Chapter 1

Broadcasting Techniques for Video-on-Demand in Wireless Networks

Duc A. Tran[♮] and Thinh Nguyen[♯]
[♮]Department of Computer Science, University of Dayton, Dayton, OH 45469
[♯]School of EECS, Oregon State University, Corvallis, OR 97331

Keywords: Video on demand, video streaming, ad hoc networks, wireless networks, mobile networks, periodic broadcast.

Contents

1.1 Introduction

Today's wireless technologies such as IEEE 802.16 (a.k.a., WiMAX) [3] for long-haul communications and IEEE 802.11 (e.g., WiFi) [15] and Bluetooth [5] for short distances are widely deployed. While WiFi is suitable for a small local wireless network, WiMAX (Worldwide Interoperability for Microwave Access) allows for communications over tens of kilometers and, thus, can be used to provide a high-speed wireless "last-mile" service to local networks. As beneficiaries, users can move freely without disconnection from the network. Wireless networks are, therefore, an excellent infrastructure for important applications such as disaster relief efforts and military collaborative networks.

Being wireless and mobile also means that users may enjoy ubiquitous entertainment services. For example, they may play game or watch videos of their interest online wherever they are. This chapter focuses on technologies that enable mobile video-on-demand (VOD) services. A VOD system is an interactive multimedia system working like cable television, the difference being that the client can select a movie

from a large video database stored at a distant video server. Unlike other video services such as pay-per-view (PPV) and video in demand (VID), individual VOD clients in an area are able to watch different programs *whenever* they wish to, not just in time as in VID or pre-scheduled as in PPV. A VOD system is, therefore, a realization of the video rental shop brought into the home.

A mobile wireless VOD a system has many practical applications. For instances, airlines could provide VOD services in airport lounges to entertain passengers on their own PDA (personal digital assistant) while they are waiting for a flight; a museum could provide video information on the exhibits on demand over the wireless network; in education, a university could also install such a system on campus to allow students to watch video recorded earlier from lectures they were not able to attend.

1.1.1 VOD on the Internet

VOD services are already available on the Internet. News video clips can be rendered on demand on most Web media outlets (e.g., cnn.com, espn.com). Video commercials in business areas ranging from automotive to real estate to health and travel can also be played on demand via a product of Comcast called ComcastSpotlight [7]. The major server providers for VOD deployment include Motorola On-Demand Solutions [19], SeaChange International [23], and Concurrent Corp. [8]. Informa Telecom[1] predicted a revenue of more than 10.7 billion US dollars from VOD services offered to more than 350 million households by 2010. The future of VOD business is very bright.

The designs for current VOD systems can be categorized into three main approaches: client/server, peer-to-peer, and periodic broadcast. They are described below:

Client/server [10,11,14,31]: The video content is stored at the server. Each client connects to the server and plays the requested video from the server, independently from other clients' transactions. The server workload may be heavy if there are many client requests. Proxy servers can be deployed to reduce this load. Alternatively, clients requesting the same video can be grouped into a single multicast sent by the server. The client/server approach is used in most commercial VOD products nowadays because of its simplicity and central management.

Peer-to-peer (P2P) [20, 25, 26, 32]: The motivation for P2P is due to the bottleneck problem at the server side. In this approach, not only the video server, but also clients can provide the video content to each other, thus saving server bandwidth. For example, using P2P, a TiVo subscriber could, not only replay a video previously recorded in his or her TiVo box, but potentially play a video recorded in some other subscriber's box without requesting the TV central server. Most P2P VOD systems are implemented in laboratory settings, but we expect them soon to be in the market.

Periodic broadcast [1, 12, 22, 24, 28]: For a popular video, it is better for the server to proactively and periodically broadcast it to all the clients, rather than send it out reactively upon a client request. The advantage of the broadcast approach is that

[1] http://www.informatm.com

the server bandwidth required is constant and the system can satisfy any number of clients. It avoids the bottleneck problem of the client/server approach and the service vulnerability of the P2P approach.

These three approaches are like "apple and orange" when it comes to comparison because each approach is effective for some subset of VOD applications. Video popularity is known to practically follow a 80/20-like rule of thumb [9]; that is, most clients would be interested in only a few popular videos. As such, periodic broadcast should be the best design for transmitting popular videos to a large number of clients, while client/server and P2P techniques are better suitable for non-popular videos or for videos requested by a small client population.

1.1.2 VOD for Mobile Wireless Users

When deployed to a wireless environment, the success of existing VOD designs may not remain. The first reason is due to the limitation of wireless bandwidth. Because the most widely used form of wireless communications in a local area is using IEEE 802.11 technologies (a, b, or g), the network bandwidth shared by all the users covered by an access point is typically no more than 54Mbps. Therefore, no more than 36 MPEG-1 video streams can be delivered simultaneously to this local area. Taking into account signal inference and distance factors, the effective number of such streams would be much fewer.

The second reason is due to the limited coverage of wireless transmission. An 802.11-enabled host can only reach other devices within 100m of its radius, while that radius is 10m for Bluetooth. If a user is too far away from any access point, how can it get the video service? Fortunately, wireless hosts are nowadays able to concurrently participate in multiple connections: with the access point in the infrastructure-based mode and with a nearby host in the ad hoc mode. Therefore, it is possible that a distant user could get the video service from the access point via several intermediate intermediate hosts. The problem is that significant amounts of bandwidth and energy of the intermediate mobile hosts are consumed. We do not have this problem with the typical Internet.

So, we have the following two questions:

1. What should be the architecture for a mobile wireless VOD system?
2. What should be the communication protocol for a client to download a video from the video server?

To cope with the limited wireless bandwidth, the client/server approach should not be the first design option for VOD in a local wireless area. Also, as each wireless transmission is a broadcast where every host can hear, it is natural to think that it would be more efficient if we adopt the broadcast approach. It would be best if we apply the broadcast approach to the most popular videos and the client/server approach only for ad hoc unpopular video requests.

