Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 4, Issue. 5, May 2015, pg.218 – 224

RESEARCH ARTICLE

Virtualization Technique to Optimize Cloud Resources for Delivering Secured and Simplified IPTV Service

[1]Prof. K.N. Shedge, [2]Mukesh B Thakare, [3]Nikhil B. Dhanrale, [4]Smita V. Shirsath
Final Year, Computer Engineering, S.V.I.T. Nashik, Pune University, India
[2] mukeshthakare77.mt@gmail.com
[3] dnikhil07@gmail.com
[4] smita.vk1992@gmail.com

Abstract—Optimizing cloud-based services can take advantage of statistique multiplexing across application to yield operator saving to the significant cost. But achieving similar benefits with real-time services, this real-time services are challenged. In this paper, we search to reduce a provider's cost of real-time IPTV through an optimizing IPTV structure and through intelligent time shifting of delivery services. We choose benefit of the differences in the deadlines join with Live TV versus video on demanding [VoD] these services also effectively multiplexing.

We provide a generalized framework to support multiple services for computing the amount of resources needed, outwith missing the boundary for any services. We build the problem as an virtualization or optimization formulation that note generic cost function. We observe multiple forms for the cost function to mirror the different pricing options. To answer to this formulation move the number of servers required at different time moment to support these services. We implement a plain mechanism for time shifting scheduled task in a system and study the transformation in server load using true traces from an operational IPTV network. Our research result show that we reduce the load by 23% compared to possible. We also provide that there are interesting problems in planning mechanisms that permit time shifting of load in such environments.

Keywords—VoD, IPTV, ICC, Live TV

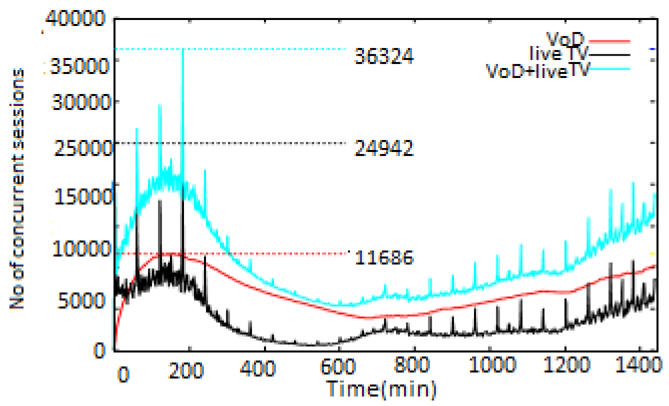
INTRODUCTION

The IP- based video broadcasting becomes more popular, the IP-based video delivery demands placed upon the services providers resources have dramatically raise. Service providers common supply for the peak demands of each service opposite the subscriber population. After all, provisioning for peak demands leaves resources under used at all other periods. This is particularly evident with instant channel change requests in IPTV. The IPTV, Live TV is simply multicast from servers using IP multicast, with one group-set per TV channel. VoD is also supported by the services provider, A unicast stream used server with each request being served. When users change channels watching live TV, we provides additional functionality the channel change takes quickly effect. Each channel change, the user has to join the multicast group joined with the channel and wait for enough data to be buffered before the video is played, this can take some time yet. A result ,there have been many try to supported instant channel change by check the user perceived channel switching latency with the typical ICC implemented on IPTV system the content is delivered at an advanced rate using a unicast stream from the server. The playout buffered is filled quickly and thus retain switching latency small. When the playout buffer is filled up to the playout point, the set top-box turn back to receiving the multicast stream.

Instant channel change adds a requirements that is proportional to the number of users concomitant initiating a channel change event. Operational data display that there is dramatic burst load placed on servers by related channel change requests from consumers. This results in big peaks occurring on every half-hour and hour boundaries and is frequently significant in terms of both bandwidth and server I/O capacity .This requirement is served by a big number of servers grouped in a data center for serving single channels and are scaleup as the number of clients increased. After all this demand is transient and normally only lasts several second, possibly upto a pair of minutes. As a result, majority of the servers dedicated to live TV sit idle outside the burst period. our achieving goal in this paper is to take advantage of the difference in workloads of the different IPTV services to good utilized the deplored server. For example, instant channel change workload is very bursty with a big peak to media ratio, VoD has a relatively supported load and imposes not so estric to delay bounds. More importantly. It offers chance for the services provider to supplying the VoD content in prevision and potentially out of order taking advantages of the buffering available at the receivers. We search to minimize the resource requirements for supporting the services by taking advantage of statistique multiplexing across the different services in the sense, we search to satisfy the peak of the sum of the requirements of the services, some what than the sum of the peak demand of each single services when they are handled independently virtualization offer us the capacity to share the server resource across these services.

