DOKUZ EYLUL UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

3G WIRELESS MULTICASTING SERVICE DESCRIPTION, DISCOVERY AND TRANSPORT

by
Zeki YETGİN

April, 2008
IZMIR

3G WIRELESS MULTICASTING SERVICE DESCRIPTION, DISCOVERY AND TRANSPORT

A Thesis Submitted to the
Graduate School of Natural and Applied Sciences of Dokuz Eylül University
In Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy in Computer Engineering

by
Zeki YETGİN

April, 2008
IZMIR

Ph.D. THESIS EXAMINATION RESULT FORM

We have read the thesis entitled **"3G WIRELESS MULTICASTING SERVICE DESCRIPTION, DISCOVERY AND TRANSPORT"** completed by **Zeki YETGİN** under supervision of **Prof. Dr. Tatyana YAKHNO** and we certify that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Doctor of Philosophy.

Prof. Dr. Tatyana YAKHNO

Supervisor

Assoc. Prof. Dr. Yalçın ÇEBİ

Committee Member

Asst. Prof. Dr. Zafer DİCLE

Committee Member

Prof. Dr. Alp KUT

Jury Member

Asst. Prof. Dr. Öznur ÖZKASAP

Jury Member

Prof.Dr. Cahit HELVACI
Director
Graduate School of Natural and Applied Sciences

ACKNOWLEDGMENTS

I would like to thank Gamze SEÇKİN for her supervision during this work. She has not only encouraged my research but also has been a good role model with her personality.

I would like to thank my committee members Yalçin ÇEBİ, Zafer DİCLE for their support throughout the work and Tatyana YAKHNO for her supervision at the last stage of the work. My special thanks to Alp KUT who encouraged me for this topic.

I would like to thank TUBITAK and Vidiator Technology (US) for the financial support for this thesis.

I dedicate this thesis to my mother Ümmü who encouraged me to be my best.

Zeki YETGİN

3G WIRELESS MULTICASTING
SERVICE DESCRIPTION, DISCOVERY AND TRANSPORT

ABSTRACT

Wireless Multicasting is a technology that enables data and multimedia services to be delivered from a single source to a group of mobile receivers particularly for the actors in the broadcasting and telecommunication world. Although multicasting has been extensively researched in the past, the wired IP Multicast model has not picked up due to various limitations. The new generation wireless counterpart of this technology is receiving tremendous interest from all over the world.

In this work, first we have provided a survey of recent technological improvements for wireless multicasting in both cellular and broadcast world. Then, one of the 3G wireless multicasting architecture, 3GPP's MBMS (Multimedia Broadcast Multicast Services) in UMTS (Universal Mobile Telecommunication System) networks, is investigated with a focus on reliable download mechanism. We have provided an end to end download prototype for MBMS. Our prototype, called MBMS legacy download, also covers an implementation of a Service Discovery Architecture. As a unique contribution the thesis provides the gain of using progressive download instead of legacy download and proposes ways to increase the gain for streamable multimedia files for MBMS. With progressive download, downloadable media can be streamed earlier after some waiting time, while the downloading still continues in the background. First we provide optimizations of the parameters for an efficient MBMS legacy download. Then based on these optimizations, we provide experimental analyses to show the gain in using progressive download in MBMS. Finally in order to further increase the progressive download performance, we apply our application layer interleaving strategy to our MBMS download systems and give a performance comparison of the legacy, interleaved and progressive download delivery. This work has has been fully funded by TUBITAK and Vidiator Technology US under the project EEEAG 104E163.

Keywords: 3G, MBMS, Interleaving, Progressive Download.

ÜÇÜNCÜ NESİL KABLOSUZ ÇOKLU DAĞITIM SERVİS TANIMI, KEŞFİ VE İLETİMİ

ÖZ

Kablosuz çoklu dağıtım tek bir kaynaktan, hareketli bir grup alıcıya, veri ve çoklu ortam dağıtımını özellikle telekomünikasyon ve yayımsal dünya aktörleri için mümkün kılan bir teknolojidir. Çoklu dağıtım geçmişte kapsamlı bir şekilde çalışılmasına rağmen, kablolu IP çoklu dağıtım modeli birçok kısıtlamalardan dolayı başarılı olamamıştır. Bu teknolojinin yeni nesil kablosuz sürümü dünyanın her yerinden olağanüstü bir ilgi almıştır.

Bu tezde ilk olarak hücresel ve yayımsal dünyada, kablosuz çoklu dağıtımın son teknolojik gelişmelerini ortaya çıkardık. Sonra, UMTS (Universal Mobile Telecommunication System) ağlarda 3. nesil kablosuz çoklu dağıtım mimarilerinden biri olan 3GPP'nin MBMS (Multimedia Broadcast Multicast Services) teknolojisini güvenilir yükleme üzerinde durarak inceledik. MBMS'in uçtan uca yükleme prototipini geliştirdik. Bu prototipimizi MBMS kalıt yükleme olarak isimlendirdik. Prototipimize aynı zamanda servis keşif mimarisinin bir uygulamasını da ekledik. Prototipi daha da geliştirerek aşağıdaki yenilikleri tezde sunduk. Tez MBMS'te duraksız çoklu ortam için, kalıt yükleme yerine gelişimsel yükleme kullanımının getirdiği kazancı sunar ve bu kazancı attırmak için yeni metotlar önerir. Gelişimsel yükleme ile yükleme işlemi arka planda devam ederken, belli bir bekleme zamanından sonra yüklenebilir çoklu ortam dosyaları duraksız olarak oynatılabilir. Önce verimli bir MBMS kalıt yükleme için parametrelerin en iyilemelerini gösterdik. Sonra bu en iyilemelerin üstüne, MBMS gelişimsel yükleme kullanılarak elde edilen kazancı göstermek için deneysel analizler sunduk. Son olarak gelişimsel yükleme verimliliğini daha da artırmak için, uygulama katmanında "interleaving" mekanizması kullandık. Kalıt Yükleme ile Gelişimsel ve "Interleaved" Yükleme arasında verimlilik kıyaslaması yaptık. Bu çalışma TUBITAK ve Vidiator Technology US firması tarafından EEEAG 104E163 proje numarası altında desteklenmiştir.

Keywords: 3G, MBMS, Serpiştirim, Gelişimsel Yükleme.

CONTENTS

Page

CHAPTER ONE

INTRODUCTION

1.1 Introduction

The recent development in multimedia applications with the parallel progress in transport technologies has brought real-time or non-real time multimedia distribution in the form of multicasting or broadcasting for both wired and wireless environments. Multimedia distribution is rapidly evolving with effective multimedia compression techniques and higher speed wireless networks. Hence mobile services are getting better with the decreasing delivery cost day by day. Such mobile services include streaming, downloading and progressive downloading (BenQ Mobile, 2006) services for on-demand video, mobile TV, short clips for news, football results, software updates and more. IP multicasting that refers to IP layer wired multicasting case has not succeeded due to many limitations. The new generation wireless counterpart of this technology is receiving tremendous interest from all over the world. The term "Broadcast" refers to the ability to deliver content to all users. Known examples are radio and TV services, which are broadcasted over the air, such as terrestrial or via satellite, or over cable networks. Multicasting, on the other hand, refers to services that are solely delivered to users who have joined a particular multicast group. Both multicasting and broadcasting are synonyms regarding to communication in that they deliver data over point-to-multipoint communication where data packets are transmitted from a single source to multiple destinations.

Today many mobile operators launched streaming type services such as mobile TV services, which allow mobile users to watch TV on their mobile terminals or downloading type services such as MMS. In Europe, a number of operators have launched sports information services that push short video clips to the mobile terminals. Vodafone (Germany, the Netherlands), TIM (Italy and Greece), Three (Italy and Sweden) and Sprint (US) have all launched mobile TV services and continue to do so in

different countries. Multimedia services are still offered to consumers over point-to-point wireless connections. Large scale media distribution makes this point-to-point service delivery inefficient especially for wireless networks. Furthermore, the cost of point-to-point services is expensive. Although the technology already realized the service delivery over one-to-many multicast channels, with sufficient quality of service, consumer market is not deployed yet due to it's still in transition from 2.5 G to 3G. Several technologies that provide multicasting for 3G wireless networks are 3GPP MBMS (Third Generation Partnership Project Multimedia Broadcast and Multicast System; 3GPP TSG 26.346, 2007), 3GPP2 BCMCS (Third Generation Partnership Project 2 Broadcast and Multicast System; 3GPP2 BCMCS, 2005), DVB-H (Digital Video Broadcast for Handhelds), and MediaFLO among others (Mobile TV WG, 2006). Beside scability problem, each service type has its own problems. While real time services expose delivery of packets in a timely manner, downloading type of services requires reliable delivery of content, where the same content is sent reliably to a large number of users.

Recently, 3GGP and 3GGP2 began addressing broadcasts / multicast services in GSM/WCDMA and CDMA2000 respectively. 3GPP is currently introducing support for IP multicasting services into the UMTS architecture namely the multimedia broadcast and multicast service (MBMS). Using these standards, multimedia services such as audio, video and TV-like services as well as large software updates could be provided to thousands of users simultaneously in a point-to-multipoint manner. In 3GPP2 the same work item is called Broadcast and Multicast Service (3GPP2 BCMCS, 2005). They have much common functionality where Open Mobile Alliance Broadcasting (OMA BCAST) is working around. OMA BCAST is working on the specification of broadcast/multicast related service-layer functionalities that can be applied to mobile and non-mobile digital broadcast networks. For instance, OMA BCAST addresses content protection, service and program guides, and transmission scheduling.

Although 3G mobile networks are much more powerful than that of existing traditional networks, they have still limitations in the transmission of larger files or

streams to a large number of users. Some broadcast technology such as DVB-H come up with more powerful multicast streaming or multicast downloading. Trial tests of DVB-H platform have been already started at some countries. The main limitation in this type of broadcast networks is that they have only unidirectional communication architecture with no back channel support. Because of this problem, DVB convergence of the broadcast and mobile services group (ETSI TS, 2007) has recently begun specifying the protocols and the codecs above IP so that a new work item to converge both networks into a more powerful hybrid network with back channel support is started. However more serious criticism of deployment of DVB-H in mass market is that network requirements and related deployment cost for providing coverage comparable to that of mobile networks. In most countries virtually all suitable DVB-H spectrum is being used by analog or digital TV services. Even if the spectrum in this band were made available for DVB-H, in many countries this spectrum is assigned to TV services only. It means it cannot be used for other types of IP datacast service. Because of such limitations in broadcast world primarily in DVB-H, we believe that 3GPP MBMS will take off earlier than DVB-H.

Our research focuses on reliable download including progressive download in MBMS. The downloading in MBMS is based on FLUTE (File Delivery over Unidirectional Transport) protocol (Paila T. & Others, 2004). FLUTE is a protocol used to deliver files, particularly over unidirectional systems from one sender to many receivers. Since FLUTE uses an unreliable transport protocol, an application layer FEC is coupled with FLUTE to recover from packet losses, making a reliable service. The most popular FEC codes are Raptor codes, as initially introduced by Shokrollahi A. (2003) and Reed Solomon codes (Rizzo L., 1998). For the MBMS system, Raptor codes have been selected due to their high performance, relative to others. Consequently, 3GPP has mandated the support of Raptor codes (3GPP TSG 26.346, 2007) for their terminals that use the MBMS service.

There are two ways considered to play the media in MBMS download delivery. First is to wait for the download to complete and then play the media. Second is to wait for some initial startup time and then play the media while downloading it. First one is

called Legacy Download while the latter is called Progressive Download. Hence progressive download is important in that it reduces the waiting time substantially, which will be demonstrated in this work. There are two important parameters for MBMS download; transmission cost and waiting time. Minimum waiting time, called downloading time optimization, and minimum FEC cost, called Transmission cost optimization, with reliability requirement are the targets for an acceptable service. Transmission cost optimization minimizes the FEC overhead required for a reliable download. However downloading time optimization minimizes the initial startup delay as well as the download duration. In this thesis we provide three contributions to decrease the waiting time as well as FEC overheads for a reliable and efficient MBMS download service: i) Progressive Download Approach ii) Downloading Optimizations and iii) Application Layer Interleaving.

1.2 Progressive Download Approach

Today MBMS has two delivery modes which correspond to streaming and downloading services over point to multipoint bearers. Downloading mode can be also referred to as “download and play” mode when the content includes continuous media such as audio, video and presentations. Downloading mode consumes less radio resources despite its longer time of service consumption with respect to streaming. In this thesis we focus on progressive download which combines the advantages of streaming and download in terms of time and bandwidth. Progressive download can be referred to as “play while download”. With progressive download, downloadable content can be streamed sooner, after some initial startup delay, also called waiting time in the thesis, while the downloading still continues in the background. By its nature, MBMS progressive download is a software overlay on top MBSM download mode.

We believe that MBMS should enable the use of progressive download for three reasons; the first reason is while the media content is being downloaded in the background the user is waiting for the download to complete. Instead of waiting for the download to complete the user experience can be enriched if media play started earlier. The second reason is the optimization of radio resources. Download mode uses less bit rate compared to streaming and progressive download provides the benefits of

download in terms of bit rate utilization. The download and play mode allows download of media contents at much lower bit rates than the streaming bit rates. The third reason, adding progressive download capability to any multicast delivery system will not require many changes in the infrastructure or software components since progressive download will utilize the existing download delivery mode.

MBMS Progressive download is still an open issue in 3GPP and related discussions are postponed to future MBMS releases. One of the issues is the gain to be obtained from having progressive download. Our aim in the thesis is to show that using progressive download compared to legacy download we have satisfactory gain with respect to waiting time. We target progressive download of small 3gp multimedia files, instead of big files, which require long waiting time that is not acceptable for user experience. We considered constant bit rate encoding since variable bit rate makes waiting time prediction difficult in MBMS progressive download and requires more capability at receiver side. We consider enhanced AAC+ and H.264 AVC (3GPP TSG 26.346, 2007; ITU-T Recommendation H.264, 2005) coding with total 128 kbps media play rate.

1.3 Downloading Optimizations

During the MBMS FLUTE transport, a file is partitioned into source blocks (SBs), each of which is encoded in FEC layer and then carried as a set of symbols in Multicast IP datagrams over the IP backbone to the destination network. IP datagrams are mapped to SDU (Service Data Unit) blocks and each SDU packet is mapped to RLC (Radio Link Layer) blocks across the UMTS core network. Each RLC block is carried as PDU (Protocol Data Unit) packets to receivers in the Radio Access Network. This partitioning and mapping process requires allocating proper block sizes wherever they are sent throughout the route from sender to a destined multicast area. Furthermore, the sizing considerations in the IP network (IP packet size), core network (SDU and PDU size) and FEC Layers (SB size) all affect the cost of the download reliability and hence there should be a combination of the size choices that lead to a target-optimized result, such as the reliability with minimum FEC overhead and minimum waiting time with reliability.

1.4 Interleaving Approach

Another technique to increase efficiency of the MBMS download as well as progressive download delivery is the use of application layer interleaving. Interleaving can be used in digital communications systems to enhance the error correcting capabilities of FEC mechanism. Interleaving changes the transmission order of symbols in an attempt to minimize the loss of symbols belonging to the same source block. In practice, packet losses occur as error bursts. One lost packet may cause one or more consecutive packets to be lost. The interleaving mechanism can substantially reduce the negative effects of packet losses that belong to the same FEC block, thus providing an increase in download efficiency. Interleaving transmission strategy is important in that if not properly selected it may cause randomization of source blocks which prevents progressive download.

1.5 The Problem Definition

Discussions related to progressive download in MBMS are postponed to future MBMS releases. One of the important reason, among many, is there is no clear work that show our gains from having progressive download in MBMS. The main problem addressed in this thesis is to show the possible gains from having progressive download in MBMS and to show our contributions to improve the gains further by our optimizations and our application layer interleaving strategy. We studied four solutions to reduce the waiting time and FEC overhead for reliable MBMS downloads in the thesis. The reductions are identified as gains from MBMS Download optimizations, gains from Application Layer Interleaving, gains from Progressive Download and gains from the Interleaved-Progressive Download in MBMS. Gain is described in terms of waiting time and FEC overhead for full reliability. So optimizations are based on waiting time and as well as on transmission cost. To the best of our knowledge these topics have not been studied in the literature and our work is providing a leading path for future research.

1.6 Main Contributions

Four contributions are provided to decrease the waiting time of the MBMS download service in the thesis. First the work provides optimizations for efficient and reliable download services for 3GPP's MBMS that also supports progressive downloading. Since MBMS download mechanism uses unreliable multicast, Forward Error Correction (FEC) is used to recover from packet losses. Reed Solomon FEC coding is used in our work as underlying protection method. Two optimizations; downloading time optimization and FEC overhead optimization are introduced to investigate an efficient and reliable MBMS download service. Experimental analyses are provided to show the gain in downloading time as well as the gain in FEC overhead from our optimizations. Trading between the two optimizations is investigated under MBMS network conditions. Instead of considering only FEC cost optimization as legacy MBMS downloads do, downloading time optimization is recommended for efficient MBMS download services.

Second, based on the optimizations, Reliable Download analyses with and without interleaving in MBMS is studied to provide the gain in FEC overhead as well as the gain in download duration from the application layer interleaving approach in MBMS. Then a performance comparison of the legacy and the interleaved download delivery is provided.

Third, based on the optimizations, we provide the progressive download approach to provide the gain in download duration from the progressive download instead of legacy download for streamable media files for MBMS. For this, a legacy MBMS download system optimized for waiting time to play the media is provided first. Then a progressive MBMS download system is provided to compare the gain in waiting time.

Finally, we combined the approaches in order to further increase the system performance, hence we applied our application layer interleaving strategy to our MBMS Progressive download system, so called Interleaved Progressive Download, and gave a performance comparison of the legacy and the interleaved progressive download delivery.

The results encourage the usage of progressive download instead of legacy download mechanism where the data file is streamable, for improved user experience for 3G wireless multicasting systems. The results of this study will also provide guidelines to designers to fine-tune MBMS download service parameters for an efficient and reliable download service.

1.7 Scope of the Thesis

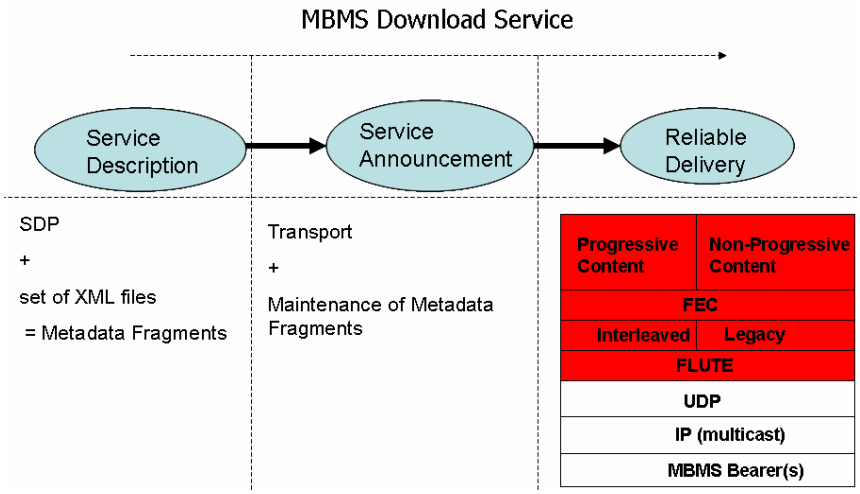


Figure 1.1 Scope of the PhD work.

The scope of the work aimed in the thesis is shown in Figure 1.1. We have provided an end to end download prototype for MBMS. Our prototype, called MBMS legacy download, also covers an implementation of a Service Discovery Architecture.

Closely related to service descriptions is their announcement (push) to subscribers. There must be a way for subscribers to learn service descriptions so that they can join and start playing the multicast media or start downloading the multicast data. Service discovery is a mechanism for subscribers to get service descriptions before the start of the service. MBMS does not restrict the delivery method of service descriptions. It can be via MBMS multicast download sessions such as broadcast or multicast over FLUTE

protocol or any other means such as cell-broadcast, http, even via emails. Delivering the service descriptions for a session is vital but not enough alone for a successful deployment of service discovery/announcement mechanism onto a mass. During the course of time, service descriptions may change, expire or corrupt, so suitable metadata structures are needed to maintain service discovery/announcement process. Currently MBMS delivers service descriptions as a set of metadata fragments each of which coupled with a metadata envelope that has a time-validity and other properties to maintain the actual metadata fragment. Our prototype covers generation of these metadata fragments, their maintanence and their transport to subsribers.

The protocol stack for our MBMS download systems are shown in Figure 1.1. The layers above the FLUTE protocol that inlude interleaving, FEC and progressive content are considered in the application layer. With the interleaving and progressive downloading methods the IP / UDP / FLUTE packet may contain interterleaved or progressive content or the interleaved progressive content where combination of both methods is applied.

