CMPE 49 FINAL PROJECT REPORT

Ahmet Faruk Çelimli, Sinem Kafiloglu, Fatih Alagöz
Department of Computer Engineering, Boğaziçi University
Istanbul, Turkey
Email: {ahmet.celimli, sinem.kafiloglu, fatih.alagoz}@boun.edu.tr

I. INTRODUCTION

Mobile traffic in network is increasing. Video contents generate most of this traffic. When the increase in network traffic is taken into consideration, it seems unlikely that our current host based network will supply with people's expectations. Academicians thinks that a content based network can solve possible future problems. Our motivation in this study to propose a popularity based caching algorithm for content based network that will reduce latency, while increasing local hit ratio compared to least recently used(LRU) caching and least frequently used(LFU) caching.

In second section, we will describe the network devices used in the system. Moreover default system parameters and assumptions will be explained.

In third section, pseudo code of the proposed caching algorithm will be written. After that, in order to help readers to ease understanding of how algorithm works, sample scenario for both base and enhancement content will be explained.

In fourth section, we will compare latency, $P^{SQ}(base)$, $P^{HQ}(base)$, $P^{HQ}(enh|base_{loc})$ and $P^{HQ}(enh|base_{D2D})$ values that we get from our algorithm to LRU and LFU results.

In fifth section, we will briefly describe the results we obtain. Then we will state possible future studies regarding this subject.

II. SYSTEM MODEL

In this architecture, we used six components.

1) Device: Devices are components that has caches and communicates with each other. They are distributed in a circular area whose radius is 300 meters. Each device manages its own cache according to chosen caching algorithm.
2) Base Content: These are standard quality video contents. Each user request must contain a base content. Their size is determined according to exponential distribution. Each of them has a certain popularity according to zipf distribution.
3) Enhancement Content: These are extra video contents that enhance video quality. Their size is determined according to exponential distribution. Each of them has a certain popularity according to zipf distribution.

4) Channel: Channels are enable devices to communicate with each other. Each channel has a bandwidth, a temporary owner and a release time. This release time dynamically updated according to its current owner.
5) Primary User: Primary users have priority in communication over secondary users. Each primary user requests a base layer content over a predefined frequency.
6) Secondary User: Secondary users can request only base layer content or both base and enhancement layer content. They don't request a content from a specific frequency channel as opposed to primary users. They can receive a content from any idle channel at the request moment.

Parameter	Explanation	Value
BaseCache	Device cache capacity for base layer contents	0.8 Gbits
EnhancementCache	Device cache capacity for enhancement layer contents	0.2 Gbits
Rc	Radius of circle that contains devices	300 meters
Nd	Number of devices	500
Simulation Length	Determines simulation length	1200 sec
Repeat	Multiplies the simulation length	100
S	Skewness parameter for content popularity	0.8
BaseAvg	Mean value of base layer contents' size	25 Mbits
EnhancementAvg	Mean value of enhancement layer content' size	5 Mbits

TABLE I. Default system parameters

In this study, we didn't use a base station or a queue for dropping requests. We assumed that at the beginning devices' caches are filled with contents. Also we assumed that capacity of a channel doesn't change due to weather and geographical conditions.

III. CACHING ALGORITHM

The following figure shows the pseudo code of caching algorithm we proposed in this study in detail.