To cope with the limited wireless coverage, we should allow sharing of video contents among the users. For instance, instead of playing a video through multiple hops

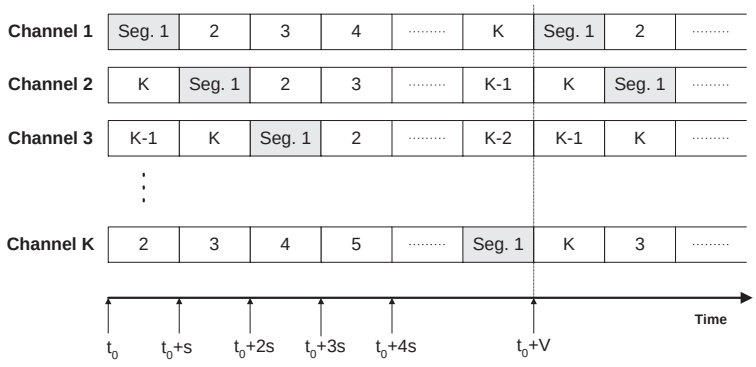

Figure 1.1: Broadcasting video segments at the server: s is the duration of a segment, t_0 the start time of server broadcast, and V the duration of the video

from an access point, we hope to play the video or part of it in some existing users nearby. In other words, we should adopt the P2P approach for this kind of users.

In the next section, we review existing periodic broadcasting techniques for VOD on the Internet and their potential for deployment in a mobile wireless network. We then introduce MobiVOD, a mobile VOD technique that combines the strength of periodic broadcast and P2P approaches.

1.2 VOD Periodic Broadcast Techniques

Cable TV and Satellite TV follow the broadcast approach, in which TV programs are broadcast at prescheduled time slots. This simple broadcast schedule is not designed for VOD because if a TV client wants to watch a movie instantly, the movie will not be shown until the scheduled time. A TiVo box can offer VOD, but only *locally*. What if the client wants a movie not pre-recorded in the TiVo box? Periodic Broadcast (PB) techniques are aimed for that service.

In the PB approach [1, 12, 22, 24, 28], a video is divided into several segments, each repeatedly broadcast on a separate communication channel from the server. A client receives a video by tuning to one or more channels at a time to download the data. The broadcast schedule at the server and playback synchronization protocol at the client ensure that the broadcast of the next video segment is available to the client before the playback of the current segment runs out. Many other PB techniques have been proposed, but the most popular ones are Staggered Broadcast [18], Skyscraper Broadcast [12], Pyramid Broadcast [28], Pagoda Broadcast [22], Harmonic Broadcast [16], Permutation-based Broadcast [1], and Fast Broadcast [17]. We review these techniques in this section. For other PB techniques, a comprehensive survey can be found in [13].

1.2.1 Staggered Broadcasting

Staggered Broadcast (SB) [18] is a simple periodic broadcast technique. It works as follows. Without loss of generality, we focus on a single video. This video is partitioned into K equally sized segments $\{s_1, s_2, ..., s_K\}$. The duration of each segment is $s = V/K$, where V is the duration of the entire video. Given the video consumption rate r (bps), we allocate a server bandwidth of $r \times K$ for the video[2]. This bandwidth is divided into K equal channels, each repeatedly broadcasting the video with a transmission rate equal to the consumption rate. The scheduling of these broadcasts is illustrated in Figure 1.1.

The playback procedure at a client follows the simple algorithm below:

Algorithm 1.2.1 (Staggered Broadcasting) *Client Playback*

1. *Tune in a random channel i. Suppose that channel i is currently broadcasting segment s_h*
   (a) *If $h = K - 1$, let $j = i$ and GOTO Step (2)*
   (b) *Else, compute*

$$j = \begin{cases} i + h - K, & \text{if } i + h - K > 0 \\ i + h, & \text{otherwise} \end{cases} \tag{1.1}$$

   - *Channel j must be currently broadcasting segment s_K and about to broadcast segment s_1.*

2. *Wait until channel j starts broadcasting segment s_1 and join this channel when that time comes*
3. *Play the video data received from this channel and quit when the video is finished playing*

A client joins only one broadcast channel at any time, thus the client bandwidth required is no more than the playback rate. In addition, the rendering at the client only consists of receiving a video packet, decoding, and displaying it. Therefore, the computational complexity required for playback is minimal.

A disadvantage of SB is a high service delay. If a client requests the video during the broadcast of segment s_1, this client has already missed the already-broadcast packets belonging to s_1 and must wait until the next broadcast of this segment. For instance, in Figure 1.1, if a client requests at time $t_0 + s + \delta$ ($0 < \delta < s$), it must wait until time $t_0 + 2s$ to start downloading segment s_1. Hence, the service delay is s - δ; in the worst case, it is s. Supposing K = 5 channels, which is likely to be the case with a IEEE 802.11g video server and MPEG-1, broadcasting a 60-minute video results in a worst-case delay of $s = V/K$ = 60/5 = 12 minutes.

[2]We assume that the server has enough bandwidth for this allocation

1.2.2 Harmonic Broadcasting

Similar to SB, Harmonic Broadcasting (HB) [16] divides each video into K equally sized segments $\{s_1, s_2, ..., s_K\}$. The server bandwidth for a video is also divided into K channels, however with different bandwidth allocation. Specifically, each segment s_i is broadcast repeatedly on channel i at rate r/i (bps), hence the name "Harmonic". As such, it takes longer to download a later segment than a previous segment.

At the client side, it has to tune in all K channels and receive the data for all the K segments. The client starts the playback as soon as it can download the first segment. Before that, all the received segments are stored in the client cache. The cache space required can be up to 40% of the video [16]. The good thing is that, because of the Harmonic rate allocation, it is guaranteed that by the time a segment is finished playing, the next segment is either in the cache or readily available from its broadcast channel. Therefore, the playback is smooth.