In this paper, we goal 1) using a cloud computing services with virtualization to dynamically shift the resources in real time to handle the ICC workload.2) the change in the workload Avanti of time and preload VoD content on STBs, damit facilitate the shifting of resources from VoD to ICC during the burst.3) solve a general cost optimization problem formulation outwith having to careful model each and every parameters setting in a data center easier this resource shift.

In virtualized environment thus ICC is managed by a set of VMS used to be created to handle VoD requests.With the capacity to spawn VMs quickly. We can shift server from VoD to handle the ICC requirements in a matter of a minimum seconds. Use that by being able to known the ICC burst fig[1] shows the action of the VoD load and the average load for both services. In prevision of the ICC load. We can search to accelerate delivery of VoD content to the users STBs and shift the VoD requirements off from the ICC burst interval. This leave also ensure that VoD user will not notice any impairment in their delivered quality of service as the playout can be from the local STB cache



In preparation work on this topic, we analyzed the large number of server that are desired to services jobs with a strict boundary constraint. We also consider non-causal data. Of the jobs arrive at the each instant. This paper, we assume a generalized cost function for the servers. The cost of servers in this creation can be a function of time, load. Etc. our aim is to find the number of servers at each time instant by minimizing this generalized cost function while at the same time shifting all the boundary constraints. We established the server ability region formed by servers at each time period such that all the jobs arriving meet their boundary. The defined as the reference such that for any server order pair with integer entries inside this reference. All boundaries can be met and for any server order pair with integer entries outside this reference, there will be at minimo one request that misses the boundaries. We shows that for any server ordered paired with integer entries inside the server ability reference, an earliest boundary first strategy can be used to distributed all request without missing their boundary/deadline. This denotation of previous results in the written works where the number of server are fixed at the all time. The server ability reference is formed by lineal constraints, and thus this reference is a polytope.

We displayed the examples of cost function for calculating the number of server in part namely, the large and piecewise lineal convex cost function. We set up a number of simulations to detect the effect of varying firstly, the ICC time and secondly, the VoD defer tolerance on the total number of servers desired to accommodate together workload. Our locating indicates that potential server bandwidth savings of [25%-30%] can be actualized by anticipating the ICC load damit smoothing the VoD load ahead of the ICC burst. Endly show by means of a faithful feigns implementing both these services in this part of section , that meticulously choice of look ahead smoothing window can help to meet aggregate the additional VoD load. Eventually our approach only requires a server complicated that is sized to meet the demands of the ICC load, which no boundary flexibility, and we can totally mesh the desire for any additional servers for boundaries with the VoD load.

RELATED WORK

There are mainly three threads of related work namely cloud computing scheduling with deadline constraints and optimization. Cloud computing use internet based computing. Where by shared tool of configurable computing resources can be providing and same infrastructure support multiple services. Due to its nature of provide or serving computationally thorough applications, cloud infrastructure or organization is specific suitable for content delivery application. Normally Live TV and VoD services are operated loyal servers, while this paper considers the choice of multiple operating services by anxiety rebalancing of resources in real time same cloud organization.

Arrival of requests that have to be provided by certain boundary have been deeply studied. For a given set of processors and incoming task characterized by arrival time and necessity to complete by certain boundary, EDF(Earliest Deadline First) timelines the tasks such that each job completed by the boundary. In this paper, there are manifold sets of services providing tasks.each of these services and representative with distinct boundary. For a fix entity of processors, EDF is ideal timeline. In this paper, we catch the region formed by server tuple so that all the representative misses boundary.

Optimization theory is a mathematical method for figure out the most lucrative or small beneficial choice outer of the set of alternatives . dynamic optimization is a sub branch of optimization theory that deals with optimizing the demand control variables of discrete time dynamic system. In this paper we consider possibility optimization where the optimal parameter controls parameter with finite look ahead are to be find. More explicitly, we know the arrival pattern of the IPTV and VoD request with their boundary in the future. We wish to find the number of servers to use ateach time so as to insignificant the cost function. We derive closed form solution for many cost functions.

OPTIMIZATION FRAME WORK

Un IPTV services provider is commonly involved in delivering maximum real time action, such as live TV , VoD and in a few cases, a network based DVR services. The services has boundaries for delivery. Which possibly slightly different, that the playout buffer at the consumer does not under run, resulting in a user perceived insufficient. In this part, we analyze the amount of resources required when multiple real time services eith boundaries are deployed iv cloud infrastructure.

