DESIGN AND IMPLEMENTATION OF A CONSONANT BROADCASTING ARCHITECTURE FOR LARGE-SCALE VIDEO STREAMING

LIU WING CHUN

A Thesis Submitted in Partial Fulfillment
of the Requirements for the Degree of
Master of Philosophy
In
Information Engineering

July 2003

ACKNOWLEDGEMENT

First and foremost, I would like to express my gratitude to my supervisor Prof. Jack Y. B. Lee. His passion in research and continuous support has guided me on the completion of this thesis and has inspired me on developing research skills.

I also treasure helps and funs from my friends, including Raymond, Sai, Yamaha, Wai, and Rudolf over these two years. And I would like to thank the dedicated love from my family and Belinda.

ABSTRACT

Video streaming systems that can serve hundreds to thousands of concurrent users are already widely available. However, to deploy metropolitan-scale video streaming services for potentially millions of users, current video streaming systems are still limited in capacity, and expensive in cost. To tackle this challenge, researchers have proposed novel broadcasting schemes to deliver video streaming services to virtually unlimited number of users using network multicast and client-side caching. This study presents a novel Consonant Broadcasting (CB) scheme utilizing network multicast that outperforms all known periodic broadcasting schemes. For example, with a client access bandwidth constraint of twice the video bit-rate and a total system bandwidth equal to ten times the video bit-rate, the proposed CB scheme can reduce the maximum startup latency experienced by users by 96%, 95%, and 72% compared to Skyscraper Broadcasting, Greedy Disk-Conserving Broadcasting, and Staircase Data Broadcasting respectively, which are among the current state-of-the-art periodic broadcasting schemes. Moreover, we devise a dynamic version of CB that performs well at both heavy and light loads, and can support the streaming of both constant bit-rate (CBR) and variable bit-rate (VBR) videos using comparable resources respectively. This thesis presents the principles of Consonant Broadcasting, analyzes its performance, addresses practical implementation issues, and reports experimental results to verify CB's feasibility, practicality, and performance.

摘要

今時今日，能支援數以千計用戶的視頻點播 (VoD) 系統已經十分普及。但當面對系統的規模不斷擴大時，伺服器便需要升級來增加容量，成本十分昂貴。爲了克服這個難題，研究人員利用互聯網組播 (multicast) 及用戶端快取 (caching) 技術，建議出不同的週期性廣播技術 (periodic broadcasting)。在本論文中，我們研究一種突破性的週期性廣播技術，名爲和協式廣播技術 (Consonant Broadcasting)，比起現時的週期性廣播技術更有效利用網絡資源。例如，在用戶端只有兩倍視頻 (video bit-rate) 及系統只有十倍視頻的環境下，比起現時的 Skyscraper Broadcasting、Greedy Disk-Conserving Broadcasting 及 Staircase Data Broadcasting，和協式廣播技術 (Consonant Broadcasting) 能分別減低用戶端達百分之九十六、百分之九十五及百分之七十二的等候時間。我們並由和協式廣播技術衍生出新的動態廣播技術以適應不同的系統負荷及變數視頻 (VBR) 的視頻廣播。本論文中，我們展述和協式廣播技術之系統架構，並分析其效能。與此同時，我們解決其實際運作時所遇到的問題及詳錄實驗結果，以驗證此技術的可行性和效能。

Contents

Chapter 1

INTRODUCTION

Among the various applications in multimedia systems, video streaming poses one of the greatest challenges. Extensive researches by many prominent researchers in the past decade have since significantly advanced video streaming technologies to the point where systems capable of streaming thousands or even tens of thousands of concurrent video streams are now commercially available. Nevertheless, the capacity of current video streaming systems still falls short of that needed in bringing video streaming services to the mass population, with potentially tens of millions of users.

A fundamental limitation in most of the current video systems is the network transmission model employed. Specifically, many existing video systems stream video data over data networks using unicast, i.e., one-to-one, network transmission. Consequently, the capacity and bandwidth requirements of the video servers as well as the network will increase proportionally with the user population, rendering it impossible to achieve economy-of-scale at the system level.

This observation has motivated researchers to turn the focus from unicast-based architectures to multicast-based architectures for building large-scale video streaming systems. Unlike unicast, a multicast video stream can be received by an arbitrary number of connected users without incurring additional resource

requirements at the upstream backbone network and the video servers. As a result, the costs of the video servers and the network backbone can be effectively amortized over all the users, and thus enabling one to achieve the critical economy-of-scale to make video streaming technically and financially feasible for the metropolitan.

In recent years, a number of pioneering researchers have proposed new video streaming architectures based on network multicast. We can classify them into closed-loop and open-loop architectures. For closed-loop architectures, the multicast video streams are dynamically scheduled according to the user arrival pattern. The general principle is to merge users arrived at different time instants receiving separate video streams to share just a few video streams. Notable examples include the Streaming Tapping scheme proposed by Carter and Long [1], the Patching scheme proposed by Hua et al. [2], the Controlled Multicast scheme proposed by Gao and Towsley [3], and the Dyadic Stream Merging scheme proposed by Coffman et al. [4]. There are also other schemes [5-9] and all these video streaming architectures can significantly reduce the system resources required compared to unicast-based systems.

By contrast, open-loop architectures, also known as periodic broadcasting, have fixed schedules for all video streaming channels irrespective of the user arrival pattern. A new user will receive video data from one or more of the pre-scheduled video channels to sustain continuous playback. Notable examples include the Pyramid Broadcasting scheme proposed by Viswanathan and Imielinski [10], the Skyscraper Broadcasting scheme proposed by Hua and Sheu [11], the Greedy Disk-Conserving Broadcasting scheme proposed by Gao et al. [12], the Staircase Data Broadcasting scheme proposed by Juhn and Tseng [13], the Harmonic Broadcasting scheme proposed by Juhn and Tseng [14], the Poly-harmonic Broadcasting scheme

proposed by Paris et al. [15] and the Pagoda Broadcasting scheme proposed by Paris [16] et al. There are also other schemes [17-20] as well and the interested readers are referred to the study by Hu [21] for a comprehensive study of the existing periodic broadcasting schemes.

Comparing closed-loop and open-loop architectures, the performance (e.g. startup latency) of closed-loop architectures depends on the system load (i.e. user arrival rate), and generally the performance deteriorates with higher system load. By contrast, open-loop architectures have invariant performance irrespective of the system load. Consequently, at light system load closed-loop architectures can achieve better performance while open-loop architectures perform better at high system load.

In this study, we present a novel scheme that addresses six fundamental challenges in current closed-loop and open-loop architectures. First, we propose a new open-loop architecture called Consonant Broadcasting (CB) that outperforms all existing open-loop architectures. For example, with a client access bandwidth constraint of twice the video bit-rate and a total system bandwidth equal to ten times the video bit-rate, the proposed Consonant Broadcasting can reduce the maximum startup latency by 96%, 95%, and 72% compared to Skyscraper Broadcasting, Greedy Disk-Conserving Broadcasting, and Staircase Data Broadcasting, which are among the current state-of-the-art open-loop architectures.

Second, we show that CB can be applied to systems with any client access bandwidth constraint. Third, we address the practical issue of limited availability of multicast channels (e.g. multicast IP addresses) by extending CB to a Grouped Consonant Broadcasting (GCB) architecture. Fourth, we have successfully implemented a system prototype using off-the-shelve computing and network hardware, thereby proving CB's feasibility, practicality, and performance.

Fifth, we extend CB to a novel Dynamic Consonant Broadcasting (DCB) that combines the virtues of both closed-loop and open-loop architectures such that its efficiency is not limited to a specific range of loads. Instead, DCB outperforms the current state-of-the-art closed-loop and open-loop architectures for a very wide range of system load (e.g. from 0.001 to 10 customers/sec).

Finally, while most open-loop/closed-loop architectures are designed for streaming constant-bit-rate (CBR) videos, we further extend CB to support the streaming of both CBR as well as variable-bit-rate (VBR) videos. More surprisingly, streaming VBR videos using DCB does *not* incur substantially more bandwidth than streaming CBR video of the same average bit-rate. Given that VBR encoded videos can achieve visual quality similar to CBR encoded versions using as little as half the average bit-rate [23], we can potentially *reduce* the system resources required by streaming VBR videos instead of CBR videos.

The rest of the thesis is organized as follows: Chapter 2 reviews some previous works on periodic broadcasting. Chapter 3 presents and analyzes the Consonant Broadcasting scheme. Chapter 4 evaluates the performance of the Consonant Broadcasting scheme and compares it to the current state-of-the-art periodic broadcasting schemes. Chapter 5 presents a Grouped Consonant Broadcasting scheme to tackle the problem of limited multicast channels. Chapter 6 addresses two practical implementation issues and present experiment results obtained from a system prototype. Chapter 7 presents the principles and evaluates the performance of Dynamic Consonant Broadcasting scheme. Chapter 8 extends the Consonant Broadcasting scheme for VBR videos streaming. Finally, Chapter 9 concludes the thesis.

Chapter 2

RELATED WORKS

Fundamental to all periodic broadcasting schemes are four design dimensions. First, a video title is divided into a number of smaller segments according to a *data partition scheme*. Second, the system (i.e. server and network) bandwidth is divided into a number of logical channels according to a *bandwidth partition scheme*. Third, a predetermined and fixed *broadcasting schedule* defines when the server should broadcast (or multicast, we will use these two terms interchangeably in the rest of the thesis) a video segment over which logical channels. Fourth, a client *reception schedule* defines when a client should receive video data from which logical channels.

Different designs of the four design dimensions result in different tradeoffs between the three system resources, namely system bandwidth, client access bandwidth, and client buffer requirement. Clever designs of the four design dimensions can result in significant resource savings compared to current unicast-based video streaming systems. More importantly, the resource requirements and performances of these periodic broadcasting systems are independent of the system scale. In other words, the same system can potentially serve an unlimited number of concurrent users, as long as the network infrastructure can accommodate them. This property is instrumental to deploying metropolitan-scale video streaming

Table 1. Summary of notations.

Symbol	Definition
L	The length of the video (sec)
b	The playback rate of the video (Mbps)
K	The total number of logical channels
L_i	The size of the i^{th} video segment (sec)
N	The total number of video segments
B	The total network bandwidth (Mbps)
B_i	The network bandwidth for the video segment L_i (Mbps)
C	The client access bandwidth constraint (Mbps)
T	The maximum startup latency (sec)
H	The maximum client buffer requirement (Mb)

services as it reduces the per-user system cost when more users are added, thereby allowing the service provider to achieve the crucial economy of scale.

In this chapter, we review some of the existing periodic broadcasting schemes [11-13,15-16], and present some known performance bounds. Due to space limitation, interested readers are referred to the study by Hu [21] for a more comprehensive study and comparison of the existing periodic broadcasting schemes. We summarize in Table 1 the notations used throughout this chapter.

2.1 Fixed-Segment Fixed-Bandwidth Schemes

In fixed-segment fixed-bandwidth schemes, a video title is divided into fixed-size video segments. These segments are then broadcast over a group of fixed-bandwidth channels according to its broadcasting schedule. A notable example is the Pagoda Broadcasting scheme [16] proposed by Paris et al. in 1999. A video is divided into N fixed-sized video segments, based on the number of channels K, obtained from

solving the equation $N = 4 \cdot (5^{\frac{K}{2}-1}) - 1$ if K is even or $N = 2 \cdot (5^{\frac{K-1}{2}}) - 1$ if K is odd. Each video segment is then broadcast over a fixed-bandwidth channel according to its broadcasting schedule at a defined broadcasting frequency.

The client receives from the beginning of a video segment as soon as it encounters it in any of the broadcasting channels. In the worst case, the client has to receive data from all channels simultaneously. The maximum startup latency T is equal to the broadcast duration of the first video segment L_0.

2.2 Variable-Segment Fixed-Bandwidth Schemes

Variable-segment fixed-bandwidth schemes (e.g. [10-12]) divide a video title into variable-size video segments for broadcast over fixed-bandwidth network channels (e.g. b Mbit/sec). A notable example is the Skyscraper Broadcasting scheme [11] proposed by Hua et al. in 1997 as an improvement to the Pyramid Broadcasting scheme proposed by Viswanathan and Imielinski [10]. Unlike the Pyramid Broadcasting scheme, where the video segment sizes increase according to a geometric series, the Skyscraper Broadcasting scheme divides a video title into N video segments according to a predefined data partition function. They also limited the maximum video segment size to a given length W to reduce the client buffer requirement. The network bandwidth B is then divided equally into N channels (i.e. same as the number of video segments), each with a bandwidth equal to the video bit-rate b. Video segment L_i (i=0,1,…,N−1) is then repeatedly broadcast over channel i.