An advantage of SB is that it requires reasonable server bandwidth, which is equal to $\sum_{i=1}^{K} \frac{r}{i} = r \sum_{i=1}^{K} \frac{1}{i}$. This is much less than Kr – the server bandwidth required by SB. On the other hand, the client bandwidth requirement is very high, same as the server bandwidth. For example, the client would need 240 channel tuners if desiring a service delay of 30 seconds for a 2-hour video. This is practically not desirable. Also, the service delay of HB is as long as as that of SB and equal to V/K – the duration of a segment. Later techniques such as Cautious Harmonic Broadcasting and Quasi-Harmonic Broadcasting [21] attempt successfully to erase the long service delay of HB, but the caching and bandwidth cost at the client side remains substantial.

1.2.3 Fast Broadcasting

SB broadcast all segments of each video on every channel. HB broadcast each segment on a different channel. Unlike these two strategies, Fast Broadcasting (FB) [17] allows to broadcast more than one segment on each channel.

FB divides each video into n equally-sized segment. The server bandwidth for a video is also divided into K equal-rate channels as in SB. The broadcast schedule at the server is as follows:

- Channel 1 broadcasts segment s_1 repeatedly
- Channel 2 broadcasts 2 segments, s_2 and s_3, repeatedly one after another
- Channel 3 broadcasts 4 segments, s_4, s_5, s_6, s_7 repeatedly one after another
- Channel i ($i > 3$) broadcasts 2^{i-1} segments $s_{2^{i-1}}$, $s_{2^{i-1}+1}$, ..., s_{2^i-1} repeatedly one after another

The number of segments n therefore must be $n = 1 + 2 + 2^2 + ... + 2^{K-1} = 2^K - 1$.

For playback, the client listens to all the channels and stored the segment downloaded but not yet played into the client cache. Therefore, the client bandwidth required is as large as the server bandwidth (K channels) and the client cache is also significant. The service delay is the wait time for the first segment. In the worst case, this delay is $V/n = V/(2^K - 1)$. An advantage over SB and HB is that this delay decreases exponentially as K increases.

1.2.4 Pagoda Broadcasting

Similar to FB, Pagoda Broadcasting (PaB) [22] allows to broadcast more than one segment on each channel. PaB differs from FB in the selection of segments to broadcast on each channel.

PaB divides each video into n equally-sized segment. The server bandwidth for a video is also divided into K equal-rate channels. Each channel is logically divided into time-slots, each for a duration of a segment. The time-slots are indexed as $slot_0$, $slot_1$, $slot_2$, ...

At the server side, channel 1 broadcasts s_1 repeatedly $\{s_1, s_1, ...\}$. We consider channel i. There are two cases:

i is even: Suppose that s_z is the earliest segment not broadcast on channels 1, 2, ..., $i-1$. It will be broadcast in every slot $slot_{jz}$ ($j = 0, 1, ..$). All the other even-indexed slots will be equally allocated to segments s_{z+1}, s_{z+2}, ..., $s_{3z/2-1}$. All the other slots will be equally allocated to segments s_{2z}, s_{2z+2}, ..., s_{3z-1}.

i is odd: Supposing that s_z is the first segment broadcast on channel $i-1$, the earliest segment not yet broadcast is segment $s_{3z/2}$. $s_{3z/2}$ is broadcast on channel i in every slots $slot_{j(3z/2)}$ ($j = 0, 1, ..$). Every third slot is equally allocated for segments $s_{3z/2}$ to s_{2z-1}. The remaining slots are equally allocated to segments s_{3z} to s_{5z-1} in such a way that each pair of consecutive sets of $3z/2$ slots contains exactly one instance of each of these $2z$ segments.

Therefore, channel 2 will be broadcasting in the order s_2, s_4, s_2, s_5, s_2, s_4, s_2, s_5, ..., channel 3 will be broadcast in the order s_3, s_6, s_8, s_3, s_7, s_9, s_3, s_6, s_8, s_3, s_7, s_9, ...

Similar to HB and FB, PaB requires the client to tune in all channels to download data and store yet-to-played data into the client cache. The playback starts as soon as the client receives the first segment. [22] proved the following relationship between the number of segments n and the number of channel K:

$$n = \begin{cases} 4(5^{k-1}) - 1, & \text{if } K = 2k \\ 2(5^k) - 1 & \text{if } K = 2k+1 \end{cases} \tag{1.2}$$

The service delay is due to the wait time for the first segment on channel 1. Therefore, it decreases exponentially as K increases.

1.2.5 Pyramid Broadcasting

All the aforementioned broadcasting techniques partition each video into segment of equal size. On the contrary, Pyramid Broadcast (PyB) [28] partitions each video into K segments of increasing sizes: (hence, the name "pyramid")

$$s_i = \begin{cases} V(\alpha - 1)/(\alpha^K - 1), & \text{if } i = 1 \\ s_1 \times \alpha^{i-1}, & \text{otherwise} \end{cases} \tag{1.3}$$

Suppose that the server bandwidth allocated for each video is B, which is divided into K channels. Thus, the broadcast rate at the server is B/K on each channel. At this rate, channel 1 broadcasts repeatedly segment s_1, channel 2 broadcasts repeatedly

segment s_2, and so on. The client has two video loaders that can download data from two channels concurrently. The video playback is done as follows:

Algorithm 1.2.2 (Pyramid Broadcasting) *Client Playback:*

1. *Tune in channel 1 and start download the first segment s_1 at the first occurrence and play it concurrently*
2. *Set $i = 1$*
3. *While ($i \leq K$)*
   (a) *As soon as segment s_i is started to be played, also tune in channel $i + 1$ to download segment s_{i+1} at the earliest possible time and store it into a buffer*
   (b) *Once finishing playing segment s_i, switch to play segment s_2 from the buffer*
   (c) *Set $i := i + 1$*
4. *End While*

