VIDEO CONTENTS PRIOR STORING SERVER FOR LTE NETWORKS

Tony Tsang[1] and Ho Fong Fun [2]

[1] Department of Computer Science, Chu Hai College of Higher Education, Hong Kong

[2]Centre of International Education, Hong Kong College of Technology, Hong Kong

ABSTRACT

One of the most important multimedia applications is Internet protocol TV (IPTV) for next-generation networks. IPTV provides triple-play services that require high-speed access networks with the functions of multicasting and quality of service (QoS) guarantees. Among 4G mobile ac-cess networks, LTE networks are regarded as among the best solutions to meet higher bandwidth demands. In this paper, we propose a new architecture for multicasting live IPTV traffic in 4G LTE networks. The proposed mechanism involves assigning a unique logical link identifier to each IPTV channel. To manage multicasting, a prior storing server in the Base Station and in each mobile network unit, mobile phone, is constructed. In this work, we propose a partial prior storing strategy that considers the changes in the popularity of the video content segments over time and the access patterns of the users to compute the utility of the objects in the prior storage. We also propose to partition the prior storage to avoid the eviction of the popular objects (those not accessed frequently) by the unpopular ones which are accessed with higher frequency. The popularity distribution and ageing of popularity are measured from two online datasets and use the parameters in simulations. Simulation results show that our proposed architecture can improve the system performance and QoS parameters in terms of packet delay, jitter and packet loss.

KEYWORDS

Internet protocol TV, Wireless Network, Mobile Network, QoS parameters

1. INTRODUCTION

Long Term Evolution (LTE) provide wireless communication of high speed data between data terminals and mobile phones. It base on GSM/EDGE and UMTS/HSPA network technologies. The LTE provide higher speed of wireless data networks and higher capacity compared with 3G Net-work. Although LTE provide fast network to us nowadays, mobile traffic will rise dramatically in the coming years, it may cause network congestion which affect quality of user experience. There-fore, mobile network carriers must increase their network capacity investment (eg. transmission equipment, software and hardware).

Today, most of mobile traffic is caused by mobile video transmission and the demand for mobile video is expected to increase further. The required bit rate of mobile video content is much higher than the rate required by other type of mobile content. Hence rate of data transfer will obviously increase in the future. The mobile carriers are necessary to handle video delivery in the mobile network.

To handle video delivery, the mobile carriers can improve the Video-on-Demand (VoD) system. A typical VoD system consist of a large number of distributed serving nodes, connected via dedicated or shared links, each of which is further built up with a lot of servers, and possibly provides service to end users in its own domain. Traditional VoD system, e.g.. IPTV, simply replicate the entire video library across all the serving nodes, which are becoming infeasible with

DOI : 10.5121/ijwmn.2015.7503

the current growth of online video library. On the contrary, collaborative VoD systems allow serving nodes to share their disk capacities through in-system links, and are becoming the mainstream systems currently.

In the area of media communication, video adaptation is an emerging field. The basic idea of video adaptation is to degrade the data rate of video requests with acceptable loss of quality, so as to reduce the network congestion or the end-user stall. It aims to maximise the quality on experience (QoE) of all user equipment (UEs).

Another method to handle explosive growth of Video-on-Demand (VoD) service, large number of geographically distributed serving nodes are built up, connected via a high-bandwidth backbone, each of which consists of a lot of servers to store videos and provides VoD service to users in its own domain. We can share the capacities among the nodes, and make them collaborative work. Thus, the capacity of the system is greatly increases. In project, we investigate the video prior storage problems in practical LTE and we suggest solution to improve the system performance and QoS parameters.

2. PROPOSED ARCHITECTURE

To manage Video Content, a prior storing serve in the eNode Base Station communicate with Mobile Phone through LTE Network. In this structure, we propose a partial prior storing strategy that considers the changes in the popularity of the video content segments over time and the access patterns of the users to compute the utility of the objects in the prior storage. We also propose to partition the prior storage to avoid the eviction of the popular objects (those not accessed frequently) by the unpopular ones which are accessed with higher frequency. Simulation results show that our proposed architecture can improve the system performance and QoS parameters. The distributed local video content servers allow operators to economically alleviate the inherent storage and network bandwidth limitation, proportionally distribute the subscriber load and service demand. In this work, an efficient prior storing management scheme is investigated for distributed IPTV local video content services. Video Content prior storage is a fundamental strategy for improving the performance and quality of service perceived by consumers.