The client always caches video data from the beginning of a video segment (instead of from anywhere in between). The client begins by caching data from the next broadcast of the video segment L_0. Then it caches the subsequent video segments L_i in the order of i=1,2,...,N−1 at the earliest time after it started playing back the video segment L_i. The client receives data from up to two channels simultaneously and the maximum startup latency T is equal to the broadcast duration of the first video segment L_0. The client buffer requirement is equal to $L_0 b(W-1)$ [11].

Another notable example is the Greedy Disk-Conserving Broadcasting scheme [12] proposed by Gao et al. in 1998. It is a greedy algorithm that minimizes the number of server channels needed to guarantee a given maximum startup latency T and client I/O bandwidth requirement. Unlike the Skyscraper Broadcasting scheme, GDB allows the client to receive video segments from n−1 channels simultaneously, where n is defined as the order of this scheme (denoted as GDBn). Again the maximum startup latency T is equal to the broadcast duration of the first video segment L_0 and the client buffer requirement is equal to $L_0 \cdot b \cdot (f^C_{GDB(n)}(N)-1)$ [12], where $f^C_{GDB(n)}(N)$ is the largest video segment size.

2.3 Fixed-Segment Variable-Bandwidth Schemes

Alternatively, we can broadcast fixed-size video segments over variable-bandwidth channels. Notable examples include the Harmonic Broadcasting scheme proposed by Juhn and Tseng [14] in 1997 and the Poly-harmonic Broadcasting scheme [15] proposed by Paris et al. in 1998.

In the Poly-harmonic Broadcasting scheme, the video title is partitioned into N equal-size video segments. Given the desired startup latency T and a control parameter m, one can choose N by solving the equation $T = (m \cdot L)/N$. The network bandwidth B is then divided into N channels (i.e. same as the number of video segments), with the bandwidth for channel i equal to $B_i = \frac{b}{m+i}$, $i = 0,1,...,N-1$. Video segment L_i is then repeatedly broadcast over channel i. The client on the other hand, is required to cache video segments from *all* channels simultaneously once it enters the system.

The Poly-harmonic Broadcasting scheme can achieve near-optimal performance when m is large. However, it suffers from two limitations. First, as the client must receive all channels simultaneously, the client's access network bandwidth requirement is very large (same as the server bandwidth requirement). Clearly this may not be practical in most wired systems where the access bandwidth is substantially more limited than server bandwidth (e.g. ADSL, cable modem). Second, using a large value of m, while improves performance, will generate a huge number of video segments, each requiring its own network channel for transmission. For some types of network (e.g. IP multicast) this may become a bottleneck as the number of network channels is limited (e.g. IP multicast addresses). Both of these limitations are addressed by the Consonant Broadcasting scheme investigated in this study.

2.4 Variable-Segment Variable-Bandwidth Schemes

The final type of broadcasting scheme is to have both variable segment size and variable channel bandwidth. Juhn and Tseng proposed the first variable-segment

variable-bandwidth scheme called Staircase Data Broadcasting [13] scheme in 1997. In Staircase Data Broadcasting, a video is first partitioned into N equal-size video segments, based on the number of channels K, derived from the equation $N = \sum_{j=0}^{K-1} 2^j = 2^K - 1$. The network bandwidth B is then divided equally into K channels, with the same bandwidth b for the i^{th} logical channel. For each video segment L_i, it is further divided into 2^i continuous video sub-segments for i=0,1,…,K−1. Similarly, each logical channel i is further sub-divided into 2^i sub-channels, each with a bandwidth of $b/2^i$. Finally, each sub-segment is then broadcast repeatedly over a separate sub-channel.

The client begins by receiving data from the first occurrence of the beginning of video segment L_0 at time t_0. The 2^i continuous video sub-segments $L_{i,j}$, j=0,1,…,2^i−1, within channel i (i=0,1,…,N−1) are then cached at time $t_0 + (L \cdot j) / N$. The client access bandwidth requirement is equal to $2b$, the maximum startup latency T is equal to the broadcast duration of the first video segment, and the client buffer requirement is bounded by 25% of the size of the video.

2.5 Performance Bounds of Periodic Broadcastings

Common to all periodic broadcasting schemes, the key system parameters are startup latency, network bandwidth, client access bandwidth, and client buffer requirement. Different schemes can be considered as achieving different tradeoffs among these four parameters, and thus the natural question is whether bounds on the system's performance exist.

This question has been investigated independently by Hu [21], Birk and Mondri [22], and the authors. Although the approaches and the derivations are different, we all arrive at the same results. Specifically, given a startup latency of T, it can be shown that the minimum network bandwidth needed for any periodic broadcasting scheme, is given by

$$B = b \cdot \ln(\frac{L+T}{T}) \tag{1}$$

assuming there is no constraint on the client access bandwidth.

Additionally, for any optimal periodic broadcasting scheme achieving the performance bound in (1), it can be shown that the client buffer requirement is equal to

$$H(t') = \begin{cases} t' \cdot b \cdot \int_0^L \frac{dt}{t+T} = t' \cdot b \cdot \ln(\frac{L+T}{T}), & 0 \le t' < T \\ t' \cdot b \cdot \int_{t'-T}^L \frac{dt}{t+T} = t' \cdot b \cdot \ln(\frac{L+T}{t'}), & T \le t' \le T+L \end{cases} \tag{2}$$

where t' is the elapsed time after the client entered the video streaming system and t is the time relative to the start of video. The upper bound of this client buffer requirement is 37% of the video size. Note that this is only a sufficient condition so it is still possible for a periodic broadcasting scheme to achieve lower client buffer requirement at the expense of increased latency or bandwidth.

Chapter 3

CONSONANT BROADCASTING

In this chapter, we present a new open-loop periodic broadcasting scheme - Consonant Broadcasting (CB), which outperforms all known open-loop schemes. More significantly, CB can be used in networks with limited client access bandwidth, which is the norm in typical metropolitan broadband networks. Fig. 1 shows CB's broadcasting schedule and reception schedule. We divide a video title into N equal-size segments and repeatedly broadcast them in separate variable-bandwidth multicast channels, i.e., video segment L_i is multicast in the i^{th} logical channel, for i=0,1,…,N–1. Thus CB belongs to the category of fixed-segment variable-bandwidth schemes. We assume the video is constant-bit-rate encoded and thus the playback duration for each video segment is the same, denoted by U seconds.

To determine the bandwidth for the logical channels, we need to first set a target latency T in multiples of video segment duration U and the number of segments N in the following equation:

$$T = \frac{m \cdot L}{N} \tag{3}$$

where m is a configurable parameter to tradeoff between performance and system complexity. Given the same target latency T, increasing m will result in larger value of N (i.e., dividing the video into more segments of shorter duration) and this in turn will

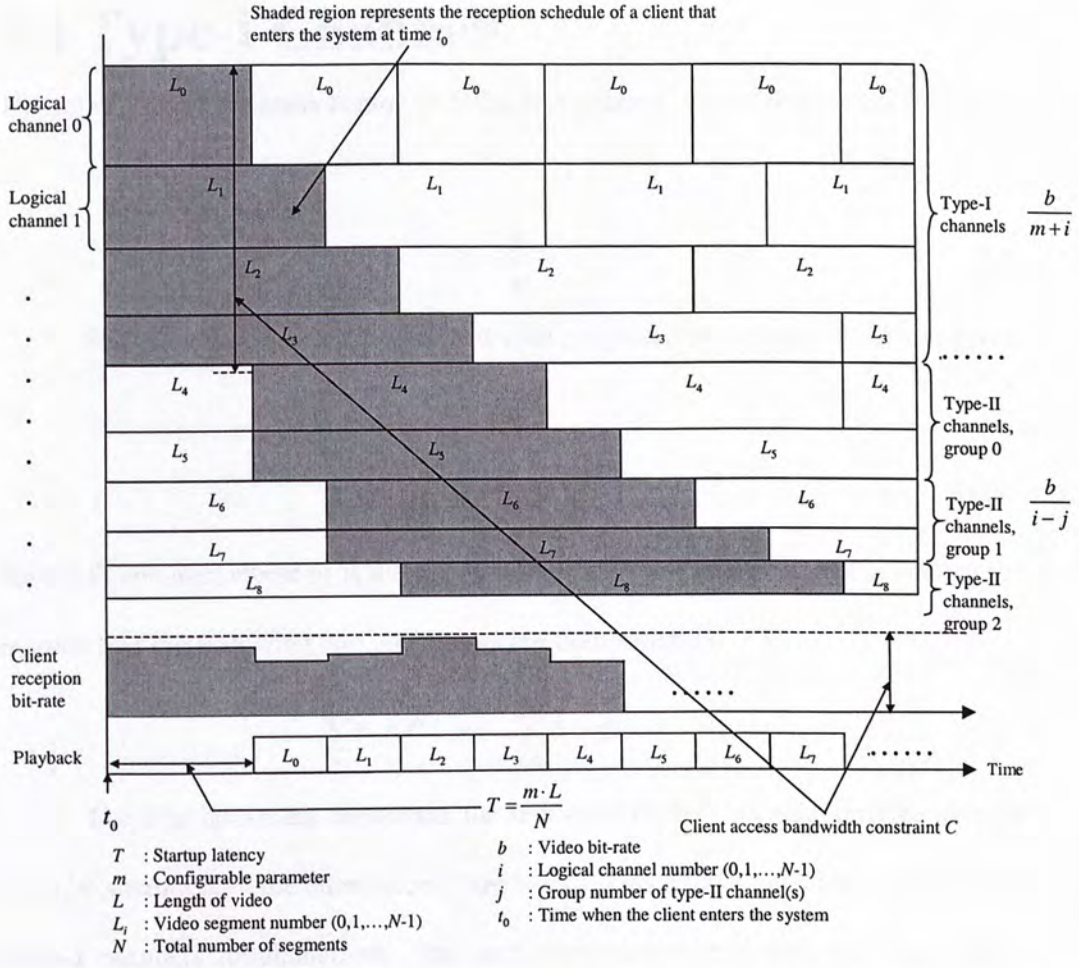

Figure 1. Bandwidth partition scheme and reception schedule in Consonant Broadcasting with m=2.

reduce the bandwidth requirement, and vice versa. While larger m is desirable from the bandwidth point of view, some network technologies (e.g. IP multicast) have limited number of logical multicast channels (e.g. multicast IP addresses) and thus m cannot be too large. We tackle this problem in Chapter 5.

Next, each video segment is multicast over a separate logical transmission channel (e.g. an IP multicast group address) in the network. There are two types of logical channels, namely Type-I, and Type-II channels. We define their respective bandwidth partition schemes and reception schedules in the following chapters.

3.1 Type-I Channels

The set of Type-I channels begins with the first channel, with a bandwidth allocation of

$$B_0 = \frac{b}{m} \tag{4}$$

Subsequent channels are allocated with progressively less bandwidth as given by

$$B_i = \frac{b}{m+i}, \quad i = 0,1,...,n_1 - 1 \tag{5}$$

for the i^{th} channel, where n_1 is the total number of Type-I channels. We can solve for n_1 such that the following two constraints are both satisfied:

$$\sum_{i=0}^{n_1-1} B_i \le C \quad \text{and} \quad \sum_{i=0}^{n_1} B_i > C \tag{6}$$

The first constraint represents the requirement that the aggregate bandwidth must be smaller than the client access bandwidth. This allows the client to receive all Type-I channels simultaneously. The second constraint represents the requirement that we should allocate as many channels as the client access bandwidth will allow maximizing utilization of the client access bandwidth available.

It is worth noting that if we remove the client access bandwidth constraint C, the number of Type-I channels n_1 will simply equal to N, i.e., all channels are of Type-I. In this special case, the bandwidth partition scheme in (5) will be identical to the Poly-harmonic Broadcasting scheme [15]. Therefore, Poly-harmonic Broadcasting can be considered as a special case of Consonant Broadcasting when there is no client access bandwidth constraint.

Fig. 1 illustrates the operation of Type-I channels (channel 0 to 3). When a client enters the system to start a new video stream, it will immediately start caching

data from all Type-I channels simultaneously. The client can start playback after a latency of T seconds as the first video segment L_0 will be completely received by then.