1.8 Organization of the Thesis

The thesis is organized as follows; chapter two gives a brief overview of MBMS download delivery and its main components; FEC and File Delivery over Unidirectional Transport (FLUTE) Protocol. It discusses gains from MBMS progressive download and gains from interleaving with providing existing works. In the third chapter FLUTE protocol is investigated in detail while FLUTE usage in MBMS is provided in chapter four. The system models that our work is based including an analytical model to formulize the problem are provided in chapter five. In chapter six we show the experimental results of four MBMS download system proposed in this thesis: (i) Legacy downloads with waiting time and transmission cost optimizations, (ii) Progressive download, (iii) Interleaved download, and (iv) Interleaved Progressive. Then we compared proposed systems with legacy download and give a conclusion and future directions in the final chapter. Our ptototype and its enhancements for our solutions are provided in Appendix A while Appendix B shows figures from the intermediate works during the experimental analyses.

CHAPTER TWO

BACKGROUND AND RELATED WORK

2.1 Background

Third Generation (3G) is the generic name for next-generation mobile networks such as the Universal Telecommunications System (UMTS) or IMT-2000; 3G wireless networks offer faster data transfer rates than current networks. As indicated in Table 2.1, the first generation of wireless (1G) networks is analog cellular. The second generation (2G) networks are digital cellular, featuring integrated voice and data communications. 2.5G networks offer incremental speed increases. 3G networks offer dramatically improved data transfer rates, enabling new wireless applications such as streaming media. Services and their speeds in each phase of this evolution are given in Table 2.1.

Table 2.1 Service types and their speeds in 3G (CNET Asia).

	1G	2G	2.5G	3G	3.5G	4G and beyond
Technology	AMPS	GSM CDMA	GPRS 1xRTT EDGE	UMTS 1xEV-DO	HSDPA (upgrade for UMTS) 1xEV-DV	WiMax*
Speeds	n/a	Less than 20Kbps	30Kbps to 90Kbps	144Kbps to 2Mbps	384Kbps to 14.4Mbps	100Mbps to 1Gbps
Features	Analog (voice only)	Voice; SMS; conference calls; caller ID; push to talk	MMS; images; Web browsing; short audio/video clips; games, applications, and ring tone downloads	Full-motion video; streaming music; 3D gaming; faster Web browsing	On-demand video; videoconferencing	High-quality streaming video; high-quality videoconferencing; Voice-over-IP telephony

At the time this thesis is being written, the deployment of 4G, which is a combination of broadband wireless and cellular wireless, has just started. It is forseen that the near future will focus on 4G technologies and challenges. The 1990s marked

the arrival of two digital networks: Code Division Multiple Access (CDMA), popular in the United States and a few other countries; and GSM, the dominant technology in Europe. These 2G networks replaced the analog communication with the digital one. By further upgrading existing components in 2G network and by adding packet-switching features GPRS technology progressed.

The Global System for Mobile Communications (GSM) is a wireless network system is used at three different frequencies: GSM900 and GSM1800 are used in Europe, Asia, and Australia, while GSM1900 is deployed in North America and other parts of the world.

GPRS is an enhancement to existing GSM networks that introduces packet data transmission, enabling "always on" mobility. This means that users can choose to be permanently logged on to e-mail, Internet access and other services, but do not have to pay for these services unless sending or receiving information. It is a new non-voice value added service that allows information to be sent and received across a mobile telephone network. It supplements today's Circuit Switched Data and Short Message Service. GPRS allows customers to maintain a data session while answering a phone call, which is a unique and exclusive feature to GSM technologies. GPRS also provides an "always-on" data connection where users don't have to log on each time they want data access, and the packet architecture means they only pay for the data itself rather than for the airtime used to establish a connection and download data.

EDGE (Enhanced data rates for GSM evolution) upgrades to GPRS systems that require new base stations and claim to increase bandwidth to 384 kbps. HSCSD (High-speed circuit-switched data) software upgrade for cellular networks that gives each subscriber 56K data.

CDMA is a cellular technology widely used in the world. There are currently three CDMA standards: CDMA One, CDMA2000 and W-CDMA. CDMA One and CDMA2000 are widely used in North America while W-CDMA is used in Europe,

Asia and Australia. CDMA technology uses UHF 800Mhz-1.9Ghz frequencies and bandwidth ranges from 115Kbs to 2Mbps.

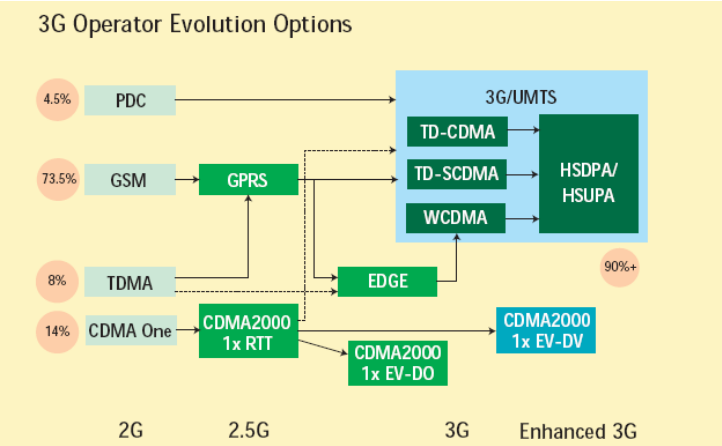


Figure 2.1 3G evolution paths (UMTS Forum, 2005).

CDMA One also known as IS-95, is a 2nd generation wireless technology and supports speeds from 14.4Kbps to 115K bps. CDMA2000, also known as IS-136, is a 3rd generation wireless technology and supports speeds ranging from 144Kbps to 2Mbps. In general CDMA technology spreads voice calls across several wireless spectrums, making for more reliable connections that are much harder for hackers to intercept. More importantly, CDMA and GSM networks are also capable of sending a sliver of data along with voice signals, making possible for such features as text messaging (SMS), caller ID, and conference calling.

Figure 2.1 shows different evolution paths for 3G systems. Wideband Code-Division Multiple Access (W-CDMA), also known as IMT-2000, is a 3rd generation wireless technology and supports speeds up to 384Kbps on a wide-area network, or 2Mbps locally. UMTS is a standard that will provide cellular users a consistent set of technologies no matter where they are located worldwide. UMTS utilizes W-CDMA technology. EDGE is the result of a joint effort between TDMA operators, vendors and carriers and the GSM Alliance. TDMA is used by Digital-American Mobile Phone Service (D-AMPS), GSM and Personal Digital Cellular (PDC). However, each of these systems implements TDMA in a somewhat different and incompatible way.

Crucially, 3G/UMTS has been specified as an integrated solution for mobile voice and data with wide area coverage, Universally standardized via the Third Generation Partnership Project (www.3gpp.org). Symmetry between uplink and downlink data rates when using paired (FDD) spectrum also means that 3G/UMTS is ideally suited for applications such as real-time video telephony – in contrast with other technologies such as ADSL where there is a pronounced asymmetry between uplink and downlink throughput rates. Ongoing technical work within 3GPP will see further increases in throughput speeds of the WCDMA Radio Access Network (RAN). High Speed Downlink Packet Access (HSDPA) and High Speed Uplink Packet Access (HSUPA) technologies are already standardized and are undergoing network trials with operators in the Far East and North America. Promising theoretical downlink speeds as high as 14.4 Mbps (and respectively 5.8 Mbps uplink), these technologies will play an instrumental role in positioning 3G/UMTS as a key enabler for true 'mobile broadband'. Offering data transmission speeds of the same order of magnitude as today's Ethernet based networks that are a ubiquitous feature of the fixed-line environment, 3G/UMTS will offer enterprise customers and consumers all the benefits of broadband connectivity whilst on the move (UMTS Forum, 2005, s.4).

Building on current investments in GSM/GPRS, 3G/UMTS offers mobile operators significant capacity and broadband capabilities to support greater numbers of voice and data customers –especially in urban centres – plus higher data rates at lower incremental cost than 2G. The choice of eight out of the world's ten biggest operators who have been awarded licenses to launch 3G services, UMTS represents the natural evolutionary route from 2G to 3G for more than 90% of the world's mobile users – spanning 1.2 billion GSM customers as well as subscribers to second generation TDMA and PDC networks. Taking use of radio spectrum in bands identified by the ITU for Third Generation IMT-2000 mobile services and subsequently licensed to operators, 3G/UMTS uses a 5 MHz channel carrier width to deliver significantly higher data rates and increased capacity compared with second generation networks. This 5 MHz channel carrier provides optimum use of radio resources, especially for operators who have been granted large, contiguous blocks of spectrum – typically ranging from

2x10 MHz up to 2x20 MHz – to reduce the cost of deploying 3G networks. This contrasts with the 1.25 MHz channel carrier width specified for the CDMA2000 system that was developed initially to serve North American mobile markets with more limited access to large, contiguous blocks of radio spectrum than operators in Western Europe. This means that 3G/UMTS offers greater cost efficiencies in terms of carrying network traffic than other mobile technologies, allowing operators to support larger numbers of simultaneous users and offer greater data speeds. (UMTS Forum, 2005, s.3)

2.2 Current State of Wireless Multicasting

Multicasting has been extensively researched in the past (Almeroth K.C, 2000; Obraczka K., 1998; Diot C. & Others, 2000). First introduced in 1988, IP multicasting has not been as successful as WWW, a technology of the same age. The two main reasons for the failure of IP multicasting are:

1. the lack of a well defined business model and services
2. the need for network intelligence

The need for network intelligence has shifted the research focus on multicast routing and transport protocols. This shift has provided significant source for academic research, but it did not impact the success of multicasting. With the lack of a well defined business model and services, wired Internet multicast simply did not take off as expected.

The wireless multicast (Varshley U., 1999) research began around 1994. The mobility of the user and the characteristics of the wireless channel made multicasting even more challenging. Varshley (2002) provides a review of the challenges of wireless multicasting, Figure 2.2. Varshley divides wireless multicasting architectures into two as infrastructure based and ad-hoc. Infrastructure-based wireless architectures have a base-station and a fixed topology but the user nodes are mobile. For ad-hoc wireless multicasting both the routers and the users are mobile, where in most cases the nodes have routing capabilities. The challenges for each model are different. Ad hoc wireless multicasting has a very important yet very limited application area such as military or a

disaster connectivity scenario. Hence, in this project we will concentrate on infrastructure based wireless multicasting.

Issue	Current "wired" multicast	Wireless and mobile multicast	Possible ways to support wireless and mobile multicast
Type of links	Symmetrical and fixed characteristics Broadcast links in LANs	Possibly asymmetrical and/or unidirectional links of varying performance and point-to-point links in cellular and PCS	Design of new protocols to handle route asymmetry and unidirectional links without reverse-path information (possible history and prediction-based schemes)
Bandwidth	Plentiful	Limited and variable amount	Protocols to adapt membership management and routing updates to the amount of bandwidth available and user mobility
Topology	Fixed	Fixed in infrastructure-based, dynamic in ad hoc networks	Protocols for both fixed and changing topology by "sensing" topological changes
Loss of packets	Infrequent (<1%)	Frequent and variable (1%–30% based on links)	Error control with possible retransmission from neighboring user(s)
Membership changes	Only when a user leaves or joins a group	Also when a user moves to another location	Protocols with reduced over-head for managing membership
Routing	Fixed routing structure throughout the multicast session	Routing structure subject to change due to user mobility	Protocols that could dynamically adapt the routing to current structure and available resources
Security Issues	Less complex due to fixed users and wired links	More complex due to wireless links and possible use of broadcasting	Encryption and security techniques in routing and membership management
Quality of service	Individual routes can use RSVP	Due to user mobility, RSVP may cause excessive overhead	Design of new protocols for "soft" QoS under varying link conditions and mobility
Reliability	Possible use of a transport-layer protocol (such as the Multicast File Transfer Protocol)	More complex due to wireless links and user mobility; possible unwanted interaction of protocols at transport and link layers	Design of new protocols that could allow one or more different retransmission schemes at one or more protocol layers

Figure 2.2 Challenges of wireless multicasting (Varshley U., 2002).

Infrastructure based multicasting means wireless (cellular) and mobile multicasting. In the work (Gossain H., 2002), the challenges of infrastructure based multicasting are studied further but it focuses mostly on the network layer issues of wireless multicasting and does not address the impact of these issues to higher level protocols and applications. Dutta (2003) addresses the impact of wireless multicasting on streaming applications. Dutta identifies the following areas as major challenges:

1. diverse wireless network support: different networks with different characteristics should be supported

2. intradomain mobility: the user should keep it's multicast group connectivity while moving from cell to cell
3. scalability: the application should support scalable multicast groups.
4. load balancing: the application should support load balancing depending on the number of users or location of content.
5. Quality of service: the user should keep it's QoS while moving from cell to cell.

Li X. & et.al (1999) provide a comprehensive overview of video multicasting challenges and it's possible solutions. Li reviews layering and rate adaptation and provides results for tests on MBone, the well-known IP multicast overlay network. However Li fails to provide any overview for the challenges specific to wireless multicasting.

In the research of Mukhtar R. G. (2003), although wireless multicasting is not being addressed directly, Mukthar reviews the challenges for wireless traffic management. Radio link layer characteristics are identified along with higher layer transport protocol issues.

Wan T. & Subramanian R. K. (2004) provide a very good comparison of traditional QoS metrics and wireless multicast QoS metrics. It also reviews application layer QoS adaptation techniques, where feedback based adaptation with predictive adaptation is compared. However, it does not address how different wireless networks can be accommodated with the proposed model and how reliability can be added.

In addition to the work presented in the literature review, there are two major groups for wireless multicasting work:

1. Standards based wireless multicasting work
    a. 3GPP MBMS (3GPP TSG 26.346, 2007; 3GPP TSG 26.946, 2007)
    b. 3GPP2 BMCS (Wang J. & Others, 2004)
    c. OMA BCAST (2004)
    d. DVB-H

2. Commercial (proprietary) architectures: MediaFlo (Qualcom), Bamboo Mediacast, Crown Castle (About CrownCastle), etc.

MBMS is a point-to-multipoint service in which data is transmitted from a single source entity to multiple recipients. Transmitting the same data to multiple recipients allows network resources to be shared. MBMS user services can be built on top of the MBMS bearer service.

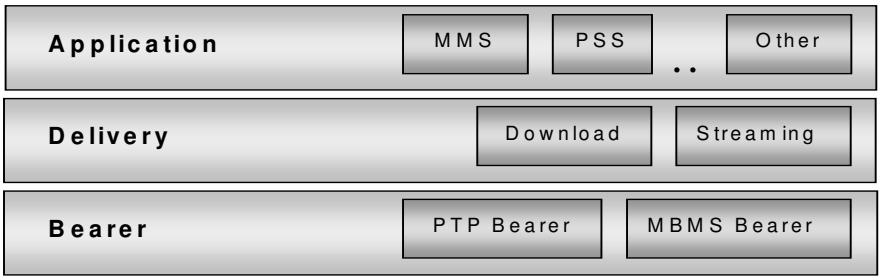


Figure 2.3 Functional layers for MBMS service delivery (3GPP TSG 26.346, 2007).

There are two delivery methods for the MBMS user services: download and streaming. Examples applications using the download delivery method are news and software upgrades. Delivery of live music is an example of an application using the streaming delivery method.

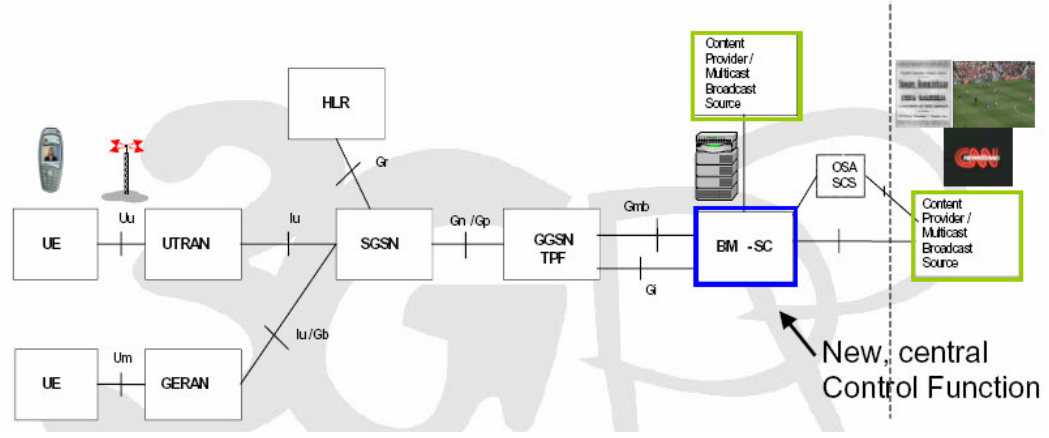


Figure 2.4 MBMS architecture as defined by 3GPP (3GPP TSG 26.346, 2007).

Functional layers for MBMS service delivery is shown in Figure 2.3 and the architecture diagram for MBMS is shown in Figure 2.4. Existing packet switched entities such as GGSN, SGSN, UTRAN/GERAN and UE are upgraded to provide MBMS bearer services. Additionally a new entity called BMSC (Broadcast Multicast Service Center) is placed as an entry point between content provider and operator network.

BMSC task is very critical in MBMS. It provides a set of functions for MBMS user services such as bearer control signalling, bearer establishment and bearer maintenance through Gmb interface, scheduling among MBMS sessions, delivery of IP datagrams through Gi interface to UEs with a specified QoS, authentication and authorization of UEs and 3rd party content providers, delivering of service descriptions to UEs, maintaining UE subscription information etc.

2.3 MBMS Download

This section gives brief overview of MBMS Download delivery and its main components FEC and FLUTE protocol. An example of the MBMS Download Service is given in Figure 2.5 where subscribers are announced previously for the upcoming service and its contents. When the service descriptions are available to the subscribers they may join the service. Joining to a service does not mean that the service starts soon. The service always starts at the time as it is previously announced to the subsribers. The service start and end times are provided in session descriptions (SDP) of the service. As shown in Figure 2.5 the MBMS download service occurs as unidirectional transport. So for the receiver, there is no way to feedback the lost packets during the download session but after the download. Associated delivery description components describe how to request the missing packets after the download session. Optionally the associated delivery procedure descriptions may provide for the server to collect statistical report from clients.

MBMS download delivery is based on FLUTE (Paila T. & Others, 2004) protocol used to deliver files particularly over unidirectional systems from one sender to many receivers. FLUTE is an IETF protocol based on Asynchronous Layered Coding (ALC)

protocol (Luby M., Gemmell J. & Others, 2002) that makes the FLUTE well scalable and hence preferable for unidirectional systems.

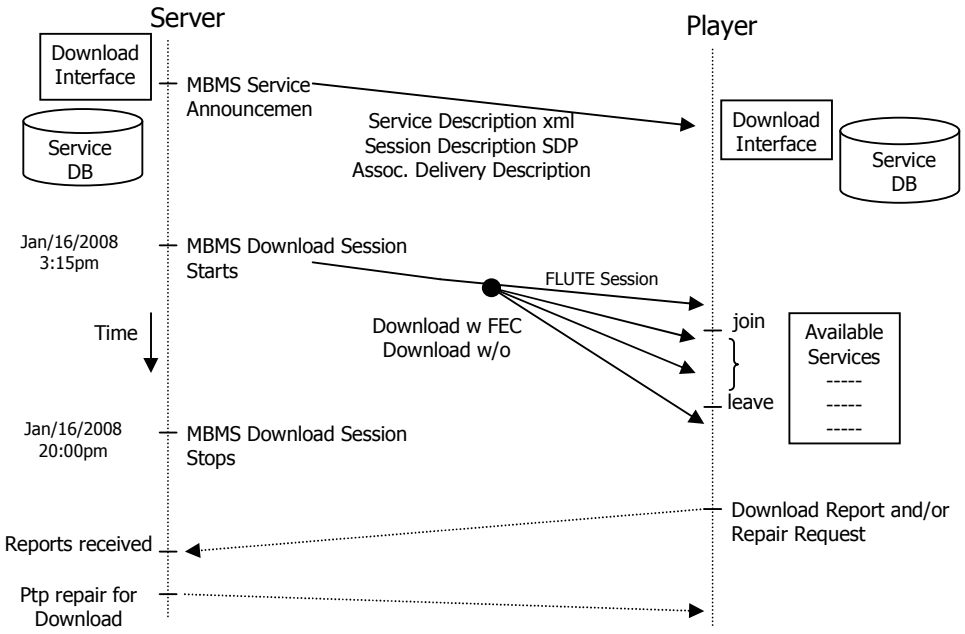


Figure 2.5 MBMS download service flow.