In our proposed architecture in Figure 1, the ISP network (operated by a single telecom provider) for the VOD services in which the video objects are located at the Video Content Server and connected to Video Content Distribution Networks and Packet Data Network Gateways. The multiple eNode Base Station are connected to Packet Data Network Gateways and connected to Prior Storage Server individually. We assume that video objects are partitioned into fixed size content segments. We found that fixed contents segmentation with a moderate size is preferred over adaptive contents segmentation owing to memory management issues. One problem with fixed contents segmentation is that the contents segment boundaries may not align with those of heavily-viewed portions, leading to prior storing inefficiency. To avoid this problem, we measure the utility of a content segment based on the number of bytes played from it, thereby capturing the utility of prior storing the contents segment. We assume that different contents segments of the same video might differ in their popularity. Changes in the popularity of a video at the segment level are also seen when the user can seek to watch different parts of the video (termed skipped viewing) so that some content segments of the video tend to have a higher popularity than the others. In this work, we use a content segment as a basic unit of a video object for prior storing and replacement. Henceforth, we use the words 'content segment' and 'object' in a prior storage interchangeably. Following, a large piece payload is used as the basic unit of a video and a content segment can be viewed as a collection of payloads. On the other hand, we propose to partition the prior store into two level prior storages. Each partitioned prior storing uses a different function to update the utility of content segments present in its storage. We also

determine the popularity distribution and ageing in popularity based on user ratings from two online datasets. Prior Storage 1 (primary prior storage) is used to prior storage a segment that is accessed for the first time (on a typical prior storage miss) and Prior Storage 2 (secondary prior storage) is used to store segments whose utility is already determined to be high (i.e., popular over a longer period of time). We use independent functions to determine the utility of segments in these two partitions because the eviction of contents segments from them should be handled differently. Finally, the multiple eNode Base Station connected to Prior Storage Server to save the data of video. On the other side, the eNode Base Station communicated with mobile phone. Therefore, the user can get the data of video. In summary, we proposed a dual level prior storage algorithm which adaptively prior storage with bi-level control and it incorporated with popularity characteristics of video on user's demand content streaming service. It is important that the Node Base Station as a caching server to communicate with mobile phone to user. By our proposed architecture can improve the LTE system performance and QoS parameters.