In general, let t_0 be the time the client enters the system, and let c_i be the playback time for video segment L_i, which can be computed from

$$c_i = t_0 + (m+i)\cdot U,\ i = 0,1,...,n_1 - 1 \tag{7}$$

As the client caches all Type-I channels immediately at time t_0, it will have completely received video segment L_i by the time s_i given by

$$\begin{aligned} s_i &= t_0 + \frac{(L\cdot b)/N}{b/(m+i)}, \quad i = 0,1,...,n_1 - 1 \\ &= t_0 + (m+i)\cdot U, \quad \because \frac{L}{N} = U \end{aligned} \tag{8}$$

which precisely meets the playback schedule c_i's and thus playback continuity is guaranteed.

3.2 Type-II Channels

Type-II channels are divided into groups of consecutive channels as shown in Fig. 1. When a client completes receiving a video segment, the corresponding channel will be released. With the increased available client access bandwidth, the client can then begin to receive a group of Type-II channels. Channels within the same group have their bandwidth allocated according to (9) and subject to the same client access bandwidth constraint. For the example in Fig. 1, at time $(t_0+(m+1)U)$, the client completes receiving video segment L_1 (releases channel 1) and then begins receiving data from channel 6 and 7. These two channels form the group 1 of Type-II channels.

It may appear that it is simpler to reallocate all the available bandwidth to a single channel instead of a group of channels. However, doing so will unnecessarily increase the bandwidth requirement because there is more than enough time to transmit the new video segment. To see why, consider video segment L_4 being

broadcast in channel 4 in Fig. 1. Channel 0 is released at time t_0+2U and video segment L_4 will be playback at time t_0+6U, thus we have 4U seconds to transmit the video segment. However, since the bandwidth released by channel 0 is equal to b/2, video segment L_4 will be transmitted completely in just 2U seconds if all the available bandwidth is allocated for this logical channel. The extra 2U seconds available are then wasted and the network bandwidth is unnecessarily increased.

We tackle this deficiency by transmitting a video segment in a *just-in-time* manner. For the previous example, we can transmit video segment L_4 using the lowest possible bit-rate, i.e., b/4, to meet the playback schedule. Then we allocate the remaining bandwidth to the next video segment using the same *just-in-time* scheduling procedure until no more video segment transmissions can be added. These channels then form a group of Type-II channels.

Let $n_{2,j}$ be the number of channels in group j, where j=0,1,..., etc. Then the bandwidth allocation for channels in group j is given by

$$B_i = \frac{b}{i-j}, \quad \text{for } i \geq n_1 \tag{9}$$

and the number of channels in group j can be determined from solving for $n_{2,j}$ in

$$\sum_{i=j+1}^{n_1+n_{2,0}\cdots+n_{2,j}-1} B_i \leq C \text{ and } \sum_{i=j+1}^{n_1+n_{2,0}\cdots+n_{2,j}} B_i > C \tag{10}$$

which represents the client bandwidth constraints.

To prove playback continuity for video segments broadcast in Type-II channels, we consider an arbitrary Type-II channel i in group j. As the client begins receiving all channels in group j at the time

$$t_0 + (m+j)\cdot U \tag{11}$$

and it takes a duration of $(U \cdot b)/B_i$ seconds to completely receive the video segment L_i, we can then compute the time s_i at which video segment L_i is ready for playback from

$$s_i = t_0 + (m+j)\cdot U + \frac{U \cdot b}{B_i} \tag{12}$$

Substituting B_i from (9) into (12) we obtain

$$\begin{aligned} s_i &= t_0 + (m+j)\cdot U + (i-j)\cdot U \\ &= t_0 + (m+i)\cdot U \equiv c_i \end{aligned} \tag{13}$$

which is equal to the playback schedule and thus playback continuity for video segments broadcast in Type-II channels is also guaranteed.

3.3 Client Buffer

As Fig. 1 illustrates, the amount of video data accumulated in the client buffer can vary during the video session. Assume a client arrives at the system at time t_0. Let t_i's be the time instants at which a change in the reception schedule occurs, e.g., when the client releases an existing channel (i.e. video segment completely received) and begins to receive data from a new group of Type-II channels. As video segments are of the same size U and channel bit-rates are integral fractions of the video bit-rate b, we can compute t_i (i=1,2,...) from

$$t_i = T + (i-1)\cdot U \tag{14}$$

In particular, at time t_i, the client begins playback of video segment L_{i-1} and begins to receive group $i-1$ of Type-II channels (see Fig. 1).

Let H_i be the amount of video data accumulated but not yet played back at time t_i. Then H_0=0, and we can compute H_1 from

$$H_1 = \sum_{i=0}^{n_1-1} m \cdot U \cdot B_i \tag{15}$$

where n_1 is the total number of Type-I channels received and the B_i's are their respective bit-rates. Similarly, we can compute H_2 from

$$H_2 = H_1 - U \cdot b + \sum_{k=1}^{n_1+n_{2,0}-1} U \cdot B_k \tag{16}$$

where the first term is the buffer occupancy at time t_1, the second term is the amount of video data consumed, and the last term is the amount of video data received from time t_1 to t_2 (i.e., U seconds).

In general, we can compute H_i ($i \geq 2$) recursively from

$$H_i = H_{i-1} - U \cdot b + \sum_{k=i-1}^{n_1+n_{2,0}+\ldots+n_{2,i-2}-1} U \cdot B_k \tag{17}$$

As both video data consumption rate and total reception rate are constant within a given time interval from t_i to t_i+1, the maximum client buffer requirement must occur at one of the time instants given by the t_i's. Hence we can determine the maximum client buffer requirement H simply by finding the maximum H_i:

$$H = \max\{H_i \mid \forall i = 0,1,\ldots\} \tag{18}$$

Chapter 4

PERFORMANCE EVALUATION

In this chapter, we evaluate performance of Consonant Broadcasting and compare it to some of the current state-of-the-art periodic broadcasting schemes, including Skyscraper Broadcasting (SB), Greedy Disk-Conserving Broadcasting (GDB), Staircase Data Broadcasting (SDB), Poly-harmonic Broadcasting (PHB), and Pagoda Broadcasting (PB). In computing the numerical results, we use a video length of L=7200 seconds (2 hours) and assume the client access bandwidth is equal to twice the video bit-rate, i.e., $2b$. For example, if the video bit-rate is 3Mbps, then the client access bandwidth is 6Mbps, within the limit of current 10Mbps Ethernet. For the existing periodic broadcasting schemes, we apply the optimization procedure proposed by the original studies [11-13,15-16] to configure their operating parameters. Interested readers are referred to the original literature for details. The following chapters compare these broadcasting schemes in terms of startup latency and client buffer requirement, with respect to the network bandwidth required.

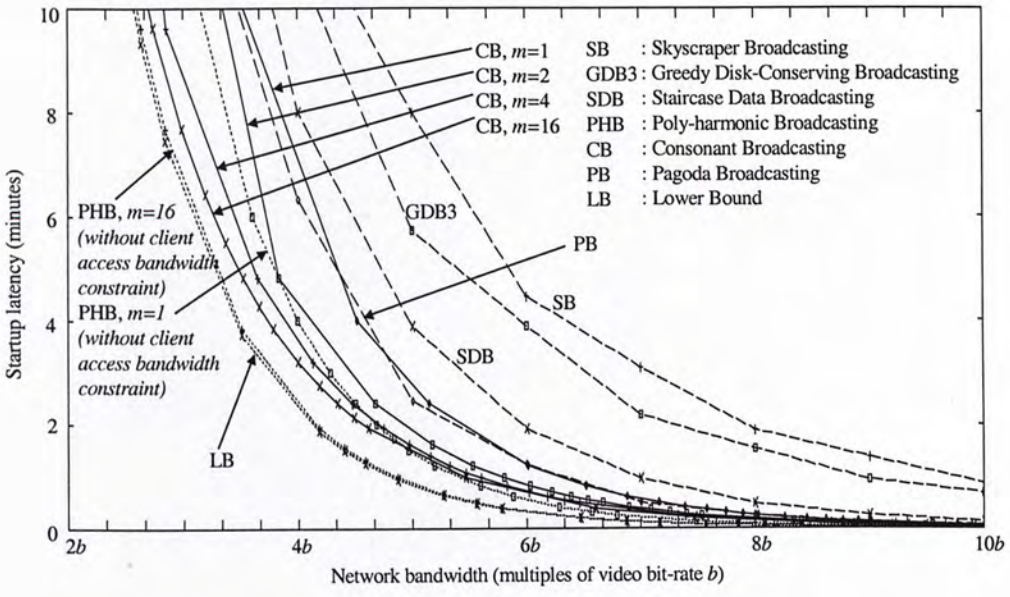

Figure 2. Startup latency versus network bandwidth at large latency range.

4.1 Startup Latency versus Network Bandwidth

Startup latency is defined as the maximum time from a client entering the system to the time video playback starts. With a client access bandwidth of $2b$, we plot in Fig. 2 the startup latency versus the network bandwidth ranging from $2b$ to $10b$.

The results in Fig. 2 show that PHB achieves the lowest startup latency, close to the theoretical lower bound when configured with large value of m (e.g. 16). Similarly, PB also achieves very good performance, comparable to CB with m=1. However, unlike the other schemes, we did not apply the client access bandwidth constraint in computing results for PHB and PB and thus the results are not directly comparable. Nevertheless, this result shows the performance loss due to limited client access bandwidth.

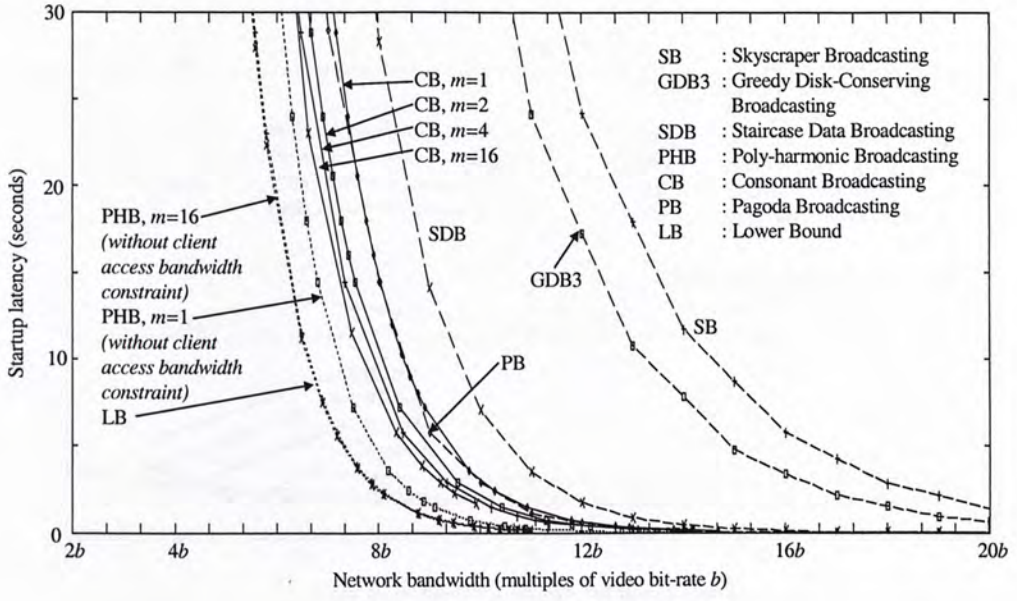

Figure 3. Startup latency versus network bandwidth at small latency range.

Except for PHB and PB, it is clear from the results that CB achieves the lowest startup latency. This is true even for m=1, which generates the least number of video segments (and hence system complexity) given the same system parameters. Increasing m can further reduce the startup latency but at the expense of higher system complexity. For a network bandwidth of $5b$, CB with m=4 achieves startup latency 81%, 74%, and 60% lower than SB, GCB, and SDB respectively.

Fig. 3 compares the startup latency of the broadcasting schemes for larger network bandwidth ranging from $2b$ to $20b$. At this range the startup latency is reduced to seconds, well within the response time required in a video streaming service. Again the observation is consistent with the results in Fig. 2, showing that CB achieving the lowest startup latency. For example, with a network bandwidth of $10b$, CB with m=4 can achieve a startup latency of only 2 seconds, which is 96%, 95%, and 72% lower than SB, GCB, and SDB respectively.