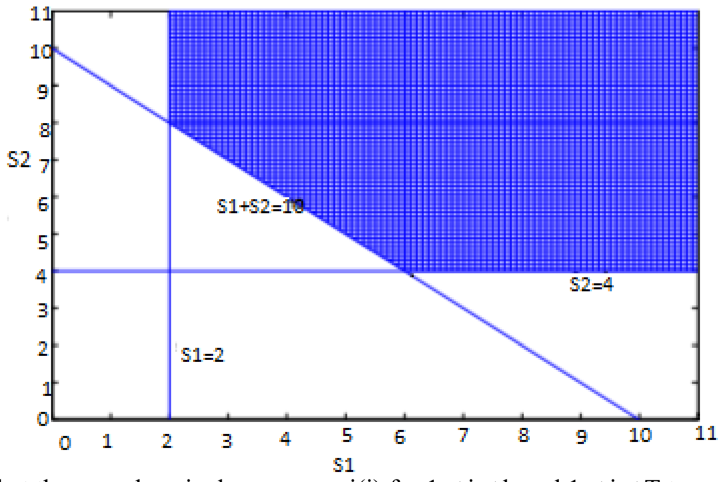
The multiple works in the past to analytically appraisal the resource necessity for serving arriving request which have dealy constraints. The studied especially in the surrounding of voice, such as delivering VoIP packets, and usually consider the arrival process is poisson. We first expand the analysis so that our results apply for any usually process and we also assume maximum services with different boundaries. Our designing algorithm computes the number of servers desired at each time based on the structural workload of request from these different services. The optimization achieving goal is to multiple cost function which depends on the server triplet such that all the boundary/ deadline constraints are happy. We also study the objects of calming the boundary constraint on the optimal cost. for example, the used to take advantage of the same server resources to deliver live TV as well as VoD . their boundary can be different, the case of VoD we can official content in the STB buffer .our research is applicable to the condition where cloud resources is are actively allocated to a services by take advantage of virtualization.

Theorem1: Given that all service requests belong to the same class, the server-capacity region is the set of all server-tuples (s1,s2,...,sT) which satisfy

$$\sum_{n=i}^{i-t+d} r(n) \le \sum_{n=i}^{i+t} s_n \qquad \forall \le i \le i+t \le T, t \ge d, \qquad (1)$$

$$\sum_{n=0}^{l}(T-n) \leq \sum_{n=T-l}^{T} s_n \forall 0 \leq l \leq T. \qquad (2)$$

Equation (1) suggests that the total number of servers needed in time window i and i + t has to be greater than or equal to the sum total of all arriving jobs in time window i and i + t that have deadlines in the same window (ignoring the boundary condition that the jobs have to depart by time- instant T, which is the end of the time interval over which we estimate the required capacity). Equation (2) indicates a boundary condition that all jobs have to be completed and delivered by the time instant T. This theorem is a special case of the following theorem.



Theorem2: Suppose that there are k arrival processes rj(i) for 1 ≤ j ≤ k and 1 ≤ i ≤ T to a queue at time i and no request is arriving for times i > T. Request rj(i) arriving at time i has a deadline of i+dj. In this case, the peak number of servers given by,

$$S = \left[1 \leq i \leq i+t \leq T, t \geq \min(d_1, \ldots.., d_k) \quad \frac{\sum_{j=1}^{k}\sum_{n=i}^{i+t-m_j} r_j(n)}{t+1}\right]$$

For all incoming request is is compulsory and adequate.

Theorem3: When none of the services can have any delay, (or dj = 0), the peak number of servers that is necessary and sufficient is given by max1≤n≤T$\sum_{j=1}^{k} r_j(n)$

NUMERICAL RESULT

A. Maximum cost function

We genrate many experiment to see the variation. That's All the figures we are shown the characteristic activally Vod time series(in blue) and instance channel change time series(in red). colaboration of VoD session with periodic Live TV session is shown by figure5. Figure 6 shows as same effect, the only argument here is an operational tail is used for Live TV. We observed that as VoD requests are defered up to 20 seconds and near about 20 seconds.