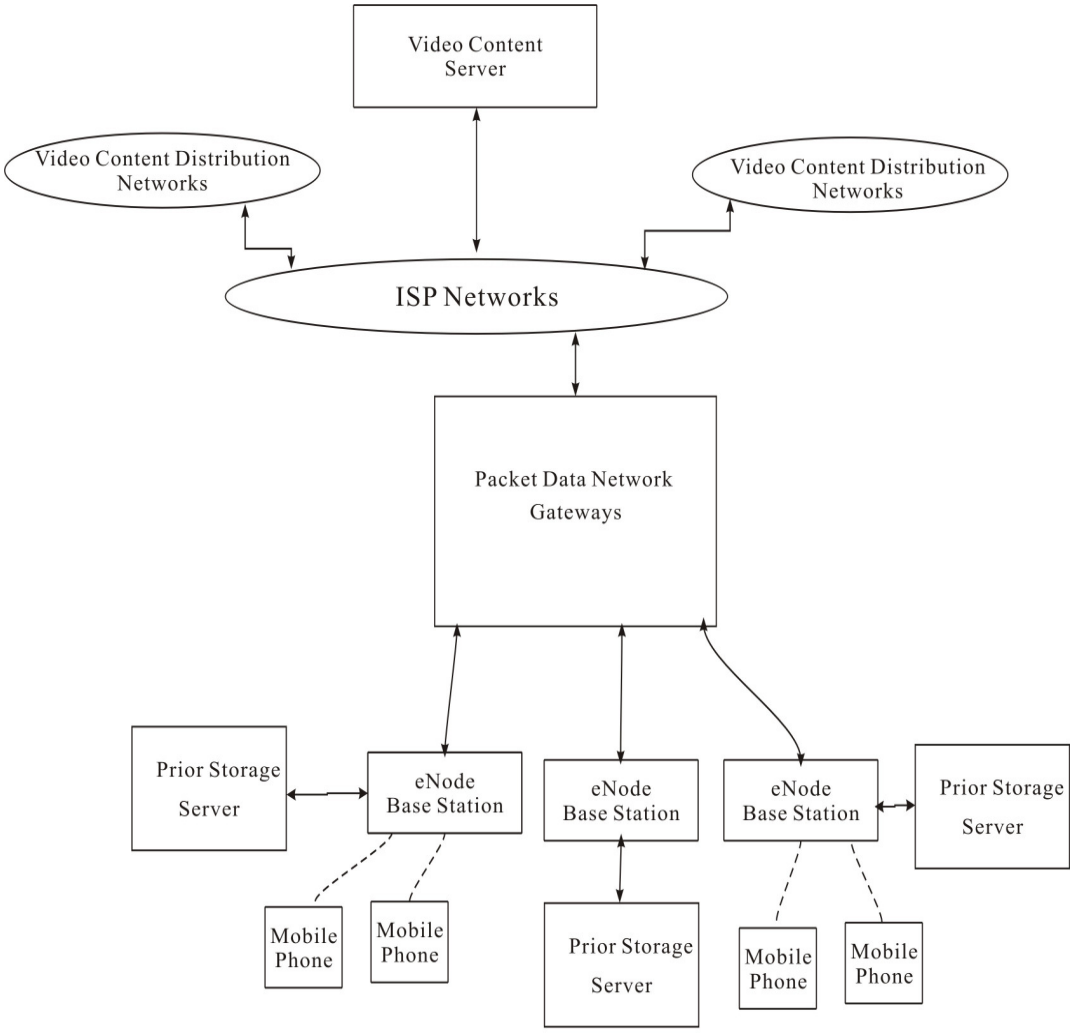

Figure 1: Network Model for Prior Storage Servers in a LTE System

3. PRIOR STORAGE FORMULATIONS AND HARDNESS

We target an LTE system depicted in previous section have local storage for video content prior storage; subscribers request video content clips through corresponding eNode Base Stations, which further pass requests to serving gateways. There has one copy of the required video content (local hit) for a serving node directly responses to a request. Otherwise, the serving gateway checks whether the file is prior storage at other serving nodes, through internal directory service. If so, it fetches the file from remote serving nodes and serves the request (in-system hit). Otherwise, the serving gateway passes the request through the Packet Data Network (PDN) gateway to ISPs' networks. The video content file is then delivered from the original server or certain Content Distribution Network (CDN) to the PDN gateway, which is in turn passed to the subscriber through LTE network (external hit).

We next formulate the system model. We use M to denote the set of serving node ($m = |M|$), and N to denote the target video content library accessible to subscribers ($n = |N|$). Serving node $i \in M$ is with storage capacity of D_i, and stores a sub-set of video content clips $S_i \subseteq N$. Video content clip $k \in N$ is a file with size s_k, representing a set of video content frames. Typically, for single video content clip, we have $s_k \quad D_i, \forall\ k \in N,\ i \in M$. The aggregate request frequency for video content k at node i is denoted by f^k_i, which can be predicted from historical statistics [1]. We note that f^k_i also represents the popularity of video content k at node i for the time period considered.

Table I summarizes important notations used in the article.

TABLE I: Basic Notations

Notation	Meaning
N	The set of accessible online video content clips
M	The set of serving nodes
D_i	The disk capacity of serving node i
S_i	The set of video content clips prior storage at serving node i
y^k_i	Indicator of prior storage video content k at node i
χ_k	Indicator of video content prior storage k in system
f^k_i	Request frequency of video content k in node i
s_k	The size of video content k
B_e	The total amount of budget for external data traffic
l_0	Latency for local hit per unit
l_1	Latency for in-system hit per unit
l_2	Latency for external hit per unit

Let indicators y^k_i and χ^k denote whether video content k is prior storage at node i and whether video content k needs to be fetched from the Internet, respectively. Here we assume that the system would never request for any video content clip from the Internet if the clip is local or in-system hit. Accordingly, we have

$$\sum_{k\in N}\sum_{I\in M} L^k_i \qquad x_k = \begin{cases} 0 & \sum_{i\in M} y^k_i > 0 \\ 1 & otherwise \end{cases} \qquad (1)$$