The α value is chosen in such a way to ensure that the playback duration of the current segment must be longer than the worst delay in downloading the next segment. This is equivalent to

$$\frac{s_i}{r} \geq \frac{s_{i+1}}{B/K} \Leftrightarrow \alpha \leq \frac{B}{rK} \tag{1.4}$$

Based on this inequality, [28] suggested two options for the α value

$$\alpha_1 = \frac{B}{r \times \lfloor \frac{B}{re} \rfloor} \text{ or } \alpha_2 = \frac{B}{r \times \lceil \frac{B}{re} \rceil} \tag{1.5}$$

where $e \approx 2.72$ is the Euler's constant. Therefore, α is close to this constant. The number of channels K is chosen to be $K_1 = \lfloor \frac{B}{re} \rfloor$ (case α_1) or $K_2 = \lceil \frac{B}{re} \rceil$ (case α_2).

The wait time before the video is started to be played equals the wait time until the first segment is first downloaded from channel 1. In the worst case, this delay is $\frac{VKr(\alpha-1)}{B(\alpha^K-1)}$. Assuming that $r = B/K$, this delay equals $V(\alpha - 1)/(\alpha^K - 1) = V/(1 + \alpha + \alpha^2 + ... + \alpha^{K-1}) \leq V/K$ because $\alpha \approx 2.72$. Therefore, PyB'delay is much better than both SB and HB's. In addition, the former is improved exponentially when K increases while the latter can only improve linearly.

On the other hand, PyB requires caching at the client side. The memory space needed for this caching is as large as $r(s_K + s_{K-1} - rKs_K/B)$. This is approximately the size of the second-last segment s_{K-1}. Since segment size increases exponentially, size s_{K-1} is significant. If α is kept around e, each client must have a disk space large enough to buffer more than 70% of the video file.

1.2.6 Permutation-based Broadcasting

Permutation-based Broadcasting (PbP) [1] is an extension to PyB in an attempt to reduce the client cache space requirement, however at a cost of more complexity. In PyB, each segment is broadcast sequentially on its corresponding channel. In PbP, each channel is further divided into P subchannels with $\frac{B}{KP}$ (bps) each. A replica of segment s_i is repeatedly broadcast on each of the P subchannels of channel i with a phase delay of s_i/P seconds. Since the subchannels are time-multiplxed on their parent channel, the client cache space required is less. The new requirement at each client is

$$\frac{rVK(\alpha^K - \alpha^{K-2})}{B(\alpha^K - 1)} \tag{1.6}$$

The service delay is simply the access time of the first segment; hence, $\frac{s_1}{P+\alpha}$. In PbP, we choose $K = \lfloor \frac{B}{Kr} \rfloor$, but its value must be in the range [2, 7]. We can choose the following values for α and P:

$$P = \lfloor \frac{B}{Kr} - 2 \rfloor \; and \; \alpha = \frac{B}{Kr} - P \tag{1.7}$$

Typically, PbP requires a client to cache about 50% of the video size, much less than that is required PyB. However, when the server bandwidth is more generous, the delay of PbP can only increase linearly as opposed to exponentially as in the case of Pyramid Broadcasting. [12] showed that PbP actually performs worse than PyB.

1.2.7 Skyscraper Broadcasting

Similar to PyB, Skyscraper Broadcasting (SkB) [12] also divides the server bandwidth for each video into K channels, each of rate B/K and broadcasting repeatedly a segment of the video. However, unlike PyB, SkB decomposes the video into K segments of the following sizes

$$s_i = \begin{cases} s_1, & i = 1 \\ 2s_1, & i = 2, 3 \\ 2s_{i-1} + 1, & i \; mod \; 4 = 0 \\ s_{i-1}, & i \; mod \; 4 = 1 \\ s_{i-1} + 2, & i \; mod \; 4 = 2 \\ s_{i-1}, & i \; mod \; 4 = 3 \end{cases} \tag{1.8}$$

That is, the sizes of the segments are

$$[1, 2, 2, 5, 5, 12, 12, 25, 25, 52, 52, ...] \times s_1$$

The technique is called "Skyscraper" because of this segment sizing[3]. SkB uses a control value W to restrict the segments from being too large. If a segment s_i becomes

[3] If we stack boxes of these sizes in the decreasing order from bottom up, the shape looks like a skyscraper. The shape for Pyramid Broadcasting looks like a pyramid.

larger than Ws_1, its size is rounded to Ws_1. This is because if we allow a segment to grow too large, a large caching space would be required for each client. The size of the first segment s_1 is chosen so that $\sum_{i=1}^{K} s_i = V$. Thus, we can control the size s_1 by changing W. A nice thing about SkB is the following relationship between the serviced delay and the control factor W:

$$delay = s_1 = \frac{V}{\sum_{i=1}^{K} \min(s_i, W)} \tag{1.9}$$

which can be used to determine W given the desired service delay.

The video segments are classified into groups, each containing consecutive segments of same size. For example, [s_1], [s_2, s_3], and [s_4, s_5] are different groups of size 1, 2, and 5, respectively. A group is called an "odd" (or "even") group if its segment size is odd (or even). The client uses two software modules: an even loader to download even-group segments and an odd loader to download odd-group segments. Each loader tunes into the appropriate channels to download its groups one at a time and in the order they appear in the video. The downloaded segments are forwarded into a buffer which can be played by the "video player" module at the playback rate.

