Algorithm for Cost-effective Distribution of VoD Contents

Sasho Gramatikov and Goce Gjorgjijoski
Faculty of Computer Science and Informatics, Ss Cyril and Methodious University in Skopje, Macedonia
Email: sasho@finki.ukim.mk and gocegj@ukim.edu.mk

***Abstract*—The demand of video contents has rapidly increased in the past years as a result of the wide deployment of IPTV and the variety of services offered by the network operators. One of the services that has especially become attractive to the customers is real-time video on demand (VoD) because it offers an immediate streaming of a large variety of video contents. The price that the operators have to pay for this convenience is the increased traffic in the networks, which are becoming more congested due to the higher demand for VoD contents and the increased quality of the videos. As a solution, in this paper we propose a hierarchical network system for VoD content delivery in managed networks and a redistribution algorithm for optimal distribution of the videos on the servers. The system monitors the state of the network and the behaviour of the users to estimate the demand for video contents and to take decision of the best placement of the videos on the streaming servers. The objectives of our work is to distribute replicas of the videos in the network in a way that the most demanded contents are placed closer to the clients so that the network is optimally used and the users' experience is improved. Our experimental results show that the redistribution algorithm is highly responsive to the change of the popularity of the videos which significantly reduces the cost of the traffic generated in the core of the network.**

***Keywords*-VoD; cost; optimization; popularity;**

I. Introduction

Video on Demand (VoD) has a potential to become one of the most popular services, due to the fact that it enables the clients to watch any video they want at any time they want, and this experience can be highly personalized. However, this kind of service has a significant impact on network performance. It requires a dedicated unicast flow of data for every request from the clients, which significantly increases the traffic in the network. Two of the main challenges associated with this service are the network architecture and the strategies for placement of the video contents.

One of the initial approaches that has been considered for solving the problem of network congestion is placing servers in various points of the network and caching replicas of certain videos so that they are closer to the users and they can serve more clients. One of the most accepted concepts that use this approach are the Content Delivery Networks (CDN) [1]. As the videos started to dominate in the internet traffic, the CDNs become a convenient solution for hosting such contents and many of the approaches for distribution of web contents emerged as an acceptable solution. The research in the CDNs was mainly concentrated in solving the replication problem which consists of determining the number of replicas that have to be made for a given video and the servers where to be placed in order to optimize certain cost and quality of service [2][3].

Many VoD solutions move towards development of new architectures in the reliable managed networks. These networks are a convenient solution because their size and capacity can be adjusted according to the number of the subscribed users. The IPTV [4] emerged as one of the most implemented solution by many operators. Because of its growing popularity, the problem of the optimal network utilization and the provision of an improved quality of service is a main concern for many research works. Following this direction, the authors in [5] propose an algorithm for optimal placement of video contents for various IPTV services based on the popularity of the contents without considering the state of the network. Few different algorithms for replication and placement of VoD contents within a cluster of media servers based on the user request pattern are proposed in [6][7].

In this paper we propose a solution for optimal and efficient content delivery in a managed network. We have developed a hierarchical content delivery system that implements replication algorithm for redistribution of the contents in the system. The main objectives of our proposed system are to keep the popular content items close to the clients concentrating the traffic to the outer bands of the network. We use the number of requests on every server and the state of the network as main parameters for reaching our objective. The model is highly responsive to the user behaviour and network conditions, following the dynamic of the popularity of the offered video contents.

The rest of this paper is organized as follows. In Section II, we describe the proposed systems architecture and the interactions between its entities. In Section III, we define a set of parameters essential for the system. In Section IV we present the principles of the redistribution algorithm. The experimental results obtained by different simulation scenarios are shown in Section V. Finally, we give our conclusions in Section VI.

II. System Description

For the purpose of optimal video delivery, we developed a network model capable of serving large amount of streaming requests and managing the network according to the users' behaviour. It consists of streaming servers responsible for serving the clients and management servers which are responsible for the automatic content distribution and service selection.

The streaming servers serve the requests for videos of the clients. They can host and serve only part of the entire video library offered to the clients. From a structural point of view, the streaming servers are organized as an n-tier hierarchical architecture (Figure 1). Starting from the top of the hierarchy, every level downwards is closer to the clients. The CR server contains all the contents that are offered by the operator and unlike the rest of the streaming servers, it does not serve clients. It serves as an entry point for new videos in the system and as an origin point for distribution of replicas to the streaming servers down the hierarchy. The clients in the system use a PC or STB to view the content items. These devices have internal buffer that stores the received packets for decoding and prevents interruptions when there is network congestion. The management servers are represented by the Automatic Content Movement (ACM) server and the Service Selection (SS) server.