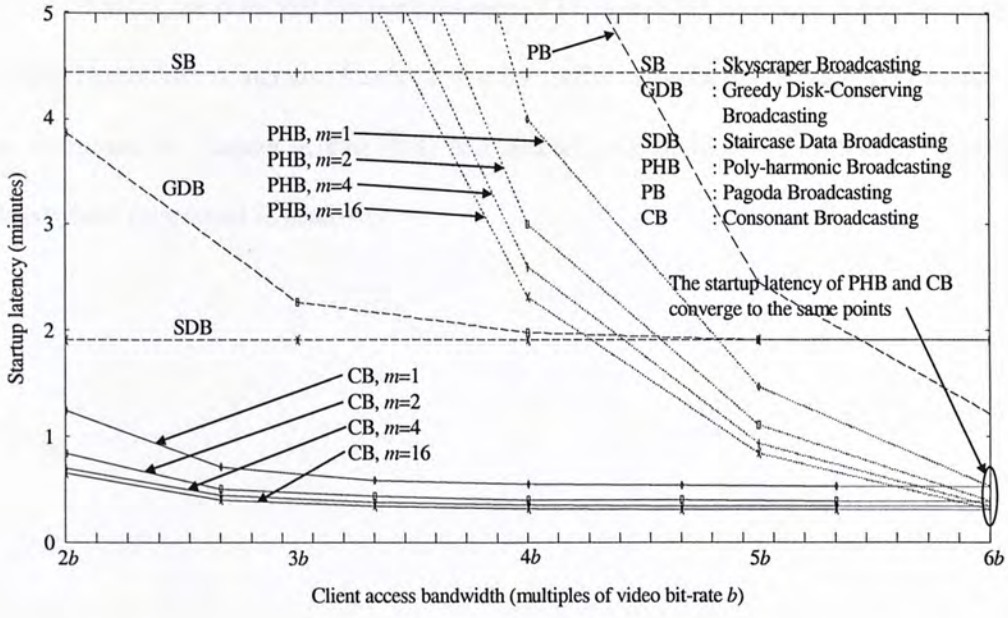

Figure 4. Startup latency versus client access bandwidth (network bandwidth = 6*b*).

4.2 Startup Latency versus Client Access Bandwidth

Fig. 4 plots the startup latency versus the client access bandwidth ranging from 2*b* to 6*b*, where *b* is the video bit-rate. The network bandwidth is equal to 6*b*. There are three observations.

First, CB clearly outperforms the other schemes, especially when the client access bandwidth is low. This is a significant property as the client access network in practice will likely have substantially lower bandwidth than backbone networks. Second, the performance of PHB and PB degrade significantly when the client access bandwidth is reduced. This is because both broadcasting schemes require a client access bandwidth that equals to the network bandwidth. Therefore in case the client access bandwidth is the bottleneck, the network bandwidth in fact cannot be fully utilized, leading to the performance degradation.

Finally, we note that the performance of PHB and CB converge when the client access bandwidth is increased to $6b$, i.e., same as the network bandwidth. This verifies, as discussed in Chapter 3, that PHB is a special case of CB when the client access bandwidth constraint is removed.

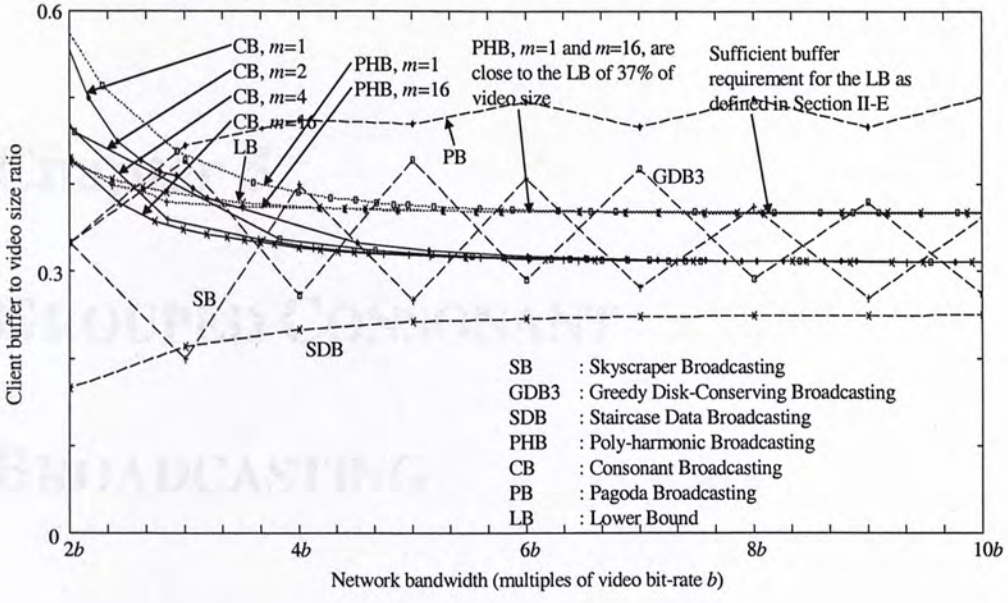

Figure 5. Client buffer to video size ratio versus network bandwidth.

4.3 Client Buffer Requirement

Fig. 5 plots the maximum client buffer requirement versus the network bandwidth, ranging from $2b$ to $10b$. The client buffer requirement is normalized and expressed as the ratio of the size of the video title. For example, a ratio of 0.3 means that the client buffer must be large enough to store up to 30% of the whole video title.

We can observe from Fig. 5 that the maximum client buffer requirement for all the schemes are comparable, and varies within a range from 0.2 to 0.5. For example, at a network bandwidth of $5b$, the maximum client buffer requirements are 27%, 43%, 24%, and 32% for SB, GDB, SDB, and CB (with m=4) respectively. The only broadcasting scheme that consistently achieves lower client buffer requirement is SDB. Therefore the client buffer requirements of these broadcasting schemes are comparable.

Chapter 5

GROUPED CONSONANT BROADCASTING

The previous chapters show that CB outperforms existing state-of-the-art periodic broadcasting schemes. In particular, the performance continues to improve for larger values of the system parameter m in (3). The tradeoff, however, is increased system complexity in terms of the number of channels required for broadcasting the video segments.

For example, given a client access bandwidth constraint of $2b$ and network bandwidth of $8.98b$, CB with m=4 can achieve a startup latency of only 5.76 seconds but this requires 5,000 network multicast channels. In networks with limited number of multicast channels (e.g. group addresses in IP multicast), this requirement can become a significant bottleneck. To tackle this problem, we present in the following chapter a Grouped Consonant Broadcasting (GCB) scheme to dramatically reduce the number of network channels required, with a small tradeoff in performance.

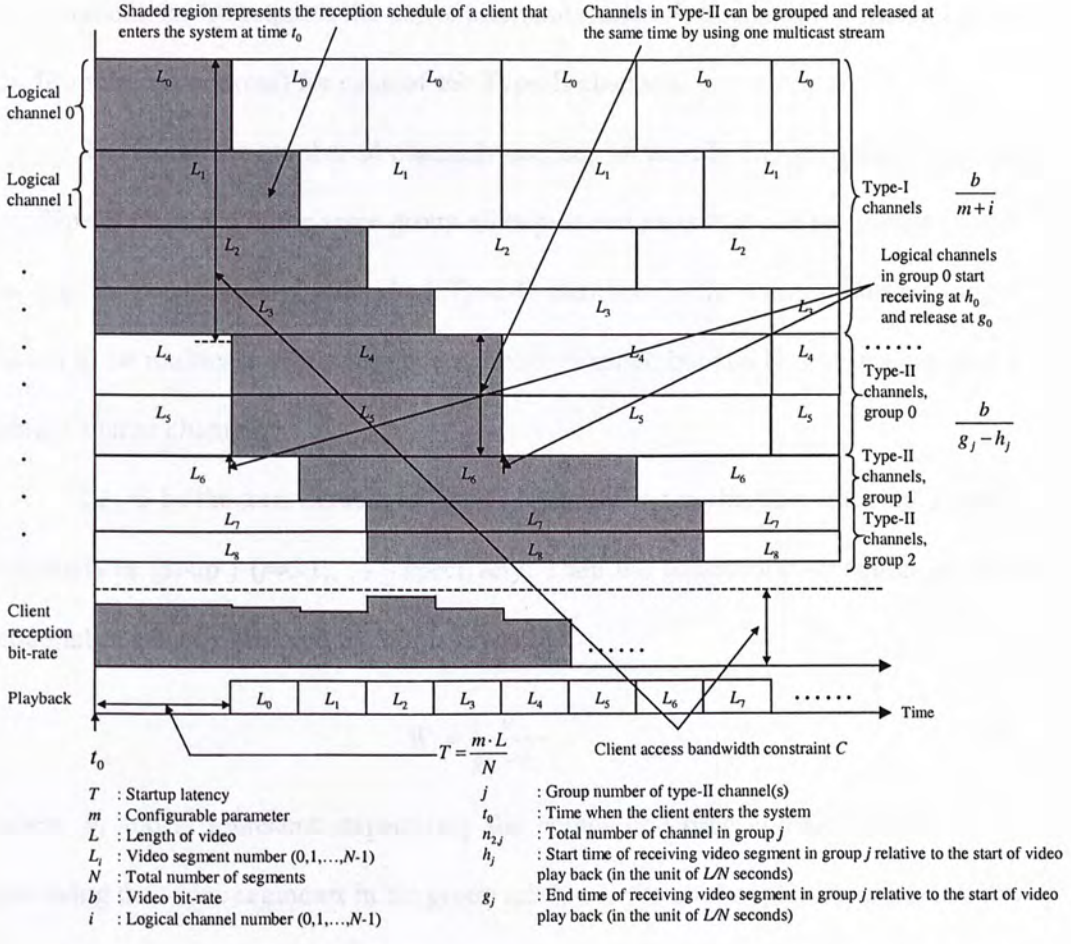

Figure 7. Bandwidth partition scheme and reception schedule in Grouped Consonant Broadcasting with m=2.

5.1 Bandwidth Partitioning and Reception Schedule

Type-I channels in GCB is the same as the original CB as defined in (5) and (6). The difference is in the design of the Type-II channels. In CB, reception of Type-II channels in the same group begins at the same time but ends at different times due to the just-in-time scheduling principle. While this technique can reduce the bandwidth

requirement, it also requires the use of a separate network transmission channel (e.g., an IP multicast address) for each of the Type-II channels.

To reduce the number of channels needed, we modify CB such that reception of Type-II channels in the same group all begins and ends at the same time as shown in Fig. 7. Consequently, individual Type-II channels in the same group no longer needs to be multicast over a separate network channel, but can be transmitted over a single shared channel.

Let n_1 be the total number of Type-I channels and $n_{2,j}$ be the number of Type-II channels in group j (j=0,1,...) respectively. Then the bandwidth allocation of each channel in group j, denoted by W_j, is given by

$$W_j = \frac{b}{g_j - h_j} \tag{19}$$

where g_j and h_j represent respectively the completion time and the start time for receiving the video segments in the group relative to the start of playing a video in the unit of U seconds, and are given by

$$g_j = \begin{cases} n_1 & \text{,for } j = 0 \\ n_1 + n_{2,0} + ... + n_{2,j-1} & \text{,otherwise} \end{cases} \tag{20}$$

$$h_j = \begin{cases} j & \text{,for } j \le n_1 \\ n_1 + n_{2,0} + ... + n_{2,j-n_1-1} & \text{,for } j > n_1 \end{cases} \tag{21}$$

We can determine the number of channels in group j from solving for $n_{2,j}$ in

$$\sum_{i=h_{j+1}}^{n_1+n_{2,0}+...+n_{2,j}-1} B_i \le C \text{ and } \sum_{i=h_{j+1}}^{n_1+n_{2,0}+...+n_{2,j}} B_i > C \tag{22}$$

where $B_i = W_j$ for all i's in the range $g_j \le i < g_{j+1}$.

To prove playback continuity for video segments broadcast in Type-II channels, we consider an arbitrary Type-II group, say group j, comprising video

segments $\{L_i \mid g_j \le i < g_{j+1}\}$. As the client begins receiving all channels in group j at time

$$t_0 + (m + h_j) \cdot U \tag{23}$$

and it takes $(U \cdot b)/W_j$ seconds to completely receive the video segments, the time s_j at which all video segments in the group is ready for playback can be computed from

$$s_j = t_0 + (m + h_j) \cdot U + \frac{U \cdot b}{W_j} \tag{24}$$

Substituting W_j from (19) into (24) we obtain

$$\begin{aligned} s_j &= t_0 + (m + h_j) \cdot U + (g_j - h_j) \cdot U \\ &= t_0 + (m + g_j) \cdot U \\ &\le t_0 + (m + i) \cdot U, \text{ for } g_j \le i < g_{j+1} \end{aligned} \tag{25}$$

which is equal to or earlier than the playback schedule and thus guaranteeing playback continuity.