The client bandwidth and cache requirement is as follows:

$$cache\ space = (W-1)s_1 B/K \tag{1.10}$$

$$bandwidth\ required = \begin{cases} 0, & W = 1\ or\ K = 1 \\ 2B/K, & W = 2\ or\ K = 2, 3 \\ 3B/K, & \text{otherwise} \end{cases} \tag{1.11}$$

[12] showed that SkB is able to retain the low latency of PyB while using significantly less buffer space (20%) than that required by PbP.

1.3 MobiVoD: A Mobile Wireless VOD Solution

Despite many periodic broadcast designs for the typical Internet-based networks, when applied to a wireless network, they may not be directly applicable. Since mobile wireless clients are usually of limited resource, some of these techniques (HB, FB, PaB) are not well suitable because they require significant client bandwidth and caching space. So is PbP because it incurs complex client playback. PyB is better in terms of client bandwidth but its client caching requirement remains very high. The two techniques potential for efficient deployment in a large-scale wireless environment are SB and SkB. We summarize the client resource requirement of these periodic broadcast techniques in Table 1.1.

None of these periodic broadcast techniques can provide true VOD because their service delay is non-zero. Between SkB and SB, the former provides better service delay. On the other hand, SkB is more complex and requires that the client be capable to download at a rate twice as large as the playback rate and have caching space enough for approximately 10% of the video length. For current wireless architectures, SB seems a better choice because of its simplicity. In this section, we discuss how SB

Table 1.1: Typical client requirement in Periodic Broadcasting solutions (r is consumption rate)

Solution	Caching space	Bandwidth
Staggered [18]	0% of video	$1 \times r$
Skyscraper [12]	10% of video	$2 \times r$
Pyramid [28]	75% of video	$\geq 4 \times r$
Permutation-based [1]	20% of video	$\geq 2 \times r$
Pagoda [22]	45% of video	$\geq 5 \times r$
Harmonic [16]	40% of video	$\geq 5 \times r$
Fast [17]	50% of video	$\geq 6 \times r$

can be adapted to work for a large-scale wireless networks. As wireless and memory technologies continue to advance, SkB could be the better choice in the future. The technique discussed here, called MobiVOD and proposed in [27], is also applicable to SkB (with a slight change).

The main problem with SB is its service delay. MobiVOD is an adaptation of SB, that erases this delay by leveraging video content sharing between the wireless clients.

1.3.1 Network Architecture

Illustrated in Figure 1.2, the system architecture for MobiVOD consists of three components: *video server*, *clients*, and *local forwarders*. The video server stores video files. Clients are the mobile users (devices or the people who use them), who subscribe for the VOD service provided by our system. Because the only way to communicate with the clients is via wireless transmissions, it is not possible for the video server to transmit a video to clients located in a too-wide geographic area. Therefore, we may deploy a scatter of local forwarders $\{LF_1, LF_2, .., LF_k\}$. A local forwarder LF_j is a stationary and dedicated computer and used to relay the service to LF_j's transmission coverage range. This area is called a local service area.

The local forwarders and clients are referred to as "nodes". Each of the nodes is equipped with a wireless network interface card or two, so that they can receive data from their local forwarder and, at the same time, form among themselves an ad hoc network. The rationale for this multi-connectivity is that MobiVOD allows clients to share and exchange video information with each other, directly without going through a broker node like the server or any local forwarder.

Every local forwarder receives the video packets from the server, either via a wired broadband connection or a wireless broadband connection like WiMAX. This local forwarder then broadcasts the packets to its local coverage, for example, using IEEE 802.11. If a client is within the service area of a local forwarder, the former can receive the video packets.

The topology for disseminating the video packets from the server to every local forwarder can be a star rooted at the server or any overlay topology connecting the server

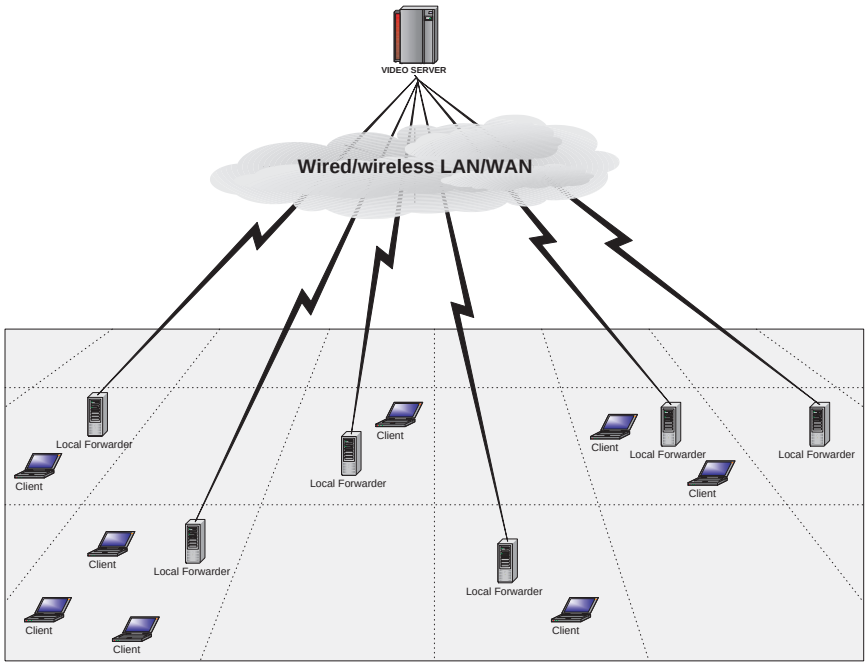

Figure 1.2: MobiVoD System Architecture: Server, Local Forwarders, and Clients

and all the local forwarders. Depending on what working environment the system is running in be the locations of the local forwarders determined. For instance, on a campus or at an airport terminal, the local forwarders should be geographically uniformly distributed. In a big building with closed-door halls, however, we should install a local forwarder in each hall.

