A Cache Prefetch Policy based on User's Temporal-and-Social Behavior for Content Management in Wireless Access Networks

Cleomar Márcio Marques de Oliveira, Igor Monteiro Moraes and Célio Albuquerque
Laboratório MídiaCom, PGC - TCC, Instituto de Computação
Universidade Federal Fluminense, Niterói - RJ - Brazil
cleomarmarques@id.uff.br, igor@ic.uff.br, celio@ic.uff.br

***Abstract*—In this paper, a new policy is proposed for Store and Drops cache content in the Wireless Access Networks nodes. The proposed policy select content that can be dropped and new content to be cached in a network node, on predefined time periods at each day and with pre-established time duration each one, repeated at each day. The temporal aspects and users social behavior that connect to the node for decision making are considered. An algorithm selects new content, in the same proportion, from those categories historically requested ones in these time periods on previous day. These new content selection purpose is to cache them in current day, at the corresponding time periods, to increase the content request hit ratio according to this policy. Simulation results against established FIFO, LRU, LFU and RANDOM policies shows that this proposed policy hit ratio is 2.46 times higher than the others, with an average hit ratio of 13.1% here versus an average hit ratio 5.325% of the above cited policies, in the evaluated scenarios.**

I. Introduction

On the Mobile Edge Computing context, [1], a fast growing has been observed on smart wireless devices use to access the Internet through Wireless Access Networks. The same growth is predicted in a near future, with a traffic volume exponentially growing, as CISCO claims [2]. Wireless access traffic is expected to account for 77.49% of Internet traffic by 2022 and, together with new applications requirements, will cause a high bandwidth demand and possible Wireless Access Networks congestion at the Internet edge.

Due to this large data volume, new functions have been proposed to meet network resources demands by mobile users in these networks, such as multimedia content caching on the edge, for example. Popular content, which are typically the most requested ones, cause overloads in these access networks due to the high demand for bandwidth and low latency needs. One drawback perceived in those networks is the latency growth and an unsatisfactory Quality of Experience (QoE) by the mobile users, even in content centric (ICN, NDN or CCN) architectures [3]–[5].

Considering the high and variable network latency for multimedia traffic, mobile devices playback may suffer interruptions and quality degradation. We introduce a policy to select and cache multimedia files through statically computing the most content categories requested by mobile users and when they did it in a recent past in wireless access networks, in order to reduce the network latency and to improve the QoE to these mobile users. The content selection criteria proposed in this work is based on users social behavior and on the time they typically request them. These content are classified by categories, similar to the ones found in YouTube [6] or NetFlix.

The most requested content on previous days, their categories and the time period each one had been requested are computed and ordered by requests number. These data are stored at its respective time period. After that, the **HASSAN-S** algorithm, proposed in this work and described in Section III-B, selects new content to cache in the current day from a content set on a network cache. These new selected content are stored in cache at its respective time period, based on the previously most requested content categories, and proportionally to the number of times each one had content requested. These new selected content belongs to the same categories as those requested on previous days, as explained, but are different from these ones and are stored in its respective files and cached at the time period beginning on the current day.

This new policy expected result will be to contribute to better save networks resources, since the most popular content may be fetched from the local network cache, instead from a geographically distant content provider. Artifacts on video playout, such as disruptions or quality degradation, shall be avoided since the network latency will be reduced, as the content may be stored into the edge wireless network access node cache when requested by the mobile area users.

The average hit rate resulting in the simulation from the performance processing of the 4 algorithms was 5.325% and the HASSAN-D algorithm was 13.1%.

The remainder of this work is organized as follows. In Section II, related works are presented. In Section III the policy guidelines are described, including it's main algorithm pseudo code, how the mobile users requests were modeled to be processed in the evaluation and the algorithms summarized description. Simulation results and scenarios description are presented in Section IV. In Section V we present considerations and challenges.