5.2 Client Buffer Requirement

Compared to CB, GCB generally requires more client buffer because all but the first video segments in a Type-II group are not transmitted in a just-in-time manner. Instead, they are transmitted at a higher rate so that reception can be completed at the same time as the first video segment. Consequently, these video segments are completely received before the time for playback and thus occupy more client buffer.

Specifically, the client will playback video segment L_i at time $t_0 + (m + i) \cdot U$, where t_0 is the time the client entered the system. We define H_i as the amount of video data received but not yet played back at time t_i as defined in (14). As channel switching occurs only at the time instants $t_0 + (m + h_j) \cdot U$ for j=0,1,..., we only need

to consider the buffer occupancy at these instants. We can compute H_{h_j} recursively from

$$H_{h_j+1} = \begin{cases} \sum_{k=0}^{n_1-1} m \cdot U \cdot B_k & ,\text{for } j = 0 \\ H_{h_{j-1}+1} - (h_j - h_{j-1}) \cdot U \cdot b + (h_j - h_{j-1}) \cdot \sum_{k=h_j}^{n_1+n_{2,0}+\ldots+n_{2,j-1}-1} U \cdot B_k & ,\text{for } j > 0 \end{cases} \quad (26)$$

where in the second case the first term is the buffer occupancy at time $t_{h_{j-1}+1}$ (i.e., when the last channel is released), the second term and the last term are the amount of video data consumed and the amount of video data received from time $t_{h_{j-1}+1}$ to t_{h_j+1} (i.e., $(h_j - h_{j-1}) \cdot U$ seconds) respectively. The maximum client buffer requirement can then be computed from $H = \max\left\{H_{h_j} \mid \forall j\right\}$.

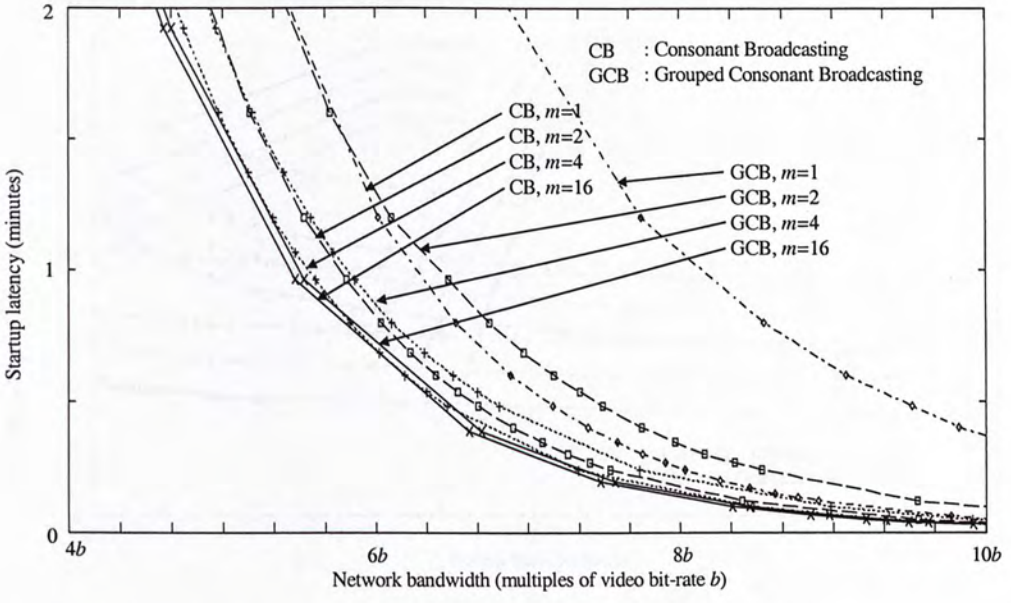

Figure 8. Startup latency versus network bandwidth.

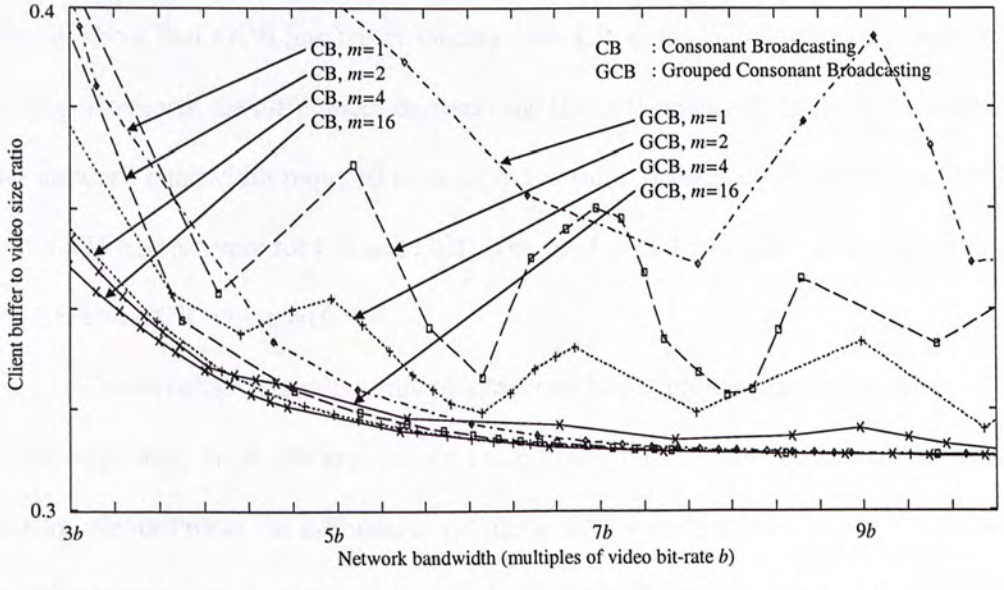

Figure 9. Client buffer to video size ratio versus network bandwidth.

5.3 Performance Tradeoffs

As most Type-II channels in GCB are transmitted at higher than necessary bit-rate, we can expect it to require more network bandwidth as well as client buffer to achieve the

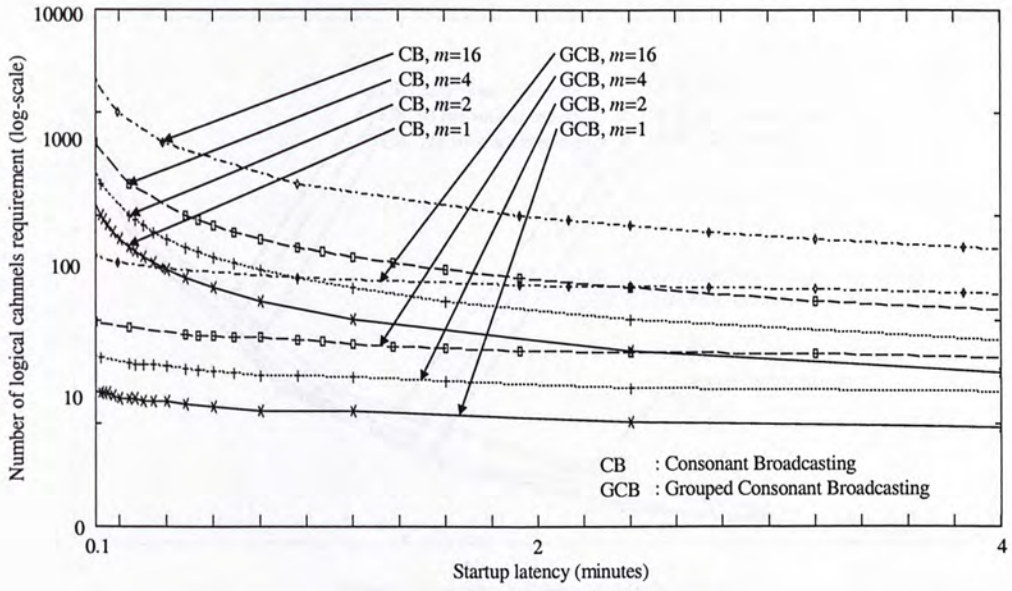

Figure 10. Number of logical channels requirement (log-scale) versus startup latency.

same latency in CB. We first consider the bandwidth tradeoff in Fig. 8. The results clearly show that GCB has larger latency than CB at the same network bandwidth setting. However, the differences decrease significantly when m is large. For example, the network bandwidth required to achieve the same latency of 15 seconds is $7.48b$ and $8.47b$ respectively for CB and GCB with m=2, and $7.14b$ and $7.23b$ respectively for CB and GCB with m=16.

We also observe similar tradeoffs in client buffer requirement as shown in Fig. 9. As expected, GCB always require more client buffer than CB under the same setting. Nevertheless the differences are again significantly smaller when m is large. Another observation is that variation in the client buffer requirement with respect to network bandwidth is substantially larger in GCB. Thus more careful planning is needed to strike a balance between client buffer requirement and network bandwidth requirement.

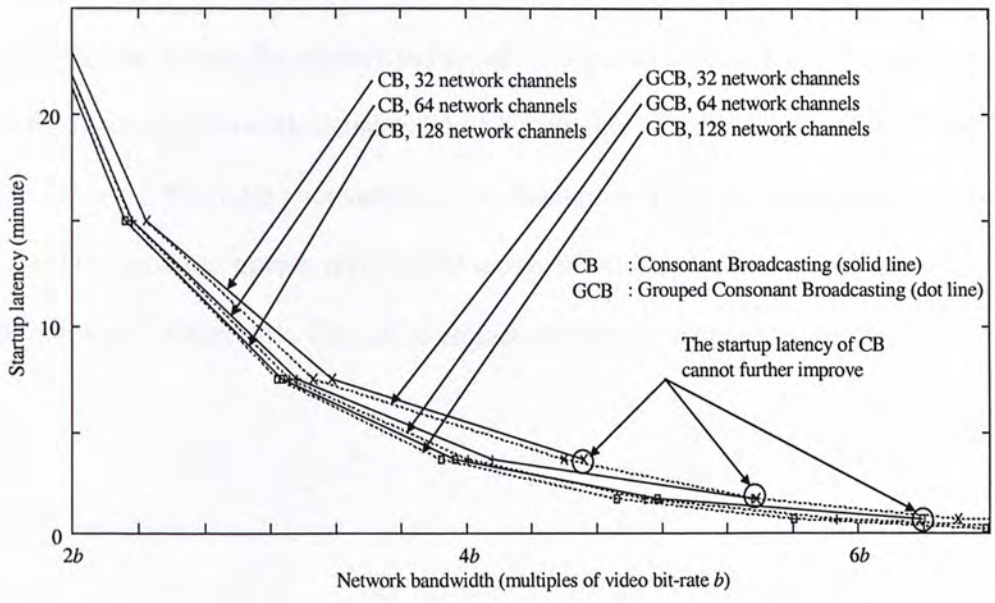

Figure 11. Startup latency versus network bandwidth for fixed number of network channels.

In contrast to the two tradeoffs, GCB gains in terms of the number of network transmission channels required. Fig 10 plots the number of channels required to achieve a given latency for CB and GCB. The results clearly show the significant reduction achieved by GCB. For example, with a latency constraint of 15 seconds, CB and GCB requires 1920 and 104 channels respectively with m=4, and 7680 and 434 channels respectively with m=16. The tradeoffs in network bandwidth and client buffer requirement in these two cases are 5.47% and 2.76% respectively for m=4, but only 1.27% and 1.28% respectively when m=16

On the other hand, if the number of network channels available is the limiting factor, then GCB can achieve lower network bandwidth requirement than CB for a given startup latency by observing the result in Fig 11. For example, given a maximum of 64 channels and startup latency of 3.75 minutes, GCB requires 4b

network bandwidth, which CB requires 4.12*b*. Moreover, in Fig 11, we observe that, by further increasing the network bandwidth, CB cannot achieve lower startup latency as the number of network channels limits the number of video segments *N* defined in (3), however, GCB does not subject to this limitation. Thus GCB will be particularly useful for networks having very limited supply of network transmission channels (e.g. IP multicast). Otherwise, CB can be employed to achieve better performance.

Chapter 6

IMPLEMENTATION AND BENCHMARKING

To prove the feasibility of CB/GCB as well as to verify the theoretical performance results, we implemented using C++ a system prototype running on Red Hat Linux 7.0. We employ UDP over IP multicast as the network transmission protocol and set up a testbed with off-the-shelve PCs connected by an IP-multicast-ready FastEthernet switch.

The video server sends each of the channel or (group of channels in GCB) in CB/GCB using UDP over a separate IP multicast address. Each video packet carries 1,400 bytes of video data and an 8-byte header comprising sequence number. The video packet size is chosen to match Ethernet's maximum frame size to prevent datagram fragmentation.

