Expansion of Cloud Resources for Transmitting IPTV Services through Cloud

[1]V.Anusha, [2]U.Usha Rani

[1]M.Tech Research Scholar, Department of CSE,
[2] HOD, Department of CSE
Priyadarshini Institute of Technology & Science, Chintalapudi, India

Abstract—*Virtualized cloud-based services can take advantage of statistical multiplexing across applications to yield significant cost savings to the operator. However, achieving similar benefits with real-time services can be a challenge. In this paper, we seek to lower a provider's costs of real-time IPTV services through a virtualized IPTV architecture and through intelligent time shifting of service delivery. We take advantage of the differences in the deadlines associated with Live TV versus Video-on-Demand (VoD) to effectively multiplex these services. We provide a generalized framework for computing the amount of resources needed to support multiple services, without missing the deadline for any service. We construct the problem as an optimization formulation that uses a generic cost function. We consider multiple forms for the cost function (e.g., maximum, convex and concave functions) to reflect the different pricing options. The solution to this formulation gives the number of servers needed at different time instants to support these services. We implement a simple mechanism for time-shifting scheduled jobs in a simulator and study the reduction in server load using real traces from an operational IPTV network. Our results show that we are able to reduce the load by 28% (compared to a possible 33%). We also show that there are interesting open problems in designing mechanisms that allow time-shifting of load in such environments.*

Keywords: Virtualized cloud-based services, IPTV, Video-on-Demand

I. INTRODUCTION

As IP-based video delivery becomes more popular, the demands placed upon the service provider's sources have dramatically increased. Service providers typically condition for the peak demands of each service across the subscriber population. However, provisioning for peak demands leaves resources underutilized at all other periods. This is particularly evident with Instant Channel Change (ICC) requests in IPTV.

In IPTV, Live TV is typically multicast from servers using IP Multicast, with one group per TV channel. Video-on- Demand (VoD) is also supported by the service provider, with each request being served by a server using a uncast stream. When users change channels while watching live TV, we need to provide additional functionality to so that the channel change takes effect rapidly. For each channel change, the user has to join the multicast group associated with the

channel, and wait for enough data to be buffered before the video is displayed; this can take some time. As a result, there have been many attempts to support instant channel change by mitigating the user perceived channel switching latency [1], [2]. With the typical ICC implemented on IPTV systems, the content is delivered at an accelerated rate using a uncast stream from the server. The play out buffer is filled quickly, and thus keeps switching latency small. Once the play out buffer is filled up to the playout point, the set top box reverts back to receiving the multicast stream.

ICC adds a demand that is proportional to the number of users concurrently initiating a channel change event [1]. Operational data shows that there is a dramatic burst load placed on servers by correlated channel change requests from consumers (refer figure 1). This results in large peaks occurring on every half-hour and hour boundaries and is often significant in terms of both bandwidth and server I/O capacity. Currently, this demand is served by a large number of servers grouped in a data center for serving individual channels, and are scaled up as the number of subscribers increases. However this demand is transient and typically only lasts several seconds, possibly upto a couple of minutes. As a result, majority of the servers dedicated to live TV sit idle outside the burst period. Our goal in this paper is to take advantage of the difference in workloads of the different IPTV services to better utilize the deployed servers. For example, while ICC workload is very burst with a large peak to average ratio, VoD has a relatively steady load and imposes “not so stringent” delay bounds. More importantly, it offers opportunities for the service provider to deliver the VoD content in anticipation and potentially out of- order, taking advantage of the buffering available at the receivers.