978-3-903176-23-2 © 2019 IFIP

II. Literature Review

The mobile users social and temporal behavior aren't very addressed together in the literature in Wireless Access Networks. Some approaches were found in the ICN architecture, but in the TCP/IP traditional centralized network architecture, which focus on address instead of content, it's very poor. Some mobile-related factors compared to the traditional computation offloading techniques, such as device energy, bandwidth utilization cost, network connectivity, mobility, context awareness, location awareness and users social behavior only recently have been considered.

An approach in [7] considers the access costs to Base Stations (BS) based on the availability of storage space and the network bandwidth disposal to cache content in some BS near to the requesting users and, dynamically, assign these content requests to a proper BS content replica in cellular networks. They presented an optimization model through an algorithm online for caching and to dynamically associate content to users content requests, but don't consider the latency and the effects on multimedia content.

In this approach, [8], the authors have considered a distributed caching in the wireless networks nodes in the near geographical area and in the additional device-to-device communications to help the popular content distribution to the interested ones. These mobile devices were named helpers and grouped in clusters. The simulations results have shown an total data throughput improve of 400 to 500%, but they don't consider the QoE and the problems which we want to avoid.

In [9], the authors have proposed to combine the streaming protocol (ABR) in Wireless Networks and Caching in Radio Access Networks (RAN) to improve QoE and increase the cache video capacity. Additionally, they had proposed an algorithm to rates transmission adaptation and specific video coding by its own characteristics. This may contribute to avoid the problems for video streaming transmission.

In [10] the authors have considered the changes in communications due to Smart phone growth and the users habit of uploading a large volume of data through the devices themselves. They also verified that these operations occur systematically in a few locations geographically distinct and different for each user.

They proposed a cellular network architecture for scheduling these file submissions in order to optimize network resources and to better distribute these operations over time. Points were defined as Drops Zones, from where such files could be transmitted in due time, depending on the availability of the network. To cover the entire American territory, 963 were considered sufficient. They consider uploading files by the mobile users and schedule them in network free time. We consider downloading content immediately when they are requested by mobile users.

In [11], the authors considered the high costs and poor QoE perceived by users to obtain popular content and proposed a system that leverages the ability of the current mobile devices to connect them over a mobile network, in function of their near locatizations, to obtain such content directly from one or some of those devices that already have them in memory or that require them, reducing the latency and the costs of its obtaining by the others on that mobile network that also have interest ourselves. This approach may avoid the problems, but depend on the another local mobile users who don't spent long time in the same geographical location and maybe, don't agree to connect their devices in a mobile network to help another users.

III. Policy Description

This new caching policy, through previously prefetching selected content into the cache, consider the content categories, [6], most requested on previous days, in different day time pre-established periods **fhda(n)**. They was statistically computed by counting and classifying the most requested ones by categories **vc(y)** and selecting other ones from the same categories, but different from those it selves.

To do so, we have created fictitious users **UE(i)**. To simulate requests, users **UE(s)** had been randomly chosen from this **UE(i)** set. In that file, each user has **k** preferred content categories registered, established randomly too, to be used at users requests simulation on new content selection to cache. We had randomly established **vc(y)** different video files content categories to be processed by the algorithms.

