense function defined in the previous expression, is commonly used to characterize the statistical time varying nature of the wireless fading signal.

Further, since the problem has been solved with dynamic programming, and it only functions with a finite number of states, we consider a *discretized Rayleigh distribution*.

The discretization of the channel distribution depends on the number of the states.

Assuming that the number of states is k, the *cumulative distribution function (CDF)* $F(h)$, is divided in k equal probabilities like in the figure below.

Each k^{th} interval on the y-axe represents the following segment of CDF:

$$\frac{l-1}{k} < F(h) \leq \frac{l}{k} \qquad l = 1,2, \dots k$$

For each of them, we consider h_l as the average of all the h values that respect the previous expression. In this manner, the cost function assumes values $c_k \quad 0 \leq k \leq \infty$ with equal probabilities $\frac{1}{k}$.

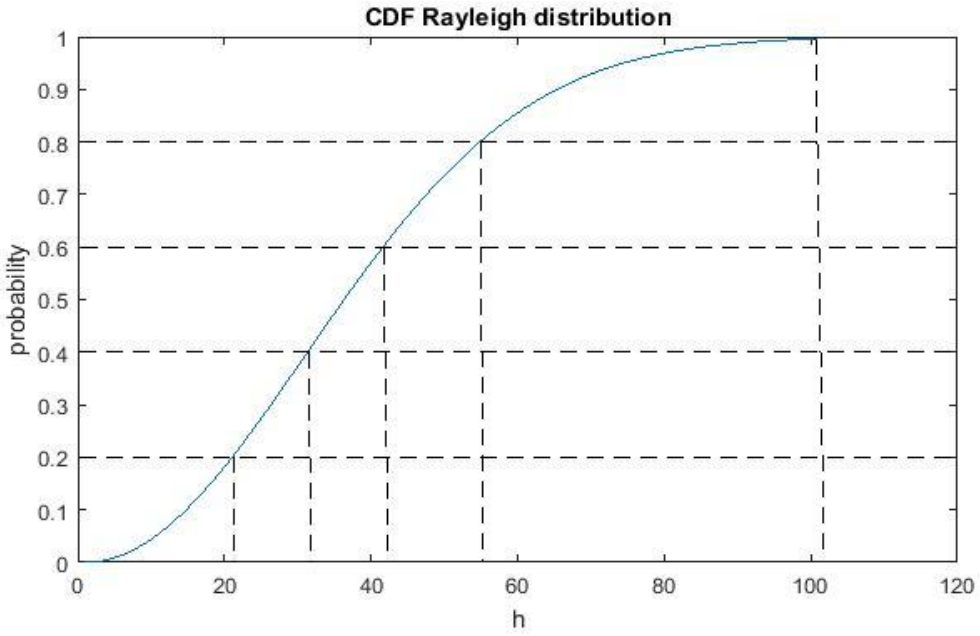

Figure 5-1: Rayleigh distribution discretization

The user's demand process is described by the following two state Markov chain:

$$P = \begin{bmatrix} p & 1-p \\ p & 1-p \end{bmatrix} \qquad p \in [0,1]$$

In the next figures we will show the average costs per content evaluated over a large number of files. The costs are analysed with the following approaches:

- *Reactive approach* – that measures the cost per file only when the user requests it.
- *Proactive approach* – that is the optimal algorithm presented in the previous chapter.