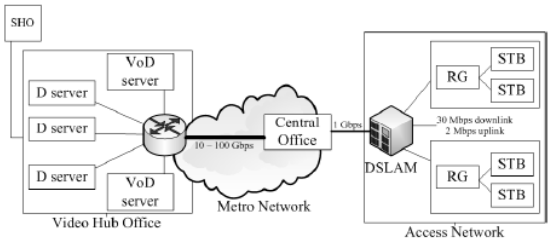


Fig. 2. Typical IPTV architecture

We seek to minimize the resource requirements for supporting the service by taking advantage of statistical multiplexing across the different services - in the sense; we seek to satisfy the peak of the sum of the demands of the services, rather than the sum of the peak demand of each service when they are handled independently. Virtualization offers us the ability to share the server resources across these Services. In this paper, we aim a) to use a cloud computing infrastructure with virtualization to dynamically shift the resources in real time to handle the ICC workload, b) to be able to anticipate the change in the workload ahead of time and preload VoD content on STBs, thereby facilitate the shifting
of resources from VoD to ICC during the bursts and c) solve a general cost optimization problem formulation without having to meticulously model each and every parameter setting in a data center to facilitate this resource shift. In a virtualized environment, ICC is managed by a set of VMs (typically, a few VMs will be used to serve a popular channel). Other VMs would be created to handle VoD requests. With the ability to spawn VMs quickly [3], we believe we can shift servers (VMs) from VoD

to handle the ICC demand in a matter of a few seconds. Note that by being able to predict.

The IPTV provides the following ways to watch television through over air broadcasts and cable signals. The broadcast TV, an antenna picks up radio waves to transmit pictures and sound to your television set. The cable TV, wires connect to your TV & these wires run from your house to the nearest cable TV station and it acts as one big antenna

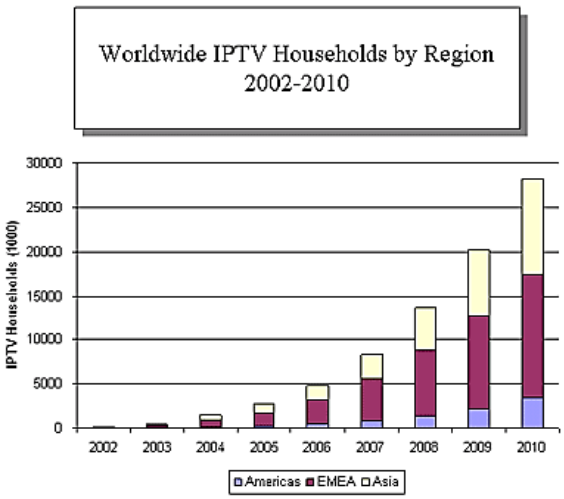
- **TV and Content Head End** - Where the TV channels are received and encoded. Also other content (Video's) are stored at Head End. MTNL has signed agreement for this with M/s Aksh Broadband.
- **Delivery network** - Which is MTNL's Broadband network and MTNL's telephone line? (Landline).
- **Set Top Box (STB)**
  The Set Top Box is required at the customer location for converting the IP signal back to TV signal. The STB shall be connected between MTNL Broadband Modem and customer's TV. The STB shall be provided by M/s Aksh in case of existing Triband. M/s Aksh shall also provide single port ADSL modem, in case of IPTV provisioning on either existing landline or on copper pair without landline.

In this paper, we aim a) to use a cloud computing infrastructure with virtualization to dynamically shift the resources in real time to handle the ICC workload, b) to be able to anticipate the change in the workload ahead of time and preload VoD content on STBs, thereby facilitate the shifting of resources from VoD to ICC during the bursts and c) solve a general cost optimization problem formulation without having to meticulously model each and every parameter setting in a data center to facilitate this resource shift

Live TV, VOD and IPTV through virtualization

	Live TV, VOD &ITV	IPTV through virtualization
Data Transfer	Use general internet	Use dedicated, private network
Whole Geographical Reach	Can be access from anywhere in the globe	Limited by service provider
Service quality &Quantity	Not guaranteed	Guarantees high quality audio and video
Data Access Mechanism	A PC with media player	Set-Top-Box most of the time
Content Generation use	Use own content	Provided by existing TV broadcasters

Growth of IPTV



2. IPTV THROUGH VIRTUALIZATION

2.1. IPTV through Virtualization Strategy

In recent years, cloud storage service has become a faster profit growth point by providing a comparably low-cost, scalable, position-independent platform for clients' data. Since cloud computing environment is constructed based on open architectures and interfaces, it has the capability to incorporate multiple internal and/or external cloud services together to provide high interoperability.