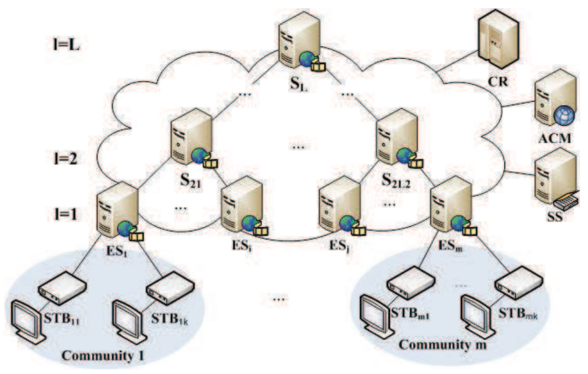

Fig. 1. Model architecture

The ACM server has a central role in the system. It communicates with all the servers, monitors the system, takes redistribution decisions and issues commands to the servers. The ACM monitors the state of the network by periodically issuing commands to the streaming servers. Upon reception of the state information from all the streaming servers, it forwards it to the SS server for redirection purposes. Whenever it detects that there are overloaded servers, it runs an algorithm for content redistribution. Using popularity data for the contents in the recent past, previously obtained from the SS server, the algorithm decides whether a replica of a content item should be moved to another server, removed or left as it is. The execution of the algorithm results with a new distribution of the content items in the system which is deployed by execution of the set of removal, replication and movement commands issued by the ACM server. Along with issuing the commands, the ACM server sends the new availability of the contents to the SS server. The SS server's role is to accept the clients' requests and to redirect them to the most appropriate streaming server. When the best server is chosen, the SS forwards the address to the client, which in turn resends the request to the indicated server. The whole process of handling the situation when there is no replica on any server is fully shown in Figure 2, where the numbers attached to the arrows mark the sequence of each

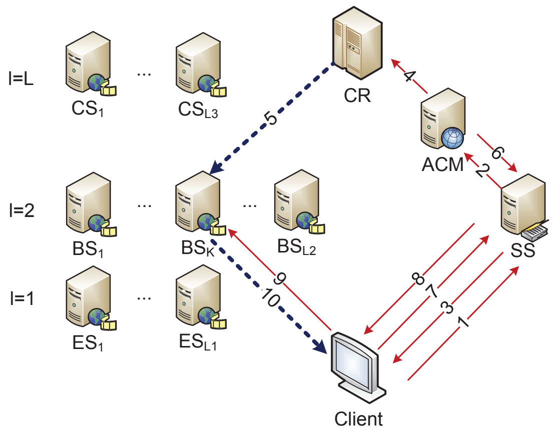

Fig. 2. Process of serving a client request for video

action. After the client makes a request for an unavailable content (1), the SS servers asks the ACM server (2) to issue a replication command for the missing content item to the CR server and sends a response to the client (3) indicating it to retry after a time that is long enough for a sufficient part of the content to be provided from the other servers. Because of the large size of the multimedia objects, to avoid long waiting time for complete distribution, the replicas are pushed to the servers with data rate higher than the streaming rate and the streaming is initiated when there is enough buffered streaming data on the server where the replica is pushed. The ACM server chooses the best server to host the replica and issues a push command to the CR server (4). After the delivery of the content has been initiated (5), the ACM server informs the SS server about the new location of the replica (6). Later, when the client resends the request (7), it is redirected to the new streaming server (8). Once the client has the address of the server that can best serve it, it makes a request (9) and immediately initiates a streaming session (10).

The redirection strategy implemented on the SS servers has an objective to redirect each request to the appropriate server so that the streaming traffic is concentrated in the lower levels of the network, closer to the clients. For this purpose we have chosen a very simple redirection strategy: we choose the nearest available server that is not overloaded, and that has the requested content.

III. System parameters

In this section we define the parameters for estimation of the state of the system and parameters that will be used by the redistribution algorithm. The content delivery network that we propose has a total of S streaming servers, placed in one of the L different levels of the hierarchy. With $U(s)$ is defined the streaming capacity of the server s, which represents the allowed number of simultaneous streams at any given time. Every server s has its storage capacity $SC(s)$, which is the number of videos that can be placed on that server. The availability of content c on server s is represented by the matrix $A(s, c)$. The videos have the same average duration of d time units and are streamed with Standard Definition (SD) quality. The clients are requesting the videos every w time unit.

