RESEARCH ARTICLE OPEN ACCESS

Capacity Managed Adaptive Videostreaming Based On Peer Cache Adaptation Mechanism

Seema Safar[1], Sudha S K[2]
M.Tech Scholar in Computer & Information Science,
Sarabhai Institute Of Science & Technology, Trivandrum, Kerala
Associate Professor,
Department of Computer Science & Engineering,
Sarabhai Institute Of Science & Technology, Trivandrum, Kerala

Abstract
To limit the crash against users demand for smooth video, clear audio, and performance levels specified and guaranteed by contract quality, an innovative approach to face these challenges in streaming media is considered. Here a new idea of boosting the capacity of seed servers to serve more receivers in peer to peer data streaming systems is focused .These servers complement the limited upload capacity offered by peers. The peer requests for a data segment is handled by the server or another peer with a seeding capacity of any finite number with a local cache attached in each peer, which enable the peer to temporarily store the data once requested, so it can be directly fetched by some other node near to the peer without accessing the server there by improving the performance of rendering data. The capacity of the cache in each peer can be designed based on popularity of the segment in cache. Once the peers are cached the peer , data segment request are handled by performing a distributed hash table search strategy, and seed servers boost the capacity of each peer based on utility to cost factor computed each time till it exceeds the seeding capacity. Apart from this selfish peers connected in system can be traced to check for unfaithful peers. This system efficiently allocates the peer resources there by considering the server bandwidth constraints.
Keywords—peer to peer streaming systems, resource allocation, scalable video streams

I. INTRODUCTION

Streaming data is a term applied to the compression and buffering techniques that allow one to transmit and view data in real-time through the Internet. Instead of downloading an entire data file, users can download small portions of data files from the Internet. Many engineers utilize streaming, but for numerous applications data cannot be generated or acquired fast enough. In these situations, engineers must compromise by using a slower sample rate to transfer data over the bus or by sampling at the necessary high-speeds for the short periods of time that on board instrument memory allows. Neither sacrifice is desirable.

Video streaming addresses the problem of transferring video data as a continuous stream. With streaming, the end-user can start displaying the video data or multimedia data before the entire file has been transmitted. To achieve this, the bandwidth efficiency and flexibility between video servers and equipment of end-users are very important and challenging problems. The three fundamental challenges in video streaming: unknown and time-varying bandwidth, delay jitter, and loss, must be addressed in video streaming.A typical video streaming system consists of an encoder, a distribution server with video storage,a relay server and end-users that receive the video data. The distribution server stores the encoded video data and begins to distribute the data at the client's demand. Users can watch the video whenever and wherever by accessing the server over the networks. Encoding and distribution is carried out in real time in the case of live distribution and may not be performed in real time for on-demand type of applications.

For video encoding, there are two ways to compress the video signals: non-scalable video coding and scalable video coding. In non-scalable video coding, the video content is encoded independent of actual channel characteristics. In this method, coding efficiency is the most important factor and the compression is optimized at a pre specified rate. The main problem with this method is that it is difficult to adaptively stream non scalable video contents to heterogeneous client varying communication

channels .This is especially true for wireless applications. On the other hand, with scalable video coding, video needs to be encoded only once, then by simply truncating certain layers or bits from the single video stream, lower qualities, spatial resolutions and/or temporal resolutions could be obtained. As an ultimate goal, the scalable representation of video should be achieved without impact on the coding efficiency ,i.e., the truncated scalable stream should produce the same reconstructed quality as a single-layer bit stream in which the video was coded directly under the same conditions and constraints, notably with the same bit-rate.

For the distribution of video bit streams, the video server and relay server are generally responsible for matching the output data to the available channel resources and ultimately the client's device capabilities. For non-scalable video data, the server may transcode the bit stream to reduce the bitrate, frame rate or spatial resolution. Alternatively ,it may select the most appropriate bit stream from multiple pre-encoded streams having different quality, spatial resolution, etc. Considering loss characteristics of the networks, the servers may also add error resilience to the output bit stream. Overall complexity in the system, including servers and clients is another important consideration. In recent years, several P2P streaming systems have been designed and deployed for large scale user communities.

Most of these systems, however, still use non scalable video streams. Among the different challenges that need to be dealt with for running an efficient P2P streaming system, focus on efficient management of the resources of seed servers is done. These servers are needed in high-quality P2P streaming systems to make up for the limited upload bandwidth of peers compared to their demanded download rates. For example, while an average-to-good quality video stream requires about 1-2 Mbps bandwidth, the typical upload capacity of home users with DSL and cable connections is often less than a few hundred kilobits per second.