ALC is an instantiation of Layered Coding Transport (LCT) (Luby M., Gemmell J. & Others, 2002) for FLUTE. LCT manages how to transport an object identified by Transport Object Identifier (TOI) within a session identified by Transport Session Identifier (TSI). ALC inherits LCT with asynchronous and FEC coding selection as underlying coding technique. Finally the FLUTE protocol inherits the ALC and provides capabilities carried in-band or via FDT instances (File Delivery Table) to signal the properties of the file including FEC coding descriptions and map them to the ALC protocol.

With FLUTE, the only built-in reliability mechanism is provided by FEC mechanism. FEC provides reliable delivery of media content by appending repair symbols to the original data called source block prior to transmission across communication network. If some symbols are lost during transmission FEC allows

receiver to use repair symbols to recover the original source block without retransmission.

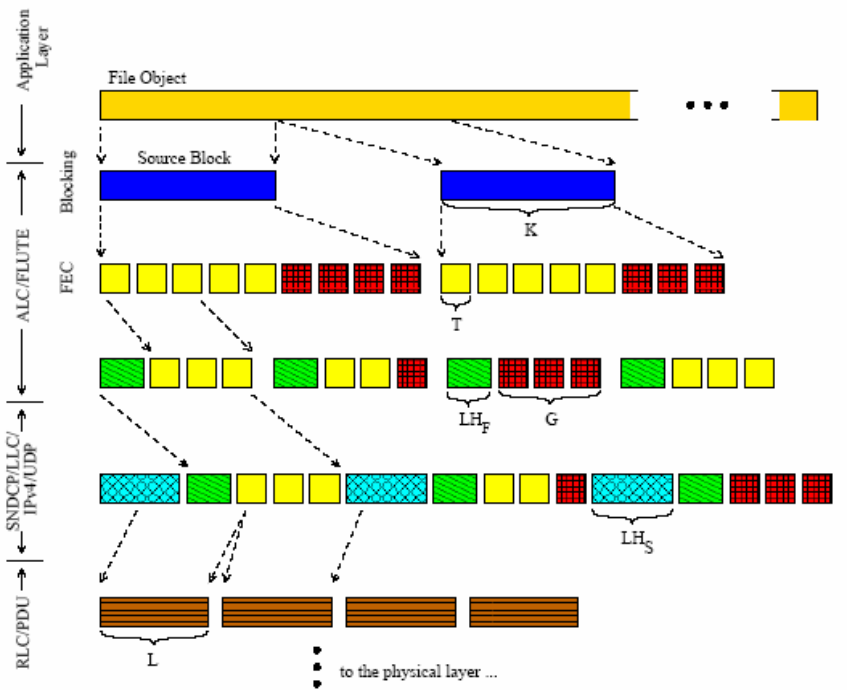


Figure 2.6 MBMS download protocol stack.

Figure 2.6 (Gasiba T. & Others, 2006) shows a detailed view of the protocol stack for MBMS download delivery. A file in application layer, called object in ALC terminology, is partitioned into source blocks (SB) each of which is further divided into *"k"* source symbols of size *"T"* each. Each SB is forwarded to the FEC/FLUTE layer, where Raptor FEC encoding is individually applied to each source block. The result is an encoding block (EB) of *"n"* encoding symbols, *"n - k"* of which is repair symbols. Each encoding symbol is identified by the couple: a source block number (SBN) and an encoding symbol identifier (ESI). A group of G consecutive encoding symbols that share the same SBN is appended to an ALC/FLUTE header (LH_F =16 bytes). The result is a FLUTE packet with payload P = G×T. The FLUTE header contains FEC payload ID that is the ESI and SBN of the first symbol, Transport Session Identifier (TSI), Transport Object Identifier (TOI), as specified by FLUTE protocol. User datagram protocol (UDP) over IP is used to distribute the FLUTE packets.

Further headers are appended to the original packet at the UDP, IP, Logical Link Layer (LLC)/Subnetwork Data Convergence Protocol (SNDCP) or Packet Data Convergence Protocol (PDCP). At the Radio Link layer (RLC), Protocol Data Units (PDU) are obtained and forwarded to the physical layer as usual, where the data is encoded with a convolutional code, is interleaved and transmitted in small bursts. Due to the lower layer interleaver, the RLC-PDUs can be assumed to be lost with link loss rates as high as 10% or even more. (Gasiba T. & Others, 2006).

2.3.1 Forward Error Correction

Some part of the original data may be lost during transmission. A FEC scheme allows receiver to use the additional redundant data to recover the original data without retransmission. FEC codes are divided into two sub categories:

1. Systematic FEC codes: An (n, k) systematic FEC block code preserves the k source symbols and appends (n –k) repair symbols.
2. Nonsystematic FEC Codes: An (n, k) non-systematic FEC block code creates n encoding symbols from k source symbols without necessarily preserving all the source symbols.

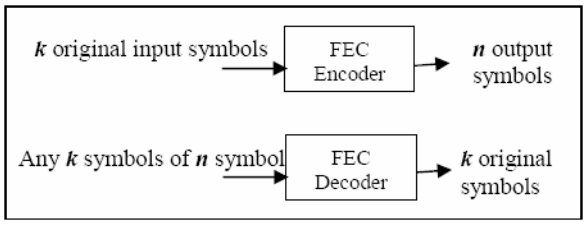


Figure 2.7 (n, k) FEC block code.

Figure 2.7 shows a FEC block code which is specified as an (n, k) code, for each "*k"* input symbols the encoder produces *"n"* output symbols. A source block is a fragment of the original object (media). Each source block contains several source symbols. A block of *"k"* source symbols constitute a source block. Decoding algorithms allow the recovery of the *"k"* source symbols from any set of the *"n"* received symbols. While

"k" is a number of source symbols, *"n"* is a number of encoding symbols, *"n – k"* is the number of repair symbols which are encoding symbols that are not source symbols.

2.3.1.1 Reed Solomon versus Raptor

Important factors of FEC mechanisms are their encoding/decoding efficiency and their time complexity. Algorithm time complexity particularly affects the processing ability of limited handsets. Raptor FEC scheme computational complexity is about *O(1)* time to generate an encoding symbol and *O(k)* time to decode a message of length *"k"*. Reed Solomon encoding algorithm computational complexity depends on the current source block length (k) and number of encoding symbols (n) generated for the relevant source block. These parameters are carried by the FEC Object Transmission Information (FEC OTI) to receiver side to execute the decoding algorithm.

Raptor provides linear encode/decode time (Luby M., 2005). Raptor is a fountain code, i.e. as many symbols as needed can be created unlike Reed Solomon, which has a block size of 255 symbols. Raptor decoding time is independent of packet loss patterns. However, Reed Solomon decoding time is loss dependent. Raptor is based on irregular low-density parity-check code (LDPC), since the LDPC codes allow data transmission close to the theoretical maximum (Luby M., 2005). Both Raptor and Reed Solomon codes are systematic so the original source symbols are sent intact from sender to receiver.

2.3.2 Interleaving Effect on FEC performance

With application layer interleaving, FEC performance increases greatly. This fact can be observed in the example in Figure 2.8. The cross sign indicated that the symbol is lost. Without interleaving all symbols belonging to source blocks will be sent sequentially in the order of source block number. That is, symbols of SB1 are sent first, symbols of SB2 next and so on. With interleaving, symbols in a block are sent in different times in a changing order of source block numbers. There can be many different ways of changing transmission order of symbols. One way is the

randomization of the transmission order of all symbols. In the example in Figure 2.8, different transmission strategy is used to show the effect of interleaving.

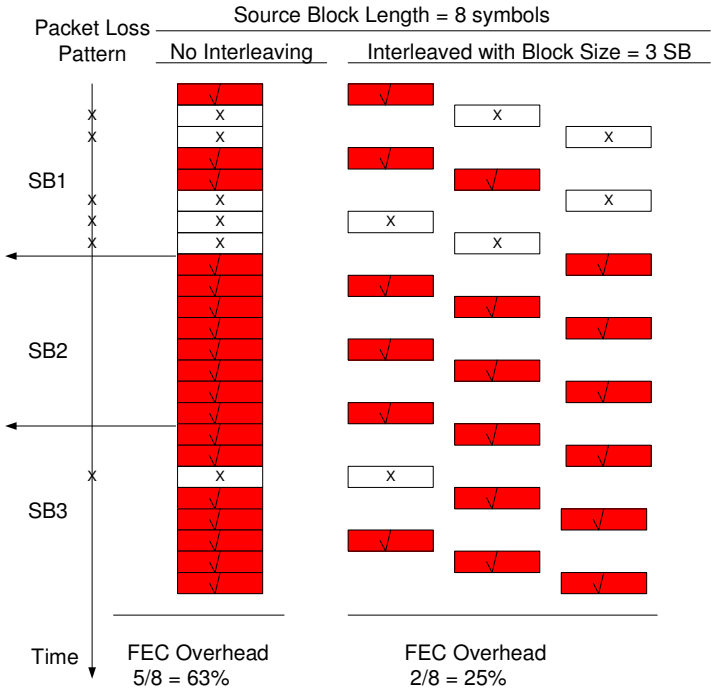


Figure 2.8 Interleaving effect on FEC performance.

After each transmission by skipping next symbol, 3 symbols from SB1, 3 symbols from SB2 and 2 symbols from SB3 are sent in first round, 3 symbols from SB1, 2 symbols from SB2 and 3 symbols from SB3 are sent next round, 2 symbols from SB1, 3 symbols from SB2 and 3 symbols from SB3 are sent in final round. The result is a reduction in necessary FEC overhead from 63% to 25% for reliability.

2.4 Related Work

Recently progressive download of 3gp media files in MBMS is studied in 3GPP working groups (BenQ Mobile, 2006). According to 3GPP PSS (packet-switched streaming service) specification (3GPP TS 26.234, 2007*),* PSS clients already support progressive download of 3GP media files with HTTP connection over TCP/IP. However, PSS is based on IETF RTSP (Real Time Session Protocol)/SDP and requires

bidirectional connectivity between sender and clients. Progressive download in MBMS is still in debate and currently few works (BenQ Mobile, 2006; Yetgin Z., Seckin G., March 2008; Yetgin Z., Seckin G., April 2008) exist to support MBSM progressive download.

Yetgin Z., Seckin G. (2007) studied MBMS downloading optimization for minimum FEC overheads for both Reed Solomon and Raptor FEC. In MBMS, FEC mechanisms have been studied on two layers, namely physical layer and application layer, in a complementary way. The tradeoff in applying one or the other or suitable combinations of the two is addressed in (Luby M., Watson M., Gasiba T., Stockhammer T., 2006; Watson M. & Stockhammer T., n.d). Interleaving in MBMS is also studied on these two layers. On the physical layer, Turbo coding with interleavers is used as a standard in 3GPP. Turbo codes emerged in 1993 (Berrou C., Glavieux A., & Thitimajshaima P., May 1993) and have since increased its popularity in communications research. In (Rekh S., Rani S.S. & Shanmugam A., 2005), some of those works are referred and the behavior of Turbo codes for various interleaver size and structure is analyzed. Luby M., Watson M., Gasiba T. & Stockhammer T. (October 2006) investigated the tradeoffs between the assignment of physical layer resources for UMTS turbo code and application layer resources for the MBMS download delivery service.

MBMS download delivery has already been analyzed in 3GPP working groups. Reed-Solomon codes with and without interleaving (Siemens, March 2005; Yetgin Z., Seckin G., 2007) and Raptor codes, also investigated in (Luby M., Watson M., Gasiba T., Stockhammer T., and Xu W., January 2006; Watson M. & Stockhammer T., n.d) for MBMS, are used in these analyses. Generally interleaving mechanism above FEC layer is studied as random transmission of symbols (Siemens, March 2005). However random transmission strategy disables the progressive download. Our recommendation is to use an interleaving strategy that also enables progressive downloading. This issue is also addressed in the work by Yetgin Z., Seckin G. (2007). The strategy used to support progressive download, just after sending each source block, it's repair symbols are sent and a group of consequitive blocks are interleaved at a time.

From the MBMS service delivery point of view, progressive download is a download because it uses MBMS download delivery mode based on FLUTE protocol. From the end user point of view, it is streaming because while the download continues in the background, media can be played in parallel with the downloading. Download delivery mode is very cost effective compared to streaming in that less bandwidth consumption is required in downloading since there is no real time requirement. As an alternative to the streaming, Siemens AG (August 2006) presents an intermediate delivery mode that constitute an actual bridge between streaming and downloading and based on MBMS download delivery mode, called "Preload Delivery Mode", for multimedia of limited duration. However, adding new network elements and requirement for a standardization effort are some of the critics, which make the proposed approach difficult to be deployed in practice.

Jenkac H., Stockhammer T., Xu W. (March 2006) provides an asynchronous and reliable solution for streaming, conceptually more suitable for progressive download. The proposal stands on partitioning of the media into segments and segment protection with fountain codes over FLUTE. Partitioning of the media into segments is not a new concept and is previously studied. In harmonic-broadcast based approaches such as Pyramid Broadcasting, studied by Hu A. (2001) and Engebretsen L., Sudan M. (2002), such segments are mapped onto different channels and mainly proposed for streaming type of applications. The drawback of these approaches is their expectation from the receiver to be capable of handling many channels in parallel. Repetition of these segments with FEC redundant-symbols over FLUTE is proposed by Jenkac H., Stockhammer T., Xu W. (March 2006) and Jenkac H., Stockhammer T., Xu W., Abdel Samad W. (May 2006) for streaming applications on a few channels where each segment is repeated at different frequency. By repeating the early parts of the media more frequently, users are expected to catch the overall stream from the beginning with acceptable initial startup delay. The solution can be equally or more suitably applied to the progressive download where missing the initial portions similarly causes the streaming to fail.

For the reliability issues, FEC is the only built-in mechanism in FLUTE. MBMS download reliability has already been analyzed in 3GPP working groups. Generally existing works analyses the download reliability to discover the minimum FEC overhead among a small set of the sizing parameters such as source block size, symbol size, IP packet size, SDU (Service Data Unit) and PDU (Protocol Data Unit) size and so on. However, Yetgin Z., Seckin G. (2007) studied minimum waiting time with minimum FEC overheads to discover the reliability among a large set of sizing parameters. According to MBMS specification, 3GPP TSG 26.346 (2007), the FEC repair symbols are sent after all the source blocks are sent to the clients. However, this approach is not suitable for progressive download application, since lost packets are recovered after the file download and client must wait for all source blocks to be sent. As studied in (Siemens, March 2005) Random transmission strategy of FEC repair symbols is not suitable from the same reason for progressive download too. Reliable download analyses supporting progressive download is recently studied by BenQ Mobile (2006) and Yetgin Z., Seckin G. (October 2007).

Apart from FEC reliability, some of the works use data carousal technique over FLUTE, Peltotalo J., Peltotalo S., Harju J. (July 2005), in which files are transported in loops and missing portions can be caught in next loops. This approach is not useful for playing the stream for the receiver who missed a portion in current loop and waiting for the following loop to recover. So once a portion is missed, the receiver has to wait the loop that serves the missed portion. This means the receiver can download but cannot play it during the download. However, data carousal technique is still important in that receivers missing a loop can still have a chance to use this technique for progressive download in subsequent loops.

Another way to guarantee the download reliability apart from FEC is to use MBMS repair procedure, one of the associated delivery procedures as defined in MBMS, 3GPP TSG 26.346 (2007), in which missing portions can be requested over ptp (point to point) or ptm (point to multipoint) repair sessions are configured. Again, this is not suitable for progressive download, since this procedure starts after the session ends or transmission of the object is finished.

In our analyses in the thesis, we try to find optimum combination among many that leads to minimum waiting time. Then, we provided an analysis of the progressive download delivery considering Reed Solomon and Raptor to find the best gain in waiting time for a large set of system parameters under various MBMS network conditions. Finally, we provided an analysis of the interleaved download delivery considering Reed Solomon and Raptor to find the best gain in FEC overhead for a large set of system parameters under various MBMS network conditions.

Four MBMS download systems are proposed in this thesis: (i) Legacy downloads with waiting time and transmission cost optimizations, (ii) Progressive download, (iii) Interleaved-legacy download, (iv) Interleaved-progressive download. Then we compared proposed systems with legacy download and give a conclusion in final section. Sequentially we did following steps:

1. Optimizations for Legacy-Download delivery are done to explore Reed Solomon FEC protected MBMS from the progressive download point of view.
2. Analysis for the Interleaved-Download delivery is done to find the gain in downloading time and transmission cost.
3. Analysis of the Progressive-Download is done to find the gain in waiting time.
4. Analysis for the Interleaved-Progressive-Download delivery is done to find the possible gain in waiting time and transmission cost.
5. Comparision of the legacy and the proposed download systems are provided.

CHAPTER THREE

FLUTE PROTOCOL

3.1 Overview

In multicast networks, especially in wireless environments, scability is the primary issue. For downloading in such environments, a necessity of reliability brings an extra challenge. So there is a requirement to achieve transport of content delivery in a unidirectional manner while preserving reliability and scability at the same time.

Typical multicast data streams are sent using UDP. TCP is not used because it is designed for one-to-one unicast streams of data. Multicast data streams sent over UDP are inherently unreliable because UDP does not provide guaranteed delivery or retransmission of lost packets. Lost packets in UDP-based multicast data streams cannot be detected or recovered, unless reliability is provided by the upper layer protocol.

There are many protocol standards that provide reliable multicast at the transport or application layers. Existing reliable multicast protocols fall into the following four categories (Microsoft, 2003):

1. Negative acknowledgement (NACK)-only

Receivers use NACK packets to request, from the sender, the retransmission of missing packets in the multicast data stream. NACK-only protocols do not require any additional support from routers in the network.

2. Tree-based acknowledgement (ACK)

Receivers use positive acknowledgments to indicate multicast data packets that are successfully received.

3. Asynchronous Layered Coding (ALC)

Senders provide forward error correction (FEC) with no messages from receivers or the routers of the network.

4. Router assist

Receivers use NACK packets. Routers in the network assist with retransmitting lost packets.

Router assist category adds network-centric requirements while other categories require bi-directional connectivity between sender and receivers. FLUTE and NORM are two IETF (The Internet Engineering Task Force) protocols. FLUTE is an ALC based protocol and it differs from other categories in that no messages are required from receivers to senders. That is, it requires a connection from sender to receivers but does not require a connection from receivers to sender. FLUTE has unidirectional property, fewer requirements, less overhead reliable transmissions, most scalability and interoperability.

NORM is designed to provide reliable transport of data from one or more sender(s) to a group of receivers over an IP multicast network. “The primary design goals of NORM are to provide efficient, scalable, and robust bulk data (e.g., computer files, transmission of persistent data) transfer across possibly heterogeneous IP networks and topologies.….NORM is a protocol centered around the use of selective NACKs to request repairs of missing data.” (Adamson B. & Others, 2004).

IETF Reliable Multicast Transport Working group (RMT WG) believes that variety of applications and orthogonal requirements these applications exhibit makes a "one size fits all" protocol unable to meet the requirements of all applications.

3.1.1 Target Environment

One of the desing goals for FLUTE is its massive scability to a large number of multicast receivers particularly in wireless environments. FLUTE is applicable to in both fixed networks such as IP multicast and wireless networks such as MBMS in UMTS and DVB-H in real broadcast networks as well as in satelite networks. Besides

its scability, it can also provide reliability through re-transmission besides Forward Error Correction. However, FLUTE re-transmission reliability is not used in MBMS.

FLUTE can be used for both multicast and unicast content delivery, but it is primarily designed for unidirectional multicast content delivery. FLUTE can support IP4 and IP6 environments. There is no IP specific part in FLUTE headers. FLUTE supports Any Source Multicast (ASM) model in which many senders concurrently send to same multicast group and receiving side discriminate packets by looking at source addresses and Source Specific Multicast (SSM) models in which there is only one sender source in a multicast session.

FLUTE provides reliability using the FEC building block. This will reduce the error rate but does not guarantee a complete success. “Because, FLUTE does not provide a method for senders to verify the reception success of receivers, the specification of such a method is completely application specific.” (Paila T. & Others, 2004). For example in MBMS, after the session, a receiver may request missing parts that was not downloaded with associated delivery procedures.

3.1.2 FLUTE Basics

FLUTE is an IETF standardized protocol for unidirectional delivery of files over UDP protocol, which is particularly suited to multicast networks. FLUTE is built on top of the Asynchronous Layered Coding protocol instantiation as shown in Figure 3.1 that uses LCT (Layered Coding Transport Protocol) to carry transport parameters such as session identifier and it allows receiver to discriminate among packets by TOI (transport object identifier). So a FLUTE packet can be destined for a specific TOI in a specific session identified by TSI (Transport Session Identifier) by using LCT session and object concept. However LCT alone can not handle the transport of objects, because the size of objects is unknown to it. It can generally transport binary objects of finite or indeterminate length. All files are referred as objects in LCT concept and hence in this document.