We call such a distributed cloud environment as a multi- Cloud (or hybrid cloud). Often, by using virtual infrastructure management (VIM) , a multi-cloud allows clients to easily access his/her resources remotely through interfaces such as Web services provided by Amazon EC2. There exist various tools and technologies for multicloud, such as Platform VM Orchestrator, Vmware spheres, and Ovirt. These tools help cloud providers construct a distributed cloud storage platform (DCSP) for managing clients' data. However, if such an important platform is vulnerable to security attacks, it would bring irretrievable losses to the clients.

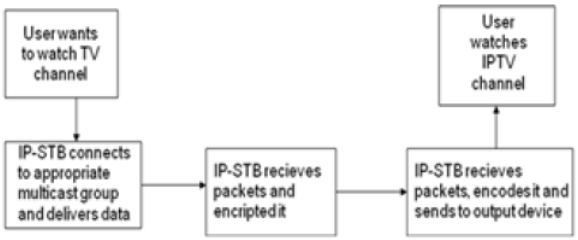
Provable data possession (PDP) (or proofs of retrievability (POR)) is such a probabilistic proof technique for a storage provider to prove the integrity and ownership of clients' data without downloading data. The proof-checking without downloading makes it especially important for large-size files and folders (typically including many clients' files) to check whether these data have been tampered with or deleted without downloading the latest version of data. Thus, it is able to replace traditional hash and signature functions in storage outsourcing. Various PDP schemes have been recently proposed, such as Scalable PDP and Dynamic PDP. However, these schemes mainly focus on PDP issues at untrusted servers in a single cloud storage provider and are not suitable for a multi-cloud environment.

2.2 IPTV Services

IPTV virtualization provides the following services

1. Live TV
2. Video on Demand (VOD)
3. Interactive TV

IPTV-STB Operation



Interactive television is a form of media convergence, adding data services to traditional television technology.

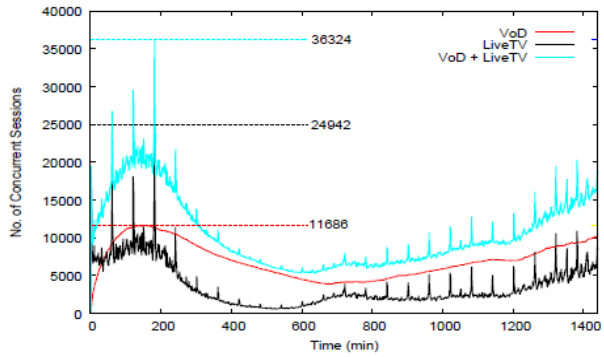


Fig. 1. LiveTV ICC and VoD concurrent sessions vs time, ICC bursts seen every half hour

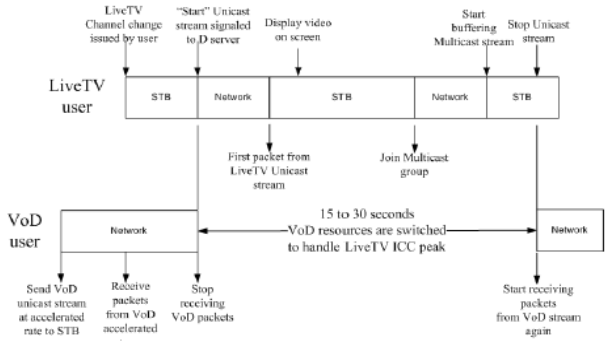


Fig. 2. LiveTV ICC and VoD packet buffering timeline

3. COMPUTATION OF RESOURCES FOR SERVICES WITH DEADLINE CONSTRAINT

There have been multiple efforts in the past on modeling to estimate the resource requirements for serving arrivals within a certain delay constraint, especially in the context of voice processing, including VoIP assuming Poisson processes [5]. We are different from these efforts since our results apply for any general arrival process. Our optimization algorithm computes the minimum number of servers needed based on the sum of the peaks of the composite workload. We also examine the amount of server resources required as the deadline constraint is varied. We then examine the benefit of multiplexing diverse services on a common infrastructure, and show how by dynamically adjusting the resources provided to a particular service while delaying the other service can bring significant savings in resource requirements in comparison to provisioning resources for each of the services independently. This would reflect the amount of 'cloud resources' required with multiple real-time services in the cloud infrastructure.