B. Convex unitwise linear cost function

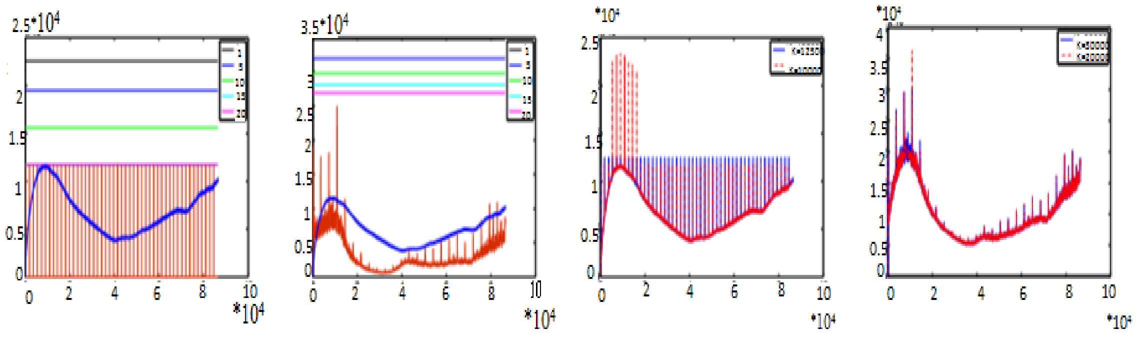
Figure 7 and 8 shows multiple servers needed when a unitwise linear cost function is performed. There are various values of k set a synthetic Live TV of amplitude ≈12000 and operational VoD measurement track. Many number of part s that need to be served with a above cost is larger when k in less. We also need more jobs to served with more cost when value of k is less.

SIMULATION

To state the effectiveness of our proposal in true circumstances of supporting IPTV services, we followed the adjustment mechanism in a custom event driven simulator. We analyze our approach using the VoD & ICC requests seen in a large scale operational IPTV service. The user request traces were collected over 24 hours at a set of VHOs in single state for both VoD & ICC. Overall our data more than 18 million VoD and ICC requests in that time span. Our result states that we can get a proper reduction in peak server load by adjusting the deadlines of VoD requests in anticipation of ICC request. Here we are giving the details of our experiment-

A. Experiment Setup

Generally we second-hand a tradition event driven simulator to perform the experiments. The simulators models the clients and the servers but abstracts, and symbolize by a simple link, the multifaceted network that typically attachs them. We outlook each video is consisting of chunks. We can observe today most profitable streaming systems employ. When a client asks for a video it sends the demand to the server & in response identifies the chunks in video. We set each chunk to be up to 30 seconds long. Each client then requests N chunks in parallel from the server. In our experiments we nominally consider N=25. The server then schedules these requests according to their deadlines & transfers it before the deadline. Upon receiving a chunk, the client requests the next outstanding chunk.We modelled ICC request also as requests for video, we assumed that each ICC request results in 15 seconds of unicast video transferred to the client. As a result, each ICC request results in a request of one 15 second chunk of video that has to be delivered instantly.
It is found that sometimes, there is a sudden burst in ICC request every 30 minutes just for a short duration. We termed this the ICC burst window. We try to minimize the load due to these ICC bursts by advancing the transfer of previously scheduled VoD request. The process we use is depicted. Assuming that the ICC burst window lasts from time t+s to t+b, we start the adjustment process at an earlier point, at time t. In the window from t to t+s, which we call the smoothing window, we advance the jobs already scheduled in the ICC burst window. Thus these jobs are served prior to their deadline. However, in this process we make sure that room for the increased number of ICC requests expected in the burst window should be there. Note that we do not take any special action for the new VoD requests that arrive after the adjustment process has began at time t,but before the end of burst window. As a result, these requests will result in some continued VoD load during the burst window. One could however implement a more proper scheme.



The load depends on many factors If the environment warrants it the reduction. Initially we need to analyze the time of occurring of burst. Next we need to predict how long the burst will show its effect. Also we have to perform to see how many VoD jobs arranged in the burst window have to be moved. And the final time is to start the adjustment process and its time period for averaging the load also plays a important role. We can easily study each of these effects in our experiments.