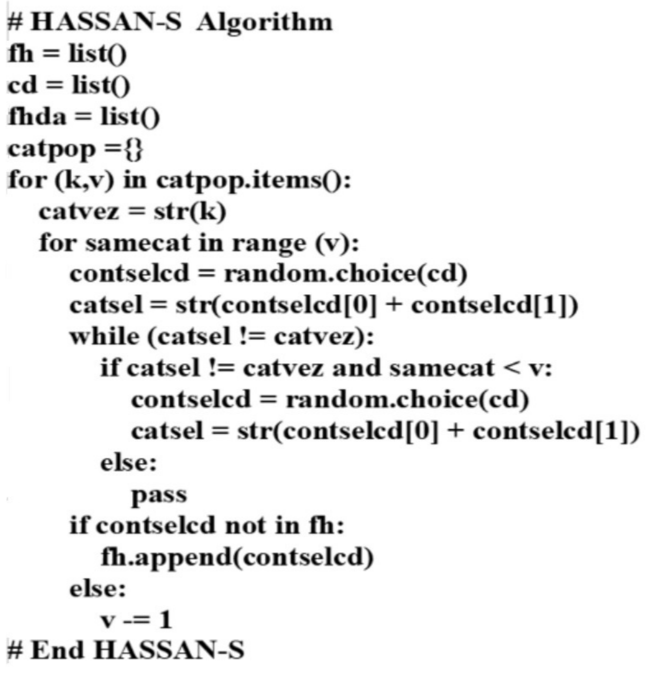
On each time period beginning in the current day, a new content file selected is cached and we simulate the performance of this policy against the LRU, FIFO, LFU and RANDOM drop policies. Another algorithm, **HASSAN-D**, described on section III, was implemented to drop content from the cache as necessary on the simulation process, according to this policy guidelines.

Those four established policies algorithms above don't process the same content file as algorithm HASSAN-D does, they process another subset of content randomly selected from the content set **CD(u)** on a network node, in a number equal to those selected by **HASSAN-S** algorithm, but different from them. These content selected to be processed by the four above cited algorithms are loaded in main memory at the simulation beginning, as it usually occur in the networks cache, and process them with its own drops policies. They are cached as was requested by users, if not yet into the cache.

The most requested content on previous days, its categories and the times each one had content requested are computed and ordered, considering in which previous day time period **fhda(n)** it was requested. These data are stored in its respective time period file. Then **HASSAN-S** algorithm selects new content from a content set **CD(u)** in a network node.

These new selected content are cached in the current day at its respective day time periods **fh(n)**, based on the **categories** most requested previously by users and proportionally to the number of times each one had content requested. These new selected content belongs to same categories **vc(y)** of those requested ones on previous day, but are different from those ones. They are stored on its respective time period files and cached at it's respective beginning on the current day. This allows the **HASSAN-S** algorithm performance evaluation and this policy results.

This **prior content selection** procedure to cache in its respective current day time periods is this **policy key point**, corresponding to the social and temporal users behavior. The HASSAN-S algorithm is presented below:



```
# HASSAN-S  Algorithm
fh = list()
cd = list()
fhda = list()
catpop ={}
for (k,v) in catpop.items():
   catvez = str(k)
   for samecat in range (v):
      contselcd = random.choice(cd)
      catsel = str(contselcd[0] + contselcd[1])
      while (catsel != catvez):
         if catsel != catvez and samecat < v:
            contselcd = random.choice(cd)
            catsel = str(contselcd[0] + contselcd[1])
         else:
            pass
      if contselcd not in fh:
         fh.append(contselcd)
      else:
         v -= 1
# End HASSAN-S
```

Figure 1: Hassan-S Algorithm.

HASSAN-S - Algorithm Files and Variables Summary Description.: **fh**: Selected content set to cache by prefetching on current day period, Fig. 4.
fhda: Previous day time period content set.
cat-pop: Dictionary. k Key = content category, v Value = same content category occurrences number in fhda set.
k,v: var key and value in cat-pop dictionary.
contselcd = var. Selected content from cd.

Fig. 1 shows Hassan-S Python algorithm code to select new content to cache, based on the content categories historical requested on previous days and same time periods.

That previous content selection, which considers the social and temporal users behavior through their requests, is similar to that behavior found in [10], where users make bulky files uploads, such as multimedia files, in a few usual places.

The same behavior is assigned here to them for their content downloads purposes. Users usually order them at their respective custom times, and in the same geographical region as well. The expectation is that the fulfillment of theirs requests can get better results, due to the new content prefetching, according to their personal preferences on their recent past content requests, at the same geographic area.