To be able to decide whether to replicate or to delete a given content c on a specific server s, the redistribution algorithm uses two parameters: the replication metric RM and the deletion metric DM, defined as

$$RM(c,s) = (1 - A(s,c))\,(C - rank(c) + 1) \cdot \cdot \left(S - \sum_{i=1}^{S-1} (S-i) A(i,c) \right) \tag{1}$$

$$DM(c,s) = A(s,c)\mathrm{rank}(c) \sum_{i=1}^{S-1} (S-i) A(i,c) \tag{2}$$

where $A(s, c)$ has a value 1 if content c is available on server s, or 0 if otherwise, S is the number of servers in the system, C is the size of the video library, and $\mathrm{rank}(c)$ is a function that gives the rank of content c. The video with the lowest value of the rank is the most popular one.

IV. Redistribution algorithm

The goal of the redistribution algorithm is to reorganize the contents in the system by moving them as close as possible to the clients in order to optimize the network utilization according to the users behaviour. It can be run whenever the system detects increase of the streaming utilization of the servers. The redistribution algorithm is designed to work with one server at a time and it is executed in two phases: calculating parameters, and replication and deletion. The algorithm repeats this phases for every server in the system. In the first phase, the algorithm calculates the deletion and replication metric according to (1) and (2) for every content in the system. This values are only used for the server that is processed at that time by the algorithm.

In the second phase, the algorithm first finds the IDs of the contents with the highest replication and deletion metrics. After choosing the content with the biggest replication metric, the algorithm checks whether there is enough free storage space on the server to place the content. If there is enough space, the content is copied to the server, and the new replication and deletion metrics are calculated. If there is not enough free space to copy the content with the highest replication metric on the server, the algorithm chooses the content with the highest deletion metric as a candidate for deletion from the server. Before this content is deleted, the algorithm calculates the replication metric that that content will have if it is deleted from the server. Then, the algorithm compares the calculated replication metric for the content chosen for deletion and for the content chosen for replication. If the calculated replication metric for the content chosen for deletion is bigger than the replication metric of the content chosen for replication, then the content chosen for deletion is not deleted from the server, and the content chosen for replication is not replicated to the server. If the calculated replication metric for the content chosen for deletion is smaller than the replication metric of the content chosen for replication, then the content chosen for deletion is deleted from the server, the content chosen for replication is copied to the server, and the new replication and deletion metrics are calculated for both contents. This phase of the algorithm is repeated C times, where C is the number of contents in the system. The description of the second phase of the algorithm with a pseudo-code is presented in Table I

```
for $i = 1, i < C$
   maxRMid = max(RM)
   maxDMid = max(DM)
   stored = 0
   for i=1, i $<$ C
      stored = stored + $A(s, i)$
   end
   if stored $>=$ SC(s)
      A(s,maxDMid) = 0
      tmpRM = (S-calc(maxDMid))(C-rank(maxDMid)+1)
      if tmpRM $<$ RM(maxRMid)
         DM(maxDMid) = 0
         RM(maxDMInd) = tmpRM
         A(s,maxRMid) = 1
         DM(maxRMInd) = rank(maxRMid)·calc(maxRMid)
         RM(maxRMid) = 0
         delete(maxDMid,s)
         copy(maxRMid,s)
      else
         A(s,maxDMid) = 1
      end
   else
      A(s,maxRMid) = 1
      RM(maxRMid) = 0
      DM(maxRMid) = rank(maxRMid)·calc(maxRMid)
      copy(maxRMid,s)
   end
end
function tmp = calc(id)
   tmp = 0
   for i=1, i¡(S-1)
      tmp = tmp + (S-i)A(i,id)
   end
   return tmp
end
```