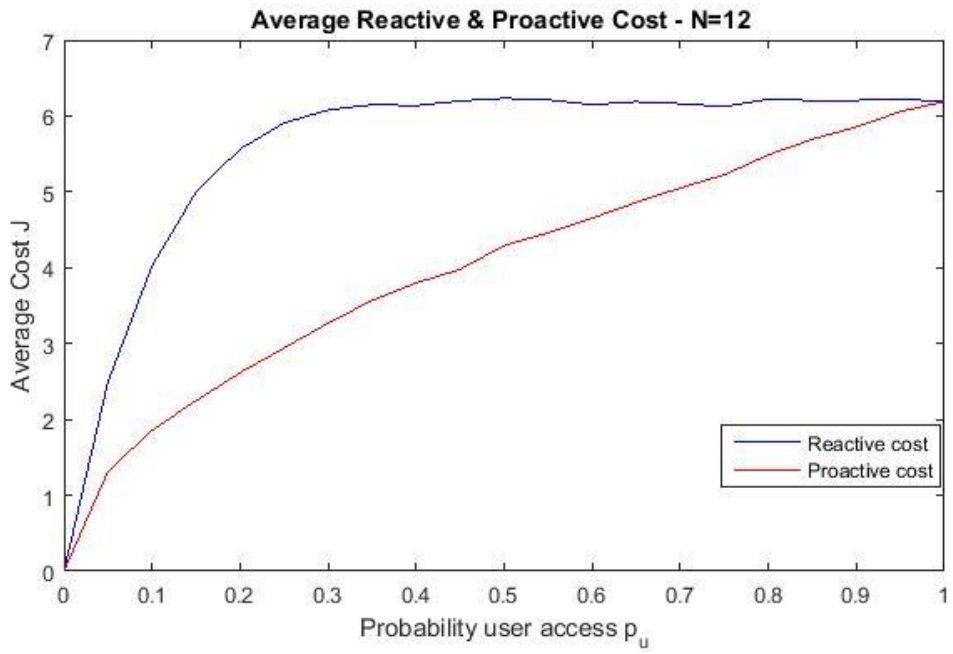

Figure 5-2: Reactive and Proactive cost for a channel with 5 states and content life-time N=12

In the figure 5.2 we present the simulation results of the average cost function with respect to the probability that the user accesses his application. The costs, for the iterations, are generated starting from a Rayleigh distribution with parameter $\sigma = 0.6$.

In particular in this figure (5.2) a comparison between the red curve, which is the optimal average cost, and the blue curve, which is the average cost paid by the user for every access without using any intelligence, is presented.

The most evident outcome is the large gap, in terms of cost, among the intelligent performance and the reactive one. This is due to the fact that the proactive caching algorithm decides to download when the channel cost is the best, in accordance with the stochastic processes involved.

Looking at the figure, it is easy to understand that when the user does not access (p=0) the cost is J=0.

While, if the user accesses with p=1, there is no possibility to do proactive caching, because it is sure that the file will be downloaded. Hence, the proactive curve converges to the expected cost, because the cost distribution for all the files is i.i.d.

If the user accesses very rarely, i.e. for very small value of p, the reactive curves is under the saturation value because, iterating the algorithms for a large number of files, some contents expire without being requested at all, during their lifetime N.

Moreover, the gain provided by the proactive cost is very significant for probabilities $p \in [0.1, 0.7]$, because the algorithm may wait in order to choose a *cheaper* time-slot to download.

Instead, if the user is more likely to access ($p \geq 0.7$), the gain of the proactive solution decreases, because the user's behaviour strongly affects the performance of the algorithm.