To begin a video session, the video client software joins the corresponding multicast groups as defined by the CB/GCB algorithm by sending out an IGMP Join Group requests. The IGMP request will be handled by the network switch and thus has no effect on the video server. Upon receiving the IGMP request the network switch will begin forwarding to the client packets of the requested multicast group.

The client then resequences the received packets based on the sequence number embedded in the packet header. Once a video segment is completely received, the client will send an IGMP Leave Group request to the network switch, which then stops the forwarding of data belonging to the requested multicast group.

6.1 Practical Issues

Implementing the system prototype reveals a number of practical issues not found in the theoretical model. First, the time in joining and leaving an IP multicast group is not precise but subject to delay variations in packet transmission and request processing. In joining a multicast group, the client may experience a small delay before data packets of the video segment are received. In a small local area network such as our experimental test-bed, the channel switching latency is in the order of 10^{-3} seconds. Given that the client has buffered video segment L_0 comprising multiple seconds of video data before commencing playback, this channel switching delay can readily be absorbed. For larger network that involves multicast routing the channel switching delay will be larger and thus extra measures (e.g. increasing the amount of prefetch data before playback commence) may be needed to prevent playback starvation during channel switching.

Similarly, the client may not be able to leave a multicast group immediately after receiving a video segment, and thus additional duplicate data packets may continue to arrive at the receiver. The client can detect and discard these duplicate packets. Alternatively, the client can simply process them normally as there is no harm overwriting existing data with the same data. However, these duplicate data do incur additional bandwidth usage at the client access link (see Section 6.2).

Another aspect where the implementation deviates from the theoretical model is in data transmission. Specifically, we have thus far modeled the transmission of video data as a continuous bitstream, i.e., using a fluid-flow like model. In practice the server must packetize video data into discrete UDP datagrams for transmission. In our implementation, we use a datagram size of 1,408 bytes (1,400-byte video data plus 8-byte header) excluding UDP and IP headers. Thus with a configuration of N=1,000 in our experiments, the inter-packet transmission time can vary from the order of 10^{-2} seconds (0.02 seconds etc.) to a few seconds (5.7 seconds etc.) depending on the broadcast duration.

Our experiments show that the large deviations in the inter-packet transmission time can result in substantial variations (~20% bit-rate variation averaged over one-second interval) in the bit-rate of the aggregate network traffic. We tackle this problem in GCB by combining all video segments within the same group into a single data block, and then perform packetization for the combined data block instead of individually packetizing each video segment for transmission. Our experiments show that this can reduce the bandwidth variation to negligible levels (order of 10^{-1} percentage bit-rate variation averaged over one-second interval) without any impact on other parts of the system.

6.2 Experimental Results

We conducted extensive benchmarking experiments to collect three performance results, namely startup latency, aggregate bit-rate of all channels, and peak aggregate reception bit-rate to compare against our theoretical calculations. In all experiments, we use the system parameters of L=4401, C=2.84, m=2 and b=1.42. The video is a MPEG-1 encoded system stream multiplexing one video stream with one audio

Table II. Comparison of theoretical and experimental results (with m=2).

Config	Latency		Aggregate Bit-rate of all channels			Peak aggregate reception bit-rate		
N / N_G*	Computed	Measured	Computed	Measured	Difference	Computed	Measured	Difference
50/19	176.04	176.10	5.8208	6.11	+4.97%	2.84	2.98	+4.93%
80/21	110.03	110.05	6.8302	7.17	+4.97%	2.84	2.98	+4.93%
100/22	88.02	88.04	7.2779	7.64	+4.98%	2.84	2.98	+4.93%
200/28	44.01	44.05	8.6611	9.10	+5.07%	2.84	3.00	+5.63%
500/33	17.60	18.00	10.6277	11.16	+5.00%	2.84	3.05	+7.39%
800/37	11.00	11.20	11.5828	12.16	+4.98%	2.84	3.11	+9.51%
1000/38	8.80	9.00	12.1096	12.71	+4.96%	2.84	3.09	+8.80%

* N and N_G are the number of video segments and number of channels respectively.

stream. We conducted benchmarks for a total of 7 GCB system configurations, with number of video segments N ranging from 50 to 1,000. For each configuration, we obtain the performance data by averaging data collected from 20 benchmark runs.

We first consider startup latency that is measured from within the client software. The results show that the experimental results agree closely with the theoretical calculations. The minor differences are likely due to network delay and software processing delay. Next, we measured the aggregate network bit-rate of all channels using a hardware protocol analyzer connected to the Ethernet switch's mirroring port, which forwards all packets passing through the switch. The measured results exhibit a consistent 5% increase in bandwidth usage compared to the theoretical calculations. This increase is due to the header overheads in the application-layer protocol (8 bytes), UDP (8 bytes), IP (24 bytes), and Ethernet (18 bytes). With a UDP datagram payload of 1,400 bytes, the combined header overhead is equal to (8+8+24+18)/1458=4%, which closely matches the measured results.

Finally, we measure the aggregate reception bandwidth usage in the client access link, again using a hardware protocol analyzer. Unlike the aggregate network

bit-rate, the reception bit-rate is not constant and does vary depending on which video segments are being received. Nevertheless, we are more interested in the peak bandwidth usage and thus we measure the maximum bandwidth usage averaged over a 10-second window. The results show similar header overhead induced bit-rate increases (~5%) for configurations with N up to 200. For larger values of N the differences widen further to up to 9.51%. Our study of the log data shows that two factors lead to the bit-rate increase. First, larger values of N result in more frequent channel switching, and as discussed earlier in Section 6.1, there is some delay from the time the client leaves a multicast group to the time the network switch stops forwarding the multicast data. This results in some duplicated data being transmitted to the client, only to be discarded by the client's operating system. The second reason is due to the specific network switch we used in the experiment. Our results show that there seems to be bugs in the switch's hardware, resulting in some random multicast data transmitted to the client *after* the switch has pruned the multicast tree. This specific problem is easy to escape notice because the random multicast data will be discarded by the client's operating system (as the client has left the multicast group already) and thus will not case any data transmission or application error. We expect this problem to be resolved in future revisions of the switch hardware.

Chapter 7

DYNAMIC CONSONANT BROADCASTING

Given that open-loop and closed-loop architectures perform well at heavy-load and light-load conditions respectively, in this chapter, we further extend the original CB to a unified architecture called Dynamic Consonant Broadcasting (DCB) that can perform well under all load conditions. There are two key principles to achieving this goal.

First, the virtue of closed-loop architectures is that the transmission schedules adapt to the request arrival patterns such that network bandwidth are utilized only when needed, thus reducing resource requirements at light loads. Second, the virtue of open-loop architectures is that the transmission schedules are optimized for heavy loads and are kept unchanged, thereby achieving constant performance irrespective of the request arrival patterns.

The Dynamic Consonant Broadcasting (DCB) architecture integrates these two virtues into a unified architecture to achieve both the infinite scalability of open-loop architectures and the efficient bandwidth utilization of closed-loop architectures. The principle is to dynamically schedule the transmission of video

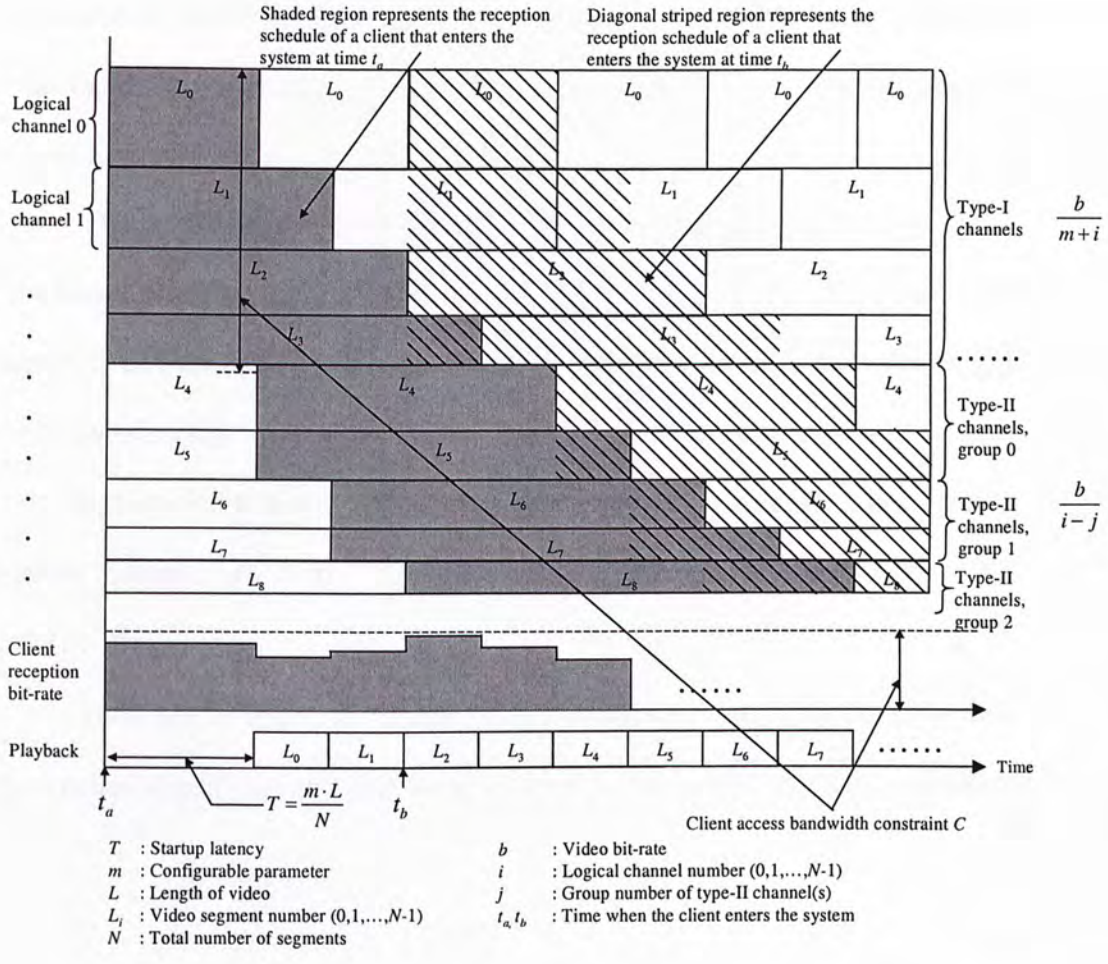

Figure 12. Bandwidth partition scheme and reception schedule in Dynamic Consonant Broadcasting with m=2.

segments based on a *virtual transmission schedule*. We first present the virtual transmission schedule in Section 7.1 and then present the dynamic scheduling algorithm in Section 7.2. Finally, we evaluate its performance in Section 7.3.

7.1 Virtual Transmission Schedules

In DCB, transmissions of video segments are scheduled dynamically according to request arrivals. In other words, the server will not transmit any video data until clients are admitted into the system, thus similar to closed-loop

architectures. However when scheduling the transmission, DCB does not schedule them in arbitrary time instants. Instead, DCB schedules transmission according to a virtual transmission schedule as shown in Fig. 12.

This virtual transmission schedule is based on the broadcasting schedule of Consonant Broadcasting (CB) defined in Chapter 3, which employs fixed-size video segments and variable bit-rate logical channels. We first divide a video title into N equal-duration segments of playback duration U seconds, thus equal-size in each video segment, Each of the video segments is then broadcast repeatedly in a separate logical channel, i.e., video segment L_i is broadcast in logical channel i, for i=0,1,…,N–1.

There are two types of channels in the virtual transmission schedules. The first type, called Type-I channels, begins with the first channel with a bandwidth allocation of

$$B_0 = \frac{b}{m} \tag{27}$$

where b is the video bit-rate and m is a configurable parameter to tradeoff between performance and system complexity.

Subsequent Type-I channels are allocated with progressively less bandwidth given by

$$B_i = \frac{b}{m+i}, \quad i = 0,1,...,n_1 - 1 \tag{28}$$

for the i^{th} channel, where n_1 is the total number of Type-I channels. The value of n_1 is obtained from solving the following constraints:

$$\sum_{i=0}^{n_1-1} B_i \le C \quad \text{and} \quad \sum_{i=0}^{n_1} B_i > C \tag{29}$$

where C is the client access bandwidth limit. These two constraints ensure that the client access bandwidth can be fully utilized but not exceeded.