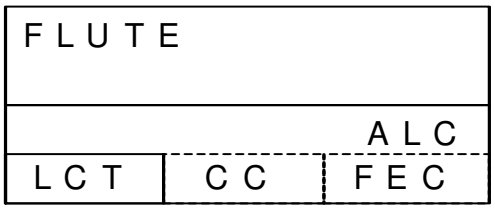


Figure 3.1 Building block structure of FLUTE.

FLUTE overlays its own headers so that each object with TOI is further described by file attributes such as file size, file name, encoding and as well as by transport descriptions such as FEC description parameters. All the files that are to be transported in a FLUTE session must be described in File Description Table (FDT). FDT table is an XML file stored local to sender. Example of a FDT table is given in Figure 3.2 for two files: A.3gp and B.txt. If a file is not described in FDT, it does not belong to that session. FDT is transported as FDT Instances with TOI=0. A FDT Instance can describe one or more files in FDT. FDT Instance carries a running index of files and their essential reception parameters in-band of a FLUTE session.

3.1.3 File Description Table (FDT)

TOI	File-Name (with URL)	Content-Type (MIME-type)	Expire	File-size	FEC-OTI Information (Enc. Id, Ins. Id, Enc. Specific Info)	Content-Encoding	Transfer Length	Complete
1	A.3gp	-----	-----	-----	-----	-----	-----	-----
2	B.txt	-----	-----	-----	-----	-----	-----	-----

Figure 3.2 Symbolic example of a FDT.

TOI is used as an index to FDT table that matches file description (saving) parameters such file name possibly with URL, Content type that identifies the type of content as a MIME type. Expire means reference to those object descriptions is valid until its expiry time. While file size shows the size of file before any encoding (if exist) is applied, transfer length shows the size of the object in compressed form such as

"gzipped". If any FEC coding is used for an object, it can be described in FDT. We will describe FEC and other file description parameters later in this section.

3.1.4 Sending FDT

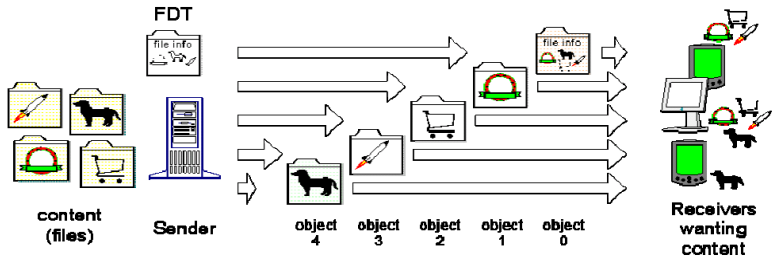


Figure 3.3 Four files described by one FDT instance.

FDT can be sent completely in one FDT Instance, in this case optional parameter "complete" can be used to indicate no more different file parameters will be described in any upcoming FDT Instance. In Figure 3.3, FDT describing four files are sent with one FDT Instance. Each file description in FDT can be separately sent in different FDT Instance. Figure 3.4 shows an example of this case. Four files are described in four FDT Instances in Figure 3.4.

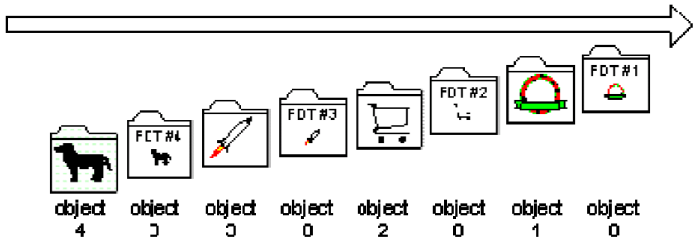


Figure 3.4 Four files described by four FDT instances.

Many combinations can be created. One FDT instance carries at least one file description at most complete FDT. During the session, FDT Instances can appear at any moment of time. However, it is recommended FDT Instances should be sent prior to beginning of transmission of actual object that it describes.

At any moment of time during the session, FDT Instances can be duplicated as shown in Figure 3.5, or re-described by any subset, superset of a FDT Instance. However, any FDT Instance cannot change file descriptions with a specific TOI that previously sent in any other FDT Instance. Figure 3.5 shows a datacarausal approach where session files are sent in a loop until session ends.

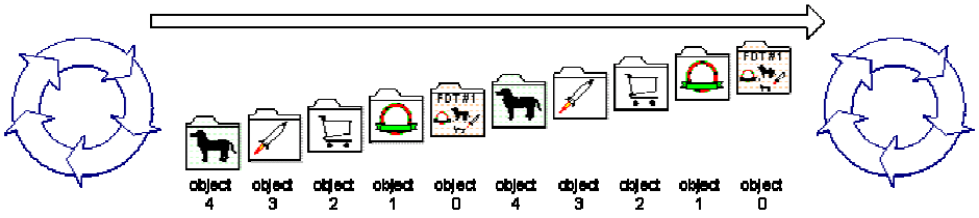


Figure 3.5 Four files described by one FDT Instances in a loop.

3.1.5 Flute Session Concept

A FLUTE session is actually an ALC/LCT session consisting of one or more ALC/LCT channels sharing a single sender. Sender's IP address and TSI (Transport Session Identifier) uniquely identify a FLUTE session. Each channel in a session is defined by a tuple of sender IP address and an address associated with the channel. Channels are used with a suitable congestion control building block. In the case of multiple channels, multiple rate receiver-driven congestion control protocol is used. A receiver must join to a channel in order to receive data sent to the channel by the sender. Similarly the receiver leaves the channel to stop receiving data packets from the channel.

3.1.6 General FLUTE Protocol Flow

At the beginning, both sender and receiver side know session description parameters so that sender knows when and which multicast group it will send to, similarly receiver knows sender IP address, the time to join to session, and other session parameters. However, receiver does not need to know file parameters in advance of session startup. FLUTE protocol allows receiver gradually to learn file descriptions required for download. This is achieved by FDT Instances as stated before. Diagram in Figure 3.6

shows an example that a multicast session is taking place from sender to receiver. Sender has already sent a FDT Instance describing only *A.3gp* and now sending source blocks belonging to *A.3gp* after which FDT instance describing the file *B.3gp* will be sent next.

Sender has 3 files A.3gp and B.txt, which are described in FDT.xml file. Partitioning of A.3gp is achieved by a blocking algorithm that depends on FEC Scheme used. Blocking algorithm partitions the file into source blocks so that sizes of source blocks are as close each other as possible. In the diagram shown in Figure 3.6 source blocks are symbolically indicated as cells. Thus, a source block, SBi, is converted to an encoding block, EBi, by FEC encoder. Encoding block EBi includes same or larger number of encoding symbols than the source block SBi.

Now sender has to transmit the EBi. However, sender may not send a complete encoding block at once. So a group of G consecutive encoding symbols in the EBi, called ESG (Encoding Symbol Group), are placed into FLUTE payload with a marker in the FLUTE header that shows the starting index, j, of the ESGi in the encoding block, called FLUTE Payload ID = (i, j) here to indicate starting index j in Encoding Block i.

Now constructed FLUTE packet is transmitted over UDP/IP within a FLUTE session. Receiver behavior is quite simple, when it receives a FLUTE packet carrying ESGj, it checks which TOI owns the packet. If TOI is zero, it means, ESGj belongs to a FDT Instance so receiver will update current FDT database. Otherwise, ESGj belongs to a file, so receiver decodes and possibly uncompresses the ESGj if it received enough number of encoding symbols and then process results in original source block i, as Figure 3.6 shows.

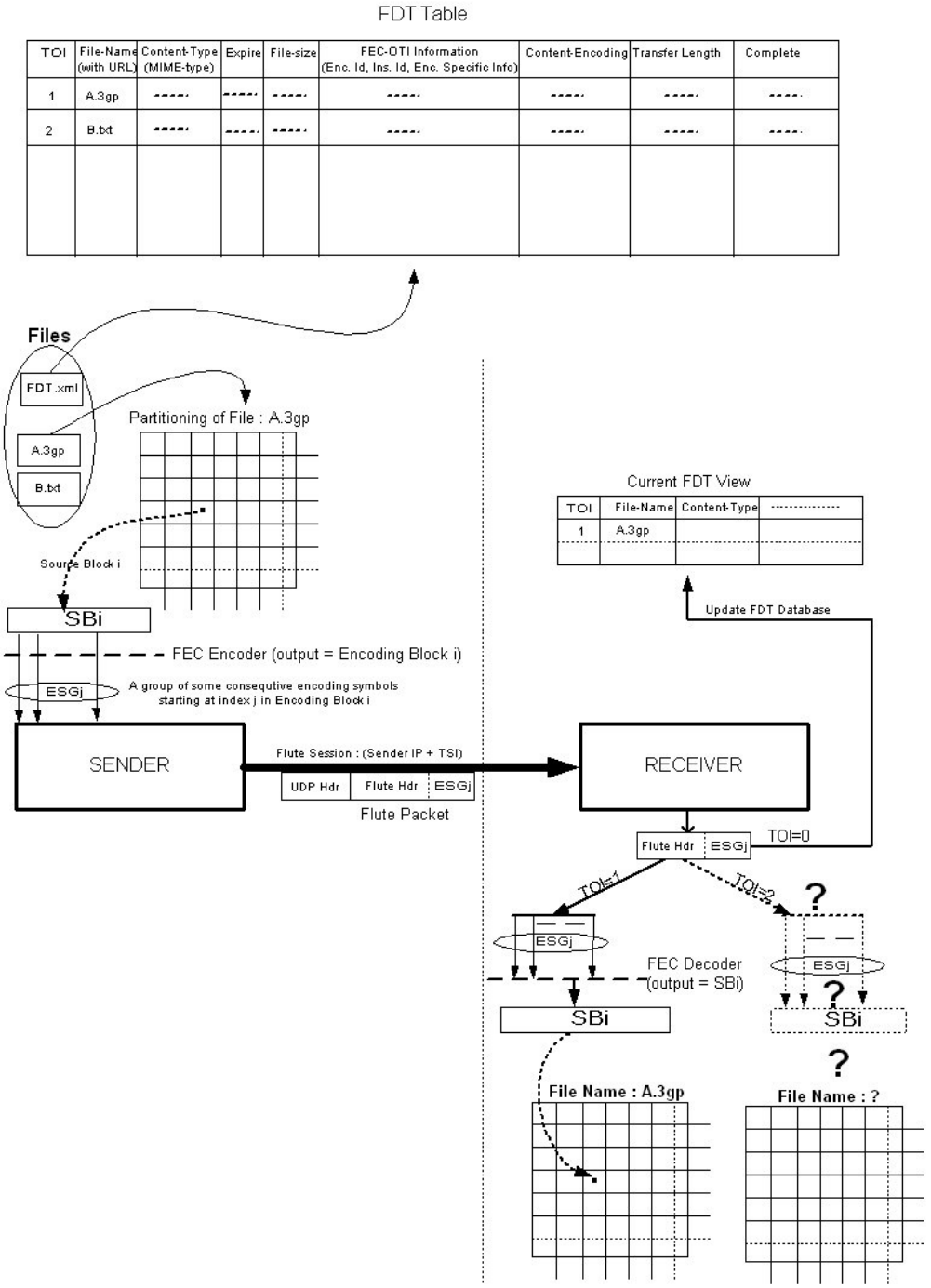


Figure 3.6 Example of the FLUTE flow diagram.

Meanwhile, if a packet comes with TOI=2, receiver behaviour is completely application dependent. Because receiver doesn't know file description information of TOI 2.

Simplest solution is to ignore the packet. Another possible solution is to put packet to a buffer, wait for a FDT Instance that describe it. Hence in Figure 3.6, the question marks indicate that the receiver behaviour is application dependent.

3.1.7 Flute Packetization/Depacketization

3.1.7.1 Partitioning

Partitioning of a file is achieved by a blocking algorithm that depends on FEC Scheme used. Blocking algorithm computes file fragmentation structure. It decides how to partition the file to source blocks so that sizes of source blocks are as close to each other as possible. It must be divided in such a way that first number of source blocks are of the same larger length, and remaining second number of source blocks are of the same smaller length as shown in Figure 3.7. $Z=Z_L+Z_S$ shows the number of source blocks that file is partitioned into. Z_L shows the left larger part, Z_S shows the rigth smaller part. K_L is the number of source symbols in a source block that is at left side (larger part) K_S is the number of source symbols in a source block that is at right side (smaller part). At any moment of time, a source block includes either K_L or K_S source symbols. Blocking algorithm computes above parameters Z_L, Z_S, K_L and K_S depending of the FEC Scheme used.

With FEC, “the data stream is transformed in such a way that reconstruction of data object does not depend on the reception of specific data packets, but only on the number of different packets received.” (Luby M., Gemmell J & Others, 2002). FEC requires that the object is partitioned into source blocks that sufficient enough in size to be able to be stored and processed in FEC buffer. Source Block consist of K source symbols. Each source block is independently given to FEC algorithm as input. For each source block that given to FEC algorithm, some N redundant symbols are generated, which is called repair symbols. When FEC decoder gathers any K of K+N encoding symbols, it can recover the original source block.

3.1.7.2 Packetization

FLUTE Packetizer receives packet payload as input creates a complete FLUTE Packet as output. In Figure 3.7, FLUTE packet is shown within an IP packet. FLUTE payload is actually a group of consequtive encoding symbols (ESG) identified by a single ID, ESI (Encoding Symbol ID).

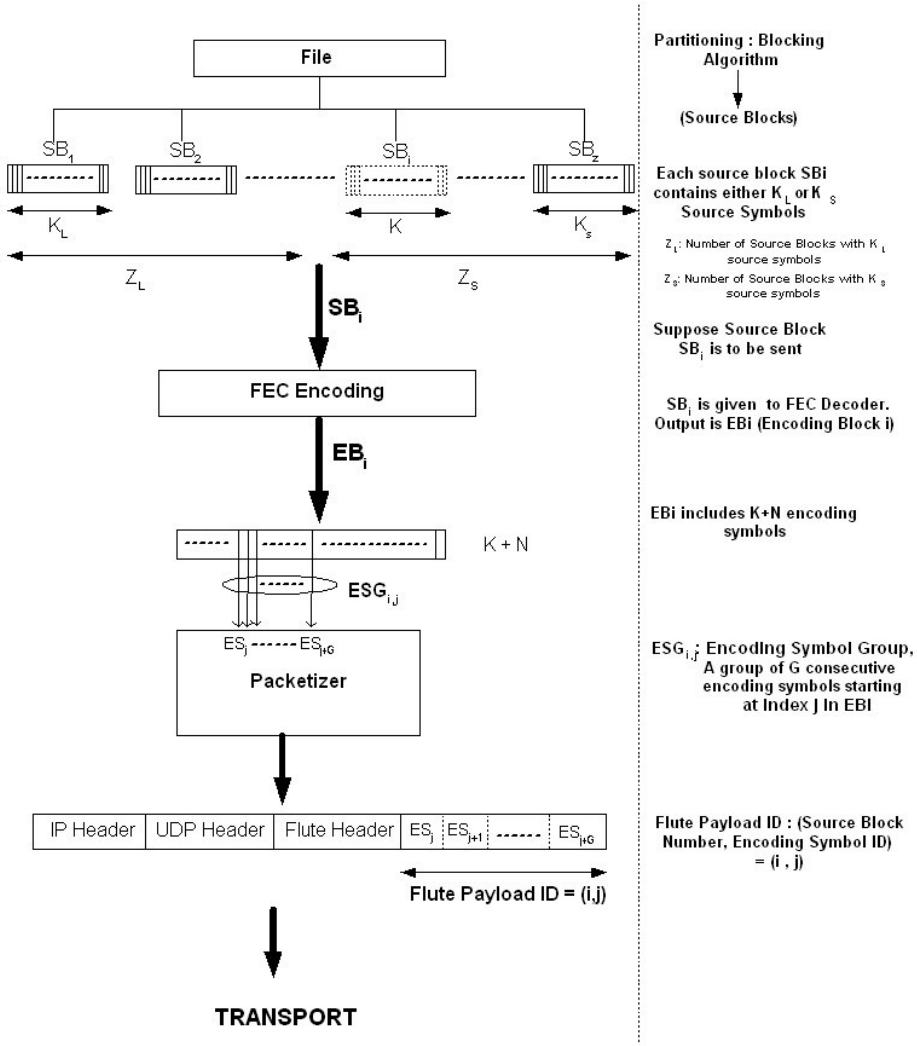


Figure 3.7 FLUTE packetization.

ESI is the starting index of the first symbol of ESG in the Encoding Block (EB). ESI together with the SBN (Source Block Number) describes the FLUTE Payload ID. For

example in Figure 3.7, the ESG = EB_j, EB_{j+1},... EB_{j+G} is referred as ESG_j, and its ESI is j. With SBN = i, FEC Payload ID becomes (i, j).

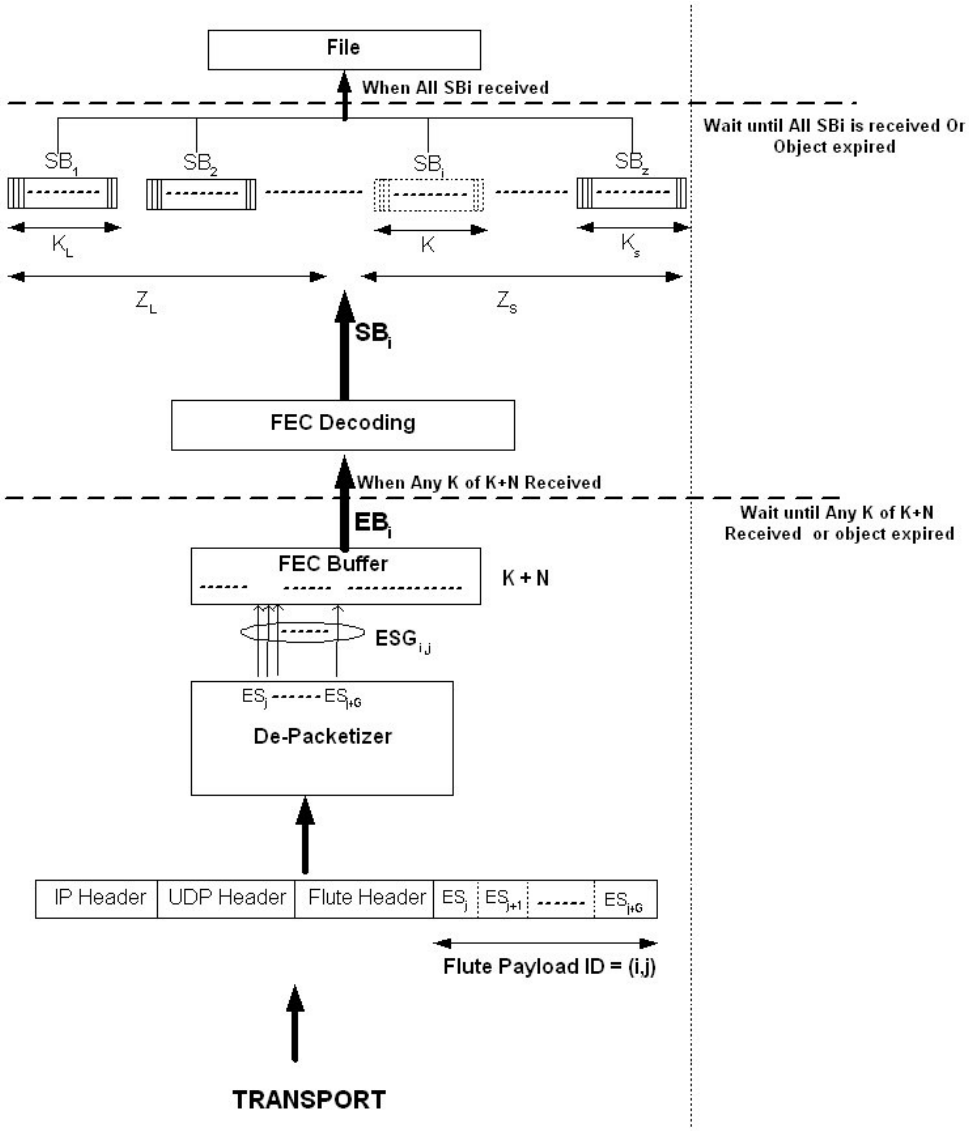


Figure 3.8 FLUTE de-packetization.

3.1.7.3 De-Packetization

FLUTE De-Packetization process is shown in Figure 3.8. Receiver receives an IP/UDP packet that includes FLUTE Packet as UDP payload. FLUTE payload is

extracted and its content ESG, a group of consequtive encoding symbols identified by ESI (Encoding Symbol ID), placed into FEC Buffer. When enough encoding symbols are collected (K of K+N), process is delivered to FEC decoding. FEC decoder decodes the Encoding Block EBi and restores the original Source Block, When all source blocks are received, file reception is completed.

3.2 FLUTE Details

> FLUTE is built on top of the Asynchronous Layered Coding protocol instantiation, a massively scalable reliable content delivery protocol that combines the Layered Coding Transport (LCT) building block, a congestion control building block and the Forward Error Correction (FEC) building block to provide congestion controlled reliable asynchronous delivery of content to an unlimited number of concurrent receivers from a single sender. (Luby M. & Others, 2002).