We assume that many of these mobile users are in the habit of accessing content of their preferences in user's available periods: during breakfast, displacement to work and after lunch, for example. Files from the same categories are correlated and defined as similar files, content or the same genre, according to [6].

As these content categories examples, there are the short film series offered by providers, such as Netflix. They are categories like Romance, Adventure, Cop, etc ..., as well as sports videos or even artistic presentations. Events have their fans too, who order and watch them when they can, due to the possibility presented by cellular applications which allow to watch these content in parts.

Another fact, already mentioned, is the need to optimize the use of network resources consumption, to reduce the costs of obtaining these files, even in high capacity networks such as the predicted next 5G Networks.

Save networks resources is necessary, since the users demands will continue growing exponentially as CISCO claims [2] and it is expected to traffic account for 77.49%. There are also legacy systems to consider with burden on back haul links and long latency bandwidth utilization costs, working as bottlenecks in the entire Network and resulting in its low global performance.

A. Requisitions Modeling

In order to implement the mobiles users content requests according to this policy guidelines, we have modeled the test on a scenario as follows: A fictitious users file UE(i), where i ranges from 0 to 999, was created with k different preferred content categories from those on content file, randomly assign to each one. We assume that (k) equal to 2 for our performance evaluation process purpose.

A UE(s) randomly users set were chosen from those aforementioned UE(i) users as a sample to generate the cache requests. We have established (s) equal to 50.

The users simulation requests take in account these k users assigned preferred content category to select new content to cache at its respective current day fh(n) time period beginning. We have established (n) equal to 10.

The fhda(n) corresponding previous day time period files store the most users content requested on previous days and the categories occurrence numbers of each requested content computed.

The 10 current day time periods corresponding files store new preselected content files, choose by applying this politic social and temporal criterion to choose these new content to cache on each respective fh(n) time period beginning on current day.

They will be processed **only** by this policy main requests processing HASSAN-D algorithm. The others four policies algorithm will process another randomly content file instead, with others chosen content from the same content set CD(u), without any social or temporal users behavior previously established criterion, as usually occurs in the Networks.

Those content categories are represented by two 2 letters. The sequential numbers following these two letters represent the different available videos content in each established categories, vc(y), were y ranges from 3 to 50. They are novel content category videos, but obviously different from each other. The content **fi18** and **no15**, for example, are respectively, **fiction** and **novel** content categories representation and the numbers represent each ones in its category.

The cache frame size may be selected on the process beginning, when we had used the sizes 16 and 30 for this parameter for convenience only in this simulation process. They are vectors data structure position numbers which store content representation symbols.

B. Summarized Algorithms Description

Two algorithms were projected and implemented according this policy strategy for new content selection to cache in the respective current day time periods beginning, and for drop cache content files, respectively: **HASSAN-S and HASSAN-D**. A main algorithm was implemented to process users requests in the cache for the others four algorithms.

The **HASSAN-S** algorithm selects (n) new content from those available on content set. Each new different selected content belongs to the same categories and in proportional number as the same previous categories content requested in the respective previous day time period.

The content selection process for the FIFO, LRU, LFU and RANDOM policies algorithms had no previous content selection criterion. Its randomly chosen from CD content set. Only this policy algorithm process user's content requests cached from a preselected content file and compute the hit and miss when their content were referred on main memory on requests processing algorithm. A aforementioned file is only for this policy processing. The other four policies algorithms process another content file, without no content selection criterion.

Summarized Algorithms Implementation Strategies:

FIFO : As the predefined cache capacity has been completed, the item at the "0" index position, the list "head", is dropped. The new item is inserted at the same list at the bottom position.

LRU : The strategy to determine the LRU item was to reorder the page items into the cache, whenever one of them was referenced, placing it at the queue end. Thus, at the time of substitution, the LRU item was the one in the list "head", at index "0" position.