Finally, as the client cannot begin video playback until the first video segment is completely received, the maximum latency, denoted by T, is given by

$$T = \frac{Ub}{B_0} = m \cdot U = \frac{m \cdot L}{N} \tag{30}$$

Next, Type-II channels are divided into groups of consecutive channels as shown in Fig. 12. When a client completes receiving a video segment, the corresponding channel will be released and the client can then begin to receive a new group of Type-II channels. Channels within the same group have their bandwidth allocated according to (31) and subject to the same client access bandwidth constraint.

Let $n_{2,j}$ be the number of channels in group j, where j=0,1,..., etc. Then the bandwidth allocation for channels in group j is given by

$$B_i = \frac{b}{i-j}, \quad \text{for } i \geq n_1 \tag{31}$$

and the number of channels in group j can be determined from solving for $n_{2,j}$ in

$$\sum_{i=j+1}^{n_1+n_{2,0}\cdots+n_{2,j}-1} B_i \leq C \text{ and } \sum_{i=j+1}^{n_1+n_{2,0}\cdots+n_{2,j}} B_i > C \tag{32}$$

Again (32) represents the client access bandwidth constraint.

7.2 Dynamic Broadcasting Schedules

The previous virtual transmission schedules represent the set of valid transmission schedules of which active video data transmission can occur. To determine *when* to activate *which* video transmissions, we need an algorithm to dynamically schedule the transmissions according to the request arrivals.

Let's reconsider the broadcasting schedules in CB in Fig. 12. Initially, all logical channels are inactive, i.e., not transmitting video data. When a client enters the system at say time t_a, the system actives a series of video segment transmissions according to the reception schedule of the client as defined in the CB. Whenever a video segment is completely received, the corresponding logical channel will become inactive again until the next client arrival.

Let $\{t_{i,on}, t_{i,off}\}$ be the time instants when the i^{th} logical channel can be activated and deactivated respectively. For Type-I channels, $\{t_{i,on}, t_{i,off}\}$ are defined as

$$\{t_{i,on}, t_{i,off}\} = \{t, t + (m+i) \cdot U\} \tag{33}$$

where t is the client arrival time (i.e. $t = t_a$). Similarly, $\{t_{i,on}, t_{i,off}\}$ in Type-II channels are defined as

$$\{t_{i,on}, t_{i,off}\} = \{t + (m+j) \cdot U, t + (m+i) \cdot U\} \tag{34}$$

where j is the group number of the Type-II channels as defined in CB. Note that a channel will be activated only if it is a part of the transmission schedule for an active client.

When another client enters the system at say time t_b, DCB will reactive the corresponding logical channels if they have become inactive. Otherwise, it simply extends the active duration of the logical channel until time $t_{i,off}$ for the new client. In this way the video transmissions during the overlapping duration are shared by the two clients, thus resulting in resource savings. When the system load increases, more and more of the broadcasting schedules will be activated. In extremely heavy loads, all schedules may be activated and DCB then degenerates into CB.

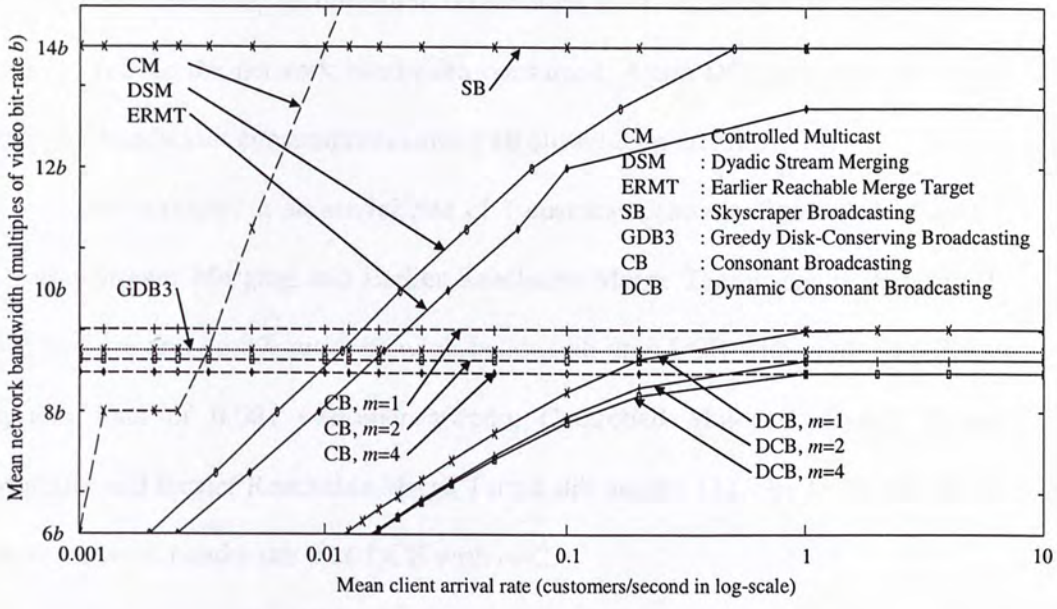

Figure 13. Mean network bandwidth versus client arrival rate.

7.3 Performance Evaluation

As to evaluate the performance of DCB, we study the relation between system load and network bandwidth consumption. Fig. 13 plots the mean network bandwidth consumed by various architectures to achieve a mean startup latency of 5 seconds with $2b$ client access bandwidth *under* client arrival rates ranging from 0.001 to 10 customers/second. Note that SB and GDB3 cannot be configured for arbitrary mean startup latency so we choose the closest mean startup latency of 5.85 seconds for SB and 7.05 seconds for GDB3

As expected, open-loop architectures have constant startup latency irrespective of the arrival rate. DCB (with m=2) outperforms both SB and GDB3, especially when the arrival rate is small. At high arrival rate, the performance of DCB converges to CB as all the logical channels are activated.

The closed-loop architectures, on the other hand, can benefit from lower arrival rates to reduce the network bandwidth consumed. Again DCB achieves the lowest network bandwidth consumption among all closed-loop architectures.

For example, at an arrival rate of 1 customers/second, Controlled Multicast, Dyadic Stream Merging and Earlier Reachable Merge Target require respectively 645%, 58% and 46.8% more network bandwidth than DCB with m=2. At a lighter system load of 0.001 customers/second, Controlled Multicast, Dyadic Stream Merging and Earlier Reachable Merge Target still require 112.7%, 77.2% and 59.5% more network bandwidth than DCB with m=2.

Moreover, as the DCB schedules its broadcasting schedule based on its of CB, we can simply apply the GCB algorithm defined in Chapter 5 to DCB tackling the problem of limited number of multicast channels and its performance can be evaluated accordingly.

Chapter 8

VARIABLE-BIT-RATE VIDEO STREAMING

As compared to constant bit-rate (CBR) encoding, variable bit-rate (VBR) encoding can produce more consistent visual quality. Moreover, Tan et al. [23] found that VBR-encoded video can achieve visual quality comparable to the CBR-encoded version at half the video bit-rate. Thus if we can stream VBR videos using less than half the network bandwidth compared to streaming CBR videos, we can potentially reduce the bandwidth requirement and at the same time achieve more consistent visual quality. In this chapter, we extend the CB to support streaming of videos encoded using variable-bit-rate (VBR) encoding algorithms. The following chapter presents transmission schedule of VBR video in Section 8.1 and then present the proof of playback continuity in Section 8.2. Finally, we evaluate its performance in Section 8.3.

8.1 Transmission Schedules

To construct the transmissions schedules of CB for streaming VBR videos, we divide a video title into N equal-duration segments, each of playback duration U seconds.

Each of these video segments is then broadcast repeatedly in a separate logical channel, i.e., video segment L_i is broadcast in logical channel i, for i=0,1,…,N−1.

However, unlike the CBR counterpart, the video segments of VBR videos are in general of different sizes. Thus we need to re-design the virtual transmission schedules of both Type-I and Type-II channels to accommodate the segment size differences.

We first consider Type-I channels. Let $b(t)$ be the video playback bit-rate of the video at playback point t. Then we allocate to the first channel a bandwidth of

$$B_0 = \frac{\int_0^U b(t)dt}{m \cdot U} \tag{35}$$

which is a generalization of (5) with the constant video bit-rate b replaced by the average rate of this particular video segment. Similarly, subsequent Type-I channels are allocated bandwidth according to

$$B_i = \frac{\int_{iU}^{(i+1)U} b(t)dt}{(m+i) \cdot U}, \quad i = 0,1,...,n_1 - 1 \tag{36}$$

for the i^{th} channel. Again, n_1 is the total number of Type-I channels, which is obtained from solving the following constraints:

$$\sum_{i=0}^{n_1-1} B_i \le C \quad \text{and} \quad \sum_{i=0}^{n_1} B_i > C \tag{37}$$

The maximum latency, denoted by T, is then given by

$$T = \frac{\int_0^U b(t)dt}{B_0} = m \cdot U = \frac{m \cdot L}{N} \tag{38}$$

which is the same as the CBR case.

Next we consider Type-II channels. Let $n_{2,j}$ be the number of channels in group j, where j=0,1,…, etc. Then the bandwidth allocation for channels in group j is given by

$$B_i = \frac{\int_{iU}^{(i+1)U} b(t)dt}{(i-j)\cdot U}, \quad \text{for } i \geq n_1 \tag{39}$$

which is a generalization of (9). Similarly, the number of channels in group j can be determined from solving the client access bandwidth and client buffer constraints:

$$\sum_{i=j+1}^{n_1+n_{2,0}+\ldots+n_{2,j}-1} B_i \leq C \text{ and } \sum_{i=j+1}^{n_1+n_{2,0}+\ldots+n_{2,j}} B_i > C \tag{40}$$

8.2 Playback Continuity

Given the dynamic nature of the streaming algorithm, the natural question is whether the system can guarantee that the client can receive video data in time to sustain continuous playback. To answer this question, we consider an arbitrary client who entered the system at time t_0.

First we consider the playback continuity for video segments received from the Type-I channels. Let c_i be the playback time for video segment L_i. Now since the playback duration is the same (U seconds) for all video segments, c_i is given by

$$c_i = t_0 + (m+i)\cdot U, \; i = 0,1,\ldots,n_1 - 1 \tag{41}$$

Observing that the client begins receiving data simultaneously from all Type-I channels from time t_0, it will receive the whole video segment L_i by the time s_i given by

$$\begin{aligned} s_i &= t_0 + \frac{\int_{iU}^{(i+1)U} b(t)dt}{\int_{iU}^{(i+1)U} b(t)dt/(m+i)\cdot U}, \quad i = 0,1,\ldots,n_1 - 1 \\ &= t_0 + (m+i)\cdot U, \quad \because \frac{L}{N} = U \end{aligned} \tag{42}$$

which is exactly the same as the playback schedule c_i's and thus playback continuity is guaranteed for all video segments received through Type-I channels.

Next we consider video segments received through Type-II channels. We consider an arbitrary Type-II channel i in group j. As the client begins receiving all channels in group j at the time

$$t_0 + (m + j) \cdot U \tag{43}$$

and it takes $\int_{iU}^{(i+1)U} b(t)dt / B_i$ seconds to receive video segment L_i, we can then compute the time s_i at which video segment L_i is ready for playback from

$$s_i = t_0 + (m + j) \cdot U + \frac{\int_{iU}^{(i+1)U} b(t)dt}{B_i} \tag{44}$$

Substituting B_i from (39) into (44) we obtain

$$\begin{aligned} s_i &= t_0 + (m + j) \cdot U + (i - j) \cdot U \\ &= t_0 + (m + i) \cdot U \equiv c_i \end{aligned} \tag{45}$$

which again equals to the playback schedule and thus playback continuity for video segments received through Type-II channels is also guaranteed.