We assume that our ICC spike occur at fixed 30 minutes intervals. We also assume that our burst window is one minute long, but we also experiment with 2 minute burst. We assume that the adjustment start 10 minute before the burst window and that all the VoD job are randomly adjusted over the smoothing window. The smoothing window is set at 10 minutes. Note that we actually want to use a larger smoothing window than the burst window to spread out the load imposed by the moved jobs; else we can view a spike in the load during the smoothing window, so that analysing the benefit of the procedure somewhat. Our basic metric is the number of simultaneous streams that servers have to serve. We use this as our metric because it directly converts to how many servers are required to supporting the total requested load. In exacting, the peak number of simultaneous stream is most applicable because it gives the minimum number of server

B. Establishing the base line

In figure 1,we individually plot the load due to VoD request only, only ICC request, and the mutual load for both VoD and ICC request. Figure 1 shows that if we did not do any adjustment, we would need to support a maximum of 36234 parallel streams of VoD and ICC requests. If we only consider ICC requests ,these goes to 24942 streams. It reuses more to 11686 streams taking into account VoD alone. remember that ICC request results in 15 seconds of data transfer and are served instantly Hence,best we can do is to go down to 24942 streams if we sustain both services, but are able to mask the VoD service wholly. This gives us a base line best case to evaluate the performance of our proposed adjustment mechanism.

C. Rescheduling jobs reduces load

For the crucial main result, we show that rearrangement jobs to an earlier deadline can be possible and that it can result in a important reduction in total load. We consider an ICC burst window of one minute a smoothing window of 10 minute. We also consider that all the VoD jobs preceding to the burst window are moved to the starting of the smoothing window. The figure 10 shows the trends for the full day to stress the performance during the high period, we put peak period for the day during which the peak number of streams are served. The result of this experiment shows that with the adjustment, we can bring the peak number of concurrent streams down from 36324 streams to 27813 streams,24% decrease. Indicating that we have successfully moved all the VoD requests needed, making way for the ICC burst to be served in the bust window. This number is very close to the load due to ICC requests alone.

As this is a considerable reduction, it is lower than the possible 31% reduction. We attribute the lower gain to the way we work out the time-shifting of the VoD load. Remind that the adjustment of serving VoD requests is done at the start of the smoothing window. However, any VoD requests that come after the adjustment is initiated cannot be rescheduled and results in load during the burst window as well. With a 10 minute smoothing window, we see quite a few of the new VoD requests after the adjustment is complete. To understand this interaction better, we study the effect of varying the size of the smoothing window next.

D. Effect of smoothing window size

The smoothing window size determines how the VoD load from the burst window is dispersed. Choosing a small smoothing window results in more exact determination of how many scheduled VoD jobs present, but could result in a load spike within the smoothing window. On the other hand, a large smoothing window allows us the average the VoD load from the burst window better, but prevents rescheduling of many new VoD sessions that arrive subsequently. VoD jobs are shifted from those 2 minutes. We vary the smoothing window from 2 minutes to 10 minutes and present the result in fig.11. We compute the effect of the smoothing window, while keeping the burst window at 2 minutes. Amusingly, we see that at the peak, using a 5 minutes smoothing window results in better performance than a 10 minute smoothing window. We attribute the improvement to the ability to rescheduled more VoD streams because of smaller smoothing window. However it is not as easy as just employing a smaller window. When we decrease the smoothing window further, to 2 minutes, the load is consistently higher than the other windows. Even more importantly, the 10 minute smoothing window consistently outperforms the other outside the peak viewing period. This is because at the peak period, the number of ICC requests is significantly higher than the VoD requests. Hence moving as many VoD requests as possible is important. At other times, the number of VoD requests is higher. Hence moving the entire VoD request to an earlier deadline increases the load at that time. This is significant; it tells us that we need a more sophisticated approach to predicting the load in burst and in choosing size of the smoothing window.

E. Effect of burst window.

We study the result of the size of the burst window by changing the burst window from 1 minute to 2 minutes, while keeping the smoothing window fixed at 10 minutes. We present the result in figure 12. Interestingly, we see that the size of the burst window has only a small role to play through the peak. Understanding the burst window is important as it tells us how long the burst is going to last. This is because the majority of the load during the peak comes from ICC requests and the new VoD sessions. This again can be known to the fact that the load in these periods is mainly due to VoD and moving more jobs is counter-productive. This is gain because we have many more VoD jobs than necessary. Finally, we see clear reductions in load after the burst window of 2 minute, but not with a burst window of 1 minute.

F. Probabilistically moving jobs

The burst window tells us the period from which we need to move the VoD jobs, and the smoothing window gives us time over which we may schedule them. However, we also need to know how many jobs to move. To capture this we probabilistically moved jobs to the smoothing window. We told the smoothing window at 10 minute and the burst window at 2 minutes but varied the possibility p of moving a job from 0.25 to 1.0 and plot the result in figure 13. We note some attractive performance. First, during the peak we see that increasing the probability of moving jobs decreases the number of simultaneous streams. This result clearly shows that we need a smarter way of figuring out how many jobs to move for this procedure to be applicable in general. However at other times decreasing the probability decreases the concurrent streams.