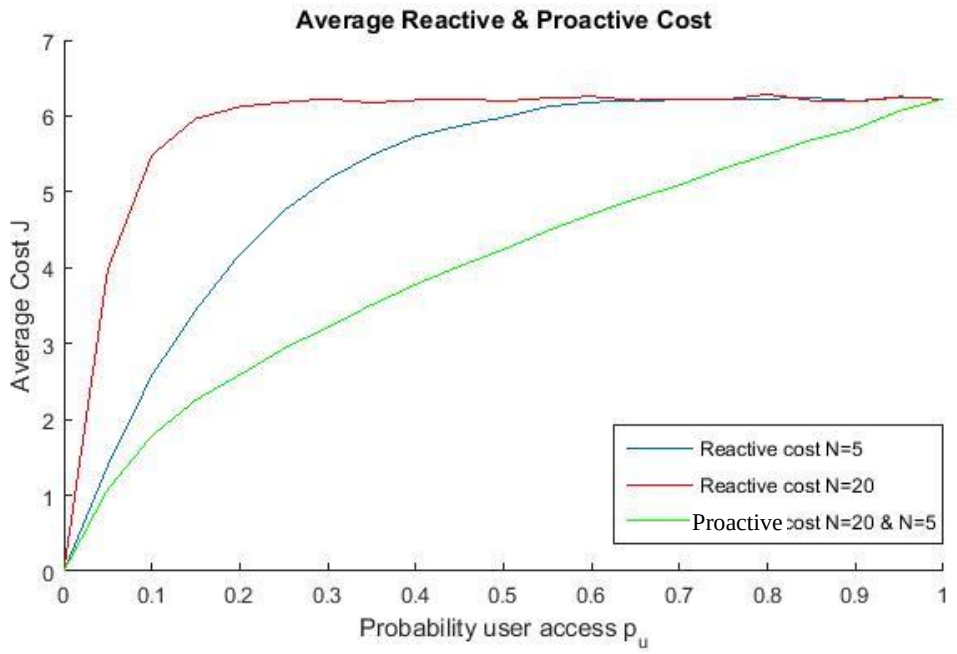

Figure 5-3: Reactive and proactive cost for a channel with 5 channel states and content's life time N=20 and N=5

Plotted in the figure 5.3 are the reactive and proactive cost for different life-times N; all the parameters are the same of the previous simulation but with N=5 and N=20.

If the user is not likely to access, analysing the reactive case, after a certain probability of accessing p, the average cost saturates. This saturation point is different for different N.

The reactive curve for N=5 saturates for ($p > 0.5$), while for N=20 the saturation takes place for a lower value of p. This happens because, when the life-time of the post is longer, the content is more likely to be requested, then the saturation point depends on N.

Hence, for a larger N, no file expires without being requested, as a result the average cost converges early.

When N=5, the saturation value is reached later because some files will never be downloaded during the freshness-time.

As regards the proactive cost, the curves, for the different life-time cases, are the same because the policy does not change too much with time.

Thus, owing to the better performance of the reactive cost when N=5 compared to when N=20, the gain of the proactive algorithm is less evident. This means that the optimal solution offers more benefits when the files are very likely to be requested.

To sum up:

- *If the life-time of each file is short*, only a few contents, of those generated, will be downloaded, so the saturation point will be reached at a higher access probability;
- *If the life-time of the content increases*, it means that the user is interested in past. As follows, more of the generated contents will actually really get downloaded and the average cost will increase over all.

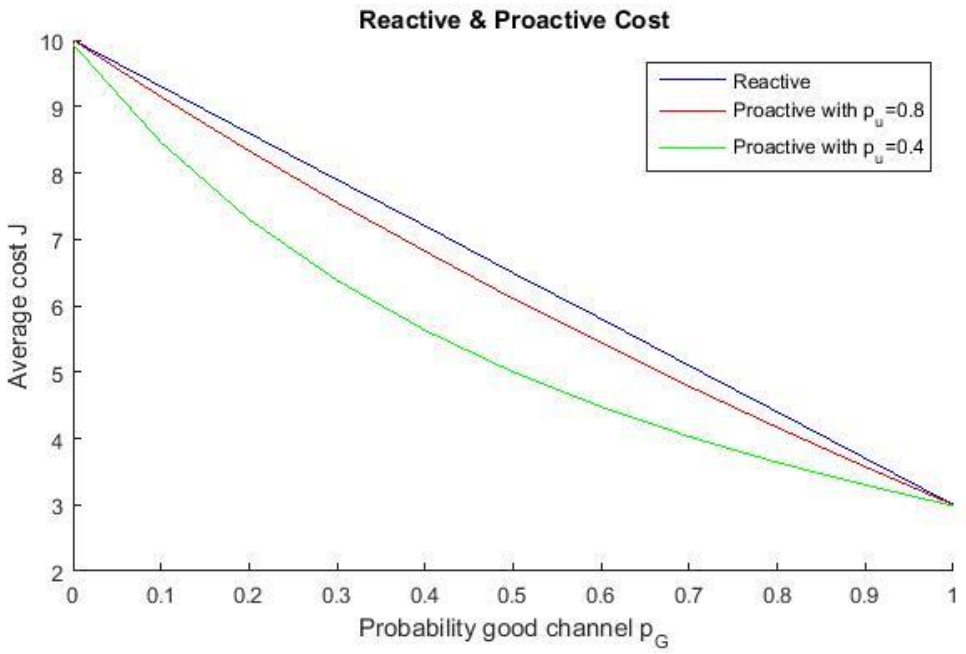

Figure 5-4: Proactive and Reactive cost for a two state channel condition plotted respect to the good channel probability.