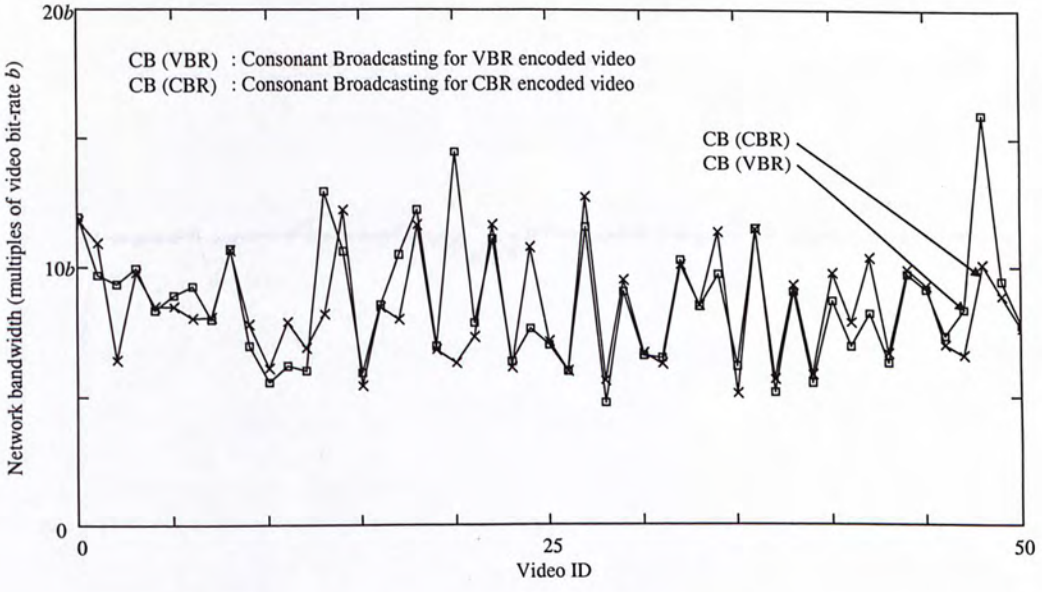

Figure 14. Network bandwidth versus video ID

8.3 Performance Evaluation

The previous results are all computed assuming CBR videos. To investigate the performance in streaming VBR videos, we extracted 300 video bit-rate traces from VBR-encoded DVDs to compute performance results for streaming VBR videos using DCB. These 300 videos have length ranging from 91 to 13,557 seconds, with average bit-rate ranging from 3.77 to 9.89 Mbps, and with peak rate as high as 65.86 Mbps.

We measure the video bit-rate traces by measuring the I/O activities when playing back the DVDs using a hardware MPEG2 decoder. Thus the traces not only account for the inherent bit-rate variations due to the encoding algorithm, but also account for the I/O behavior of the MPEG2 decoder. This also explains why the peak rate can exceed the DVD specification.

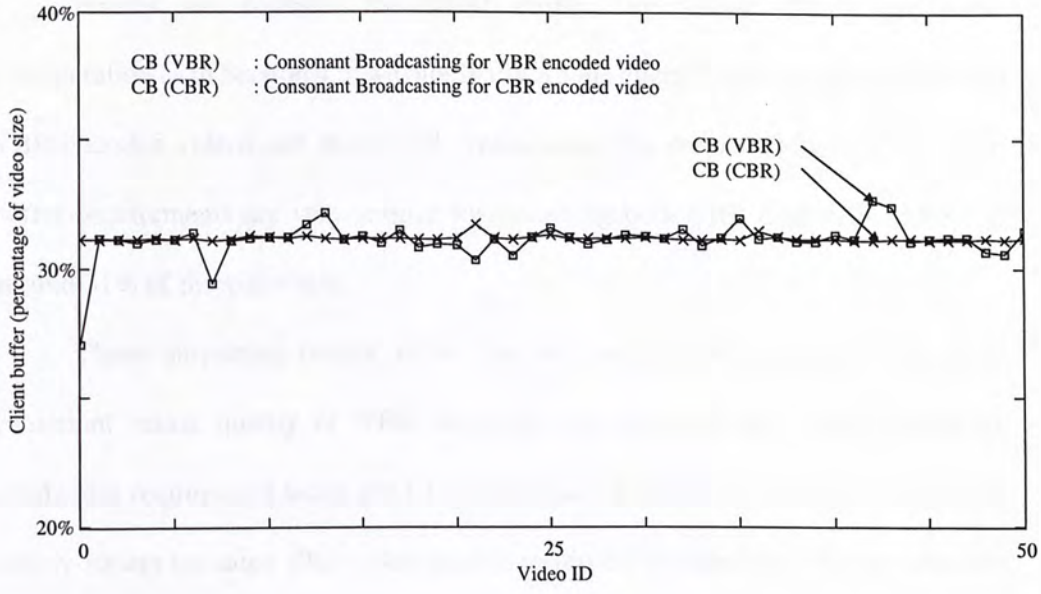

Figure 15. Client Buffer versus video ID

To our surprise, in 163 out of the 300 videos, CB in fact consumes *less* network bandwidth than streaming an equivalent CBR video of the same average bit-rate. The mean network bandwidth for streaming the 300 VBR and the equivalent CBR videos are 7.9*b* and 8.8*b* respectively. In Fig. 14 we compare the network bandwidth requirement of 50 VBR videos and their CBR equivalents. It is clear that while there are differences in the network bandwidth requirement, the differences are relatively small.

These results are in sharp contrast to established results, where streaming VBR videos often require substantially more network bandwidth to accommodate the bit-rate variations such that continuous playback can be guaranteed. The intuitive explanation is that in CB video segments are repeatedly multicast. Thus although the bit-rate of the video stream varies, the aggregate bit-rate of all video segment transmissions is constant, and thus evening out the temporal bit-rate variations.

Finally we consider the client buffer requirement. Using the same configuration as in Section 4.3, we plot in Fig. 15 the client buffer requirements for 50 VBR-encoded videos and their CBR equivalents. The results show that the client buffer requirements are very similar for streaming both CBR and VBR videos, at around 31% of the video size.

These surprising results show that we can take advantage of the more consistent visual quality of VBR encoding *and* simultaneously reduce network bandwidth requirement using the CB architecture. Moreover, service providers can simply stream the same VBR video sources originally encoded for DVD to eliminate the cost of re-encoding the video for network streaming.

Chapter 9

CONCLUSIONS

In this study, we presented a new Consonant Broadcasting (CB) scheme for multicast video streaming applications. CB can be considered as a generalization of the Poly-harmonic Broadcasting scheme with client access bandwidth constraint incorporated. Our results showed that CB outperforms all existing periodic broadcasting schemes, especially when the client access bandwidth is less than the system network bandwidth. Moreover, we extended the Consonant Broadcasting scheme to a Grouped Consonant Broadcasting (GCB) scheme to address the issue of limited number of network multicast channels. With a slight tradeoff in performance, GCB can significantly reduce the number of network channels required and thus can be readily implemented in today's network infrastructure such as using IP multicast. Our implementation clearly showed the feasibility of GCB and also verified the theoretical performance predications. Further, we devise a dynamic version of CB that performs well at both heavy and light loads, and can support the streaming of both CBR and VBR-encoded videos using comparable resources respectively. With these properties, a service provider no longer needs to choose between closed-loop architectures and open-loop architectures, and reveals an exciting opportunity for service providers as VBR encoding can be employed not only to provide a more

consistent visual quality, but also with the potential to reduce the system resources required. With growing support for network multicast in the current and the next-generation Internet, our proposed scheme will serve as a good potential candidate for building the next generation of large-scale yet cost-effective video streaming services.

BIBLIOGRAPHY

[1] S. W. Carter and D. D. E. Long, "Improving Video-on-Demand Server Efficiency Through Stream Tapping," *Proceedings of the 6th International Conference on Computer Communication and Networks*, Sep 1997, pp. 200-7.

[2] K. A. Hua, Y. Cai, and S. Sheu, "Patching: A Multicast Technique for True Video-on-Demand Services," *Proceedings of the 6th International Conference on Multimedia,* Sep 1998, pp. 191-200.

[3] L. Gao and D. Towsley, "Supplying Instantaneous Video-on-Demand Services Using Controlled Multicast," *Proceedings of the IEEE International Conference on Multimedia Computing and Systems*, Jun 1999, pp. 117-121.

[4] E. G. Coffman, Jr., P. Jelenkovic, and P. Momcilovic, "The Dyadic Stream Merging Algorithm," *J. Algorithms*, 43(2002), pp. 120-37.

[5] D. L. Eager, M. K. Vernon, and J. Zahorjan, "Optimal and Efficient Merging Schedules for Video-on-Demand Servers," *Proceedings of the 7th ACM International. Multimedia Conference*, Orlando, FL, Nov 1999, pp. 199-202.

[6] D. L. Eager and M. K. Vernon, "Dynamic Skyscraper Broadcasts for Video-on-Demand," *Proceedings of the 4th International Workshop Multimedia Information Systems*, Sep 1998, pp. 18-32.

[7] Y. Cai, K. A. Hua, and K. Vu, "Optimizing Patching Performance," *Proceedings of ACM/SPIE Multimedia Computing and Networking*, Jan 1999, pp. 204-15.

[8] Y. Cai and K. A. Hua, "An Efficient Bandwidth-Sharing Technique for True Video on Demand Systems," *Proceedings of 7th ACM International Multimedia Conference*, Orlando, FL, Nov 1999, pp. 211-214.

[9] J. Y. B. Lee and C. H. Lee, "Design, Performance Analysis, and Implementation of a Super-Scalar Video-on-Demand System," *IEEE Transactions on Circuits and Systems for Video Technology*, vol.12(11), Nov 2002, pp. 983-97.

[10] S. Viswanathan and T. Imielinski, "Metropolitan Area Video-on-Demand Service Using Pyramid Broadcasting," *IEEE Multimedia Systems*, vol. 4, 1996, pp. 197-208.

[11] K. A. Hua and S. Sheu, "Skyscraper Broadcasting: A New Broadcasting Scheme for Metropolitan Video-on-Demand Systems," *Proceedings of the ACM SIG-COMM '97*, Cannes, France, Sep 1997, pp. 89-100.

[12] L. Gao, J. Kurose, and D. Towsley, "Efficient Schemes for Broadcasting Popular Videos," *Proceedings of the 8th International Workshop on Network and Operating Systems Support for Digital Audio and Video*, Cambridge, UK, Jul 1998.

[13] L. S. Juhn and L.M. Tseng, "Staircase Data Broadcasting and Receiving Scheme for Hot Video Service," *IEEE Transactions on Consumer Electronics*, vol43(4), Nov 1997, pp. 1110-7.

[14] L. S. Juhn and L. M. Tseng, "Harmonic Broadcasting for Video-on-Demand Service," *IEEE Transactions on Broadcasting*, vol.43(3), Sep 1997, pp. 268-71.

[15] J. F. Paris, S. W. Carter, and D. D. E. Long, "A Low Bandwidth Broadcasting Protocol for Video on Demand," *Proceedings of the 7th International Conference on Computer Communications and Networks*, Lafayette, LA, USA, Oct 1998, pp. 690-7.

[16] J. F. Paris, S. W. Carter, and D. D. E. Long, "A Hybrid Broadcasting Protocol for Video on Demand," *Proceedings of the 1999 Multimedia Computing and Networking Conference*, San Jose, CA, Jan 1999, pp. 317-26.

[17] C. C. Aggarwal, J. L. Wolf, and P. S. Yu, "A Permutation-Based Pyramid Broadcasting Scheme for Video-on-Demand Systems," *Proceedings of the International Conference on Multimedia Computing and Systems*, Hiroshima, Japan, Jun 1996, pp. 118-26.

[18] J. F. Paris, S. W. Carter, and D. D. E. Long, "Efficient Broadcasting Protocols for Video on Demand," *Proceedings of the International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, Montreal, Canada, Jul 1998, pp. 127-32.

[19] L. S. Juhn and L. M. Tseng, "Fast Data Broadcasting and Receiving Scheme for Popular Video Service," *IEEE Transactions on Broadcasting*, vol.4(1), Mar 1998, pp. 100-5.

[20] J. F. Paris, S. W. Carter, and P. E. Mantey, "Zero-Delay Broadcasting Protocols for Video-on-Demand," *Proceedings of the 1999 ACM Multimedia Conference*, Orlando, FL, Nov 1999, pp. 189-97.

[21] A. Hu, "Video-on-Demand Broadcasting Protocols: A Comprehensive Study," *Proceedings of the IEEE Infocom 2001*, Anchorage, AK, Apr 2001, pp. 508-517.

[22] Y. Birk and R. Mondri, "Tailored Transmissions for Efficient Near-Video-on-Demand Service," *Proceedings of the IEEE International Conference on Multimedia Computing and Systems*, Florence, Italy, Jun 1999, pp. 226-231.

[23] W. S. Tan, N. Duong, and J. Princen, "A Comparison Study of Variable-bit-rate versus Fixed-bit-rate Video Transmission," *Proceedings of Australian Broadband Switching and Services Symposium*, 1991, pp.134-141.

[24] W.J. Liao and V. O. K. Li, "The Split and Merge (SAM) Protocol for Interactive Video-on-Demand Systems," *IEEE Multimedia*, vol.4(4), Oct-Dec 1997, pp. 51-62.

[25] W.J. Liao and V. O. K. Li, "The Split and Merge (SAM) Protocol for Interactive Video-on-Demand Systems," *Proceedings of IEEE INFOCOM*, Kobe, Japan, April 1997.