1.3.2 Client Playback

Each local forwarder (or the server in the case without local forwarders) uses Staggered Broadcasting to broadcast each video. As described in Algorithm 1.2.1, the client tunes in the channel that is to broadcast the first segment the soonest and starts the playback as soon as the first segment arrives. Therefore, the service delay can be as large as the duration of the first segment.

The idea of MobiVOD is that when a new client starts the video request and the first segment is not yet available on any broadcast channel, the new client can get this segment instantly from a nearby client who has a cache of it. Thus, we need to determine who should cache or who should not. Obviously, if a cache is very far from the new client, it is helpless because there is no efficient way for the client to download the segment in a multi-hop manner. Thus, a key design component of MobiVOD is client caching.

A MobiVOD client has the following two buffers:

- Reusable Buffer: This buffer is used to cache the first segment of the video. Therefore, the size of this buffer is that of the first segment. A client needs a reusable buffer if it is selected to cache the first segment.
- Prefetched Buffer: This buffer is used for video playback. The size of this buffer is that of the already-broadcast portion that the client misses. A client needs this buffer if it opts to make use of the first segment cached at a nearby client's reusable buffer.

Let us consider a new arriving client X who detects that it already misses the current broadcast of the first segment. For example, in Figure 1.1, the new client request the video at some time between t and $t + s$; thus, the next broadcast of the first segment is at time $t + s$. Instead of waiting for the next broadcast of the first segment, this client looks for an existing client Y in its transmission range, who has a cache of the first segment in the reusable buffer. If such Y exists, X can download and play the missing portion from Y, and, at the same time, store the packets broadcast from X's local forwarder into the prefetched buffer. Once X finishes playing the missing portion, it switches to play the data in the prefetched buffer. In this case, though X misses the current broadcast of the first segment, X still manages to watch the entire video immediately.

There are three caching strategies:

- ALL-CACHE: Every client caches the first video segment
- RANDOM-CACHE: A client randomly decides whether it should cache the first video segment
- DOMINATING-SET-CACHE: Only the clients, that belong to a dominating set of the clients, cache the first segment

ALL-CACHE

ALL-CACHE requires every existing client to cache the first segment. Since the clients are bandwidth limited, we should not let an existing client forward the cached data to more than one other client at the same time. Therefore, in choosing Y, we skip clients that have been forwarding its cache to other clients.

RANDOM-CACHE

Clearly, by caching the first segment at every existing client, we significantly increase the chance subsequent clients can join the service instantly. The tradeoff of this strategy, however, is the amount of storage space needed for caching. Alternatively, we can use selective caching, in which only a number of selected clients need to cache. The advantage of selective caching is the saving on caching space.

The simplest selective-caching algorithm is RANDOM-CACHE, in which a client decides to cache the first segment based on a probability. When a new client starts a video request, if it finds an existing client in its neighborhood who has the cache, the playback is the same as that in the case of ALL-CACHE. If no cache exists in the

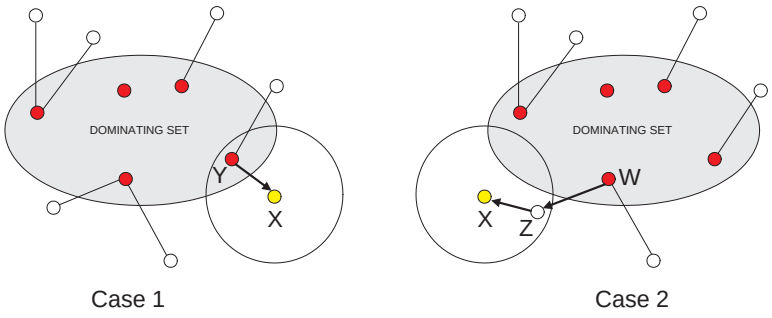

Figure 1.3: If a new client can reach an existing client, the former always finds a cache within two hops

neighborhood, the new client has to wait for the first segment to arrive from the broadcast channels as usual. Obviously, the average service delay of RANDOM-CACHE is longer than ALL-CACHE.

DOMINATING-SET-CACHE

Is there any way that a new client can always find a cache of the first segment in its neighborhood? The answer is yes. The key idea of DOMINATING-SET-CACHE (DSC) is to maintain a dominating set of all the clients. A dominating set of the set of clients is the set of clients $\{Y_1, Y_2, ..., Y_k\}$ such that for any other client X there exists Y_i which is a neighbor of X. We denote this dominating set by $DSet$. DSC requires that all clients in $DSet$ cache the first segment.

The use of a dominating set of mobile hosts was proposed in many MANET works, mostly in wireless broadcasting/routing protocols [2, 29, 30]. The playback of a new client X's arrival as follows (illustrated in Figure 1.3):

- Case 1: X is in the transmission range of a client $Y \in DSet$ and Y is not currently forwarding cache to any other client
  1. X makes use of the cache at Y and immediately plays the video as explained earlier.
- Case 2: X is in the transmission range of a client Y outside $DSet$ and not in the transmission range of any client in $DSet$
  1. Y finds a neighbor $Z \in DSet$ such that Z currently is not forwarding cache to any other client.
  2. Y downloads the broadcast portion that X misses from Z and forwards it to X. X can play the video immediately as explained earlier.
- Case 3: Neither case 1 nor case 2 holds. In this case, X waits until the next broadcast of segment 1. As soon as the first segment arrives, X starts playing it and remains on the same channel to play the rest of the video.

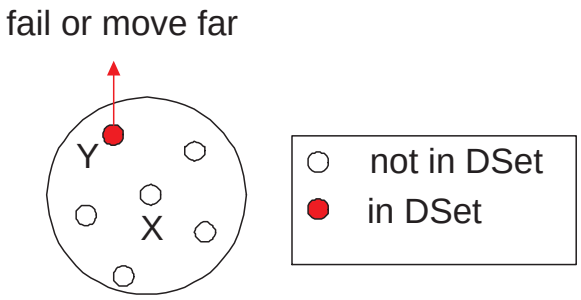

Figure 1.4: X is currently not in $DSet$. After Y fails or moves far away, X cannot be reached by any caching node. If a new client arrives close to X, the new client cannot make use of any cache and therefore has to wait until the next broadcast of the first segment from its local forwarder.