Algorithm 1 Popularity Based Caching Algorithm

```
1:  procedure CacheAlgorithm(self, content)
2:      if content is BaseContent then
3:          content_popularity ⇐ content.popularity
4:          least_popular_content ⇐ self.base_cache[0]
5:          index_for_remove ⇐ 0
6:          counter ⇐ 0
7:          for x in self.base_cache do
8:              if x.popularity <= least_popular_content.popularity then
9:                  least_popular_content ⇐ x
10:                 index_for_remove ⇐ counter
11:             counter ⇐ counter + 1
12:         if content_popularity >= least_popular_content.popularity then
13:             if self.base_capacity <= self.base_capacity_used - least_popular_content.size + content.size then
14:                 self.base_cache.pop(index_for_remove)
15:                 self.base_capacity_used ⇐ self.base_capacity_used − least_popular_content.size
16:                 self.base_cache.append(content)
17:                 self.base_capacity_used ⇐ self.base_capacity_used + content.size
18:     else if content is EnhancementContent then
19:         content_popularity ⇐ content.popularity
20:         least_popular_content ⇐ self.base_cache[0]
21:         index_for_remove ⇐ 0
22:         counter ⇐ 0
23:         for x in self.enhancement_cache do
24:             if x.popularity <= least_popular_content.popularity then
25:                 least_popular_content ⇐ x
26:                 index_for_remove ⇐ counter
27:             counter ⇐ counter + 1
28:         if content_popularity >= least_popular_content.popularity then
29:             if self.enhancement_capacity <= self.enhancement_capacity_used - least_popular_content.size + content.size then
30:                 self.enhancement_cache.pop(index_for_remove)
31:                 self.enhancement_capacity_used ⇐ self.enhancement_capacity_used − least_popular_content.size
32:                 self.enhancement_cache.append(content)
33:                 self.enhancement_capacity_used ⇐ self.enhancement_capacity_used + content.size
```

Fig. 1: Pseudo Code

Our algorithm takes content's popularities into consideration while managing a device's cache. Devices we work with has 1Gbits total cache size. We virtually divide this cache into two subgroups. First of them has 80% of total cache size and is used to store base layer contents. Second one has 20% of total cache size and is used to store enhancement layer contents.

Caching algorithm determines the popularity of the content it received as a parameter. Then it finds the content whose popularity is lowest in the cache. If the popularity of the given content is higher than the content with lowest popularity then former content should be the one that is stored in the cache.

However caching algorithm must check size constraints. If the current used size in the cache will be bigger than total cache size when the least popular content is deleted and new content is added, then cache won't change.

Let's say our secondary user makes a standard quality content request. First of all user's device checks its base content cache whether it is a local hit. Let's assume that the base layer content hasn't been stored in the base content cache. Then the device asks other devices whether or not they have this content. When it found the appropriate device, transmission of content is started. At this point, our receiver device runs its caching algorithm by passing received content as parameter.

Caching algorithm determines the type of the content. It finds out that this is a base layer content, it finds the least popular content in the device's base content cache. Then it compares two content's popularity and realize that the given content's popularity is higher. Algorithm checks cache size constraints and see that it can store this new content after deleting the least popular content. So it deletes the least popular content from its base content cache. Then it adds the new content to its base content cache.

Let's say our secondary user makes a high quality content request. When user's device check its base content cache it finds out that this base content is in the cache. So user can receive the base content from its base cache. However enhancement content cache doesn't contain enhancement content. So user's device asks other devices for the enhancement content. When it found the appropriate device, transmission of enhancement content is started. Receiver device runs its caching algorithm by passing received content. Caching algorithms determines that this is a enhancement content. So its finds the least popular content in the device's enhancement content cache. Then it compares two content's popularity. It realizes that the least popular content still has a higher popularity then the new content. Caching algorithms discards this new content and the enhancement cache of the device remains unchanged.

IV. PERFORMANCE EVALUATION

In order to compare proposed algorithm to LRU and LFU we calculated latency, $P^{SQ}_{(base)}$, $P^{HQ}_{(base)}$, $P^{HQ}_{(enh|base_{loc})}$ and $P^{HQ}_{(enh|base_{D2D})}$ values. These values are average of 10 simulation.

LATENCY

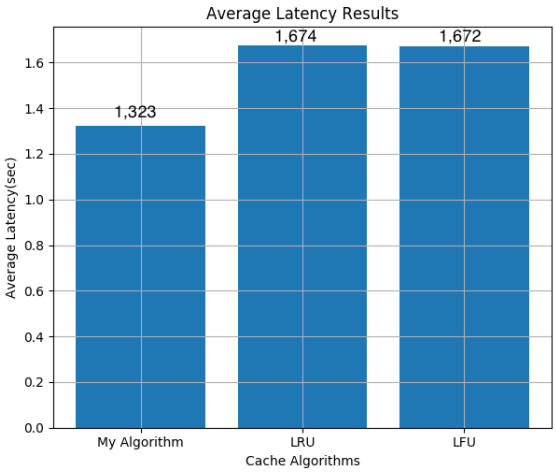

Fig. 2: Average Latency Results

According to average results we gather from 10 simulation latency in proposed algorithm is 1,323 second, in LRU is 1,674 second and in LFU is 1,672 second. Our algorithm improved latency approximately 21%.