TABLE I
Pseudo-code of the redistribution algorithm

V. Simulations and results

In this section we present the experimental results for the proposed model obtained by simulations. We developed a simulation model of the proposed system using the discrete event simulator OMNeT++ [8] and the INET library [9] that defines the network protocols used for exchange of data. The network in our simulations consists of $N = 1500$ clients and streaming servers structured in three-level hierarchy. Each level of the hierarchy is represented by one server with the streaming capacity of all the servers in the level. The representative of the servers that are closest to the clients has a streaming capacity of $U(1) = 300$ simultaneous streams, the server in the level above has capacity of $U(2) = 750$ streams and the server of the last level has the sufficient capacity to serve all the requests that cannot be served by the first two levels of servers. The servers in the first two levels have also limited storage capacity which, in our simulations, are $SC(1) = 300$ and $SC(2) = 600$ videos. The server in the last level has the capacity to store all the videos. In our systems we consider a video library of $C = 1500$ videos with average duration of $d = 80$ time units. The process of requests for videos obeys the Poisson process, such that the inter-arrival time has exponential distribution with average time of $w = 20$ time units. After the streaming is completed, the clients continue requesting strips with the same inter-arrival time. The videos are initially randomly distributed among the servers. The simulations were run within a period of $T = 1500$ time units.

We model the popularity of the video contents in the library with the Zipf-Mandelbrot (ZM) distribution [10]. It is obtained from the Zipf distribution, commonly used for modelling popularity of web pages, by introducing a shifting constant q. According to this distribution, the popularity of the item with rank i is defined as

$$f(c) = \frac{(c+q)^{-\alpha}}{\sum_{i=1}^{c}(i+q)^{-\alpha}} \tag{3}$$

where α is skew factor that defines the dispersion between popular and unpopular contents, with typical value between 0.6 and 1, and q is a shifting constant introduced to better describe the users behaviour when they request videos. This behaviour consists in that once a user watches a video it will not repeat a request for the same video, but it would rather watch another video from the library. This behaviour reduces the steepness of the Zipf distribution obtained for $q = 0$. In the simulations we use the values $\alpha = 0.8$ and $q = 10$.

For the estimation of the performances of our redistribution algorithm, we define a transport cost function obtained by the generated streaming traffic from each level in the hierarchy and the distance of the level from the clients. The transport cost of the system in a given time t is:

$$Cost(t) = \sum_{l=1}^{L} U(l,t)d(l) \tag{4}$$

where $U(l,t)$ is the streaming traffic from level l, $d(l)$ is the distance of that level from the clients, and L is the number of levels in the hierarchy. We also define a reference minimum cost function as the minimum cost that has to be paid for the streaming traffic generated in a given time t. The algorithm for the calculation of this value is shown in Table II.

```
MinCost(t)=0
for l=1, l<L
    Usum(t)=Usum(t)+ U(l,t)
end
for l=1, l<L
    MinCost(t)=MinCost(t)+max(0,Usum(t)-CU(l))*d(l)
    Usum(t)=Usum(t)-CU(l)
end
```

TABLE II
ALGORITHM FOR CALCULATION OF MINIMUM TRAFFIC COST

The performances of the system in various simulation scenarios will be measured with the relative difference of the real cost that has to be paid with a given content distribution and the minimal cost that could be achieved in the best scenario when the contents are optimally placed, compared to the minimal cost. We call this measure a relative additional cost and we calculate it according to the following expression:

$$\Delta(t) = \frac{Cost(t) - MinCost(t)}{MinCost(t)} \cdot 100 \tag{5}$$

We also define a gain function that is the average gain (savings)in transport cost obtained as a result of the implementation of the algorithm. If the simulation are run T time units and the algorithm is run in time moment τ time units, than the gain is defined as the difference of the average transport cost before running the algorithm and the average transport cost after running the algorithm relative to the average minimal cost within the whole simulation interval. The gain is calculated as:

$$\begin{aligned} G = \; & \left(\tfrac{1}{\tau}\sum_{t=0}^{\tau} Cost(t) - \tfrac{1}{T-\tau}\sum_{t=\tau+1}^{T} Cost(t)\right) \cdot \\ & \cdot \left(\tfrac{1}{T}\sum_{t=0}^{T} MinCost(t)\right)^{-1} \end{aligned} \tag{6}$$

In the first simulation scenario, we distribute the contents randomly on the servers until all the storage capacity is occupied. After 700 time units we run the algorithm in order to optimally distribute the videos according their popularity, obtained from the data collected within the simulation. The results are shown in Figure 3. The figure shows that with the random distribution of the contents, the relative additional traffic cost that has to be paid for streaming the videos is 20% higher than the minimum traffic cost. The execution of the algorithm contributes to optimal distribution of the contents and, therefore, the relative additional traffic cost rapidly falls such that the transport cost is close to the minimum cost.