When serving scalable video streams in a P2P network, the data demanded/possessed by peers gets heterogeneous. Accordingly, allocating seed resources for serving peers' requests arbitrarily, as in most of today's P2P streaming systems, will result in poor management of resources and inefficient utilization of data. Moreover, the flexibility offered by scalable video streams should be appropriately taken advantage of to best satisfy the demands of peers using a given limited resource. The distribution of multimedia streams in large scales over the Internet has been a topic of interest to academia and industry for several years. Due to the lack of IP Multicast deployment in today's Internet and the high cost of purely server-based solutions, the use of peer-to-peer (P2P) technology for Internet streaming has attracted significant attention in recent years. P2P streaming systems do not require any particular support from the Internet infrastructure, and are easy to deploy. In addition, as more peers join the system and demand the data, they also increase the streaming capacity by uploading the streams they receive. The efficient management of resources for serving scalable video streams in P2P streaming systems is required .In today's multimedia streaming systems, clients are getting more and more heterogeneous in terms of connection bandwidth, processing power, and screen resolution. For example, a user with a limited-capability cellular phone with a small screen and wireless connection, and one with a high-end powerful workstation behind cable connection can both be requesting the same video stream. An example of this is illustrated in Figure 1.1.

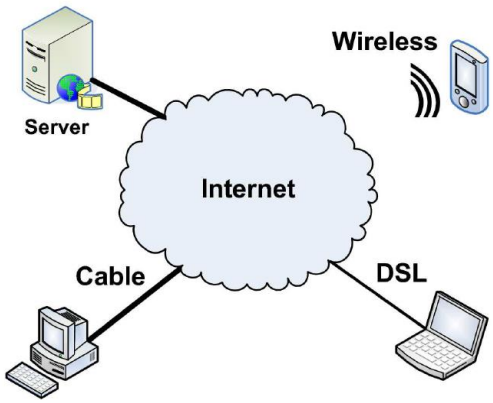

Figure 1.1: Heterogeneous receivers participating in the same streaming session.

To support a wider range of such receivers, it is preferred to encode and distribute a lower-bitrate video stream, but this will provide a low quality for everyone. By encoding a higher-bitrate stream, on the other hand, we cannot support many of the receivers. This problem may be solved by encoding and distributing multiple versions of the video, which is called simulcasting. However, a video has to be encoded many times for different combinations of decoding capabilities, connection bandwidths, and viewing resolutions. Moreover, switching among versions is not easy, because (i) for every switching, a

client has to wait, possibly for a few seconds, for the next Intra-coded frame (I-frame) of the new version, and (ii) the streams of different versions could be asynchronous. As an alternative, Multiple Description Coding (MDC) can encode a video into multiple descriptions, where the quality of the video will be proportional to the number of descriptions received. However, MDC techniques are well known for having considerable bitrate overhead.

II. RELATED WORKS

The problem of peer-to-peer streaming has been studied by numerous previous works from various angles, such as overlay construction, distribution of data availability information, and piece scheduling algorithms. Moreover, most previous works in this area consider streaming of non scalable video streams only.

A. P2P Streaming With Scalable Videos and Seed Servers

Cui *et al.*[3] and Rejaie *et al.*[16] study P2P streaming systems with scalable videos ,focusing on the tasks of peers .An algorithm is presented in [4] to be run on each peer independently to decide how to request video layers from a given set of heteroge neous senders, assuming layers have equal bitrate and provide equal video quality. Hefeeda *et al.*[4] study this problem for fine-grained scalable(FGS) videos, taking into account the rate-distortion model of the video for maximizing the perceived quality. In the framework presented in [18], the problem of requesting from a set of senders is studied. Hu *et al.* [7] design a taxation mechanism for fairness among peers with diverse download and upload bandwidths requesting a scalable video stream.

Lan *et al.* [10] present a high-level architecture for data-driven P2P streaming with scalable videos.Packet scheduling strategies for downloading scalable videos are studied. All of these works do not consider the functionalities of seed servers. Xu *et al.* study the functionality of seed servers for P2P streaming. However, their work is only for non scalable video streams, and they assume that peers upload bandwidth can only take power of 2 bitrates.The case for scalable video streams is more challenging as various sub streams need to be handled. Seed servers are assumed to always have enough capacity to serve all requests, which is not realistic.