As Figure 3.1 shows, LCT is not an overlay over CC and FEC instead it provides place holders to be aligned and compatible for CC and FEC.

3.2.1 Layered Coding Transport Block (LCT)

LCT (Luby M., Gemmell J. & Others, 2002) defines the basic transport mechanism for FLUTE. It defines session concept, channel concept and object concept in that it includes a unique TOI (Transport Object Identifier) for each object. FLUTE defines objects further to form a complete object definition by specifying download parameters for each object: object name, location, object saving parameters, etc. An LCT session consists of one or more logically grouped LCT channels sharing a common sender source in LCT headers. An LCT channel is defined by a combination of a sender and an address associated with the channel by the sender. There are currently two models of multicast delivery: Any Source Multicast (ASM) and Source Specific Multicast (SSM). LCT works with both multicast models. FLUTE adopts these transport level definition of LCT.

LCT supports transport level functions in sender or receiver applications by in-band signaling of transport parameters in LCT header. FLUTE uses the session management functionality defined in LCT. LCT includes general support for congestion control that provides multiple rate or single rate delivery to receivers but does not specify which congestion control is to be used. “LCT is also compatible with coding techniques that possibly provide reliable delivery of content” (Luby M., Gemmell J. & Others, 2002) but does not specify which coding technique to use. In LCT concept, a session can be composed of many layers each of which can be coded independently. LCT even does not require the coding technique to be used should enable reliability. ALC and hence FLUTE define more specificly some of these open issues, and finally FLUTE forms a complete protocol. ALC adopts to use a coding technique that also enables reliability by means of forward error correction.

LCT provides a number of fields and supports functionality commonly required by many protocols. For example LCT provides TSI (transport session identifier) that uniquely identify a session, TOI (Transport Object Identifir) that uniquely identify an object, Congestion Control Information (CCI) which allows the receiver to perform the required congestion control on the packets received. Default LCT header fields is given in Figure 3.9. The format of Header Extensions is given in Figure 3.10.

Within an LCT session each packet has an “LCT header”. The LCT header format in Figure 3.9 is the default format. “This format is the recommended for use by protocol instantiations to ensure a uniform format across different protocol instantiations. “ (Luby M., Gemmell J. & Others, 2002). If default LCT header is not used, position and length of each header fields must be specified. LCT header attributes are explained in following list.

- Default LCT header is of variable size. The length of the overall header is given by “HDR_LEN” field. When constructing an LCT packet, all padding and reserved (indicated as ‘r’) bits must be set to “0”. All integer types are put into “big-endian” or “network order” format.

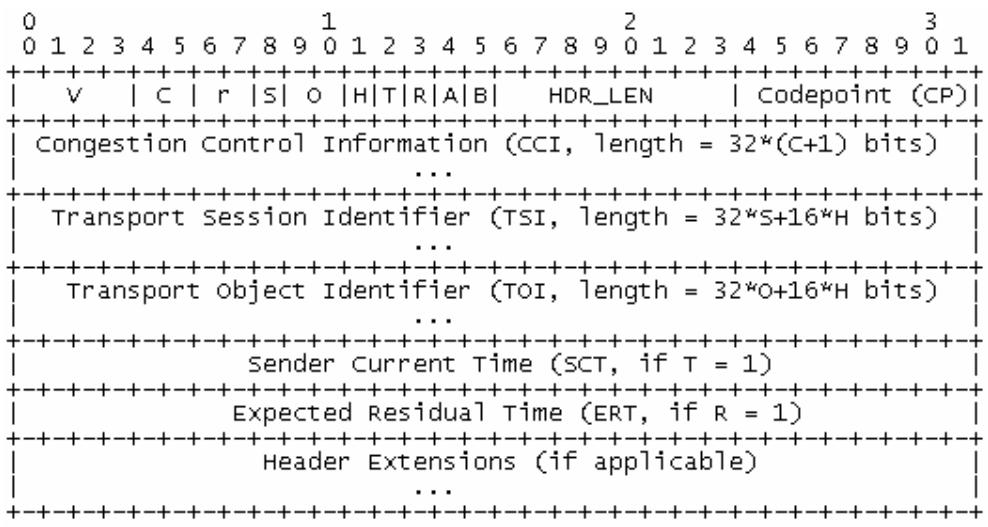


Figure 3.9 Default LCT header format (Luby M., Gemmell J. & Others, 2002).

- V (4 bits): LCT version number
- C (2 bits): Congestion Control flag. It defines the length of CCI field.

  C=0→CCI=32 bits

  C=1→CCI=64 bits

  C=2→CCI=96 bits

  C=3→CCI=128 bits
- S (1 bit): Transport Session Identifier flag. It partially defines the length of TSI field;

  S=0→TSI=0 or TSI=16 bits

  S=1→TSI=32 or TSI=48 bits
- O (1 bit): Transport Object Identifier flag. It partially defines the length of TOI field;

  O=0→TOI=0 or TOI=16 bits

  O=1→TOI=32 or TOI=48 bits
- H (1 bit): Half-word flag. It allows TSI and TOI field lengths to be multiple of 16 bits;
- T (1 bit): Sender Current Time present flag. It decides whether SCT field is present;

  T=0→SCT is NOT present

  T=1→SCT is present

- R (1 bit): Expected Residual Time present flag. It decides whether ERT field is present;

  R=0→ERT is NOT present

  T=1→ERT is present
- A (1 bit): Close Session flag. It indicates that session is about to end. Once it is set to 1 by the sender, in following packets it must be set to 1 until session is closed.
- B (1 bit): Close Object flag. It indicates that delivery of an object identified by TOI in the header is about to finish. Once it is set to 1 by the sender, in following packets for the same TOI it must be set to 1 until the object transmission is finished.
- HDR_LEN (8 bits): Total length of the LCT header that must be multiple of 32 bits.
- CP (8 bits): Codepoint. It gives coding information of LCT payload. It is similar to payload type field in RTP protocol header. In FLUTE, this field corresponds to FEC Payload ID.
- TOI (variable length) : Transport Object Identifier. Depending on the S and H bits it can be 0, 16, 32, 48, 64, 80, 96, 112 bits. In LCT each packet carries content belonging to only one object identified by TOI.
- SCT (variable length): Sender Current Time. Depending on the S and H bits it can be 0 or 32 bits. It represents the time when the packet was transmitted at the sender, local to session start time,. If SCT reaches 2^32-1, it start from zero again.
- ERT (variable length): Expected Residual Time. Depending on the S and H bits it can be 0 or 32 bits. It represents the expected time after which the current session or current object transmission is finalized, from the sender point of view. If packet containing ERT field also contain TOI, ERT applies to that object otherwise applies to session.
- CCI (variable length): Congestion Control Information. Depending on the S and H bits it can be 32, 64, 96 or 128 bits. It is used to carry congestion control information.

- TSI (variable length): Transport Session Identifier. Depending on the S and H bits it can be 0, 16, 32, or 48 bits. In LCT TSI is scoped by sender IP address. Thus, the IP address of the sender together with the TSI uniquely identifies a session. TSI must be unique among all sessions served by the sender during the period when session is active.
- Header Extensions (Variable Length) : It gives a way for upper layer protocols to add their specific header information as well as used as container for optional header fields that are not always used or have variable size, for example extended-size versions of already contained header fields.

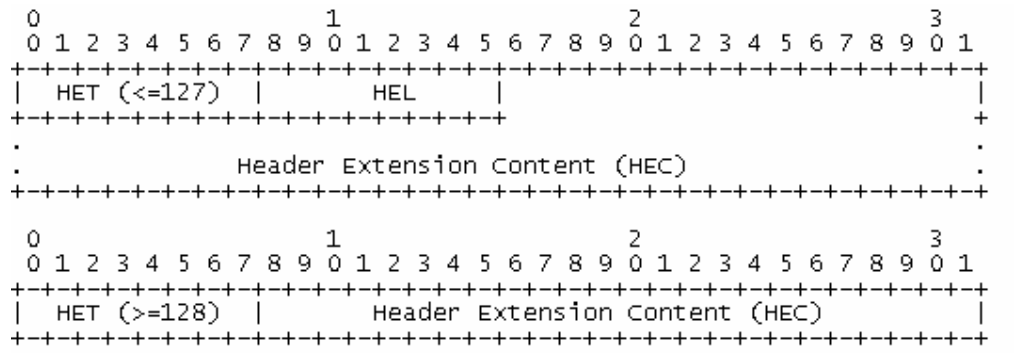


Figure 3.10 Format of extension headers (Luby M., Gemmell J. & Others, 2002).

- HET (8 bits): Header Extension Type. HET values from 0 to 127 are used for variable length header extensions. HET values from 128 to 255 are used for fixed length 32 bit header extensions. Following general header extension types that must be supported by all senders and receivers are defined:
- EXT_NOP=0: No operation extension that must be ignored by the receivers, EXT_AUTH=1: Packet Authentication Extension that is used to authenticate the sender of the packet. It must be recognized by senders and receivers. However its contents may not be able to be parsed by them.
- There may be header extension types that upper layer protocol using LCT defined. For example ALC adds FEC OTI (object transmission information) Extension header and Content Encoding Extension Headers. FLUTE adds FDT (File delivery table) extension headers.

- HEL (8 bits): Header Extension Length. The length of the whole header extension field is expressed in multiple of 32 bits.
- HEC (variable length): Header Extension Content. For fixed length header extensions, HEC is 24 bits, for variable length header extensions HEC is variable in size depending on the HEL field.
- ALC and hence FLUTE strengthen the transport concept in LCT and uses the default LCD header specified here.

3.2.2 Congestion Control Building Block

Congestion Control Building Block needed to enable co-existing of FLUTE and TCP traffic on the internet. ”FLUTE is applicable to both internet use with a suitable congestion control building block and provisioned\controlled systems, such as delivery over wireless broadcast radio systems. ” (Digital Fountains & Others, May 2004).

Multiple rate or single rate congestion control protocol can be used with LCT. For multiple rate procols, receiver driven approach is used. A session consists of more than one channel (layer) each of which is possibly coded. Sending rate of each channel does not depend on receiver current state. At any moment of time, receiver explores its available bandwidth and joins or drops channels dynamically upon the available bandwidth independently of other receivers. For single rate protocols, sender driven approach is used where a session consist of one channel and during the course of time, sending rate changes based on feedback from receivers. Reception rate of each receiver may vary dynamically but in cordination with all receivers. This approach requires feedback from receivers.

ALC protocol hence FLUTE potentially can use both single rate and multiple rate approach. Because of its massive scalability, and its suitability for reliable content delivery, feed-back free multiple rate congestition controls are adopted by ALC. Additionally by using FEC coding technique, reconstruction of an object, does not depend on the reception of specific data packets, rather than the number of different packets received, which may be gathered from multiple channels. As a result by increasing number of channels, the receiver can reduce the transfer time accordingly

without giving any concession from reliability. However if single channel is to be used, the effect of using multiple rate congestion control with single channel implies no congestion contol where the control is provided by other means.

In summary, four scenarios are possible with FLUTE: i) use of single channel and single rate congestion control protocol, ii) use of multiple channels and multiple rate congestion control protocol, iii) use of single channel without congestion control provided by ALC but possibly provided by other means, iv) use of multiple channels without any congestion control supplied by ALC but possibly by other means. Possible Congestion Control Building Blocks are Wave and Equation Based Rate Control (WEBRC) Building Block (Luby M., & Goyal V., April 2004), Receiver Driven and among others (Welzl M., & Eddy V., March 2007).

3.2.3 Forward Error Correction (FEC) Building Block

LCT does not specify which coding technique to use, and even does not require a coding technique that enables reliability such as forward error correction. But the natural definition of its layered concept advantageously can be used with a coding technique that also enables reliability. The concept of LCT can be naturally extended to reliable content delivery protocols by using forward error correction coding technique. ALC and FLUTE does not specify which FEC algorithm is to be used. But they define sufficient place holders to encompass any FEC algorithm to be used by defining FEC specific ALC extension headers.

A FEC algorithm or equivalently a FEC scheme can be specified by a tuple of FEC Encoding ID and FEC Instance ID (=0 for Fully-Specified FEC Scheme). FEC schemes can be classified into Fully-Specified FEC Scheme and Under-Specified FEC Scheme. In Fully-Specified FEC Scheme, the algorithm is fully specified and opened to other implementors. In Under-Specified FEC Scheme, the algorithm is either not fully specified or not opened to other implementors. FEC Enc. ID also specifes the format of header extension fields that carry FEC OTI in ALC packet. Different FEC algorithms need different sets of encoding parameters. So there is FEC-Enc. specific part in General FEC OTI extension header as shown in Figure 3.11.

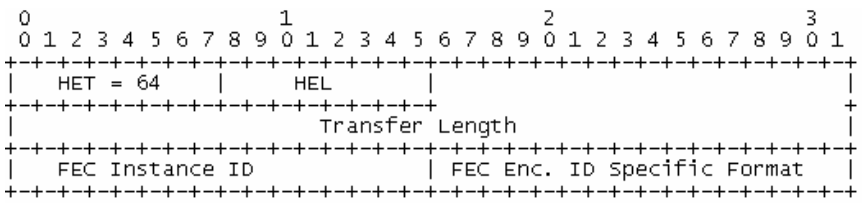


Figure 3.11 General EXT_FTI format (Luby M., Gemmell J. & Others, 2002).

LCT concept gives upper layer protocol to add their protocol specific extension headers in LCT extension concept. ALC uses FEC OTI Extension Header, the format of which is shown in Figure 3.11. FEC OTI Extension Header with FEC Enc. ID specific part carries complete FEC OTI information. FEC encoding algorihms sharing the same FEC Enc. ID uses the same FEC Enc. ID Specific Format. FEC Enc. ID is a number between 0 to 255, 0 to 127 are reserved for Fully-Specified FEC Scheme, 128 to 255 are used for Under-Specified FEC Scheme. For Fully-Specified FEC Scheme, only FEC Encoding ID is used to specify FEC algorithm with FEC Instance ID set to 0. For Under-Specified FEC Scheme, both FEC Encoding ID and FEC Instance ID are used to specify FEC codec to be used.

The use of FEC in Reliable Multicast is defined in RFC 3453 (Luby M., Vicisano L., & Others, 2002). Possible FEC Encoding algorithms are as follows:

- XOR FEC
- Reed Solomon
- Parity Checked Matrix-Based FEC
  Low Density Parity Check FEC
  Low Density Generator Matrix FEC
    — LDGM-Staircase
    — LDGM-Triangle
- Raptor FEC

With FLUTE, the default FEC is Compact-No Code FEC (NULL FEC) that does not encode or decode objects. It simply equalizes source symbols and encoding symbols.

3.2.4 FEC OTI Information

FEC OTI Information is signalled in-band either with EXT_FTI extension header and CP (code point) that carries FEC Encoding ID or using FDT where FEC OTI parameters can be specified for objects of TOI > 0. EXT_FTI format inherits from ALC and its general format is shown in Figure 3.12. The complete format of this header depends on the FEC scheme used.

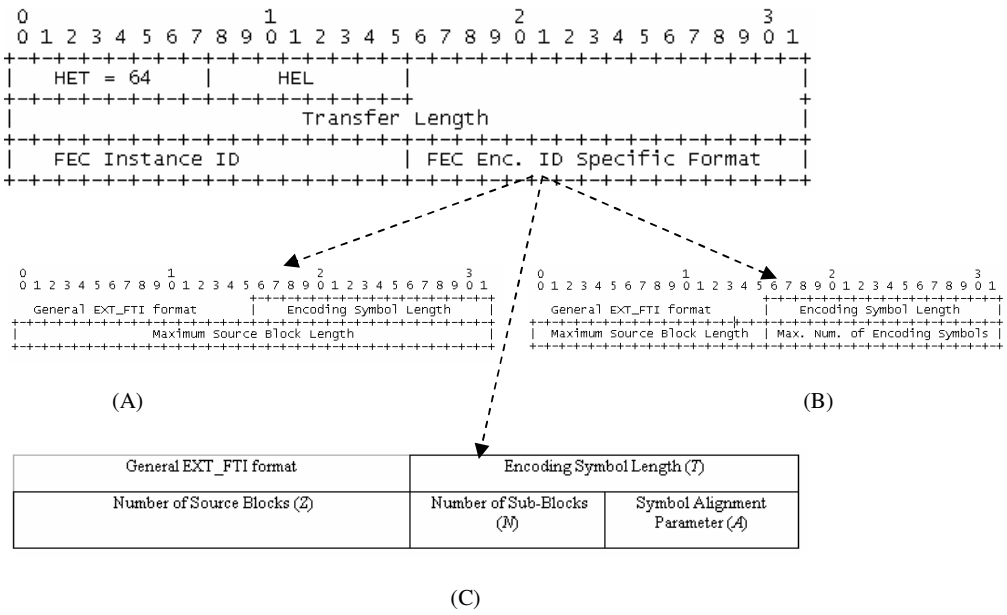


Figure 3.12 FEC OTI Examples for different FEC Encoding IDs A) FEC Enc. ID=0,128,130 specific format B) FEC Enc. ID=129 specific format C) MBMS specific format.

Figure 3.12 gives 3 examples for 3 different FEC schemes. For FDT Instance objects (TOI=0), FEC OTI information must be provided using EXT_FTI. For non-FDT instance objects (TOI>0), both EXT_FTI and FDT can be used to deliver FEC OTI information. FEC OTI Information required to be delivered is dependent on the FEC scheme used. FEC Enc. ID specifies the format of FEC Enc. specific part in General EXT_FTI format.

Following FEC information is common in all FEC Schemes:

- FEC Encoding ID

- FEC Instance ID
- Transfer Length (Object size that is being transferred).

Following FEC OTIs are dependent on FEC Scheme used:

- Encoding Symbol Length (A,B,C)
- Maximum number of source symbols in a source block (A,B)
- Maximum number of encoding symbols for a source block (B)
- Number of source blocks (C)
- Number of sub-blocks (C), when a source block size cannot be fit into FEC buffer, sub- blocks are used.
- Symbol Alignment parameter (C), ensures that Symbols in a source block and sub-symbols contained in a sub-block are multiple of Alignments in size.

3.2.5 Asychronous Layered Coding (ALC) Protocol

Asynchronous Layered Coding combines the Layered Coding Transport (LCT) building block, a multiple rate congestion control building block and the Forward Error Correction (FEC) building block to provide congestion controlled reliable asynchronous delivery to concurrent receivers. The most prominent overlays that ALC specifies over LCT concepts are FEC as coding technique and Congestion Control adaptation. Hence in ALC packet format, content type of payload is the FEC encoding type used; hence in default LCT header CP (codepoint) field carries FEC Encoding ID. ALC also forces TSI (Transport Session ID) length not to be zero. It means a session concept in LCT is more strengthen in ALC. We have already explained LCT, FEC and CC building blocks over which ALC is built. So we will not repeat them here and we have also given ALC adaptions as much as possible in previous sections.

The overall ALC packet format is given in Figure 3.13. Default LCT header is given previosly in detail in Figure 3.9. ALC packet resides in UDP payload. ALC headers are Default LCT headers and Fec Payload ID. Encoding symbols reside in ALC payload. FEC payload ID is a tuple of Source Block Number and Encoding Symbol ID. So FEC

payload ID specifies which source block packet belongs to and enables differentiation among encoding symbols by Encoding Symbol ID.

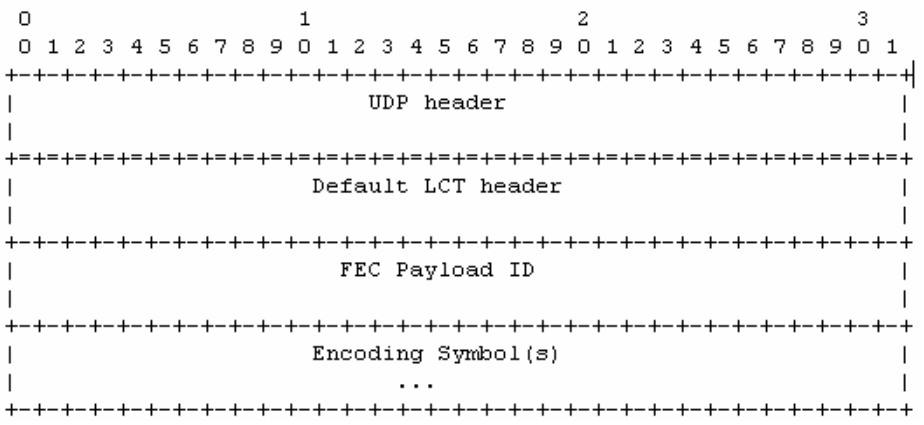


Figure 3.13 Overall ALC packet format (Luby M., Gemmell J. & Others, 2002).