In Figure 4 is shown the overall capacity utilization of the system during the simulation. It gives an inside view of the distribution of the maximum streaming capacity of the system among the streaming server in each level. In this simulation the streaming capacity of the servers are 25%, 35% and 50% of the maximum streaming capacity of the system, for the first, second, and third level servers respectively. As can be seen, without the execution of the algorithm, most of the requests are served by the server in the third level since it contains all the videos, while the rest of the servers contain randomly distributed videos. With this distribution, non of the servers reaches its maximum utilization. After the execution of the algorithm, the utilization of the server in the first level is maximized, the utilization of the server in the second level is increased close to its maximum, while the utilization of the server in the third level is considerably reduced, resulting in lower additional cost of the system.

In our next simulation scenario, we start the simulation with already optimally distributed content on the servers so that the additional cost is close to the minimum cost. Then, after 300 time units, we change the popularity of the contents.

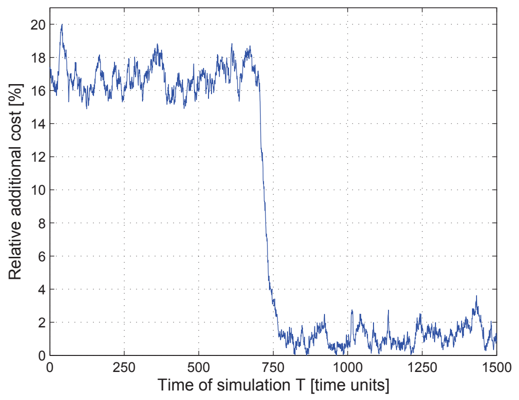

Fig. 3. Relative addtional traffic cost obtained for randomly distributed contents before and after execution of the redistribution algorithm.

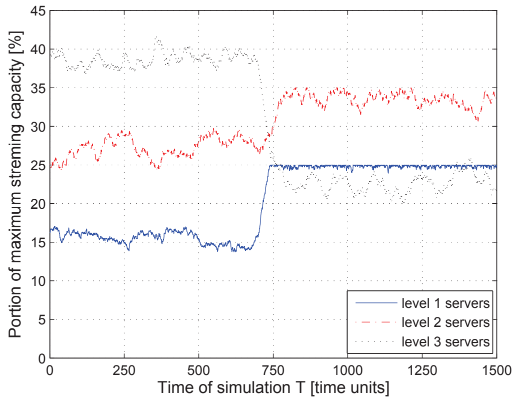

Fig. 4. Utilization of the overall streaming capacity utilization for randomly distributed contents before and after rhe execution of the redistribution algorithm.

The change of the popularity of the contents is achieved by a function that shifts the odd elements of the popularity array two positions to the right, and the even elements of the array two positions to the left. With this function, we simulate the specific nature of the videos to become more popular when they appear in the system and once they reach the peak value of the popularity, they become less popular since most of the clients watched them and would not make another request for the same videos. The unopular videos are then removed from the servers and replaced by new videos. The function for changing the popularity that we use, is a round process that after C iterations returns to the original popularity of the contents, where C is the number of contents in the system. We simulate the introduction the life cycle of one video by calling the function C times.

The behaviour of the system as a result of shifting the popularity of the videos 500 times is shown in Figure 5. We can see that at the beginning the system is in optimal state, but after the change of the popularity the relative additional traffic cost raises up to 20% and remains like that until the redistribution algorithm is run. The figure shows that in a very short period of time, the new distribution of the contents achieves to reduce the traffic cost of the system back to the same value before the popularity change.

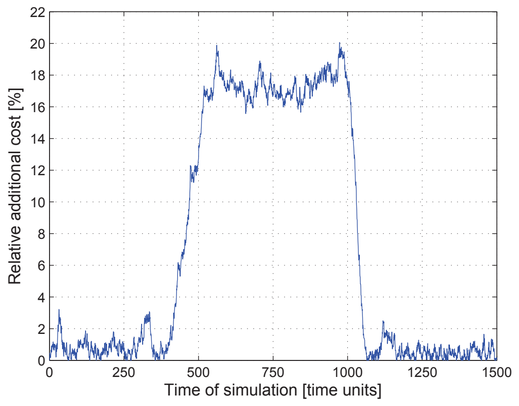

Fig. 5. Relative addtional traffic cost obtained for optimally distributed contents with change of popularity before and after execution of the redistribution algorithm.