III. PROBLEM DEFINITION

In this paper, a more practical scenario in which seed servers have finite capacity, and this finite capacity needs to be optimally allocated to requesting peers such that high quality video is delivered to all peers is considered. The peers need to be cached in order to buffer the received video layers ,so that it can share the segment it has to its requested partners there by improving the system wide utility and hence the rendered quality. A multi-layer scalable video stream is considered, which can be encoded once and can support a wide range of heterogeneous clients, who can decode it. In addition, heterogeneous clients receiving different layers can still share common layers and participate in the same overlay network, leading to a larger pool of resources. Furthermore, scalable coding has lower overhead compared to other coding techniques. Here a P2P streaming systems that: (i) deploy seed servers to complement and boost the capacity contributed by peers, and (ii) serve scalable video streams to support a wide range of heterogeneous receivers are considered.

IV. PROPOSED SYSTEM

PEER-TO-PEER (P2P) and peer-assisted streaming systems have emerged as promising approaches for delivering multimedia content to large-scale user communities. In these systems, peers contribute bandwidth and storage to serve other peers. Since the contributions from peers are often less than the capacity needed to serve high-quality streams, a number of dedicated servers are usually deployed to boost the streaming capacity. These servers are referred to as seed servers. In current P2P streaming systems, a video is encoded at a certain bitrate, typically ranging from 300 kbps to 1 Mbps.

In order to support a wider range of receivers, it is preferred to encode and serve a lower-bitrate video, but this will provide a low quality for everyone. First focus is given on the problem of efficiently allocating the resources of seed servers to requesting peers according to their demands and contributions. This allocation plays a critical role for providing a high-quality streaming service. In the proposed work recent scalable video coding technique H.264/SVC, is considered to improve this coding efficiency Congested networks and overloaded servers resulting from the ever growing number of Internet users contribute to the lack of good quality video streaming over the Internet. Caching system for streaming media utilizes its local memory and disk resources to

reduce network and server load, while also improving the video and audio quality perceived by end users. In this system peers are cached based on the popularity of the segment request and a novel approach for cache replacement is considered in case of zero free space cache. The search of the segment occurs by means of Distributed hash table (DHT).

V. METHODOLOGY

The considered P2P streaming architecture is illustrated in Figure 2.1, consists of trackers, seed servers, and peers. Peers join the system by contacting one of the trackers. The tracker receives periodic update reports from peers, informing it about their available data and capacity. This enables the tracker to monitor its network and keep track of the set of active peers, their contribution, and their data availability. Note that the tracker does not keep track of the topology of the network, i.e., the list of partners of each peer. A number of seed servers exist in the network to serve requests when there is not enough capacity in the peer population.

Our problem is to decide which subset of requests should be served by the seed servers to maximize a system-wide utility function. This problem is important because the volume of requests to be served often exceeds the seeding capacity. Allocating seeding resources optimally will lead to better utilization of seed servers, and higher video quality for users, especially during periods with excessive loads which are typically the most difficult to handle in real systems. Peers are expected to use their limited upload bandwidth for serving lower layers first so as to avoid having some peers starving while other peers are receiving highest rates. Moreover, peers try to serve as many layers as they can upload. For example, if all layers have a rate of 100 kbps and a peer has 250 kbps upload bandwidth, it will upload the two lowest layers at rate 100 kbps and the third one at 50 kbps.

Peers requests are gathered in the tracker's request queue. The tracker decides every 't' seconds, which is a few seconds, and accepts some requests (to be served by a seed server) and rejects others. Let V denote the set of video files in an on-demand session or the set of channels in a live streaming scenario. The video is divided into short time intervals, called video segments, the number of which is T_v for each video v2V. A video segment is considered an atomic unit of adaptation meaning that the number of layers

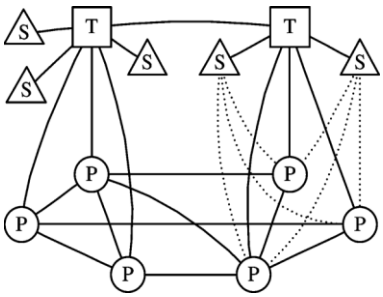

Figure 2.1 Considered P2P streaming model. T, S, and P represent trackers, seed servers and peers, respectively.