In this study we used local hit and device to device communication cases to calculate latency. D2D communication's duration does not change in different caching algorithms. So local hits make the real difference. We take latency for base contents as 0.25 second and for enhancement contents as 0.05 second. These values are significantly smaller than the service duration of D2D communication. So while our algorithm improves local hit ratio, it also decreases latency.

LOCAL HIT RATIO FOR SQ REQUEST

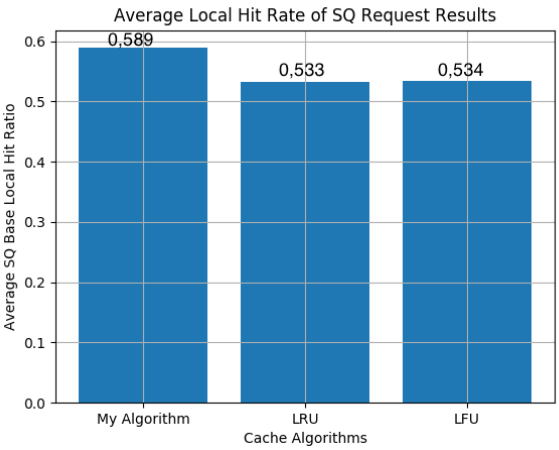

Fig. 3: Average Local Hit Rate of SQ Requests Results

According to average results we gather from 10 simulation, base local hit ratio of a standard quality request in proposed algorithm is 0,589, in LRU is 0,533 and in LFU is 0,534. Our algorithm improved hit ratio about 10%.

Proposed algorithm improves hit ratio rate due to its cache management. Secondary users' requests are in correlation with content's popularity. Proposed algorithm uses contents' popularity to manage cache. This direct relationship between requests and cache algorithm increases hit ratio.

LOCAL HIT RATIO FOR HQ REQUEST

According to average results we gather from 10 simulation (figure is in next page), base local hit ratio of a standard quality request in proposed algorithm is 0,589, in LRU is 0,534 and in LFU is 0,535. Our algorithm improved hit ratio about 10%.

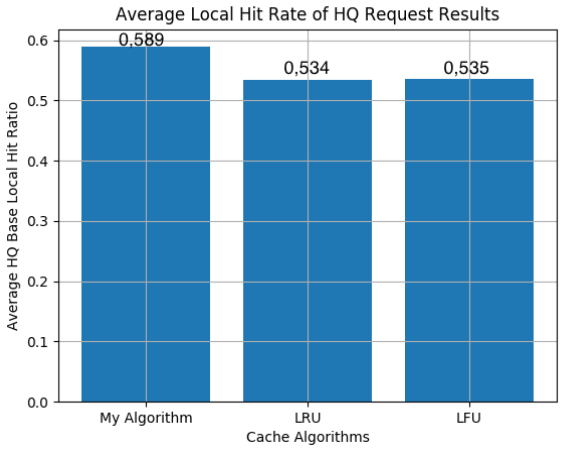

Fig. 4: Average Local Hit Rate of HQ Requests Results

The reason why proposed algorithm improves HQ request hit ratio is same with how it improves SQ request hit ratio. Our algorithm uses same logic with users' requests. So it gives better results than LRU and LFU.