G. Results summary

In this section we obtainable result from a simple adjustment mechanism we implemented. Our result shows that even our simple mechanism is able to give significant reductions in load. However there is still chance for enhancement. We showed that the load reduction is dependent on the extent of adjustment, the number of jobs moved and the period over which they are averaged. Our result show that a particular value for each of these parameters is not best across the board; instead the value chosen depends on the relative load of each of the services being adjusted. Designing such mechanisms is an opportunity for exciting future work. We believe that mechanisms to predict this relative load of each service and dynamically choose values for the parameters based on this prediction can yield further improvements.

CONCLUSIONS

We considered how IPTV service providers can leverage a virtualized cloud infrastructure and clever time-shifting of load to better utilize deployed resources. Using Instant Channel Change and VoD delivery as examples, we showed that we can take advantage of the difference in workload of IPTV services to schedule them appropriately on virtualized infrastructures. By anticipating the Live TV ICC bursts that happens every half hour we can speed up delivery of VoD content before these bursts by profiling the set top box buffer. This helps us to dynamically reposition the VoD servers to put up ICC bursts that normally last for very short time.

References

1) Otimizing cloud resources for Delivering IPTV Services through Virtualization IEEE Transactions on Multimedia,(Volume:15 , Issue: 4)June 2013
2) D. Banodkar, K. K. Ramakrishnan, S. Kalyanaraman, A. Gerber, and O. Spatscheck, "Multicast instant channel change in IPTV system," in Proceedings of IEEE COMSWARE, January 2008
3) "Microsoft tv: Iptv edition," http://www.microsoft.com/tv/IPTVEdition.mspx.
4) H. A. Lagar-Cavilla, J. A. Whitney, A. Scannell, R. B. P. Patchin, S. M. Rumble, E. de Lara, M. Brudno, and M. Satyanarayanan, "SnowFlock: Virtual Machine Cloning as a First Class Cloud Primitive," ACM Transactions on Computer Systems (TOCS), 2011.
5) V. Aggarwal, V. Gopalakrishnan, R. Jana, K. K. Ramakrishnan, and V. Vaishampayan, "Exploiting Virtualization for Delivering Cloud-based IPTV Services," in Proc. of IEEE INFOCOM (mini-conference), Shang- hai, April 2011.
6) J. A. Stankovic, M. Spuri, K. Ramamritham, and G. C. Buttazzo, Dead- line Scheduling for Real-Time Systems: Edf and Related Algorithms. Norwell, MA, USA: Kluwer Academic Publishers, 1998.
7) N. V. Thoai and H. Tuy, "Convergent algorithms for minimizing a concave function," in Mathematics of operations Research, vol. 5, 1980.
8) R. Urgaonkar, U. Kozat, K. Igarashi, and M. J. Neely, "Dynamic resource allocation and power management in virtualized data centers," in Proceedings of IEEE IFIP NOMS, March 2010.
9) C. L. Liu and J. W. Layland, "Scheduling Algorithms for Multiprogram- ming in a Hard Real Time Environment," Journal of the ACM, vol. 20, no. 1, pp. 46–61, 1973.
10) A. Dan, D. Sitaram, and P. Shahabuddin, "Scheduling Policies for an On-Demand Video Server with Batching," in Proc. of ACM Multimedia, San Francisco, CA, October 1994, pp. 15–23.
11) A. J. Stankovic, M. Spuri, K. Ramamritham, and G. Buttazzo, "Deadline Scheduling for Real-Time Systems EDF and Related Algorithms," 1998, the Springer International Series in Engineering and Computer Science.
12) L. I. Sennott, Stochastic Dynamic Programming and the Control of Queueing Systems. Wiley-Interscience, 1998.
13) D. P. Bertsekas, "Dynamic Programming and Optimal Control," in Athena Scientific, Blemont, Massachusetts, 2007.
14) G. Ramamurthy and B. Sengupta, "Delay analysis of a packet voice mul- tiplexer by the ΣDi/D/1 Queue," in Proceedings of IEEE Transactions on Communications, July 1991.
15) H. Tuy, "Concave programming under linear constraints," Soviet Math 5, pp. 1437–1440, 1964.
16) S. Sergeev, "Algorithms to solve some problems of concave programming with linear constraints," Automation and Remote Control, vol. 68, pp. 399–412, 2007, 10.1134/S0005117907030034.