When a source block is encoded into an encoding block by FEC encoder, encoding block may not be able to be put into a single ALC payload at once. So a group of G consequtive encoding symbols are placed into ALC payload and sent until encoding block is transmitted. These consequtive G encoding symbols, called encoding symbol group, identified by a single Encoding Symbol ID that is the encoding symbol ID of the first encoding symbol in the payload.

3.2.6 FLUTE Packet Format

FLUTE is built on top of ALC. ALC provides the basic trasport for FLUTE. So FLUTE is an overlay on ALC that inherits all properties of ALC and further define the object concept by in-band signaling of object properties together with the delivered object. The core of FLUTE specification over ALC is to define how object properties such as file-name, location, file-type, content encoding type, security properties, and other saving parameters are carried in-band during a session. This property is important in that it allows a FLUTE session to start without knowing the actual object that is being transferred. So a FLUTE session may gradually define object parameters during a session.

FLUTE adds two extension headers in LCT extension concept: EXT_FDT and EXT_CENC and also uses EXT_FTI inherited from ALC. FLUTE packet can be decomposed to four abstractions: Transport Information, FEC OTI Information, FDT Instance Information, Content Encoding Information. Transport Information is what ALC over LCT concept provides. Following sections provides the other abstractions.

3.2.6.1 FDT Instance Information

FDT (File Delivery Table) information is contained in an XML file local to sender. It describes all the files with corresponding file attributes and saving parameters related to delivery of files during the session. Each FLUTE session requires a FDT file local to sender and a file that is not described in FDT, cannot belong to that session. FDT file can include the following object–level information:

- TOI, uniquely identifies the file during the session. It is scoped by concerning session (obligatory)
- FEC OTI Information,(described in FEC Object Tranmission Information section)
- Transfer-Length, Object size that is transferred
- File Name and Location identified by the URL (obligatory)
- Content-Type, MIME media type of file
- Content-Length, File Size
- Content Encoding, Encoding of File if compression is applied
- Content-MD5, Message digest of file
- Additionally FDT Instances include following instance-level information that applies to FDT Instance:
- Expires, A FDT Instance is valid until its expiration time (obligatory)
- Complete, a boolean value to indicate no new FDT intances will be provided anymore until the end of the session
- Content-Type, MIME media type of files
- Content-Encoding, Encoding of Files
- FEC OTI Information

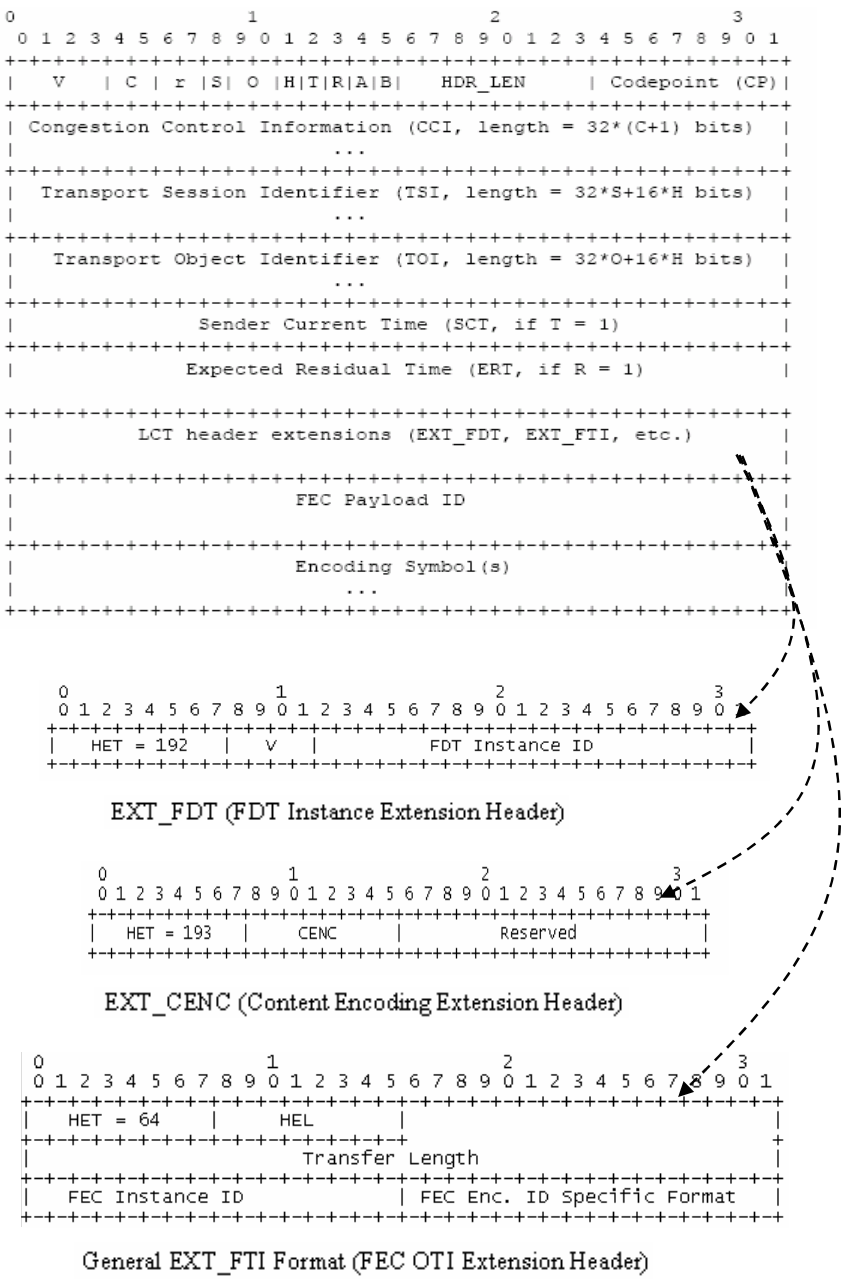


Figure 3.14 FLUTE packet format (Luby M., Gemmell J. & Others, 2002).

Each file in a FLUTE session must be associated with a TOI >0 in the scope of that session. Hence TOI and URL information are obligatory, others are optional. FDT is carried as FDT instances during the session. Each FDT instance is carried as transport objects TOI= 0. Within a FLUTE session, any TOI=0 means that packet carries a FDT instance describing one or more files in FDT. A FDT instance can be subset, superset,

dublication, or complement of any other FDT Instances. Each FDT instance is identified by FDT Instance ID. FLUTE adds FDT Instance specific extension header, EXT_FDT, with HET=192. In EXT_FDT as shown in Figure 3.14, FDT Instance ID reserves 20 bits. For each FLUTE session, FDT Instances starts from zero and it is incremented by one for each subsequent FDT Instances.

A receiver of a FLUTE session keeps a FDT database that reflects the current state of receiver knowledge of the FDT. Upon receiving a new FDT Instance, receiver updates its FDT database. In FLUTE specification, FDT Insances are recommended to be delivered before the files that are described by those FDT Instances. However receiver should assume that a FDT Instance may appear at any time during the session.

3.2.6.2 Content Encoding Information

Content Encoding Information is signalled in-band for FDT instances (TOI=0) using EXT_CENC extension header as shown in Figure 3.14. That is, FDT instance itself can be content encoded and EXT_CENC is used to deliver which content encoding algorithm is to be performed to extract original FDT instance.

For non-FDT instance objects (TOI>0), content encoding information is carried using FDT. In Figure 3.14, EXT_CENC format is shown. FLUTE identifies this extension header by giving a header extension type, HET=193. CENC (8 bits) carries content encoding algorithm used in FDT Instance payload. Possible content encoding algorithms inludes implementations of Deutsch P. (May 1996): ZLIB, GZIP, and DEFLATE.

3.2.7 FLUTE Session Descriptions

A FLUTE session is identified by a sender IP address and TSI (Transport Session Identifier). Each channel in a session is defined by a tuple of sender IP address and an address associated with the channel.

Receiver needs session descriptors to initiate a service. Service descriptors for FLUTE I-D (Walsh R., & Others, January 2007) specifies session description attributes for FLUTE. Sender must also know session descriptions in order to begin, send to and end the session. Sender must send all files described in a FDT XML file that is local to sender. Session descriptions and FDT information are explained in following sections.

3.2.7.1 Session Descriptions

Session description must be known by both sender and client side. Sender must know when, how and to whom it will send the data, similarly receiver must know when, how and from whom it will get the data. Service Description is a meta data (such as SDP or XML based) that contains required information for both side to initiate a service. For receiver it is obligatory to have a Service Description service component. To describe a FLUTE session, following transport parameters must be provided in Session Descriptions:

- The Source IP Address
- Number of channels in the session
- Destination IP Address (Multicast IP) and port number for each channel in the session
- Transport Session Identifier (TSI) that is unique in the scope of that session.

Optionally following parameters should be provided to the receiver

- Start time and End time of the session
- FEC OTI Information
- FEC Encoding ID
- FEC Instance ID
- FEC Encoding ID Specific Information (Encoding Symbol Length, Maximum number of symbols in a source block, Maximum number of source blocks, etc)
- Content Encoding Format

How service description parameters can be described is out of the scope of FLUTE specification. It can be a description syntax such as SDP or XML based. How session description information is acquired is again out of scope of the FLUTE specification. It can be via some transport protocols such as HTTP, email, SIP, Session Announcement Protocol, manual pre-confugaration,etc.

An example of Session description with SDP syntax is given in the text below that describes 2 channels identified by two IP6 addresses and two corresponding port numbers: Address1=FF1E:03AD::7F2E:172A:1E24, port1=4001 and Address2= FF1E:03AD::7F2E:172A:1E25 and port2=4002. Both channels are FEC Protected. First channel uses FEC Encoding ID=0, second one uses FEC Encoding ID 128 and Instance ID=0.

```
v=1
o=user_zeki 2890844526 2890842807 IN IP6 2201:056D::112E:144A:1E24
s=Zeki File delivery session example
i=More information
t=1160636400 1192172400
a=source-filter: incl IN IP6 * 2001:210:1:2:240:96FF:FE25:8EC9
a=flute-tsi:1
a=flute-ch:2
a=FEC-declaration : 0 encoding-id = 0
a=FEC-declaration : 0 encoding-id = 128 ; instance-id = 0
m=application 4001 FLUTE/UDP 0
c=IN IP6 FF1E:03AD:7F2E:172A:1E24
a=FEC:0
m=application 4002 FLUTE/UDP 0
c=IN IP6 FF1E:03AD:7F2E:172A:1E25
a=FEC:1
```

Source address is an IP6 address=2001:210:1:2:240:96FF:FE25:8EC9 together with session ID=1 uniquely identify the session.

3.2.7.2 File Delivery Table (FDT) Description

We have already explained FDT. Here we will explain how it is described. FDT is an XML file that contains file description entries. During the session it is sent as one ore more FDT instances, each of which describes at least one, at most the complete FDT (all files). The XML code below specifies the XML Schema for FDT Instance:

```
   <?xml version="1.0" encoding="UTF-8"?>
  <xs:schema
      xmlns="urn:IETF:metadata:2005:FLUTE:FDT"
      xmlns:xs="http://www.w3.org/2001/XMLSchema"
      targetNamespace="urn:IETF:metadata:2005:FLUTE:FDT"
      elementFormDefault="qualified">
      <xs:element name="FDT-Instance" type="FDT-InstanceType"/>
      <xs:complexType name="FDT-InstanceType">
            <xs:sequence>
                  <xs:element name="File" type="FileType"
maxOccurs="unbounded"/>
                  <xs:any namespace="##other" processContents="skip"
minOccurs="0" maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:attribute name="Expires" type="xs:string"
use="required"/>
            <xs:attribute name="Complete" type="xs:boolean"
use="optional"/>
            <xs:attribute name="Content-Type" type="xs:string"
use="optional"/>
            <xs:attribute name="Content-Encoding" type="xs:string"
use="optional"/>
            <xs:attribute name="FEC-OTI-FEC-Encoding-ID"
type="xs:unsignedLong" use="optional"/>
            <xs:attribute name="FEC-OTI-FEC-Instance-ID"
type="xs:unsignedLong" use="optional"/>
            <xs:attribute name="FEC-OTI-Maximum-Source-Block-Length"
type="xs:unsignedLong"
                  use="optional"/>
            <xs:attribute name="FEC-OTI-Encoding-Symbol-Length"
type="xs:unsignedLong" use="optional"/>
            <xs:attribute name="FEC-OTI-Max-Number-of-Encoding-
Symbols" type="xs:unsignedLong"
                  use="optional"/>
            <xs:attribute name="FEC-OTI-Scheme-Specific-Info"
type="xs:base64Binary" use="optional"/>
            <xs:anyAttribute processContents="skip"/>
      </xs:complexType>
      <xs:complexType name="FileType">
            <xs:sequence>
                  <xs:any namespace="##other" processContents="skip"
minOccurs="0" maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:attribute name="Content-Location" type="xs:anyURI"
use="required"/>
            <xs:attribute name="TOI" type="xs:positiveInteger"
use="required"/>
            <xs:attribute name="Content-Length" type="xs:unsignedLong"
use="optional"/>
            <xs:attribute name="Transfer-Length"
type="xs:unsignedLong" use="optional"/>
            <xs:attribute name="Content-Type" type="xs:string"
use="optional"/>
            <xs:attribute name="Content-Encoding" type="xs:string"
use="optional"/>
            <xs:attribute name="Content-MD5" type="xs:base64Binary"
use="optional"/>
            <xs:attribute name="FEC-OTI-FEC-Encoding-ID"
type="xs:unsignedLong" use="optional"/>
```

```
            <xs:attribute name="FEC-OTI-FEC-Instance-ID"
type="xs:unsignedLong" use="optional"/>
            <xs:attribute name="FEC-OTI-Maximum-Source-Block-Length"
type="xs:unsignedLong"
                  use="optional"/>
            <xs:attribute name="FEC-OTI-Encoding-Symbol-Length"
type="xs:unsignedLong" use="optional"/>
            <xs:attribute name="FEC-OTI-Max-Number-of-Encoding-
Symbols" type="xs:unsignedLong"
                  use="optional"/>
            <xs:attribute name="FEC-OTI-Scheme-Specific-Info"
type="xs:base64Binary" use="optional"/>
            <xs:anyAttribute processContents="skip"/>
      </xs:complexType>
  </xs:schema>
```

In the schema, there is a sing root element “FDT-Instance”. The “FDT-Element” must contain “Expires” attribute that indicates expiry time of the FDT Instance. The “File” element defined in a “sequence” structure means, one or more files can be described and each can be parsed with “File” element name. Each “File” element has same property definitions such as TOI, Content-location, Content-type,etc, which are explained before. After the “sequence” definition, there are a group of attributes that apply to the complete instance, in the document, we called it as “instance-level” attributes, for the file element attributes, we used the term “object-level” attributes. “Expires” attribute is an “instance-level” attribute because it characterizes the complete FDT Instance.

One example of the FDT based on the XML schema above is given below where two files “track1.mp3” and “track2.mp3” are described;

```
<?xml version="1.0" encoding="iso-8859-1"?>
<FDT-Instance Expires="1197446400"
FEC-OTI-FEC-Encoding-ID="129"
FEC-OTI-FEC-Instance-ID="0"
FEC-OTI-Maximum-Source-Block-Length="180"
FEC-OTI-Max-Number-of-Encoding-Symbols="246"
FEC-OTI-Encoding-Symbol-Length="948">
<File TOI="1"
Content-Location="file:///files/track1.mp3"
Content-Length="101888“
Content-Type=“audio/mp3”
Content-MD5=“Eth76GIkJU45sghK”/>
<File TOI="2"
Content-Location="file:///files/track2.mp3"
Content-Length="101888“
Content-Type=“audio/mp3”
```

```
Content-Encoding="gzip"
Transfer-Length="68552"/>
</FDT-Instance>
```

As the FDT instance shows if all the files share a group of file attribute values, they can be put into instance-level means applies to all files. FEC OTI parameters are described as instance-level. That is both mp3 files are protected with the same FEC OTI parameters. Content-protection with "md5" is applied to "track1.mp3" and content-compression with "gzip" is applied to "track2.mp3".

3.2.8 Service Delivery Model

FLUTE can support several service delivery models. Two examples of them are Push Service Model and On-Demand Service Delivery Model. A push model is a sender initiated concurrent delivery of objects to a selected set of receivers. One way a push service model can be used for reliable content delivery is to deliver a series of objects. The sender could send a Session Description announcement to a control channel and receivers could monitor this channel and join a session whenever a Session Description of interest arrives. Upon receipt of the Session Description, each receiver could join the session to receive packets until enough packets have arrived to reconstruct the object, at which point the receiver could report back to the sender that its reception was completed successfully. The sender could decide to continue sending packets for the object to the session until all receivers have reported successful reconstruction or until some other condition has been satisfied. (Luby M., Gemmell J. & Others, 2002).

For an on-demand content delivery service model, senders typically transmit for some given time period selected to be long enough to allow all the intended receivers to join the session and recover a single object. For example a popular software update might be transmitted using ALC for several days, even though a receiver may be able to complete the download in one hour total of connection time, perhaps spread over several intervals of time. In this case the receivers join the session at any point in time when it is active. Receivers leave the session when they have received enough packets to recover the object. The receivers, for example,

obtain a Session Description by contacting a web server (Luby M., Gemmell J. & Others, 2002).

There may be other reliable content delivery service models that can be supported by ALC. The description of the potential applications, the appropriate delivery service model, and the additional mechanisms to support such functionalities when combined with ALC is beyond the scope of this document. (Luby M., Gemmell J. & Others, 2002).

CHAPTER FOUR

MBMS DOWNLOAD SERVICE

4.1 MBMS Download Service

MBMS download delivery method uses the FLUTE protocol that is described in Chapter 3. However MBMS do not use all features of FLUTE, for example it uses a single channel for a download session, where in FLUTE number of channels in the session is limited but not restricted. So following sections will exhibit how MBMS selects capabilities and features of FLUTE protocol specification. A summary of MBMS download in MBMS Release 7 specification are extracted from 3GPP TS 26.346 (2007) and 3GPP TS 26.946 (2007). For up-to-date information check the current state of MBMS specification (3GPP TS 26.346 & 26.946).

4.1.1 MBMS Service Descriptions

4.1.1.1 Service Discovery/Announcement

Service Discovery / Announcement provide service description information, which may be delivered via the Session and Transmission function or via the Interactive Announcement function in MBMS. Service discovery is a mechanism for subscribers to get a list of service descriptions while service announcement is a mechanism for the sender to announce the list of service descriptions to subscribers. The latter is sender-initiated while the former is receiver-initiated process.

MBMS download delivery method or MBMS Interactive announcement function can be used to deliver service descriptions to subscribers. MBMS Interactive announcement functions provide service descriptions to the UE using HTTP. Other user service announcement and discovery mechanisms by other means than MBMS defined are also possible, for example SAP, SIP, email, SMS Cell Broadcast and so on.

4.1.1.2 Service Descriptions

Service descriptions are sets of metadata that describe sufficient number of properties of a service to be initiated by the receiver and sender. Service descriptions can be in the form of XML-based or SDP based as well as HTTP/MIME headers.

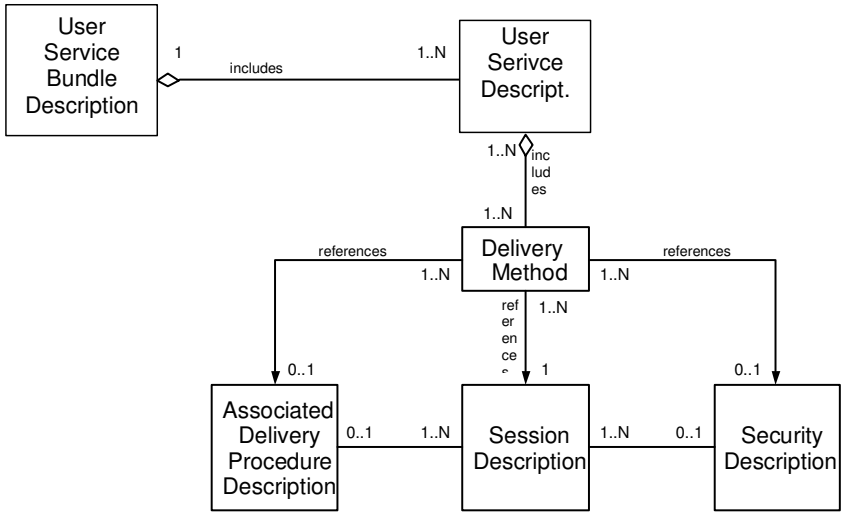


Figure 4.1 MBMS download delivery service description fragments (3GPP TSG 26.346, 2007).

A Service is described by one or more descriptive data, each of which is an identifiable block of metadata, called fragment. Each fragment describes some aspect of the service and it can be discriminated with identifiers. Figure 4.1 shows these service description fragments. Session description fragment is an SDP file that is mandatory and describes the session parameters. Others are optional XML files that describe details of service protection and associated delivery procedure. Three metadata fragments: *Associated delivery description, session description* and *security description* are referenced in a *user service description* fragment, which is another XML file.