A. Single Service

Suppose there is a sequence of time instants during which a number of requests arrive to an incoming queue, denoted by c(n) for n = 1, · · ·N. Each request has a deadline of m time units to be completely served after its arrival. In this case, a question arises as to what is the server capacity needed so that all the requests arriving at each of the n time instants are served with no request missing its deadline. When m = 0, no request can have any delay and thus the number of servers that are needed is exactly the peak of c(n) (max1_n_N c(n)). We find the trade-off between the number of servers needed and the deadline constraint tolerable. In addition we will assume that all the requests arriving within a time interval N have to be served within the same time interval. The following theorem gives the number of servers needed to serve the requests within their deadline.

Theorem 1: Suppose that the incoming arrivals to a queue at time i is ci, 1 _ i _ N. Each of the request arriving at time i has a deadline of min(i + m,N). In this case, the number of servers given by, is necessary and sufficient to serve all the incoming requests. In case there is no restriction on all the requests being served by time N, this is equivalent to lengthening the process c

$$S = \left\lceil \max \left\{ \max_{1 \le i \le i+j \le N-m} \frac{\sum_{n=i}^{i+j} c(n)}{m+j+1}, \max_{0 \le k < N} \frac{\sum_{j=0}^{k} c(N-j)}{k+1} \right\} \right\rceil, \quad (1)$$

(i) to a length N +m arrival process where c(i) = 0 for i > N.

This gives us the following corollary.

Corollary 2: Suppose that the incoming arrivals to a queue

are time i is ci, 1 _ i _ N and no request is arriving for times i > N. Each of the request arriving at time i has a deadline of i + m. In this case, the number of serversgiven by,

$$S = \left\lceil \max_{1 \le i \le i+j \le N} \frac{\sum_{n=i}^{i+j} c(n)}{m+j+1} \right\rceil, \quad (2)$$

is necessary and sufficient to serve all the incoming requests.

Corollary 3: When the service cannot have any delay, (or m = 0), the number of servers that is necessary and sufficient is given by max1_n_N c(n).

We will prove Theorem 1 in the remaining part of the section. We will first show the necessity of S servers. There are c(j) requests arriving at time j and at most S requests can leave the queue. If Pi+j n=i c(n) > (m + j + 1)S, the number of requests arriving from times [i, i+j] cannot have departed in m + j + 1 time starting from time i. Thus, some request will miss the deadline. So, Pi+j n=i c(n) _ (m + j + 1)S for all i + j _ N − m. Further, if Pk j=0 c(N − j) > (k + 1)S, the requests arriving in last k + 1 time would not have gone out of the queue. Thus, Pk j=0 c(N − j) _ (k + 1)S for all k < N. This proves that the expressions of S given Theorem 1 are necessary. We will now prove that the number of servers given in Theorem 1 is sufficient. For the achievability, we use a first in- first-out (FIFO) strategy for servicing the queue. We serve the first S packets in the queue at each time based on FIFO strategy if there are more than S packets waiting in the queue. If there are less than S packets in the queue, we serve all the requests. We will show that with S given as in Theorem 1 and using this strategy, no request will miss the deadline.