In Figure 6 we show the relative cost gain obtained with the utilization of the algorithm as result of the change of the popularity of the contents for different values of the shifting parameter. It can be clearly seen that for environments where there is very big change of the popularity, the traffic cost savings from running of this algorithm are more than 25%. Another interesting remark from this figure is that for changes of popularity obtained by shifting the content library of up to 300 positions, there is almost no cost gain with the execution of the algorithm since these changes do not make a distinctive increase of the overall traffic cost.

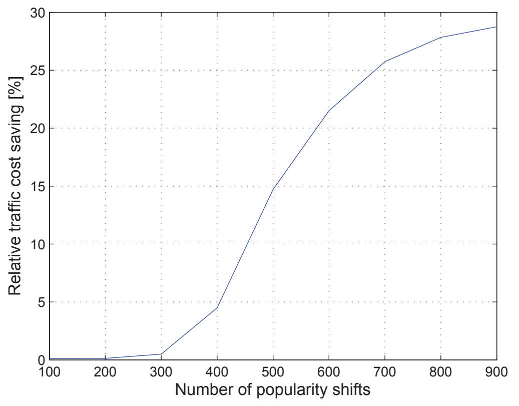

Fig. 6. Relative cost gain as a result of the change of the change of the popularity of the contents for different values of the shifting parameter.

VI. Conclusions

In this paper we propose a hierarchical system for optimal streaming and distribution of VoD contents in managed networks. The system implements a redistribution algorithm that uses the current popularity of the contents to make decisions that optimize the network utilization and improve the quality of service received by the clients, by placing the contents as close to the clients as possible. With the results from our simulations we prove that the algorithm is very responsive and reaches the defined objectives for improved quality of service and optimal network utilization. It redistributes the contents according to their popularity and thus, starting from a random distribution of the contents, it reduces the relative additional traffic cost and achieves a traffic cost savings of more than 25% in environments with very big change of popularity of the contents, maximizes the utilization of the servers that are closer to the clients, and achieves an optimal distribution in a very short period of time. There are several advantages of the optimal distribution: the traffic is concentrated to the edges of the network thus providing less congested network and a better utilization of the network resources; the time a client has to wait for a service is reduced to close to the minimum; and more clients are simultaneously served.

References

[1] R. Buyya, M. Pathan, and A. Vakali, "Content Delivery Networks," *Lecture Notes In Electrical Engineering*, p. 418, 2008. [Online]. Available: http://portal.acm.org/citation.cfm?id=SERIES11872.1457653

[2] S. Borst, V. Gupta, and A. Walid, "Distributed caching algorithms for content distribution networks," in *Proceedings of the 29th conference on Information communications*. Piscataway, NJ, USA: IEEE Press, 2010, pp. 1478–1486. [Online]. Available: http://dl.acm.org/citation.cfm?id=1833515.1833726

[3] M. Yang and Z. Fei, "A model for replica placement in content distribution networks for multimedia applications," in *Proceedings of IEEE ICC*, vol. 1, may 2003, pp. 557 – 561 vol.1.

[4] W. Simpson and H. Greenfield, *IPTV and Internet Video: Expanding the Reach of Television Broadcasting*. Elsevier Science & Technology, 2009.

[5] M. Verhoeyen, D. De Vleeschauwer, and D. Robinson, "Content storage architectures for boosted IPTV service," *Bell Labs Technical Journal*, vol. 13, no. 3, pp. 29–43, 2008.

[6] X. Zhou and C. Xu, "Efficient algorithms of video replication and placement on a cluster of streaming servers," *Journal of Network and Computer Applications*, vol. 30, no. 2, pp. 515–540, Apr. 2007.

[7] J. Dukes and J. Jones, "Dynamic RePacking: a content replication policy for clustered multimedia servers," in *Proceedings of the Microsoft Research Summer Workshop, Cambridge, England*. Citeseer, 2002.

[8] A. Varga and R. Hornig, "An overview of the omnet++ simulation environment," in *Proceedings of Simutools*, ICST, Brussels, Belgium, 2008, pp. 1–10.

[9] A. Varga. (2013) Inet framework. [Online]. Available: http://inet.omnetpp.org

[10] B. Krogfoss, L. Sofman, and A. Agrawal, "Caching architectures and optimization strategies for IPTV networks," *Bell Labs Technical Journal*, vol. 13, no. 3, pp. 13–28, 2008.