received by a peer is assumed constant during a media segment, but may vary between consecutive segments. P_v is the set of peers currently participating in the streaming session of a video v2V.At each time the tracker solves the allocation problem, there are k requests in the queue. Each request req_k is in the form $\{req_k.p, req_k.t, req_k.l_1, req_k.l_2\}$, meaning that peer $req_k.p$ is requesting layers $req_k.l_1$ through $req_k.l_2$ (inclusive) of the stream, starting at segment $req_k.t$, the peer could be receiving layers 1 through $req_k.l_1 - 1$ from other peers.An example of this is illustrated in Figure 2.2 Since req_k is for $n_k = req_k.l_2 - req_k.l_1 + 1$ layers and may be admitted partially, we break it to n_k sub-requests, denoted by reqk,jwhere $1 \leq j \leq n_k$. A sub-request $req_{k,j}$ represents a request for the j lowest requested layers, i.e.,$req_{k,j}$ corresponds to layers $req_k.l_1$ through $req_k.l_1 + j - 1$.

Figure 2.2: A request from a peer who is demanding the first five layers in total, and is receiving the first two from other peers and the next two from a seed server.

The tracker queues the requests received from peers, and solves the allocation problem for existing requests every few seconds. Serving each sub-request $req_{k,j}$ has a cost $c_{k,j}$ for seed servers which is the sum of the bitrates of the j requested layers.Letting v denote the requested video $v = v\ req_{k,p}$ in $reqk$, the

costs of *reqk*'s sub-requests is denoted by:

$$c_{k,j} = \sum_{l=req_k.l_1}^{req_k.l_1+j-1} r_{\nu,l} \quad (1 \le k \le K\ ,\ \ 1 \le j \le n_k).$$

Moreover, by admitting $req_{k,j}$, a utility (benefit) $b_{k,j}$ is gained by the system, which consists of the utility of serving the associated layers to the corresponding peer, that is,

$$\sum_{l=req_k.l_1}^{req_k.l_1+j-1} b_{self}(req_k.p, l)$$

and the utility gained when the peer shares those layers with the network, denoted by

$$\sum_{l=req_k.l_1}^{req_k.l_1+j-1} b_{share}(req_k.p, l)$$

For calculating $b_{share}(p.l)$, we need to consider the peer serving those layers (or part of them) to its partners, those partners serving (partially) to their partners, and so on. Taking these neighborhood details into account requires knowledge of the network topology, which is difficult to maintain for dynamic P2P systems. Therefore $b_{share}(p.l)$ is computed as the expected utility that the system gains when a peer shares some video layers with the network.

VI. ARCHITECTURE

In the system, the server consists of distinct data files, where each data file is encoded and is divided into data segments identified by a segment id. A peer requests a file to the server, the tracker in the server decides on the allocation of seeding capacity resource to peer. The server runs a seeding capacity allocation algorithm based on a greedy approach where each request for a segment is divided into sub segments, which is mentioned as subrequests and subrequest's cost and utility function is computed .The ratio of utility to cost ratio for each subrequests is calculated. They are sorted in decreasing order of ratio which means those requests with greater utility and cost effective one is selected and are kept in an array and served until the assumed seeding capacity is reached.

On reaching the seeding capacity the peer request gets forwarded to server directly ,where the tracker in the server decides on the control of the request of the peer to be granted. The peers connected in the system is identified and checked if the segment containing peer is not cached, otherwise the segment request is forwarded to the peer ,where the segment search in cache occurs through a DHT(distributed hash table). If a peer does not respond to segment requests, its upload bandwidth usage can be reduced. Also if a peer caches less number of segments than desired, it can save its local storage such a peer is classified as *unfaithful*. A 'faith' factor is calculated based on the total number of requests in peer and abnormal number of replies in peer. This value is compared with a threshold value (calculated based on network connectivity factors) .The peer with faith value above the threshold is considered as unfaithful peer. The requested segment is returned to the client and the client can stream the media while at the same time download the data.

System has four modules.