LOCAL HIT RATIO FOR $P^{HQ}_{(enh|base_{loc})}$

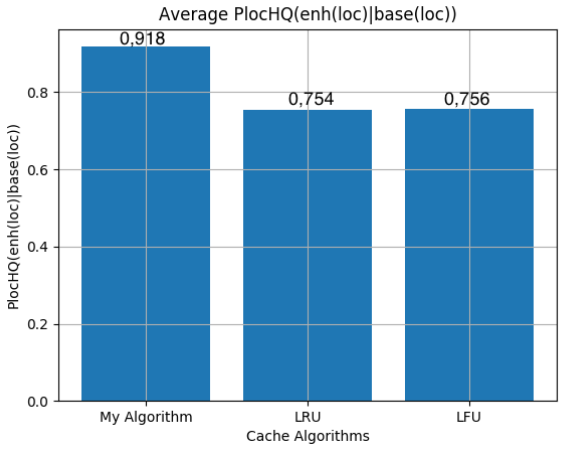

Fig. 5: Average Local Hit Rate of $P^{HQ}_{(enh|base_{loc})}$

According to average results we gather from 10 simulation (figure is in next page), ratio of enhancement is a local hit given that base is a local hit in proposed algorithm is 0,918, in LRU is 0,754 and in LFU is 0,756. Our algorithm improved hit ratio about 20%.

The reason behind is that at the beginning most of the base cache and enhancement cache is filled with same videos' contents. Each enhancement content request automatically contains a base content request. Therefore even though theoretically their can contain layer contents from different videos, in practice content of base and enhancement caches is similar. LRU and LFU has the same caching contents at the beginning. However they don't consider popularity while rearranging caches. So their base caches contains contents from videos that enhancement caches don't contain.

LOCAL HIT RATIO FOR $P^{HQ}_{(enh|base_{D2D})}$

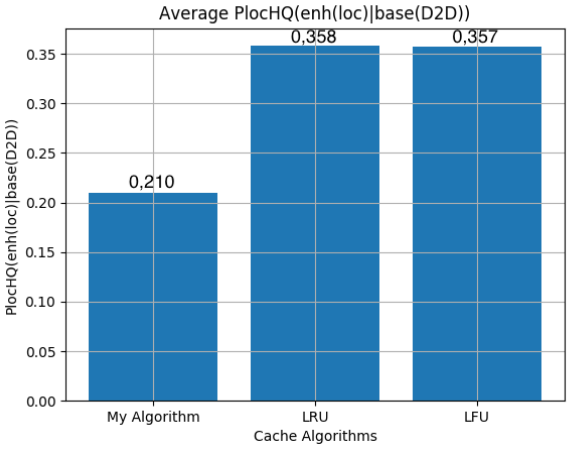

Fig. 5: Average Local Hit Rate of $P^{HQ}_{(enh|base_{D2D})}$

According to average results we gather from 10 simulation (figure is in next page), ratio of enhancement is a local hit given that base is served via D2D technique in proposed algorithm is 0,210, in LRU is 0,358 and in LFU is 0,357. LRU and LFU give better hit ratio compared to proposed algorithm about 60%.

This result isn't surprising when we consider how the proposed algorithm improves the hit ratio of enhancement is a local hit given that base is a local hit. In our proposed algorithm, contents of base cache and enhancement cache are similar. Therefore the probability of finding an enhancement content in the cache while its base content is received from another device is small. On the other hand, as explained before the differences between base and enhancement cache is higher in LRU and LFU. Therefore their hit ratio is higher compared to proposed algorithm.

V. CONCLUSIONS

The proposed algorithm in this study is implemented for a content based network. Our algorithm's cache management is popularity based. Our motivation in this study was to reduce latency, while increasing local hit ratio compared to least recently used(LRU) caching and least frequently used(LFU) caching. According to average results we managed to reduce latency approximately 21%. At the same time we increase local hit rate SQ and HQ requests about 10%. We increased rate of enhancement is a local hit given that base is a local hit about 20%. LRU and LFU only give better results in rate of enhancement is a local hit given that base is served via D2D technique. When all these results are taken into consideration, it is obvious that the proposed algorithm in this study fulfill its objectives.

Future studies can be made in order to improve both caching algorithm and the system. For now, our proposed algorithm can delete only the last element to create new space to new content. However this isn't good enough because this blocks caching a very popular and big sized content. Smarter caching algorithm can delete more than one content to open new space according to popularity of the new content.

Moreover, for now our system only have devices. If any device don't have the wanted content then the request is dropped. This isn't good for quality of experience. A base station and satellite can be added to system.

Furthermore, in our system only one transmission can be done at any specific time interval. However we can adjust our communication system to allow more than one transmission at the same frequency channel if the distance between two device pair is far enough to overcome signal interference.