One or more services may be bundled together. In this case the root fragment, *user service bundle description* will encompass all service descriptions in separate *userServiceDescription* section in XML as shown in the user service description XML example below. All services that form a bundle are described with local or remote

references of other fragments. UserServiceDescription section must include at least one delivery method, at most many. XML Schema definition of *user service bundle description* is given in (3GPP TSG 26.346, 2007, p.82).

The MBMS Release 7 schema extension provides new attributes for UEs starting and terminating behaviour of the service. An *initiationRandomization* element and *terminationRandomization* element carries the parameters to be used by the MBMS UE to randomize their initiation and/or termination operations over time. If the *initiationRandomization* element is present, all MBMS UEs shall randomize the inititation time as defined by the attributes of the elements. If the *terminationRandomization* element is present, all MBMS UEs shall randomize the termination time as defined by the attributes of the elements. (3GPP TSG 26.346, 2007, p.83).

```
  <?xml version="1.0" encoding="UTF-8"?>
  <bundleDescription
      xmlns="www.example.com/3gppUserServiceDescription"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

  <userServiceDescription
            serviceId="urn:3gpp:1234567890coolcat">
                  <name lang="EN">Welcome</name>
                  <name lang="DE">Willkommen</name>
            <name lang="FR">Bienvenue</name>
                  <name lang="FI">Tervetuloa</name>
                  <serviceLanguage>EN</serviceLanguage>
                  <serviceLanguage>DE</serviceLanguage>
                  <deliveryMethod
               accessGroupId="1"

      sessionDescriptionURI="http://www.example.com/3gpp/mbms/session1
.sdp"/>

  <deliveryMethod
      sessionDescriptionURI="http://www.example.com/3gpp/mbms/session2
.sdp"
      associatedProcedureDescriptionURI=
      "http://www.example.com/3gpp/mbms/procedureX.xml"/>

  <deliveryMethod
      sessionDescriptionURI="http://www.example.com/3gpp/mbms/session3
.sdp"
      associatedProcedureDescriptionURI=
                  "http://www.example.com/3gpp/mbms/procedureY.xml"/>

  <deliveryMethod
```

```
      accessGroupId="2"

       sessionDescriptionURI="http://www.example.com/3gpp/mbms/session4
.sdp"/>

      <accessGroup id="1">
               <accessBearer>3GPP.R6.GERAN</accessBearer>
               <accessBearer>3GPP.R6.UTRAN</accessBearer>
      </accessGroup>
      <accessGroup id="2">
               <accessBearer>3GPP.R6.UTRAN</accessBearer>
      </accessGroup>

       </userServiceDescription>
  </bundleDescription>
```

Above XML code provides an example of MBMS Service description. There is one *userServiceDescription* section; it means the bundle describes only one service. Each service has a name and identity so that it can be identified among other services. *Name* and *ServiceID* attributes are used in this purpose. *DeliveryMethod* specifies either streaming or downlading method used. A single session can be either a streaming session or a downloading session. Within the *DeliveryMethod* section, at least *sessionDescripitonURI* must be provided, and possibly other fragments *AssociatedProcedureDescriptionURI* and *ProtectionDescriptionURI* can be provided. Several Delivery Methods may refer to same *AssociatedProcedureDescriptionURI* or same *ProtectionDescriptionURI*.

Each service can provided for some radio access network such as UTRAN or GERAN. Furthermore, releases of RAN can be specified so that a client within the specified release of RAN can use the service. This feature is described by a *AccessGroup* and *AccessGroupID*. The session described by session4.sdp is provided only for 3GPP.R6.UTRAN. It means a client in Release 5 UTRAN will not be able to use the service.

4.1.1.3 Security Descriptions

The security description is referenced by the *protectionDescriptionURI* of the deliveryMethod element. A FLUTE channel can be protected using key management mechanism. A client needs to know some server addresses that managing key

distribution. Each channel in the session is mapped to a specific key ID. Since MBMS FLUTE uses single channel, there should be a single mapping of a key ID to a download channel specified by a multicast address and a port. *KeyId* element in XML does such a mapping as shown in security description example below. Requesting for actual keys that will be used for protection and their distribution by the key server, will be using MIKEY packets over point to point bearers on UDP or point to multipoint on UDP (not over FLUTE). When point to multipoint bearers are used, FEC protection can applied to MIKEY distribution channel. Using FEC requires *FEC Encoding ID*, *FEC Instance ID* and *FEC Object Transmission Information* specific to FEC Encoding ID. They are provided within *fecProtection* element in the XML. MBMS Service Protection Description Example (3GPP TSG 26.346, 2007) is given below.

```
<?xml version="1.0" encoding="UTF-8"?>
<securityDescription
    xmlns="www.example.com/3gppSecurityDescription"
    xmlns:xs="http://www.w3.org/2001/XMLSchema-instance"
    confidentialityProtection="true"
    integrityProtection="true"
    uiccKeyManagement="true">
    <keyManagement
          offsetTime="5"
          maxBackOff="10">
          <serverURI ="http://register.operator.umts/"/>
          <serverURI ="http:// register2.operator.umts/"/>
    </keyManagement>
    <keyId identity="<someMSKidA>" mediaFlow=224.1.2.3:4002 />
    <fecProtection
          fecEncodingId="130"
          fecInstanceId="0"
          fecOtiExtension="1SCxWEMNe397m24SwgyRhg==">/>
</securityDescription>
```

Both MBMS client and server request a secret key from the key servers, which are available in the specified addresses within *keyManagement* element, using same initial keys given in the security description. Since many receivers or servers may request key materials from key servers at the same time, key management procedure involves some kind of randomization that is managed by *offsetTime* and *maxBackOff*. Key servers respond them with a secret key using MIKEY packets, with which a single download channel can be protected later. (3GPP TSG 26.346, 2007)

4.1.1.4 Associated Delivery Procedure Descriptions

The Associated Delivery Procedure Descriptions is referenced by the *AssociatedProcedureDescriptionURI* of the *deliveryMethod* element. An example of MBMS Associated Delivery Description (3GPP TSG 26.346, 2007) is given below. Associated Delivery Procedure for a FLUTE session mostly means file repair procedure as well as reception reporting option. Reception reporting procedure is used to report the complete reception of one or more files. By its nature, FLUTE may not provide a complete transmission of files to a mass. Some of the files may have corrupted or lost blocks. If a receiver determines that some files are not completely received and file repair option is also available in the description, specified by *postFileRepair* element in the XML, then it starts Associated Delivery Procedure.

```
<?xml version="1.0" encoding="UTF-8"?>
<associatedProcedureDescription
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="http://www.example.com/mbms-associated-
descrition.xsd">
   <postFileRepair
          offsetTime="5"
          maxBackOff="10">
      <serverURI>http://mbmsrepair.operator.umts/"</serverURI>
      <serverURI>http://mbmsrepair1.operator.umts/"</serverURI>
      <serverURI>http://mbmsrepair2.operator.umts/"</serverURI>
   </postFileRepair>
   <bmFileRepair
sessionDescriptionURI="http://www.example.com/3gpp/mbms/session1.sdp"/
>

</associatedProcedureDescription>
```

Since many clients may request file repair procedure from repair servers about same time, some a mechanism is needed to prevent repair servers from being bottleneck. The *offsetTime* and *maxBackOff* are used in this purpose. The MBMS client Calculates a random *back-off time* and selects a file repair server randomly out of a list, specified in *postFileRepair* element and sends a *repair request* message to the selected file repair server at the calculated time.

If file repair option is available, MBMS client should wait for repair data in the defined MBMS download session, possibly current session or a different session

defined in *bmFileRepair* element. The existence of a broadcast/multicast file repair session is signalled by the inclusion of the optional *bmFileRepair* procedure in the updated Associated Delivery procedure description. This is signalled by the *bmFileRepair* element with a single "sessionDescriptionURI" attribute. In the cases where the same IP addressing is used for the broadcast/multicast repair session as the original download session, the MBMS client simply not leaves the group. Otherwise, the client must join to defined session using the specified SDP. A broadcast/multicast file repair session behaves just as an MBMS download session, and the determination of end of files and session, and use of further associated delivery procedures uses the same techniques as specified for the MBMS download delivery method. Then the file repair server responds with a *repair response* message containing the repair data, or at worst, describing an error case or some other alternative case.

4.2 MBMS Service Description Transport

Service Descriptions are transported using association of metadata envelope and metadata fragment, both of which are XML coded objects. Envelope is a metadata that give high-level description of its associated fragment so that it manages transport of service descriptions independent of the fragment syntax. “The metadata envelope and metadata fragment objects are transported as file objects in the same download session either as separate referencing files or as a single embedding file.” (3GPP TSG 26.346, 2007). A metadata envelope shall be associated with a metadata fragment by one of two methods:

1. Embedded: The metadata fragment is embedded within the metadata envelope.
2. Referenced: The metadata fragment is referenced from the metadata envelope.

The attributes for a metadata envelope and their description defined in 3GPP TSG 26.346 (2007) as follows:

- *metadataURI*: A URI providing a unique identifier for the metadata fragment. The *metadataURI* attribute is obligatory as indicated in the metadata envope schema below.

- *version*: The version number of the associated instance of the metadata fragment. The version number should be initialized to one. The version number must be increased by one whenever the metadata fragment is updated. The *version* attribute is used for this purpose and it is obligatory.
- *validFrom*: The date and time from which the metadata fragment file is valid. The *validFrom* attribute may or not be present. If not present, the UE should assume the metadata fragment version is valid immediately.
- *validUntil*: The date and time when the metadata fragment file expires. The *validUntil* attribute may or not be present. If not present the UE should assume the associated metadata fragment is valid for all time, or until it receives a newer metadata envelope for the same metadata fragment describing a validUntil value.
- *contentType*: The MIME type of the metadata fragment which shall be used as defined for "Content-Type" in RFC 2616. The *contentType* attribute shall be present for embedding metadata envelopes. The *contentType* attribute may be present for referencing metadata envelopes. For example, security description fragment uses application/mbms-protection-description.

The formal schema for the MBMS metadata envelope is defined as an XML Schema as follows (3GPP TSG 26.346, 2007);

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    elementFormDefault="qualified"
    attributeFormDefault="unqualified">
    <xs:element name="metadataEnvelope" type="metadataEnvelopeType"
minOccurs="1"
                                maxOccurs="unbounded"/>
<xs:complexType name="metadataEnvelopeType">
          <xs:sequence>
               <xs:element name="metadataFragment"
                                type="xs:string"
                                minOccurs="0"
                                maxOccurs="1">
               </xs:element>
          </xs:sequence>
          <xs:attribute name="metadataURI"
                                type="xs:anyURI"
                                use="required"/>
          <xs:attribute name="version"
                                type="xs:positiveInteger"
                                use="required"/>
```

```
            <xs:attribute name="validFrom"
                                    type="xs:dateTime"
                                    use="optional"/>
            <xs:attribute name="validUntil"
                                    type="xs:dateTime"
                                    use="optional"/>
            <xs:attribute name="contentType"
                                    type="xs:string"
                                    use="optional"/>
            <xs:anyAttribute processContents="skip"/>

    </xs:element>
</xs:schema>
```

4.3 Congestion Control

MBMS uses single FLUTE channel with single rate transport. No congestion control algorithms are needed in this case.

4.4 Content Encoding of Files

Files may be content encoded for transport, using the generic GZip algorithm described by Deutsch P. (May 1996). UEs shall support GZip content decoding of FLUTE files. However content encoding of FDT is not used in MBMS. That is, FDT Instances are not content encoded hence FDT_CENC extension header is not used at all in MBMS.

4.5 Signaling of Parameters

MBMS Download parameters are carried in FLUTE Headers including Flute Extension Headers and FDT Instances.

4.5.1 Flute Mandatory Headers (3GPP TSG 26.346, 2007)

FLUTE and ALC mandatory header fields are explained already in detailed FLUTE section. MBMS adds following additional specializations:

- The length of the CCI (Congestion Control Identifier) field shall be 32 bits and it is assigned a value of zero (C=0).

- The Transmission Session Identifier (TSI) field will be of length 16 bits (S=0, H=1, 16 bits).
- The Transport Object Identifier (TOI) field should be of length 16 bits (O=0, H=1).
- Only Transport Object Identifier (TOI) 0 (zero) will be used for FDT Instances.
- The following features may be used for signalling the end of session and end of object transmission to the receiver:

  The Close Session flag (A) for indicating the end of a session.

  The Close Object flag (B) for indicating the end of an object.
- The T flag indicates the use of the optional "Sender Current Time (SCT)" field (when T=1).
- The R flag indicates the use of the optional "Expected Residual Time (ERT)" field (when R=1).
- The LCT header length (HDR_LEN) shall be set to the total length of the LCT header in units of 32-bit words.
- For "Compact No-Code FEC scheme"], the FEC Payload ID shall be such that a 16 bit SBN (Source Block Number) and then the 16 bit ESI (Encoding Symbol ID) are given.
- For “MBMS FEC scheme”, the FEC Payload ID shall be set according to MBMS FEC Scheme definition below.

4.5.2 Flute Extension Headers (3GPP TSG 26.346, 2007)

MBMS uses FLUTE extension header fields EXT_FDT, EXT_FTI , EXT_CENC as follows:

- EXT_FTI must be included in every FLUTE packet carrying symbols belonging to any FDT Instance.
- FLUTE packets carrying symbols of files (not FDT Instances) shall not include an EXT_FTI.
- FDT Instances shall not be content encoded and therefore EXT_CENC shall not be used.

- EXT_FDT is in every FLUTE packet carrying symbols belonging to any FDT Instance.
- FLUTE packets carrying symbols of files (not FDT instances) do not include the EXT_FDT.

4.5.3 FDT Instances

MBMS uses FDT Instances elements as follows described in detail in 3GPP TSG 26.346 (2007):

- Content-Location (URI of a file)
- TOI (Transport Object Identifier of a file instance)
- Expires (expiry data for the FDT Instance)
- Content-Length (source file length in bytes)
- Content-Type (content MIME type)
- FEC Encoding ID

Other FEC Object Transmission Information specified by the FEC scheme are:

- FEC-OTI-Maximum-Source-Block-Length
- FEC-OTI-Encoding-Symbol-Length
- FEC-OTI-Max-Number-of-Encoding-Symbols
- FEC-OTI-Scheme-Specific-Info

Following FDT attributes are optinal in MBMS:

- Complete
- Content-Encoding
- Content-MD5

“The FEC-OTI-Scheme-Specific-Info FDT Instance data element contains information specific to the FEC scheme indicated by the FEC Encoding ID encoded using base64.” (3GPP TSG 26.346, 2007).

4.6 FDT Schema

MBMS uses IETF FLUTE FDT Schema as well as 3GPP FDT extensions as defined in (3GPP TSG 26.346, 2007, p.35, p.37) for FDT Instances. The extension of the IETF FLUTE FDT schema is done using the following schema definition (3GPP TSG 26.346, 2007, p.35):

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
    xmlns="urn:3GPP:metadata:2005:MBMS:FLUTE:FDT"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    targetNamespace="urn:3GPP:metadata:2005:MBMS:FLUTE:FDT"
    elementFormDefault="qualified">
    <xs:complexType name="MBMS-Session-Identity-Expiry-Type">
          <xs:simpleContent>
                <xs:extension base="MBMS-Session-Identity-Type">
                      <xs:attribute name="value"
type="xs:unsignedInt" use="required"/>
                </xs:extension>
          </xs:simpleContent>
    </xs:complexType>
    <xs:simpleType name="MBMS-Session-Identity-Type">
          <xs:restriction base="xs:unsignedByte"/>
    </xs:simpleType>
    <xs:simpleType name="groupIdType">
          <xs:restriction base="xs:string"></xs:restriction>
    </xs:simpleType>
</xs:schema>
```

Most of the data elements in the FDT instances are described previously in Chapter 3. FEC related parameters in IETF FDT Schema will be described in following sections. Grouping attribute in FDT Schema is used to logically group one or more files so that downloading of the one in the group automatically triggers other files in the same group. Most of the time files downloaded are related to each other. For example, downloading a web page component that has many references to other components, software packages, and the referencing metadata envelopes and their metadata fragments are related. A FLUTE receiver should download all the files belonging to all groups where one or more of the files of those groups have been requested. However, a UE may instruct its FLUTE receiver to ignore grouping to deal with special circumstances, such as low storage availability. The usage of the MBMS Session Identity is optional. Each MBMS session may be activated using a different MBMS

session identifier. The MBMS UE determines, based on the MBMS Session Identity value, whether the files of the upcoming MBMS download session were already received. If the files have already been completely received, the MBMS UE does not respond to the notification of the MBMS Session (3GPP TSG 26.346, 2007).

The following schema (3GPP TSG 26.346, 2007, p.37) is given as a 3GPP extension of FDT Schema, new in MBMS Release 7, it defines new elements;

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
    xmlns="urn:3GPP:metadata:2007:MBMS:FLUTE:FDT"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    targetNamespace="urn:3GPP:metadata:2007:MBMS:FLUTE:FDT"
    elementFormDefault="qualified">