In the figure 5.4 above. We analyse the cost functions for a two-states channel {Good, Bad}. The costs relative to both cases are 3 units when the channel is very good, and 10 units when the channel is very bad with N=10.

The functions are plotted with respect to the probability to stay in the Good-state p_G. The blue curve represents the reactive cost which, as expected, shows a linear trend. If $p_G = 0$ the cost paid is the maximum for each file, vice versa if $p_G = 1$, the channel will be constantly in a good state and the cost will be the minimum.

Also in this illustration, it is easy to show the benefits of the proactive algorithm. The most relevant result is that if the channel quality changes a lot, the proactive gives better results because, for a fixed user's request probability and a constant channel, there is no reason to do proactive caching.

Moreover, the red and green curves show the performance respectively for user access probability $p_u = 0.8$ and $p_u = 0.4$. The user behaviour obviously influences the gain,

which is more evident for the green curve (with $p_u = 0.4$) and less significant when the content is almost surely requested.

As already noticed in the previous section, *Problem Formulation*, observing a large number of contents, it is possible to classify the contents into two types of files.

This classification depends on the two-state Markov chain, which describes the user's demand.

Every time a *mobile user (MU)* accesses, the memory is cleared and the next contents will be generated after an Access-state. Some other files, instead, will be generated after a Not-Access state.

The next results have been simulated, with the following user's demand transition matrix:

$$P = \begin{bmatrix} p & 1-p \\ 1-p & p \end{bmatrix} \qquad p \in [0,1] \qquad (2)$$

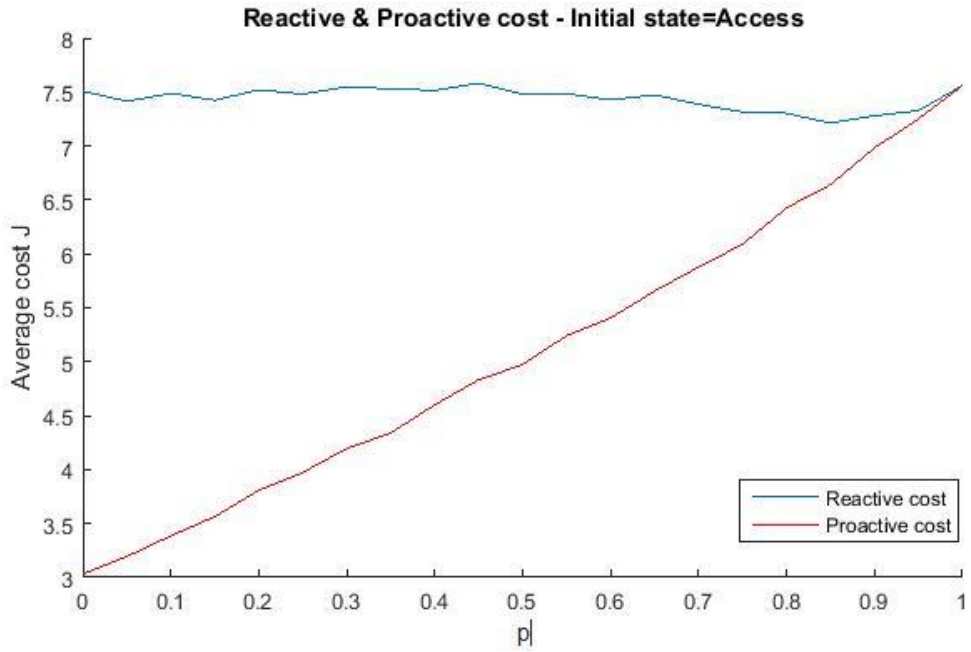

Figure 5-5: Reactive and Proactive average cost per contents generated after an access-state, with life-time N=10, plotted with respect to p, given the matrix 2.

In the figure 5.5, given the matrix defined above, the average reactive cost and the proactive one are plotted with respect to a different value of $p \in [0, 1]$. In this case, the initial state is the *Access-state*. Hence, when $p = 0$, given the matrix (2), the chain generates a sequence like $(I = A)|N, A, N, A, ...$ that means that after an access state A, there is always a not access state N.