Consider a time instant i. suppose that the last time before i that the queue became empty be j−1 (There exist such point since the queue was empty at 0 and hence this point would be last if there was nothing else in between). If i < j+m, then the packets that have arrived from j to i have not missed deadline

yet. If i _ j +m, the packets that should have departed from time j to i should be at-least Pi−m n=j c(n) and since this is _ (m+1+i−m−j)S = (i−j +1)S, these packets would have departed. So, no request from time j to i has missed deadline. This is true for all i, j that have deadline m time units away. But, after the last time j−1 when the queue becomes empty, we also need to see if all the requests have been served by time N since deadline for some packets here would be more stringent. Let j − 1 be the last time instance when the queue becomes empty. Then, from that point, number of packets that entered the queue are

PN n=j c(n). This is _ (N − j + 1)S which are packets that can depart from time j to time N. Thus, there are no packets remaining to be served after time N.

B. Extension to more services

In this subsection, we will extend the result in Theorem 1 to more than one services. Let there be k services c1(i), · · · ck(i) for 1 _ i _ N. Each of these services have a deadline associated for the requests, service cj with deadline constraint mj . In this case, the number of servers needed is given in the following theorem.

Theorem 4: Suppose that there are k arrival processes cj(i) for 1 ≤ j ≤k and 1 ≤ i ≤ N to a queue at time i. Request cj(i) arriving at time i has a deadline of min(i + mj ,N). In this case, the number of servers given by (3) at the top of next page, is necessary and sufficient to serve all the incoming requests.

In case there is no restriction on all the requests being served by time N, this is equivalent to lengthening each incoming processes to a length N + max(m1, · · ·mk) arrival process where cj(i) = 0 for i > N. This gives us the following corollary.

Corollary 5: Suppose that there are k arrival processes
cj(i) for 1 ≤ j ≤k and 1 ≤ i ≤ N to a queue at time i and no request is arriving for times i > N. Request cj(i) arriving at time i has a deadline of i + mj . In this case, the number of servers given by,

$$S = \left\lceil \max \left\{ \max_{1\le i\le i+t\le N, t\ge \min(m_1,\cdots m_k)} \frac{\sum_{j=1}^{k}\sum_{n=i}^{i+t-m_j} c_j(n)}{t+1}, \max_{0\le l<N} \frac{\sum_{j=1}^{k}\sum_{i=0}^{l} c_j(N-i)}{l+1} \right\} \right\rceil, \quad (3)$$

$$S = \left\lceil \max_{1\le i\le i+t\le N, t\ge \min(m_1,\cdots m_k)} \frac{\sum_{j=1}^{k}\sum_{n=i}^{i+t-m_j} c_j(n)}{t+1} \right\rceil, \quad (4)$$

is necessary and sufficient to serve all the incoming requests.

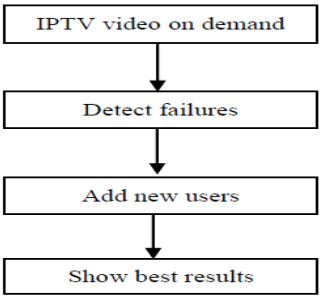
Corollary 6: When none of the services can have any delay, (or mj = 0), the number of servers that is necessary and sufficient is given by max1_n_N Pk j=1 cj(n).

The proof of necessity follows along the same lines as the proof of theorem 1. For the sufficiency, we use the strategy of Earliest Deadline Scheduling rather than FIFO which says that sort the requests by deadline and if there are less than S requests, serve all and otherwise serve the first S requests.

Similar steps prove that none of the request misses deadline with this strategy and thus a detailed proof is omitted.

4. IMPLEMENTATION OF IPTV

To check the availability and integrity of outsourced data in cloud storages, researchers have proposed two basic approaches called Provable Data Possession and Proofs of Irretrievability. Ateniese et al. first proposed the PDP model for ensuring possession of files on un-trusted storages and provided an RSA-based scheme for a static case that achieves the communication cost. They also proposed a publicly verifiable version, which allows anyone, not just the owner, to challenge the server for data possession. They proposed a lightweight PDP scheme based on cryptographic hash function and symmetric key encryption, but the servers can deceive the owners by using previous metadata or responses due to the lack of randomness in the challenges. The numbers of updates and challenges are limited and fixed in advance and users cannot perform block insertions anywhere.