Now that the video content prior storage solution is calculated and updated over certain long period of time (e.g., once a week). There are different levels of latency for delivering unit data from the Internet to the Packet Data Network gateway (for external hit), from one serving gateway to another (for in-system hit), and from a serving gateway to a subscriber (for local hit). Hence, we define l_0, l_1, l_2 for the three levels of latency above respectively, with $l_0 < l_1 < l_2$. Besides latency, external hit also causes operating expense [1]. To control the expense, we keep a budget B_e for aggregate data traffic from Internet. Usually, B_e is the traffic limit between two network vendors for certain period, i.e., weekly, monthly.

The latency for requesting video content k on node i is then expressed as:

$$L^k_i = s_k(l_0 + l_1 + l_2\chi^k - l_1 y^k_i) \quad (2)$$

Recalling the motivation of in-system prior storage, we aim to minimize the aggregate latency over all requests. The problem is then formulated as following:

$$\min \quad \sum_{k\in N}\sum_{i\in M} L^k_i \quad (3)$$

$$\sum_{k\in N} y^k_i s_k \le D_i, \forall_i \in M \quad (4)$$

$$\sum_{i\in M}\sum_{k\in N} f^k_i x_k s_k \le B_e \quad (5)$$

$$\text{var.} \quad y^k_i \in \{0,1\}, \forall k \in N, \forall i \in M \quad (6)$$

and (1). In other words, we attempt to minimize the aggregate latency over all predicted requests with a feasible video content prior storage solution, *i.e.*, disk capacity constraints (4), while keeping amount of Internet access under budget, *i.e.*, the budget constraint (5).

For brevity, we eliminate the constant part in (2) and have the equivalent problem below:

$$\max \quad \sum_{i\in M}\sum_{k\in N} f^k_i s_k (l_1 y^k_i - l_2 x_k) \quad (7)$$

subject to (1)(4)(5) and (6). We next focus on this interpolation formulation.

Unfortunately, further study shows that it is non-trivial to solve the problem (7), as it can be shown to be NP-hard.

Theorem 1: It is NP-hard to find an optimal solution to the problem (7). Proof: We first review the video content storage problem formulated for a closed VoD system in [11], where our goal is to maximize the aggregate hit ratio over the in-system video content library N':

$$\max \quad \sum_{i\in M}\sum_{k\in N} f^k_i s_k y^k_i \quad (8)$$

$$\text{s.t.} \quad \sum_{i\in M} y^k_i \ge 1, \forall k \in N' \quad (9)$$

and (4)(6).

Consider an oracle that can solve any instance of problem (7) in unit time. Apparently, the solution $\{y^k_i\}$ also implies the in-system video content library N'. With N', problem (7) is actually reduced to problem (8) by removing constant values of $\{\chi_k\}$. Accordingly, with the oracle, the max-hit problem (8) has a polynomial-time solution. Thus, we have made a polynomial time reduction from problem (7) to problem (8).

In [2], we proved problem (8) is NP-hard. Therefore, the prior storage problem studied in this paper is also NP-hard.

4. CRITICAL FACTORS AND PRIOR STORAGE SOLUTIONS

As proven to be NP-hard, the problem cannot be solved exactly in polynomial time unless P=NP. In this section, we first exploit some important factors in problem (7), and then develop a fast algorithm for the problem based on these factors.

4.1. Calculating In-system Video Content Hitting Set

Requesting a video content clip from the Internet is influenced by the distance from original servers or CDNs to the PDN gateway, the instant traffic over ISPs' networks and the traffic management of the ISPs. Different from local hit and in-system hit that we can fully control, external hit is time-costly and instable, e.g., usually $l_1 \square\ l_2$.

We note that lower frequently requested video content clips take equivalent disk space while contributing less hitting data. Therefore, to minimize the total latency in problem (3), the system should prior storage as many top popular video content clips as possible. The aggregate flow introduced by video content $k \in N$ is $\sum_{i \in M} f_i^k s_k$. Taking the budget constraint (5) into consideration, we develop the in-network video content set *N* with following steps:

1) Calculate $f_k = \sum_{i \in M} f_i^k s_k \forall k \in N$. Sort $\vec{f}$ in non-incremental order.
2) *N' = N*. For *k = 1* to |*N*|, if $B_e \geq f_k$, then $N' = N - \{k\}$, $B_e = B_e - f_k$. Otherwise, terminate and output set *N'*.