This is evident observing the average reactive cost. In fact if $p = 0$, a content with a life-time $N \geq 2$ will always be requested, since the system starts from an access state, hence, the reactive cost saturates from $p = 0$.

The average proactive cost shows a certain gain, due to the fact that, before accessing, after the initial access, for $p = 0$ there is always a not-assess state, during which the optimal algorithm may be applied.

When $p = 1$, starting from an access state, the user accesses again in the next time slot. For this reason, the proactive cost also converges to the average expected value.

In the next figure 5.6, the average costs are evaluated starting from the Initial Not-Access state.

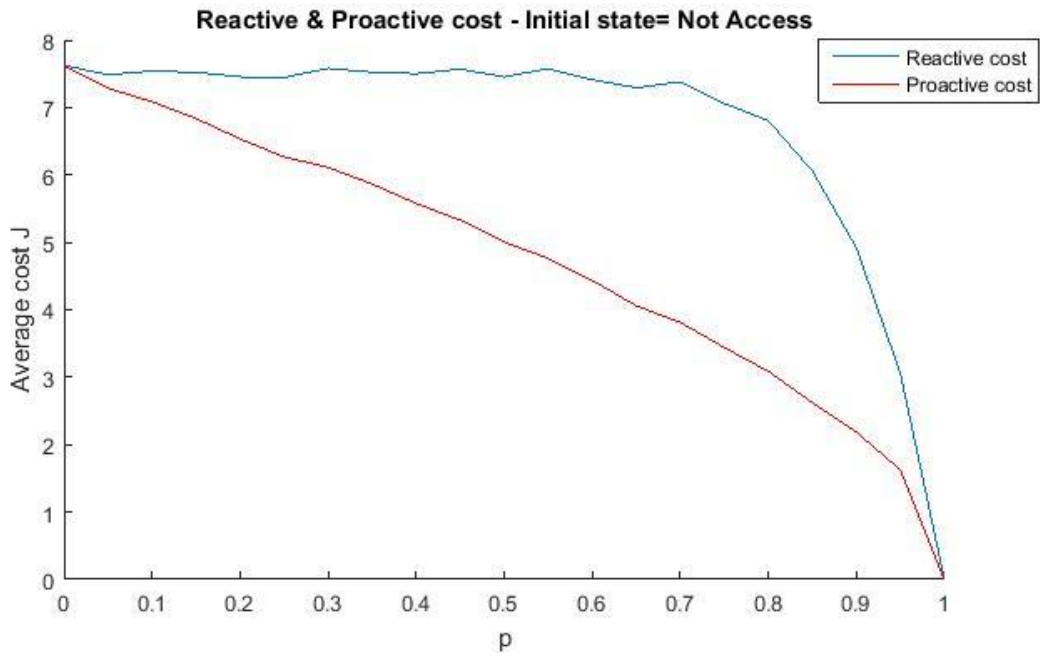

Figure 5-6: Reactive and Proactive average cost per contents generated after a Not-Access state, with life-time N=10, plotted with respect to p, given the matrix 2.

This figure 5.6 was generated using the matrix P (2) defined above. In this case when $p = 0$, starting from a N-state, the user will always be in an A-state, therefore both curves saturates at the average expected cost. By increasing p, the figure shows the benefits of the optimal algorithm.

When $p = 1$, starting from a not-access state, the user remains in the N-state. Thus, if a content is never requested, the cost is always close to zero.

The results plotted in the last two figures point out the different costs due to the different initial states.

In order to evaluate *the average cost per content*, according with the explanation given in the previous chapter, we have weighted each cost with the steady state probabilities.

For the matrix P (2), the steady state vector is $\pi = [\pi_A \quad \pi_N]$

Where π_A is the probability that the user accesses, π_N is the probability of not access.

As results, in this case, given the matrix (2), $\pi = [0.5 \quad 0.5]$, and the relative *average cost per file* is given by

$$J = 0.5 * J_A + 0.5\, J_N$$

With J_A is the cost per file after an access state and J_N is the cost after a not-access state.