LFU : The strategy to determine the LFU item was to define an auxiliary list, with the same cache page capacity and size, where instead of storing a copy of the items in the cache, the amount of references to each ones during their stay on the page are stored. These two lists, page and auxiliary list, are "tied" through their indexes in its respective lists. Thus, for an item replacement from the page, the auxiliary list lowest index value content, LFU, is searched and dropped from the page to free space. The corresponding item in the auxiliary list is also dropped and a new item with a "0" content value number is added to control its references number.

RANDOM: A random function was used to generate random numbers in range "0" to the cache memory "current size", to be used as the next drops item index when necessary.

HASSAN-D : It have used the criterion to drop the less referenced content in the cache, due to it doesn't contribute to this proposed policy guidelines performance increase.

The key to this algorithm is the input file, prepared with previously selected content, according to the history of requests from users in previous day time period.

The cache queries hits and miss count was performed for all those strategies, after each cache request simulation. This approach expectation results was that such users would request similar content from the same recent past content categories they had requested.

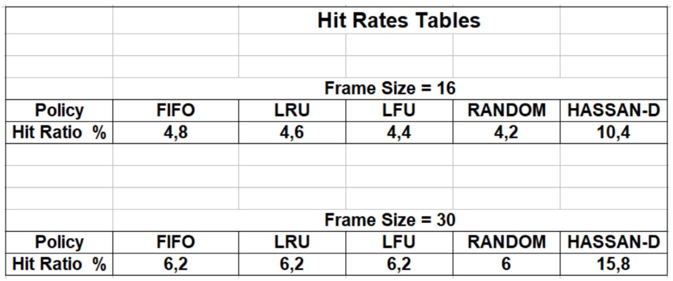
The same new content request category probability is expected to be high than others content categories in current day. A miss will occur if the requested content is not in the cache, otherwise it get a hit. At the end of the considered time period, new preselected content are cached, at the new time period beginning, dropping all previous time period cache content.

IV. Simulation Results

The simulation results presented below, Figs. 2 and 3, registered in the table and graph, were obtained from up to ten different sets of fifth randomly content sample previously selected. The used parameters to generate these results were the page size, common to all algorithms. The two input "workload.txt" and "fh.txt" content files were limited to 50 contents each one as upper bound, corresponding to each UE(s) set requests, since all these UE(s) requests may be different one from another, but repeated content requests may occur.

The hit rates table, Fig. 2, shows the average simulation hits reached from the 10 different input files processing, for each frame size stablished in the permormance evaluation. The graph, Fig. 3, shows graphically these same average results too, where each column has its corresponding color, representing to each one policy, as shown in the legend.

At the end of this evaluation, the calculations present a result of the HASSAN algorithms with performance 13.1%, 2.46 times higher than the others algorithms, with 5.325% on a average hit ratio of them, Figs. 2 and 3.



Hit Rates Tables

Frame Size = 16

Policy	FIFO	LRU	LFU	RANDOM	HASSAN-D
Hit Ratio %	4,8	4,6	4,4	4,2	10,4

Frame Size = 30

Policy	FIFO	LRU	LFU	RANDOM	HASSAN-D
Hit Ratio %	6,2	6,2	6,2	6	15,8

Figure 2: Hit Rates Tables.