The basic idea is to remove bottom frequent video content clips iteratively until the Internet budget is used up. Normally, above procedure results are in a slightly reduced video content library *N'*.

Now the remaining problem is how to prior storage the video content library *N'* among the serving nodes.

4.2. A Fast Algorithm for Prior Storing Selected Video Content Hitting Clips

With the in-system video content library *N'* selected, the objective in problem (7) can be rewritten as:

$$\sum_{i \in M} \sum_{k \in N} f_i^k s_k (l_1 y_i^k - l_2 x_k)$$
$$= l_1 \sum_{i \in M} \sum_{k \in N} f_i^k s_k y_i^k - l_2 \sum_{i \in M} \sum_{k \in N} f_i^k s_k x_k \quad (10)$$

With *N'*, Eq. (1) is equivalent to:

$$x_k = \begin{cases} 1, & k \in N, \ N' \\ 0, & otherwise \end{cases} \quad (11)$$

Hence, the second part of (10) is constant. Consequently, problem (7) equals the problem below:

$$\max \ l_1 \sum_{i \in M} \sum_{k \in N} f_i^k s_k y_i^k \quad (12)$$

subject to (4)(6)(9). Finally, the original problem (3) of minimizing aggregate latency reduces to the maximizing hit-ratio problem with a fixed in-system video content library, i.e., problem (8). Next, we adopt the α-search approach introduced in [2] to solve the rest problem.

We tend to prior storage top frequently requested video content clips with high priority so as to produce better objective value in (12). Thus, we first keep α fraction of aggregate storage capacity for top popular video content clips, while guaranteeing the coverage (9) with (1 − α) part. Then we search the optimal α to maximize the objective value. A selection of α is called "feasible" if and only if the (1−α) fraction of system capacity can cover all the rest video content clips in *N'* that is not stored with high priority.

Our algorithm consists of a *α-prior storing procedure* and a *α-search procedure*. The α-prior storing procedure further includes the following steps.

- Step 1: Store top popular video content with α fraction of system capacity;
- Step 2: Store uncovered video content clips with rest capacity;
- Step 3: Fill the rest of disk.

We next expand the procedures and steps in detail one by one.

1) Step 1 of α-prior storing: The sub-problem in step 1 is formulated below:

$$\max \quad (12)$$

$$\text{s.t.} \quad \sum_{i\in M}\sum_{k\in N'} s_k y_i^k \le \alpha \sum_{i\in M} D_i \quad (13)$$

and (4)(6). In other words, we try to pack video content clips to maximize the objective value such that no more than α fraction of aggregate capacity is used and no single disk is over-filled. We comment that problem (13) is a variant of the classical Knapsack problem [3], which is also NP-hard. Since we do not really need an optimal solution to this sub-problem, a fast greedy algorithm is more desirable instead. Algorithm 1 outlines the detail of Step 1, where D'_i denotes the current capacity left in serving node $i \in M$.

Algorithm 1: Greedy Algorithm for α-fraction Packing

1: **Init.** $S_i = \emptyset$ for $i \in M$, $D_\alpha = \sum_{i\in M} D_i, T = \{(i,k) \mid k \in N', i \in M\}$.
2: **while** $D_\alpha > 0$ and $P \neq \emptyset$ **do**
3: $(i^*,k^*) = \text{argmax}_{(i,k)\in T} f^k_i$
4: **if** $D_\alpha \ge S_{k^*}$ and $D'_{i^*} \ge S_{k^*}$ **then**
5: $S_{i^*} = S_{i^*} \cup \{k^*\}$, $D_\alpha = D_\alpha - S_{k^*}$, $D'_{i^*} = D'_{i^*} - S_{k^*}$
6: **end if**
7: $T = T \setminus \{(i^*,k^*)\}$
8: **end while**
9: **Output** $\{S_i\}$ and calculate in-system video content set $N_\alpha = \cup_{i\in M} S_i$.