DSC guarantees that if a new client can reach an existing client, there is always a cache of the missing portion within two hops. Although streaming video is a challenging problem in multihop wireless networks, DSC should work well because two hops is short and the size of the missing portion is small, thus the cache downloading is quick.

A failure may occur while a new client is downloading its missing portion from an existing node; for example, when the existing node moves far away or quits the system. The new client detects this failure by observing that it has been waiting for the next packet for some period long enough. In this case, the new client can repeat the cache search above. However, if a new cache is found, the new client just needs to download part of the missing portion, which has not been downloaded from the previous cache. Again, since the cache is less than two hops away and the missing portion is short, we expect a small probability that a client needs to switch to a new cache.

Many distributed algorithms were proposed to solve the dominating set problems in wireless ad hoc networks (e.g., [2, 6, 30]). Since mobile hosts may move or fail, these algorithms allow a mobile host to change status from "not in dominating set" to "belong to dominating set". Our situation is different. We need to decide if a client belongs to $DSet$ as soon as it joins the system. If it is in $DSet$, it will cache the first segment. If DSC decides a client is not in $DSet$, this client will not hold any cache and therefore will never belong to this set in the future. Therefore, we just use the following policy to decide whether a new client is going to cache: *Initially, there is no client and $DSet$ is empty. A new client belongs to $DSet$ if and only if no client in $DSet$ is within the transmission range of the new client.* To implement this policy, the new client X broadcasts a request and any client Y who intercepts the request will send a reply back to X if Y holds a cache. If X receives at least a reply, the new client decides that it will not cache (which also means X is not in $DSet$). If X does not receive any reply, X decides that it will cache (which also means X belongs to $DSet$).

We may find a case where $DSet$ becomes not a dominating set of clients, illustrated in Figure 1.4. The simulation results reported in [27] showed that the effectiveness of

MobiVoD remained even in such a case.

Comparison

Simulation results are provided in [27] for comparison of ALL-CACHE, RANDOM-CACHE, and DSC. In its study, ALL-CACHE provides almost true VOD services, however the storage is required for caching 20% of the video. This is a drawback that makes ALL-CACHE least desirable by current mobile clients. DSC and RANDOM-CACHE, with much less caching space occupancies, perform similarly to ALL-CACHE in terms of client bandwidth requirement, cache distance, and startup overhead. In addition, DSC and RANDOM-CACHE offer service delays much better than without caching. In deed, in most scenarios, they are more than nine times better than without caching. Between DSC and RANDOM-CACHE, DSC is more preferable in terms of service delay. On the other hand, the advantage of RANDOM-CACHE is its flexibility in choosing the number of clients who will cache.

1.4 Summary and Future Work

This chapter explored periodic broadcast techniques for video-on-demand (VOD) services in a wireless networking environment. While broadcasting is the nature of wireless communications, VOD broadcasting is not trivial to be implemented in a wireless network. This is because, unlike the Internet where one-to-one communication does not affect nodes that do not involve in the communication, transmission in a wireless environment between two nodes may interfere with transmissions by other nodes. Therefore, video broadcasts if not scheduled properly may result in significant signal loss and expensive resource consumption.

We reviewed VOD periodic broadcast techniques already designed for the Internet. We also discussed their pros and cons if adapted to work for wireless networks. Most of these techniques require significant client bandwidth and caching space, which is not suitable for mobile wireless clients. On the other hand, none of them can offer true VOD. We presented MobiVoD – a novel technique aimed to be simple so that it is feasible for wireless environment, yet providing much shorter service delay than the traditional periodic broadcasting techniques. Using today's technologies, clients can communicate wirelessly through access points, base stations, or one-to-one in an ad hoc manner with each other. The key idea of MobiVoD is to utilize the collaboration among the clients.

We implicitly assumed in this chapter that clients are homogeneous, meaning they have same capabilities (ratio transmission radius, wireless bandwidth etc.). It is better to relax this assumption so the system is more accessible to different types of clients, especially those having bandwidth less than the video consumption rate. For this purpose, MobiVoD can be extended by employing a multi-resolution or layered video coding approach [4, 20]. In a heterogeneous system, two clients are considered neighbors if and only if they are in the transmission range of each other. Each video is encoded into several "layers", including a base layer and one or more enhancement layers. The base layer provides the version of least quality, while its combination enhancement lay-

ers provide better quality. Instead of broadcasting the entire video as in pure Periodic Broadcasting, the server, and thus every local forwarder, broadcasts all layers on separate channels. A new client selects a combination of layers that best match its resource constraints and only tunes in the corresponding channels to download such layers. As for the initial portion that the client misses from the current broadcasts, it searches for a nearby client who caches a "version" of the first segment (a version is a combination of the base layer and one or more enhancement layers). If more than one such client are found, the client with the highest-quality version is selected.

Another extension of MobiVoD is to allow a client to download a cache more than two hops away. This enhancement would increase the chance of providing true on-demand services to clients, which is however suggested only when wireless bandwidth is more advanced than the current.

Bibliography

[1] C. C. Aggarwal, J. L. Wolf, and P. S. Yu. A permutation-based pyramid broadcasting scheme for video-on-demand systems. In *Proc. of the IEEE Int'l Conf. on Multimedia Systems'96*, pages 118–126, Hiroshima, Japan, June 1996.

[2] K. M. Alzoubi, P.-J. Wan, and O. Frieder. Message-optimal connected-dominating-set construction for routing in mobile ad hoc networks. In *ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, Lausanne, Switzerland, June 2002.