Advantages of IPTV

1. Provable data possession is a technique for a storage provider to prove the integrity and ownership of clients' data without downloading data.
2. It mainly focuses on PDP issues at un-trusted servers for a multi cloud storage provider and is suitable for a multi-cloud environment.
3. Scalable to support storage of data across several CSP's
4. Provides multi-prove zero-knowledge proof system
5. Introduces lower computation and communication overheads in comparison with non-cooperative approaches.

5. CONCLUSIONS

We studied how IPTV service providers can leverage a virtualized cloud infrastructure and intelligent 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 workloads of IPTV services to schedule them appropriately on virtualized infrastructures. By anticipating the Live TV ICC bursts that occur every half hour we can speed up delivery of VoD content before these bursts by prefilling the set top box buffer. This helps us to dynamically reposition the VoD

servers to accommodate ICC bursts that typically last for a very short time. Our paper provided generalized framework for computing the amount of resources needed to support multiple services with deadlines. We formulated the problem as a general optimization problem and computed the number of servers required according to a generic cost function. We considered multiple forms for the cost function (e.g., min-max, convex and concave) and solved for the optimal number of servers that are required to support these services without missing any deadlines.

We implemented a simple time-shifting strategy and evaluated it using traces from an operational system. Our results show that anticipating ICC bursts and time-shifting VoD load gives significant resource savings (as much as 24%). We also studied the different parameters that affect the result and show that their ideal values vary over time and depend on the relative load of each service. Mechanisms as part of our future work.

REFERENCES

[1] 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.

[2] "Microsoft tv: Iptv edition," http://www.microsoft.com/tv/IPTVEdition.mspx. [3] 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.

[4] 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), Shanghai,
April 2011.

[5] J. A. Stankovic, M. Spuri, K. Ramamritham, and G. C. Buttazzo, Deadline Scheduling for Real-Time Systems: Edf and Related Algorithms. Norwell, MA, USA: Kluwer Academic Publishers, 1998.

[6] N. V. Thoai and H. Tuy, "Convergent algorithms for minimizing a concave function," in Mathematics of operations Research, vol. 5, 1980.

[7] 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.

[8] C. L. Liu and J. W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard Real Time Environment," Journal of the ACM, vol. 20, no. 1, pp. 46–61, 1973.

[9] 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.

[10] 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.

[11] L. I. Sennott, Stochastic Dynamic Programming and the Control of Queueing Systems. Wiley-Interscience, 1998.

[12] D. P. Bertsekas, "Dynamic Programming and Optimal Control," in Athena Scientific, Blemont, Massachusetts, 2007.

[13] G. Ramamurthy and B. Sengupta, "Delay analysis of a packet voice multiplexer by the _Di/D/1 Queue," in Proceedings of IEEE Transactions on Communications, July 1991.
[14] H. Tuy, "Concave programming under linear constraints," Soviet Math 5, pp. 1437–1440, 1964.
[15] 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. [Online]. Available: http://dx.doi.org/10.1134/S0005117907030034

[16] D. Banodkar, K. K. Ramakrishnan, S. Kalyanaraman, A. Gerber, and O. Spatscheck, "Multicast instant channel change in IPTV system," January 2008, proceedings of IEEE COMSWARE.
[17] 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).
[18] K. K. Ramakrishnan, R. Doverspike, G. Li, K. Oikonomou, and D. Wang, "IP Backbone Design for Multimedia Distribution: Architecture and Performance," May 2007, proceedings of IEEE INFOCOM.
[19] "Microsoft tv: Iptv edition," http://www.microsoft.com/tv/IPTVEdition.mspx.
[20] G. Ramamurthy and B. Sengupta, "Delay analysis of a packet voice multiplexer by the _Di/D/1 Queue," July 1991, proceedings of IEEE Transacations on Communications.
[21] R. Urgaonkar, U. Kozat, K. Igarashi, and M. J. Neely, "Dynamic resource allocation and power management in virtualized data centers," March 2010, proceedings of IEEE IFIP NOMS.