The flow of Algorithm 1 is quite straightforward. In each iteration, we pack the most frequent pair (i,k) if the size of video content k is feasible for node i, i.e., $s_k \le D'_i$, and the aggregate packed size is under α fraction of overall capacity, i.e., $S_k \le D_\alpha$ (line 4). At the end of Algorithm 1, we

calculate the set of video content clips already stored in this phase. In next step, we attempt to cover the rest video content set $Nr = N' \setminus N\alpha$.

2) *Step 2 of α-Prior Storage*: The main purpose of this step is to guarantee the coverage, thus each video content in N_r is to be prior stored only once. We want to solve following sub-problem:

$$\max \quad (12)$$

$$\text{s.t.} \quad \sum_{k \in N_r} s_k y_i^k \le D_i', \forall i \in M \quad (14)$$

$$\sum_{i \in M} y_i^k = 1, \forall k \in N_r$$

and (6).

Further investigation shows that problem (14) is an instance of the *Generalized Assignment Problem* (GAP) [4], where the profits of items do not vary with the placement. GAP is a classical problem in combinatorial optimization, which is proved to be NP-hard and even APX-hard to be approximated. Thus, problem (14) is also NP-hard. As a quick reference, this special case of GAP can be converted into a so-called "Assigning Users to Sources Problem", which can be optimally solved by DeMaio [5] with an implicit enumeration algorithm and Srinivasan [6] with a branch and bound based algorithm. Unfortunately, both of these algorithms have high timely complexity which makes them impractical especially when the problem size is extremely large.

Since our purpose lies in the coverage of the rest video content set rather than maximizing the objective value, we adopt a greedy method for GAP in [7]. The algorithm is outlined in Algorithm 2, in which we define a desirability function, and iteratively assign the video content with the most desirability to the best node till no video content clip is left. The desirability is expressed as f_k guided by the weight function, which is set to f^k_i in our implementation.

Algorithm 2 Fast Algorithm for Problem (14)

[Step 1]: Init. $K = N_r$, $S'_i = \emptyset$ for $i \in M$.
[Step 2]: Set $Q_k = \{i | s_k \le D'_i\}$ for $k \in K$.
if $\exists k \in K$ such that $Q_k = \emptyset$ **then**
Terminate, claim "infeasible."
end if
[Step 3]: $i_k = \text{argmax}_{i \in Q} k f^k_i$, for all $k \in K$
$f_k = \min_{t \in Q_k, t \ne i_k} [f_i^k k - f_t^k], \forall k \in K$
$k = \text{argmax}_{k \in K} f_k$. Store video content $\hat{k}$ in node i_k.
Update $S'_{i_{\hat{k}}} = S'_{i_{\hat{k}}} \cup \{\hat{k}\}, D'_{i_{\hat{k}}} = D'_{i_{\hat{k}}} - s_{\hat{k}}, K = K - \{\hat{k}\}$
[Step 4]: if $K = \emptyset$ **then**
Update $S_i = S_i \cup S'_i$ for all $i \in M$.
Terminate "feasible".
else
Go to Step 2.
end if

3) *Step 3 of α-Prior Storage*: This step will be executed only when Step 2 terminates feasible. Here we want to further increase the objective value by using the remaining capacity. At each serving node $i \in M$, we want to find the video content set S''_i such that:

$$\max \sum_{k \in S''_i \subset N^r_i} s_k f^k_i$$
$$\text{s.t.} \sum_{k \in S''_i} s_k \le D'_i . \qquad (15)$$

$N^r_i = N' \setminus S_i$ is the set of video content not yet prior stored in node i through Step 1 or Step 2. This is a classical 0-1 Knapsack problem. In this work, we solve it with a greedy algorithm, which proceeds as follows:

1) Initialize $S''_i = \emptyset$.
2) $N^f_i = \{k \in N^r_i \setminus S''_i \mid s_k \le D'_i\}$.
3) If $N^f_i = \emptyset$, stop.
4) Otherwise, find $k^* = \text{argmax}_{k \in N^f_i} f^k_i$. Prior Storage video content k^*, $S''_i = S''_i \cup \{k^*\}$, $D'_i = D'_i - s_{k^*}$. Goto 2).
5) Update prior storing status, $S_i = S_i \cup S''_i$.