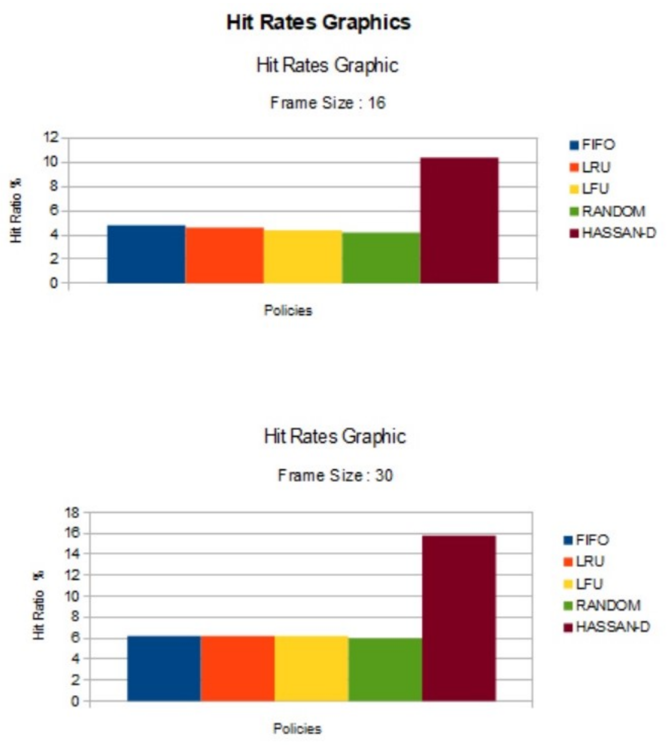


Figure 3: Hit Rates Graphics.

A. Scenarios Description

These tables and graphics presents simulation results from two scenarios: 1 - Frame Size equal to 16, meaning that the cache content store capacity are 16 simultaneous different content, and scenario 2 - Frame Size equal to 30, which meaning that the cache content store capacity are 30 simultaneous different content.

In both scenarios, we have used 10 different UE(s) set of 50 UE(i) each ones, to generate content request to the cache, registered on 10 (rq-fh) different requested content files.

In the second scenario, we have used the same UE(s) number, but the cache capacity to store grew to 30 simultaneous content.

The differences between the simulation results performance may be seen in the tables and were registered on both graphics. Those presented hits results in the two tables are an average from content requests of those 10 different content sets rq-fh(n) to the cache, for both scenarios. Each content set was a 50 content sample randomly selected from the CD set, based on previous days fhda(n) content set.

These fhda (n) files have been pre-selected considering each UE's preferred content category in each group of 10 UEs, with 50 UEs each, to be used as parameters of previous days user requests and most requested content categories, and set to allow a new content selection to cache by prefetching at different time periods beginning on each current day.

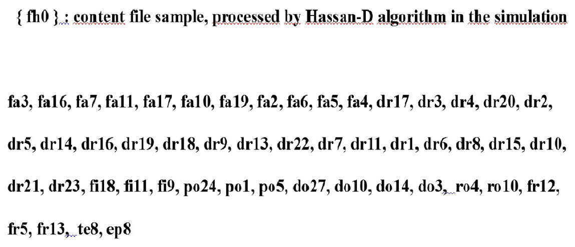
Each fh(n) selected set, ranging from 0 to 9, had differents content chosen based on the content categories in the fhda(n) sets, allowing us to obtain an average performance hit number in current day, which may gave us a consistent simulation result, as shown on tables and graphics.

Fig. 4 shows a sample file, fh(0), which was used in the simulation to represent preselected content by Hassan-S algorithm for prefetching on current day time period beginning.

The fh(0) file have some content which belong to the same content category , represented by the two equal characters, but with different numbers each ones, representing different content in the same category. This is due to the selection proportionality of new content from the same requested categories on previous day, at the same time period.

We have choosen the original Hassan-D report snippets to show content inclusions into the cache, until it be full.

In it's partial report presented on Fig. 5, it may be seen the content inclusion on cache until it be full. Then, it may be seen the cache content replacement by the algorithm, showing which content was dropped and which one was added. Hit and miss, when occurred, are showed too in this partial report. During all report, the cache content is shown, to enable better monitoring of your performance.


{ fh0 } : content file sample, processed by Hassan-D algorithm in the simulation

fa3, fa16, fa7, fa11, fa17, fa10, fa19, fa2, fa6, fa5, fa4, dr17, dr3, dr4, dr20, dr2, dr5, dr14, dr16, dr19, dr18, dr9, dr13, dr22, dr7, dr11, dr1, dr6, dr8, dr15, dr10, dr21, dr23, fi18, fi11, fi9, po24, po1, po5, do27, do10, do14, do3, ro4, ro10, fr12, fr5, fr13, te8, ep8

Figure 4: (fh0) Content File Sample.