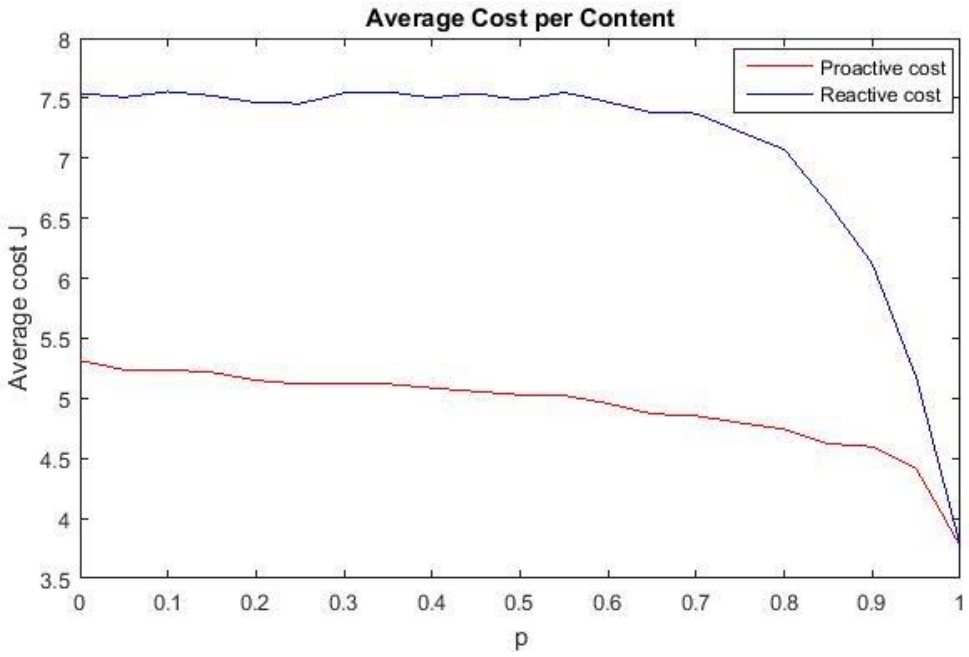

Figure 5-7: Proactive and Reactive average cost per content, weighted with the steady state probability

Hence, as described in the previous chapter, to evaluate the cost for a large number of files, starting from different initial states, the average cost per content is shown by the reactive and proactive curves in the figure 5.7.

This confirms the improvements of the proactive caching for almost all the value of p of the matrix (2), while when p converges to 1, the average cost is affected by the low value of the case with initial state=not-access.

To sum up, the results presented in this chapter demonstrate the benefits of the optimal algorithm developed in this thesis.

REFERENCES

[1] Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2014–2019

[2] Mobile traffic forecasts 2010-2020 – prepared for the UMTS forum January 2011 [http://groups.itu.int/Portals/17/SG5/WP5D/2-3%20UMTS%20Forum%20presentation%20at%20IMT-%20WS%20at%20AWG%20210311_final_v1.pdf]

[3] Ericsson Mobility Report, June 2013 [http://www.ericsson.com/res/docs/2013/ericsson-mobility-report-june-2013.pdf]

[4] Intel, Rethinking the Small Cell Business Model", White Paper, 2011

[5] Bastug, Ejder, Mehdi Bennis, and Mérouane Debbah. "Proactive Caching in 5G Small Cell Networks." (2015).

[6] Jeanette Wannstrom, masterltefaster.com and Keith Mallinson, WiseHarbor, "HetNet/Small Cells" 3GPP [http://www.3gpp.org/technologies/keywords-acronyms/1576-hetnet]

[7] David Chambers. "Data Caching Reduces Backhaul Costs for Small Cells and Wi-Fi", 23 May 2013. [http://www.thinksmallcell.com/Backhaul/data-caching-reduces-backhaul-costs-for-small-cells-and-wi-fi.html]

[8] C. Song, Z. Qu, N. Blumm, and A.-L. Barabási, "Limits of predictability in human mobility," Science, vol. 327, no. 5968,pp. 1018–1021, 2010.