N^f_i is the feasible set of video contents that can be prior stored at node i.

4) α-Searching Algorithm: For a given instance of problem (12), the objective value produced by α-prior storing is a function of α, expressed as C(α). Generally, choosing a larger α increases the hitting value but decreases the chance of finding a feasible solution in step 2, and vice versa. Furthermore, a larger α means that the system has more space for top popular video content. Accordingly, a better objective value is produced. We proved and evaluated these assumptions in [2].

   Therefore, we conclude that the optimal α is the upper bound of the feasible region, which in turn can be found via binary search in the interval [0, 1].

5. CONCLUSIONS

In order to increase data transfer rate of mobile video for LTE system, the mobile carriers should improve Video-on-Demand (VoD) System. The main problem for Video-on-Demand (VoD) System is storage and bandwidth limitation. An efficient approach to solve this problem can use distributed content prior stores and prior store popular content. In this paper, we suggest the network architecture for Prior Storage Servers in a LTE Network. In this structure, we propose a partial prior storing strategy that considers the changes in the popularity of the video content segments over time and the access patterns of the users to compute the utility of the objects in the prior storage. The distributed local video content servers allow operators to economically alleviate the inherent storage and network bandwidth limitation, proportionally distribute the subscriber load and service demand.

Moreover, we propose to partition the prior store into two level storages, it incorporated with popularity characteristics of video on user's demand content streaming service. Prior Storage 1 (primary prior storage) is used to prior storage a segment that is accessed for the first time (on a typical prior storage miss) and Prior Storage 2 (secondary prior storage) is used to store segments whose utility is already determined to be high (i.e., popular over a longer period of time). Simulation results show that our proposed architecture can improve the system performance and QoS parameters.

REFERENCES

[1] Jun He, Honghai Zhang, Baohua Zhao, and Sampath Rangarajan. A collaborative framework for in-network video caching in mobile networks. In Sensor, Mesh and Ad Hoc Communications and Networks (SECON), 2013 10th Annual IEEE Communications Society Conference on. IEEE, 2013.

[2] Jun He, Xiaoming Zhao, and Baohua Zhao. A fast, simple and near optimal content placement scheme for a large-scale vod system. In Communication Systems (ICCS), 2012 IEEE International Conference on, pages 378–382. IEEE, 2012.

[3] C. Chekuri and S. Khanna. A ptas for the multiple knapsack problem. In Proceedings of the eleventh annual ACM-SIAM symposium on Discrete algorithms, pages 213–222. Society for Industrial and Applied Mathematics, 2000.

[4] Dirk G Cattrysse and Luk N Van Wassenhove. A survey of algorithms for the generalized assignment problem. European Journal of Operational Research, 60(3):260–272, 1992..

[5] A. De Maio and C. Roveda. An all zero-one algorithm for a certain class of transportation problems. Operations Research, pages 1406– 1418, 1971.

[6] V. Srinivasan and GL Thompson. An algorithm for assigning uses to sources in a special class of transportation problems. Operations Research, pages 284–295, 1973.

[7] H.E. Romeijn and D.R. Morales. A class of greedy algorithms for the generalized assignment problem. Discrete Applied Mathematics, 103(1):209–235, 2000.

[8] R-143, “Enabling Network Throughput Performance Tests and Statistical Monitoring”, Broadband Forum, May 2008.

[9] RFC 3393, “IP Packet Delay Variation Metric for IP Performance Metrics (IPPM)”, IETF, November 2002.

Authors

Tony Tsang Tony Tsang (M’2000) received the BEng degree in Electronics & Electrical Engineering with First Class Honours in U.K., in 1992. He received the Ph.D from the La Trobe University (Australia) in 2000. He was awarded the La Trobe University Post-graduation Scholarship in 1998. Prior to joining the Hong Kong Polytechnic University, Dr. Tsang earned several years of teaching and researching experience in the Department of Computer Science and Computer Engineering, La Trobe University. He works in Hong Kong Polytechnic University as Lecturer since 2001. He works in Chu Hai College in 2015. He has numerous publications in international journals and conferences and is a technical reviewer for several international journals and conferences. His research interests include mobile computing, networking, protocol engineering and formal methods. Dr. Tsang is a member of the ACM and the IEEE.