- Seeding Capacity Allocation
- Peer Cache Adaptation
- Tracing
- Administrator

A. Seeding Capacity Allocation

Seed Capacity Allocation Algorithm

SCA(K, n[], b[][], c[][], C)
// K: number of requests, C: seeding capacity
// $n[k]$: number of sub-requests in the k-th requests
// $b[k][j]$, $c[k][j]$: utility and cost of sub-request (k, j)
// Output:
// x[]: number of sub-requests to serve from request #k
// z: total utility gained
1. x[] ⟵ *CreateArrayOfZeros*(K)
2. z ⟵ 0
3. **S**[] ⟵ all sub-requests (k, j) //a total of $\sum_{k=1}^{K} n[k]$
4. Sort **S**[] in decreasing order of utility-to-cost ratio
5. **for** $(k, j) \in$ **S**[] **do**
6. **if** $x[k] > j$ **then**
7. **continue** //subset of an already-served sub-request
8. *cost* ⟵ $c[k][j]-c[k][x[k]]$
9. *utility* ⟵ $b[k][j]-b[k][x[k]]$
10. **if** *cost* $\le C$ **then**
11. $C \longleftarrow C - cost$
12. $z \longleftarrow z + utility$
13. $x[k] \longleftarrow j$
14. **done**
15. **return** x[], z

B. Peer Cache Adaptation

This module mainly give importance to segment length, the cache capacity required for peer and the replacement of cache policy that is to be followed in case of full cache. Each peer requests segments it wants to watch. The segment search is implemented by using DHT. If a peer wants segment *j* that is not

cached at any peer, it should request it to the server. Each peer caches the segments it watched at its local storage and sends these when requested by some other peers. When a new segment needs to be cached, an existing cache should be deleted to create a memory space. DHT is a distributed system and implements a lookup function using the hash table. Given a key, DHT chooses a node that contains a corresponding value to the key. When there are *N* nodes in the steady state, each peer has the routing information of size *O*(log *N*), and each lookup can be resolved using *O*(log *N*) messages.

When peer i requests segment j, it can find peer k which manages the list of peers that store segment j. Peer k picks up a peer l from the list in a round-robin way to balance the load at each peer, and notifies the result to peer i. When a peer stores a segment, it may cache some other segments of the same video with high probability. Using this correlation property, if peer i played segment j received from peer l, it is reasonable to search peer l first for some other segments of the same video. This correlative search technique contributes to reducing much of the DHT lookup overhead.

Two cases need to be considered in deciding the segment length. First is the case that the segment length exceeds the cache capacity of a peer. A possible problem of using a long segment occurs when a peer watches a part of it and stops watching it or jumps to some other segments. In this case, the peer is not able to cache the corresponding segment completely. Second is the case that the segment length is so short to incur much overhead because of too many search requests. So the segment length should be decided considering these together. The segment length to less than 10 seconds is considered here,and use the correlative search to reduce the lookup overhead.

C. Tracing

A selfish peer may disobey the system protocol to increase its payoff. For example, if a peer does not respond to segment requests, its upload bandwidth usage can be reduced. Also if a peer caches less number of segments than desired, it can save its local storage. This module identifies unfaithful peers. Unfaithful peers may not transmit segments they have cached when requested by other peers. The distributed tracing system detects this behavior. The criterion for the tracing a peer is how well it responds to segment requests. Let $N_i req$ denote the number of segment requests of peer *i* received during the last *T* seconds, and $N_i rep$ denote the number of *normal* replies of peer *i* to these requests. A normal reply means that peer *i* transmits the requested segment successfully within a certain bounded delay. Using these,faith f_i of peer *i is defined* as $fi = N_i rep/N_i req$. If f_i is less than a *threshold*, $thr_b (thr_b \leq thr_a)$,peer *i* is eliminated from system for some time. In this case,the server informs other peers of the elimination, and they replace peer *i* with another peer in their neighbor lists and routing tables. If an eliminated peer wants to participate again, it should wait until the server accepts and informs it of other peers' information. The thr_b should be decided according to the network condition such as delay and congestion, as well as each peer's features such as processing power and bandwidth. If thr_b is too large, temporal network or peer failure may be regarded as unfaithful.

D. Administrator

The administrator module acts as a third person who deals with handling of the data files which includes uploading, downloading of the data files from server machine and peer nodes. The status of peer (join, leave or edit) is checked dynamically. The dynamic peer topology is handled and the peer nodes which join the system and leave the system is handled transparently.

VII. CONCLUSION

In this paper in order to improve the rendered video quality and to support more heterogeneous receivers an innovative video streaming system is proposed. The methodology of the system is organized in four modules, where each module is more specific in providing a crystal clear and improved client latency video streaming there by removing the selfish nodes in the network.