[3] J. G. Andrews, A. Ghosh, and R. Muhamed. *Fundamentals of WiMAX: Understanding Broadband Wireless Networking – The Definitive Guide to WiMAX Technology*. Prentice Hall Professional, 2007.

[4] S. Bajaj, L. Breslau, and S. Shenker. Uniform versus priority dropping for layered video. In *ACM SIGCOMM*, pages 131–143, 1998.

[5] Bluetooth. http://www.bluetooth.com/.

[6] Y. P. Chen and A. L. Liestman. Approximating minimum size weakly-connected dominating sets for clustering mobile ad hoc networks. In *ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, Lausanne, Switzerland, June 2002.

[7] Comcast. Comcast spotlight: Video advertising on demand. http://www.comcastspotlight.com.

[8] Concurrent. On-demand and real-time linux solutions. http://www.ccur.com/.

[9] A. Dan, D. Sitaram, and P. Shahabuddin. Scheduling policies for an on-demand video server with batching. In *Proc. of ACM MULTIMEDIA*, pages 15–23, San Francisco, California, October 1994.

[10] S. Gruber, J. Rexford, and A. Basso. Protocol considerations for a prefix-caching proxy for multimedia streams. In *Proc. of the 9th International WWW Conference*, pages 657–668, 2000.

[11] K. A. Hua, Y. Cai, and S. Sheu. Patching: A multicast technique for true video-on-demand services. In *Proc. of ACM MULTIMEDIA*, pages 191–200, Bristol, U.K., September 1998.

[12] K. A. Hua and S. Sheu. Skyscraper broadcasting: A new broadcasting scheme for metropolitan video-on-demand systems. In *Proc. of the ACM SIGCOMM'97*, pages 89–100, Cannes, France, Sepetember 1997.

[13] K. A. Hua, M. A. Tantaoui, and W. Tavanapong. Video delivery technologies for large-scale deployment of multimedia applications. *Proceedings of the IEEE*, 92(9):1439– 1451, September 2004.

[14] K. A. Hua, D. A. Tran, and R. Villafane. Caching multicast protocol for on-demand video delivery. In *Proc. of the ACM/SPIE Conference on Multimedia Computing and Networking*, pages 2–13, San Jose, USA, January 2000.

[15] IEEE-Std-802.11. Wireless lan medium access control (mac) and physical layer (phy) specification. *1999 Edition*, 1999.

[16] L. Juhn and L. Tseng. Harmonic broadcasting for video-on-demand service. *IEEE Transactions on Broadcasting*, 43(3):268–271, 1997.

[17] L. Juhn and L. Tseng. Fast data broadcasting and receiving scheme for popular video service. *IEEE Transactions on Broadcasting*, 44(1):100–105, 1998.

[18] J. B. Kwon and H. Y. Heom. Providing vcr functionality in staggered video broadcasting. *IEEE Transactions on Consumer Electronics*, 48(1):41–48, 2002.

[19] Motorola. Motorola on-demand solutions. http://broadband.motorola.com/business/ondemand/VideoOnDemand.html.

[20] V. N. Padmanabhan, H. J. Wang, P. A. Chou, and K. Sripanidkulchai. Distributing streaming media content using cooperative networking. In *ACM/IEEE NOSSDAV*, pages 177–186, Miami, FL, USA, May 12-14 2002.

[21] J. F. Paris, S. W. Carter, and D. D. E. Long. Efficient broadcasting protocols for video on demand. In *Proceedings of the 6th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS '98)*, page 127132, Montreal, Canada, July 1998.

[22] J. F. Paris, S. W. Carter, and D. D. E. Long. A hybrid broadcasting protocol for video on demand. In *ACM/SPIE Conference on Multimedia Computing and Networking*, 1999.

[23] SeaChange. Seachange international: The global leader in video on demand. http://www.schange.com/.

[24] S. Sen, L. Gao, and D. Towsley. Frame-based periodic broadcast and fundamental resource tradeoffs. In *IEEE Performance, Computing and Communications Conference*, pages 77–83, April 2001.

[25] D. A. Tran, K. Hua, and T. Do. A peer-to-peer architecture for media streaming. *IEEE JSAC, Special Issue on Advances in Service Overlay Networks*, 22(1), January 2004.

[26] D. A. Tran, K. A. Hua, and T. T. Do. Zigzag: An efficient peer-to-peer scheme for media streaming. In *IEEE INFOCOM*, San Francisco, CA, March-April 2003.

[27] D. A. Tran, K. A. Hua, and M. Le. MobiVOD: A video-on-demand system design for mobile ad hoc networks. In *In Proceedings of IEEE International Conference on Mobile Data Management (MDM 2004)*, Berkeley, CA, USA, January 19-22 2004.

[28] S. Viswanathan and T. Imielinski. Metropolitan area video-on-demand service using pyramid broadcasting. *ACM Multimedia systems Journal*, 4(4):179–208, August 1996.

[29] B. Williams and T. Camp. Comparison of Broadcasting Techniques for Mobile Ad Hoc Networks. In *ACM Symposium on Mobile Adhoc Networking and Computing (MOBIHOC 2002)*, June 2002.

[30] J. Wu. Extended dominating-set-based routing in ad hoc wireless networks with unidirectional links. *IEEE Transactions on Parallel and Distributed Systems*, 13(9):866–881, 2002.

[31] K.-L. Wu, P. S. Yu, and J. L. Wolf. Segment-based proxy caching of multimedia streams. In *Proc. of the 10th International WWW Conference*, pages 36–44, Hong Kong, 2001.

[32] D. Xu, M. Hefeeda, S. Hambrusch, and B. Bhargava. On peer-to-peer media streaming. In *IEEE Conference on Distributed Computing and Systems*, pages 363–371, July 2002.