[9] http://www.convinceandconvert.com/social-media-research/11-shocking-new-social-media-statistics-in-america/

[10] http://www.dailymail.co.uk/sciencetech/article-2300466/Smartphone-users-check-Facebook-14-times-day-admit-looking-movies.html

[11] Richard M. Karp,"On-Line Algorithms Versus Off-Line Algorithms: How much is it worth to know the future?", September 1992 http://www.icsi.berkeley.edu/pubs/techreports/TR-92-044.pdf

[12] http://www.theguardian.com/technology/2015/apr/23/facebook-4bn-daily-video-views-ads

[13] https://newsroom.fb.com/news/2014/09/news-feed-fyi-showing-more-timely-stories-from-friends-and-pages/

[14] Puterman, M.L."Markov Decision Processes – Discrete Stochastic Dynamic Programming". John Wiley & Sons, Inc., New York.

[15] http://www.socialmediatoday.com/content/facebook-posts-lifetime-even-shorter-you-thought

[16] Martijn van Otterlo, Marco Wiering, “Reinforcement Learning:State of the Art”, Springer Berlin Heidelberg , pp.3-42

[17] Howard RA (1960) Dynamic Programming and Markov Processes. The MIT Press, Cambridge, Massachusetts.

[18] John Tadrous, and Atilla Eryilmaz, “On Optimal Proactive Caching for Mobile. Networks with Demand Uncertainties”, submitted to IEEE Transactions on Networking.

[19] Anthony J. Nicholson , Brian D. Noble, “BreadCrumbs: forecasting mobile connectivity”, Proceedings of the 14th ACM international conference on Mobile computing and networking, September 14-19, 2008, San Francisco, California, USA

[20] W. Wu, A. Arapostathis, and S. Shakkottai, Optimal power allocation for a time-varying wireless channel under heavy-traffic approximation, IEEE Trans. Automat. Control , 51 (2006), pp.580–594

[21] M. Goyal, A. Kumar and V. Sharma "Power constrained and delay optimal policies for scheduling transmission over a fading channel", Proc. 2003 IEEE Infocom, vol. 1, pp.311 - 320

[22] P.Blasco and D.Gunduz. “Content-level selective offloading in heterogeneous networks: Multi-armed bandit optimization and regret bounds. Jul. 2014.

[23] Cheung, M.H., Huang, J., “DAWN: Delay-Aware WiFi Offloading and Network Selection”, DOI 10.1109/JSAC.2015.2416989, IEEE Journal on Selected Areas in Communications, 2015

[24] Andrews, J.G. Buzzi, S. ; Wan Choi ; Hanly, S.V. ; Lozano, A. ; Soong, A.C.K. ; Zhang, J.C., “What will 5G be?” Selected Areas in Communications, IEEE Journal on, pp 1065-1082. June 2014.

[25] E. Bastug, J.-L. Guénégo, and M. Debbah, “Proactive Small Cell Networks,” Proc. 20th Int’l Conf. Telecommun., Casablanca, Morocco, May 2013.

[26] E. Ba¸stug, M. Bennis, and M. Debbah, “A transfer learning approach for cache-enabled wireless networks,” ˇ arXiv preprint arXiv:1503.05448, 2015.

[27] Goebbels S. Jennen R., “Enhancements in wireless broadband networks using Smart Caching An analytical evaluation”. 2008 IEEE

[28] N. Golrezaei, K. Shanmugam, A. G. Dimakis, A. F. Molisch and G. Caire "Femtocaching: Wireless video content delivery through distributed caching helpers", Proc. IEEE INFOCOM, pp.1107 -1115 2011

[29] Francesco Malandrino, Maciej Kurant, Athina Markopoulou, Cedric Westphal, Ulas C Kozat, “Proactive seeding for information cascades in cellular networks”, INFOCOM, 2012 Proceedings IEEE, pp 1719-1727, March 2012

[30] E Bastug, M Bennis, M Debbah, “Social and spatial proactive caching for mobile data offloading” Communications Workshops (ICC), 2014 IEEE International Conference on, 581-586

[31] C. Gungor and D. Gunduz, "Proactive caching over a time-varying channel for energy minimization," submitted to IEEE Int'l Conf. on Communications, London, UK, Jun. 2015.

[32] D.P.Bertsekas, “Dynamic Programming and Optimal Control”, 3rd ed. Belmont, MA, Usa: Athena Scientific, 2005.