Complete reports like this have been generated for each of the 10 pre-selected user files to perform cache content requisitions in order to obtain an hits and miss average of these groups.

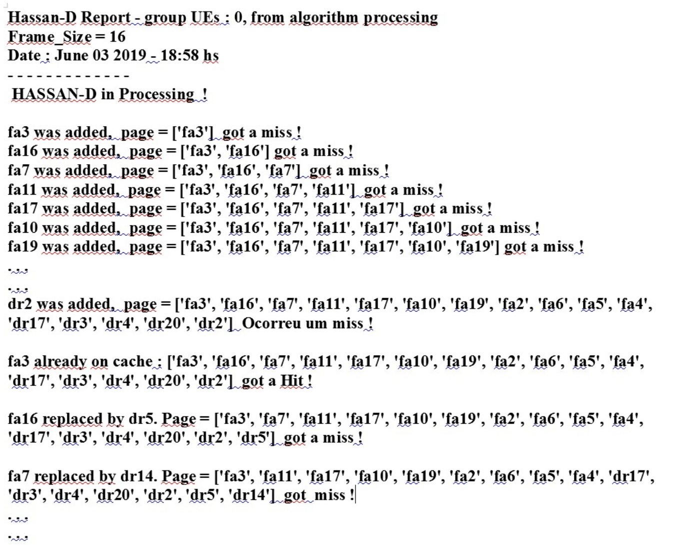
The same procedure was done for the two scenarios, whit two different frame size, as we already explained.

For the others four algorithms, we have proceeded in the same way, using its own files to compute its hits and miss. The workload(n) files were prefetched to be processed by these four algorithms, and similar report was generated to allow counting their hits and miss and to generate tables and graphs for performance comparison.

For each algorithm, a complete report like this presented in Fig. 5 was generated, for each one of the 10 different pre-selected UE(s). Group UEs(0), by example, had generated requests to be processed over the workload(0) cache content file, and so on.

V. Conclusion

The presented results are considered a good policy functioning and appropriateness indication. The HASSAN-S algorithm complete implementation, with automatic new real data set selection and cache at each different time period beginning, constitutes a future implementation challenge.



```
Hassan-D Report - group UEs : 0, from algorithm processing
Frame_Size = 16
Date : June 03 2019 - 18:58 hs
- - - - - - - - - - - - - -
 HASSAN-D in Processing  !

fa3 was added,  page = ['fa3']  got a miss !
fa16 was added,  page = ['fa3', 'fa16'] got a miss !
fa7 was added,  page = ['fa3', 'fa16', 'fa7']  got a miss !
fa11 was added,  page = ['fa3', 'fa16', 'fa7', 'fa11']  got a miss !
fa17 was added,  page = ['fa3', 'fa16', 'fa7', 'fa11', 'fa17']  got a miss !
fa10 was added,  page = ['fa3', 'fa16', 'fa7', 'fa11', 'fa17', 'fa10']  got a miss !
fa19 was added,  page = ['fa3', 'fa16', 'fa7', 'fa11', 'fa17', 'fa10', 'fa19'] got a miss !
...
...
dr2 was added,  page = ['fa3', 'fa16', 'fa7', 'fa11', 'fa17', 'fa10', 'fa19', 'fa2', 'fa6', 'fa5', 'fa4',
'dr17', 'dr3', 'dr4', 'dr20', 'dr2']  Ocorreu um miss !

fa3 already on cache : ['fa3', 'fa16', 'fa7', 'fa11', 'fa17', 'fa10', 'fa19', 'fa2', 'fa6', 'fa5', 'fa4',
'dr17', 'dr3', 'dr4', 'dr20', 'dr2']  got a Hit !

fa16 replaced by dr5. Page = ['fa3', 'fa7', 'fa11', 'fa17', 'fa10', 'fa19', 'fa2', 'fa6', 'fa5', 'fa4',
'dr17', 'dr3', 'dr4', 'dr20', 'dr2', 'dr5']  got a miss !

fa7 replaced by dr14. Page = ['fa3', 'fa11', 'fa17', 'fa10', 'fa19', 'fa2', 'fa6', 'fa5', 'fa4', 'dr17',
'dr3', 'dr4', 'dr20', 'dr2', 'dr5', 'dr14']  got  miss !
...
...
```