REFERENCES

[1] K. Mokhtarian and M. Hefeeda,A Capacity Management of Seed Servers in Peer-to- Peer Streaming Systems With Scalable Video Streams,*IEEE Transactions on Multimedia*, vol. 15, no.1, January 2013.

[2] Annapureddy, S. Guha, C. Gkantsidis, D. Gunawardena, and P. Ro driguez,Exploring VoD in P2P swarming systems,in *Proc. IEEE INFOCOM'07*, Anchorage, AK, May 2007, pp. 2571–2575.

[3] M. Hefeeda and C. Hsu,Rate-distortion optimized streaming of fine grained scalable video sequences,*ACM Trans. Multimedia Comput., Commun., Appl. (TOMCCAP)*, vol. 4, no. 1, pp. 2:1–2:28, Jan. 2008.

[4] X. Hei, C. Liang, J. Liang, Y. Liu, and K. Ross,A measurement study of a large-scale P2P IPTV system, *IEEE Trans. Multimedia*, vol. 9, no. 8, pp. 1672–1687, Dec. 2007.

[5] O. Hillestad, A. Perkis, V. Genc, S. Murphy, and J. Murphy, Adaptive H.264/MPEG-4 SVC video over IEEE 802.16 broadband wireless net- works,in *Proc. Packet Video Workshop (PV'07)*, Lausanne, Switzer- land, Nov. 2007, pp. 26–35.

[6] H. Hu, Y. Guo, and Y. Liu,Mesh-based peer-to-peer layered video streaming with taxation,in *Proc. ACM Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV'10)*, Amsterdam, The Netherlands, Jun.2010, pp.27–32.

[7] H. Kellerer, U. Pferschy, and D. Pisinger, *Knapsack Problems*. NewYork: Springer, 2004.

[8] R. Kumar, L. Yong, and K. Ross,Stochastic fluid theory for P2P streaming systems,in *Proc. IEEE INFOCOM'07*, Anchorage, AK, May 2007, pp. 919–927.

[9] X. Lan, N. Zheng, J. Xue, X. Wu, and B. Gao,A peer-to-peer architecture for efficient live scalable media streaming on Internet, in *Proc. ACM Multimedia Conf.*, Augsburg, Germany, Sep. 2007, pp. 783–786.

[10] B. Li and J. Liu,Multirate video multicast over the Internet: An overview,*IEEE Network*, vol. 17, no. 1, pp. 24–29, Feb. 2003.

[11] S. Liu, R. Zhang-Shen, W. Jiang, J. Rexford, and M. Chiang,Perfor- mance bounds for peer-assisted live streaming,in *Proc. ACM Conf. Measurement and Modeling of Computer Systems (SIGMETRICS'08)*, Annapolis, MD, Jun. 2008, pp. 313–324.

[12] K. Mokhtarian,*Efficient and secure delivery of scalable video streams*,Master's thesis, Simon Fraser Univ., Surrey, BC, Canada,2011

[13] Jongtack Kim and Saewoong,Peer Cache Adaptation for Peer-to-Peer Video-on-Demand Streaming,*JOURNAL OF COMMUNICATIONS AND NETWORKS*, VOL. 14, NO. 3,JUNE 2012

[14] X.Hei, C. Liang, J. Liang, Y. Liu, and K.Ross,A measurement study of a large-scale P2P IPTV system, *IEEE Trans. Multimedia*, vol. 9, no. 8, pp. 1672–1687, Dec. 2007**.**

[15] H. Hu, Y. Guo, and Y. Liu,Mesh-based peer-to-peer layered video streaming with taxation,in *Proc. ACM Workshop on Network and OperatingSystem Support for Digital Audio and Video (NOSSDAV'10)*,Amsterdam, The Netherlands, Jun. 2010.

[16] R. Rejaie and A. Ortega, "PALS: Peer-to-peer adaptive layered streaming,in *Proc. ACM Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV'03)*, Monterey, CA, Jun. 2003, pp. 153–161.

[17] T. Schierl, T. Stockhammer, and T. Wiegand,Mobile video transmission using scalable video coding,*IEEE Trans. Circuits Syst. Video Technol.*, vol. 17, no. 9, pp. 1204–1217, Sep. 2007.

[18] J. Lee, J. Choi, K. Park, and S. Bahk,Realistic cell-oriented adaptive admission control for QoS support in wireless multimedia networks,*IEEE Trans. Veh. Technol.*, vol. 52, no. 3, pp. 512–524, May 2003.