Figure 5: Hassan-D Partial (fh0) Content File Processing Report.

This new policy have showed that there is a gain in the wireless network resource utilization optimization.

Content requests presented results may provide a better QoE to users as showed. This is a low computational complexity policy, simple to implement, but may produces good results. It can be gradually implanted in the network cache memories.

The benefits are huge and may contribute to eliminate the interruptions and frames loss problem from video content file when playing. The entire network will also benefit from this policy due to reduction on possible core data traffic. Billing reductions may be possible too as lower data volume will pass through the network core, since many requested content may be already stored in the local wireless access network node cache.

Acknowledgment

This work was supported in part by Capes, CNPq, and Faperj.

References

[1] S. Wang, X. Zhang, Y. Zhang, L. Wang, J. Yang, and W. Wang, "A survey on mobile edge networks: Convergence of computing, caching and communications," *IEEE Access Special Section on Security Analytics and Intelligence for Cyber Physical Systems.*, vol. 5, jun 2017.

[2] Cisco, "Cisco visual networking index: Forecast and trends, 2017–2022 white paper," *Internet*, 2019, Acessed on em 02/03/2019.

[3] Y. Sun, S. Fayaz, Y. Guo, V. Sekar, Y. Jin, M. A. Kaafar, and S. Uhlig, "Trace-driven analysis of icn caching algorithms on video-on-demand workloads," *In Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Technologies*, pp. 363 – 376, 2014.

[4] C. Fricker, P. Robert, J. Roberts, and N. Sbihi, "Impact of traffic mix on caching performance in a content-centric network." *In Computer Communications Workshops (INFOCOM WKSHPS), IEEE Conference*, p. 310–315, 2012.

[5] C. Bernardini, T. Silverston, and O. Festor, "Socially-aware caching strategy for content centric networking." *In Networking Conference IEEE*, p. 1–9, 2014.

[6] YouTube Creators, "Youtube categories," *Internet*, 2018, Acessed on 02/12/2018.

[7] K. P. Naveen, L. Massoulié, E. Baccelli, A. C. Viana, and D. Towsley, "On the interaction between content caching and request assignment in cellular cache networks," *AllThingsCellular '15 - 5th Workshop on All Things Cellular: Operations, Applications and Challenges*, aug 2015.

[8] N. Golrezaei, A. G. Dimakis, A. F. Molisch, and G. Caire, "Wireless video content delivery through distributed caching and peer-to-peer gossiping," *INFOCOMM'12*, 2012.

[9] H. A. Pedersen and S. Dey, "Enhancing mobile video capacity and quality using rate adaptation, ran caching and processing," *IEEE/ACM Transactions on Networking*, vol. 24, no. 2, pp. 996–1010, 2016.

[10] I. Trestian, S. Ranjan, A. Kuzmanovic, and A. Nucci, "Taming the mobile data deluge with drop zones," *IEEE/ACM Transactions on Networking (TON)*, vol. 20, pp. 1010–1023, aug 2012.

[11] K. Thilakarathna, F. Jiang, S. Z. Mrabet, M. A. Kaafar, and G. X. Seneviratne, "Crowd-cache: Leveraging on spatio-temporal correlation in content popularity for mobile networking in proximity," *Computer Communications, 100(Supplement C)*, 2017.