    <xs:element name="Cache-Control">
          <xs:complexType>
                <xs:attribute name="no-cache" use="optional"
type="xs:boolean"/>
                <xs:attribute name="max-stale" use="optional"
type="xs:boolean"/>
                <xs:attribute name="Expires" use="optional"
type="xs:unsignedInt"/>
                <xs:anyAttribute processContents="skip"/>
          </xs:complexType>
    </xs:element>
</xs:schema>
```

A file download service may indicate the caching recommendations for a specific file or set of files that are delivered using FLUTE. The caching functionality defines three different caching directives:

- no-cache: this directive is used to indicate to the receiver not to cache a specific file (or set of files). This is probably useful in the case where the file is expected to be highly dynamic (changes to the file occur quite often) or if the file will be used only once by the receiver application.
- max-stale: this directive indicates to the FLUTE receiver that a specific file (or set of files) should be cached for an indefinite period of time, if possible. The file has no expiry date.

- Expires: this directive is used by the server to indicate the expected expiry time of a specific file (or set of files). It indicates a date and time value in the HTTP date format or in the NTP timestamp format. (3GPP TSG 26.346, 2007, p.38)

4.7 MBMS FEC Scheme Definition

MBMS uses Raptor Encoding algorithm which is a fully-specified FEC Scheme and uses FEC Encoding ID=0, and NULL FEC Encoding Algorithm which uses FEC Encoding ID=0.

4.7.1 FEC Payload ID

Figure 4.2 provides the FEC Payload ID format and its place in FLUTE packet. FEC payload ID is a tuple of Source Block Number and Encoding Symbol ID. It specifies which source block packet belongs to and enables differentiation among encoding symbols by Encoding Symbol ID.

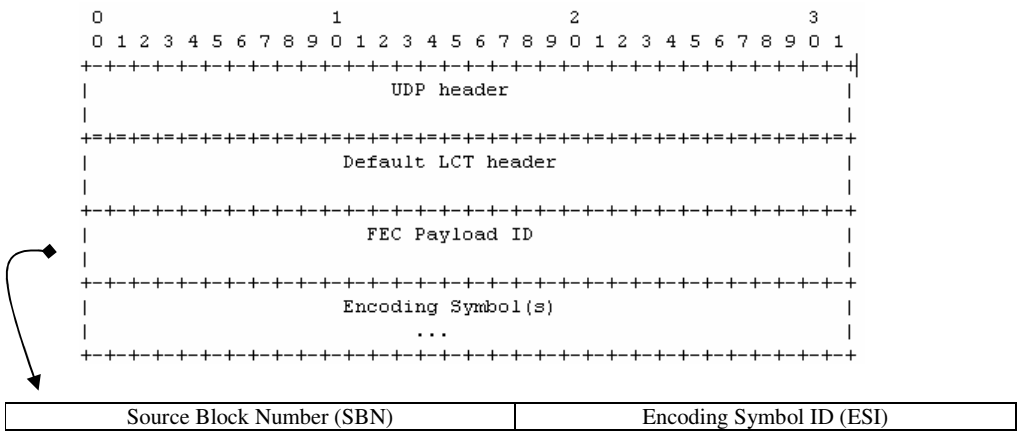


Figure 4.2 FEC payload ID in MBMS FLUTE.

"The FEC Payload ID shall be a 4 octet field defined as follows:

- Source Block Number (SBN), (16 bits): An integer identifier for the source block that the encoding symbols within the packet relate to.

- Encoding Symbol ID (ESI), (16 bits): Starting index of the first encoding symbol within a group of consecutive encoding symbols in Encoding Block." (3GPP TSG 26.346, 2007, p.38).

4.7.2 FEC Object Transmission Information (FEC OTI)

MBSM FEC Object Transmission information consists of:

- The FEC Encoding ID=0, FEC Instance ID=0 are used to define raptor encoding
- The Transfer Length (*F*), is the size of object that is being transferred
- The parameters *T*, *Z*, *N* and *A*

  T- Encoding Symbol Length

  Z- Number of source blocks

  N- Number of sub-blocks, when a source block size cannot be fit into FEC buffer, sub- blocks are used

  A- Symbol Alignment parameter ensures that symbols in a source block and sub-symbols contained in a sub-block are multiple of Alignments (A) in size.

There are two ways to communicate FEC OTI: FDT Instances or Session Description Protocol (SDP). Session Description provides FEC OTI at session level for MBMS. It means all objects can be sent using one FEC OTI configuration in SDP. However, FDT can provide FEC OTI at object-level. It means each object can be sent using different FEC OTI configuration and in that case any previously known configuration for that object will be overwritten.

General EXT_FTI format	Encoding Symbol Length (*T*)	
Number of Source Blocks (*Z*)	Number of Sub-Blocks (*N*)	Symbol Alignment Parameter (*A*)

Figure 4.3 MBMS FEC specific EXT_FTI format.

When FDT is used to communicate FEC OTI, the FEC Encoding ID will be carried in CodePoint (CP) portion of Flute packet (Figure 3.14), FEC Instance ID and Transfer

Length will be communicated in General EXT_FTI (Figure 3.14). The other parameters shall be encoded in the FEC Encoding ID specific portion of the EXT_FTI field as shown in Figure 4.3 below.

The parameters *T* and *Z* are 16 bit unsigned integers, *N* and *A* are 8 bit unsigned integers. The remaining parameters *Z*, *N*, *A*, shall be encoded as a 4 byte field within the FEC-Scheme Specific format field.

4.8 MBMS Fragmentations

4.8.1 Fragmentation of Files

Fragmentation is a mechanism that decides how to divide a file into partitions in sender side, as well as decides how to augment these partitions to the original file in receiver side. It requires two steps; First is the partitioning a file into source blocks, possibly, further dividing source blocks into sub-blocks. Second is the partitioning of a block into encoding symbols.

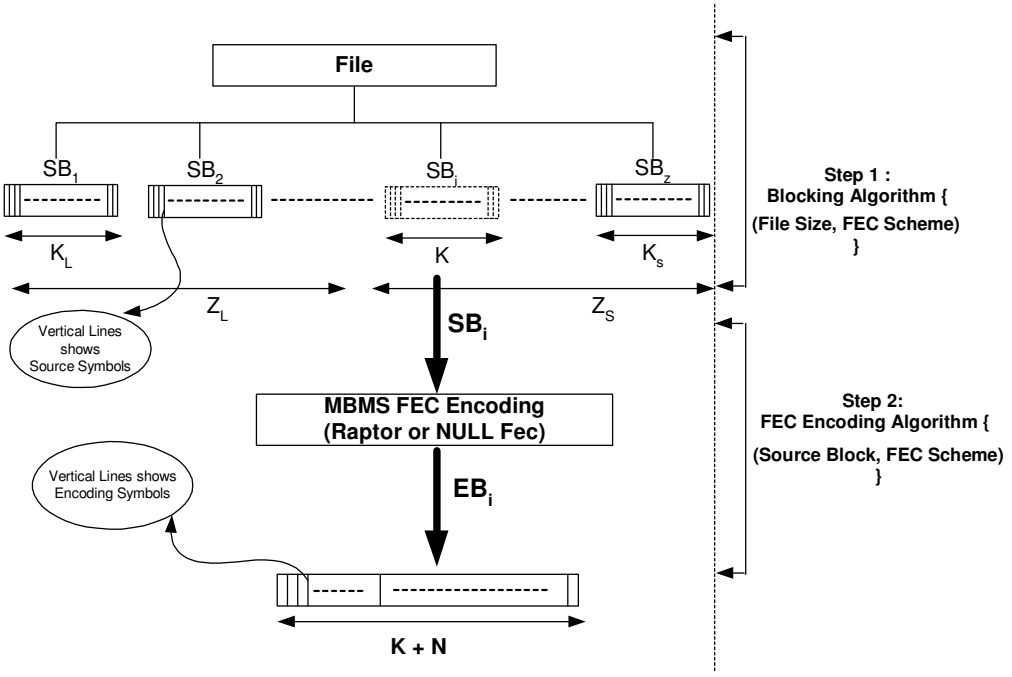


Figure 4.4 MBMS fragmentation.

Step-1 will be provided by blocking algorithms, step-2 will be provided by FEC encoding algorithms. There are two blocking algorithms; one for Compact No-Code FEC scheme that is described in FLUTE specification (Luby M. & Others, 2002), and one for Raptor-Encoding FEC scheme that will be described in later section. As Figure 4.4 shows both blocking algorithms and encoding algorithms needs FEC-encoding scheme as input parameter. It means partitioning criteria will be described by FEC Scheme and file size. While encoding criteria is defined by only FEC scheme used. Step-1 results in Z number of Source symbols, Z_L of which has K_L number of source symbols, Z_S of which has K_S number of source symbols. Step 2 results in K+N number of encoding symbols, N of which are repair symbols.

4.8.2 Blocking Algorithm

Blocking algorithms is affected by file size, FEC Scheme and some recommendations based on file size and FEC Scheme. Since MBMS uses raptor encoder, definition in Figure 4.4, Blocking_Algorithm (*File Size*, *FEC Scheme*) can be replaced by MBMS_Blocking_Alg (*File Size*, *Recommendations*), where MBMS specific recommendations will be put into a structure called *Recommendations.*

In order to apply the Raptor encoder to a source file, the file may be broken into $Z \geq 1$ blocks, known as *source blocks*. The Raptor encoder is applied independently to each source block. As Figure 4.5 shows each source block is identified by a unique integer Source Block Number (SBN), where the first source block has SBN zero, the second has SBN one, etc. Each source block is divided into a number, *K*, of *source symbols* of size *T* bytes each. Each source symbol is identified by a unique integer Encoding Symbol Identifier (ESI), where the first source symbol of a source block has ESI zero, the second has ESI one, etc.

Each source block with *K* source symbols is divided into $N \geq 1$ sub-blocks, which are small enough to be decoded in the working memory. Each sub-block is divided into *K* sub-symbols of size *T'*. Figure 4.5 shows an example source block placed into a two dimensional array, where each entry is a *T'*-byte sub-symbol, each row is a sub-block and each column is a source symbol. For example, the sub-symbol numbered *K*

contains bytes $T'\cdot K$ through $T'\cdot(K+1)-1$ of the source block. Then, source symbol i is the concatenation of the ith sub-symbol from each of the sub-blocks, which corresponds to the sub-symbols of the source block numbered i, $K+i$, $2\cdot K+i$,…, $(N-1)\cdot K+I$ as shown in Figure 4.5. That is, source symbol i is the concatenation of N number of sub-symbols in column i.

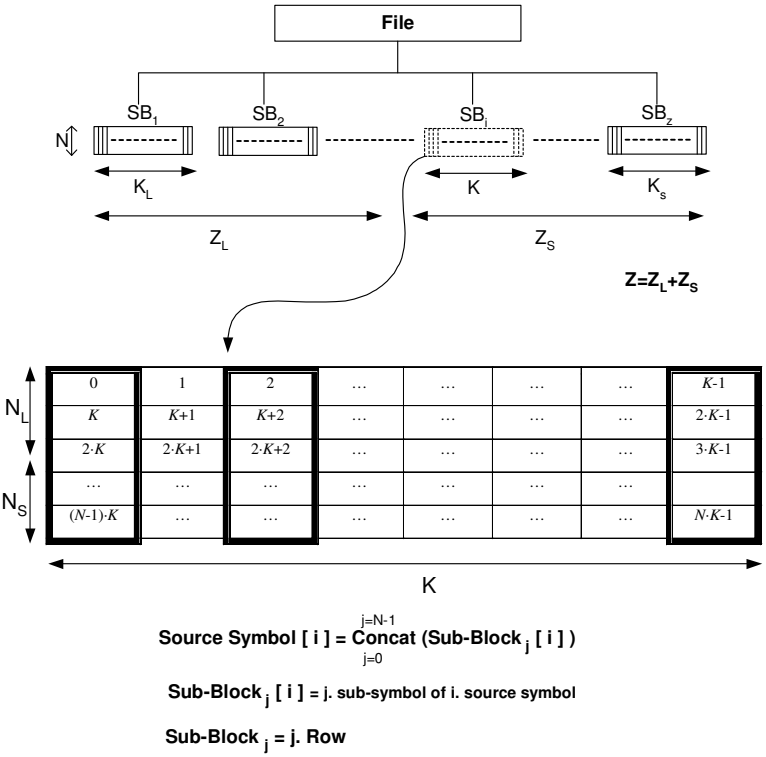


Figure 4.5 Source block construction.

In general, each source block shall be divided into $N = N_L + N_S$ contiguous sub-blocks, the first N_L sub-blocks each consisting of K contiguous sub-symbols of size of $T_L \cdot A$ and the remaining N_S sub-blocks each consisting of K contiguous sub-symbols of size of $T_S \cdot A$. The symbol alignment parameter A ensures that sub-symbols are always a multiple of A bytes.

So blocking algorithm must determine 8 quantities: Z_L, Z_S, K_L, K_S, N_L, N_S, T_L, T_S based on the five input parameters:

- F the size of the file, in bytes
- A a symbol alignment parameter, in bytes
- T the symbol size, in bytes, which must be a multiple of A
- Z the number of source blocks
- N the number of sub-blocks in each source block

These input parameters T, Z, N will be adjusted according to some parameter derivation algorithms that use some recommendations.

```
(I_L, I_S, J_L, J_S) = Partition (I, J)
  Begin
       I_L = ceil (I/J)
      I_S = floor (I/J)
      J_L = I - I_S · J
     J_S = J - J_L
  End;

Recommendatations is Struct
Begin
    W,P,A, K_MAX ,K_MIN ,G_MAX
End

[T,Z,N] = Parameter_Derivation_Alg (F, Recommendatitons)
Begin
      G = min {ceil (P ·K_MIN/F), P/A, G_MAX}
      T = floor (P/ (A·G)) ·A
      K_t = ceil (F/T)

      Z = ceil (K_t /K_MAX)
     N = min{ceil(ceil( K_t/Z)·T/W ), T/A}
End

[(K_L, K_S, Z_L, Z_S), (T_L, T_S, N_L, N_S)] = MBMS_Blocking_Alg (F,
Recommendations)
 Begin
     [T,Z,N]=Parameter_Derivation_Alg (F, Recommendations)
     K_t = ceil(F/T)
     (K_L, K_S, Z_L, Z_S) = Partition[K_t, Z ]
        (T_L, T_S, N_L, N_S) = Partition[T/A, N ]
 End ;
```

Parameter Derivation Algorithm recommends for the derivation of the four transport parameters, A, T, Z and N. This recommendation is based on the following input parameters:

- F the file size, in bytes
- W, a target on the sub-block size, in bytes

- P , the maximum packet payload size, in bytes, which is assumed to be a multiple of A
- A , the symbol alignment factor, in bytes
- K_{MAX}, the maximum number of source symbols per source block.
- K_{MIN}, a minimum target on the number of symbols per source block
- G_{MAX}, a maximum target number of symbols per packet

The values of G and N derived above should be considered as lower bounds. It may be advantageous to increase these values, for example to the nearest power of two. In particular, the above algorithm does not guarantee that the symbol size, T, divides the maximum packet size, P, and so it may not be possible to use the packets of size exactly P. If, instead, G is chosen to be a value which divides P/A, then the symbol size, T, will be a divisor of P and packets of size P can be used.

Recommended settings for the input parameters, W, A, K_{MIN} and G_{MAX} are as follows: W = 256 KB; A = 4; K_{MIN} = 1024; GMAX = 10.

4.9 MBMS Download Flow

Detailed flow diagram, shown in Figure 4.6, for the receiver side is given in (3GPP TSG 26.946, 2007, p.15). Repair procedure and reporting are also showed in detail in the diagram. Diagram identifies four states of the MBMS receiver:

- The ‘Object Reception’ state reflects the state of the MBMS UE, in which the MBMS client is receiving data for any kind of objects, files or FDT instances. TOI identifies the object. If client should receive a FDT Instance related to the current file being downloaded, it should update corresponding information for that file in FDT database. In general, a received FDT Instance results in updating current view of FDT database.
- The ‘defer file repair’ state reflects the state of the MBMS UE, in which the MBMS UE deferring the file repair request message.

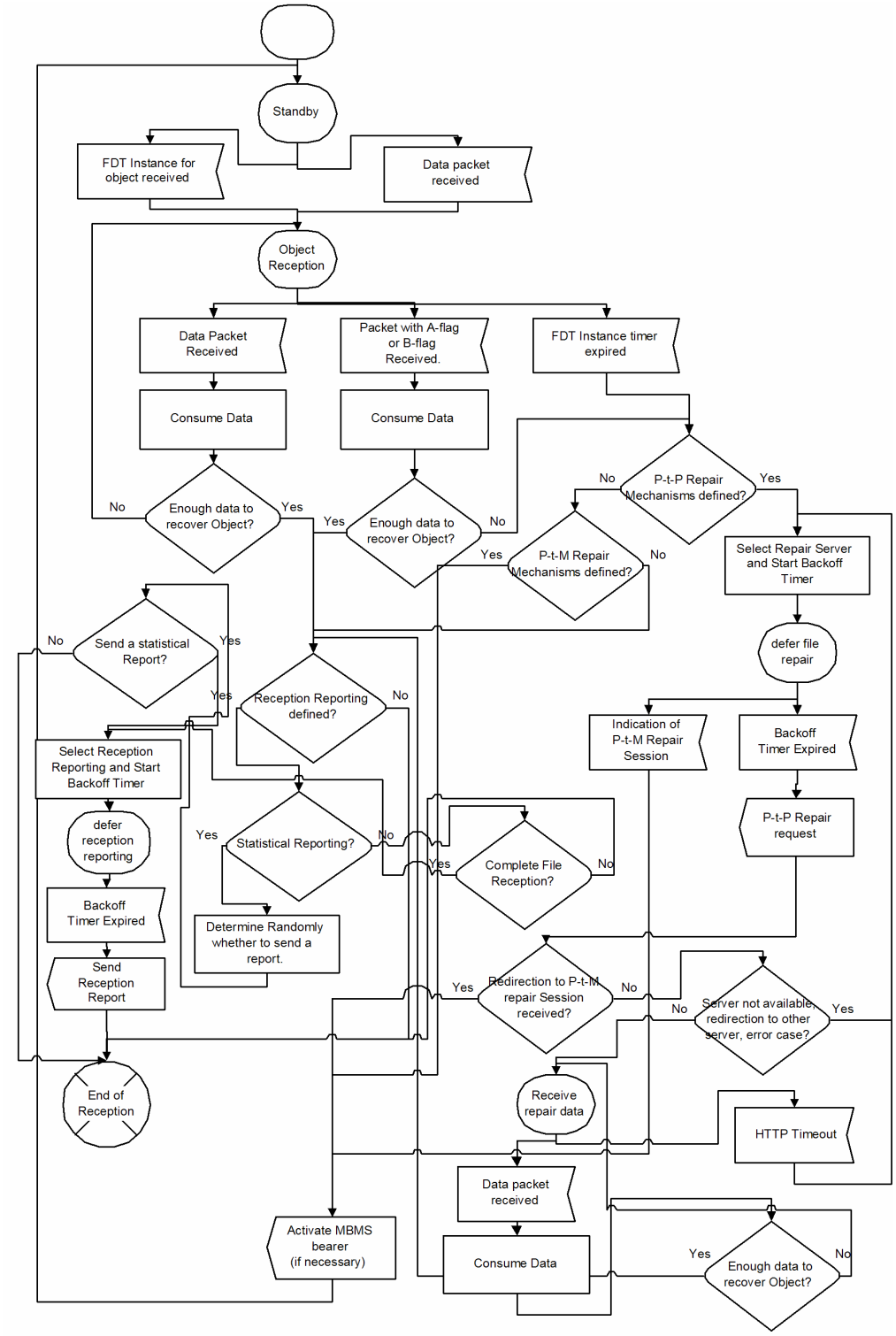


Figure 4.6 Detailed SDL diagram for the file download process of MBMS UE (3GPP TSG 26.946, 2007, p.15).

- The ‘defer reception reporting’ state reflects the state of the MBMS UE, in which the MBMS UE deferring the file repair request message. (TSG 26.946, 2007, p.15).

In stand by state, receiver is ready to receive a packet; this packet should include either FDT instance fragments or file fragments described in a previously received FDT instance. Whenever a new FDT instance is received, a timer is started for those files in that FDT instance. Each timer is possibly a separate thread that triggers FDT instance expire event so that it can decide when to start repair procedure (associated delivery) for the files in that FDT instances. However, a FDT instance may be duplicated, subset or superset of a previously received one. It means, a timer startedfor a file described in FDT instance i, can be reset if the same file (possibly with new TOI or with the old ones) is described in a newly received FDT Instance i+1. So each timer cares one or more of objects for the starting time of repair procedure of those objects.

When a timer expires, client checks which repair options are available if any. If a ptp repair is available, it starts soon a request to the selected server specified in list of the associated procedure description within the current session. Selection of server and contacting time is determined based on some randomization principle as explained before. Depending of the situation of the server and incoming request for the missing parts, client may be redirected to a pmp repair procedure. This case may occur if the requested symbols of files are also requested for many clients. Pmp repair procedure requires either new session establishment (creation of new MBMS multicast bearers) or using the current one.

When session expired or session close flag is received, same procedure above must be executed if any missing symbols in files exist. Whether everything is okey or not, client may request reporting (reception or statistical one) at the end of the repair procedure if any repair is defined and required or at the end of the session if no repair is required.

CHAPTER FIVE

SYSTEM MODELS

This section provides system models of the proposed MBMS download deliveries, which are the legacy download delivery model and the interleaved download delivery model as well as an analytical model to show how we formulate the problem and how we derive equations for Raptor.

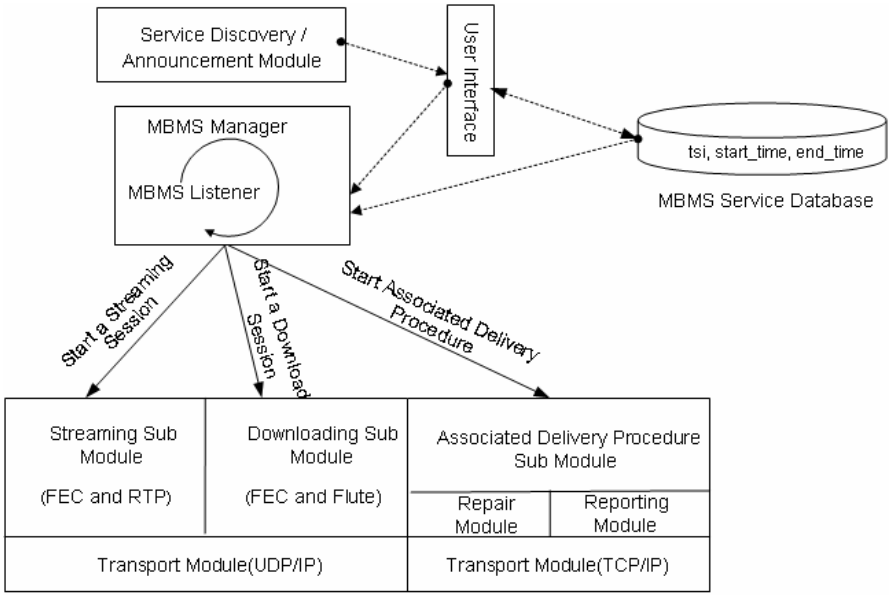


Figure 5.1 Vidiator MBMS prototype software modules.

The legacy download system is based on Vidiator (Vidiator Technology US) MBMS download prototype based on (3GPP TS 26.346, 2007; Digital Fountain, Ericsson & Others, May 2004; Nortel Networks, April 2004), which uses Reed Solomon FEC coding. System architecture of the legacy download is shown in Figure 5.1. Other proposed systems are upgraded from the legacy download system. In each step, Raptor results are derived using the experimental results of the Reed Solomon, our analytical model and the works in Siemens (March 2005) and 3GPP TSG SA WG#4 (June 2005). So for Raptor the results are approximated. To emulate MBMS link conditions we implemented a transmission rate and packet loss control module. The MBMS link

conditions are aligned with Digital Fountain, Ericsson & Others (May 2004), Nortel Networks (April 2004) and 3GPP TSG 26.946 (2007). Mobility issues are discarded, hence group management procedures joining and leaving to multicast group occurs before the session start and session end respectively.

5.1 The Legacy Download Delivery Model

The system model for the legacy download delivery is shown in Figure 5.2. In order to support progressive downloading we assumed that repair symbols are sent just after source symbols.

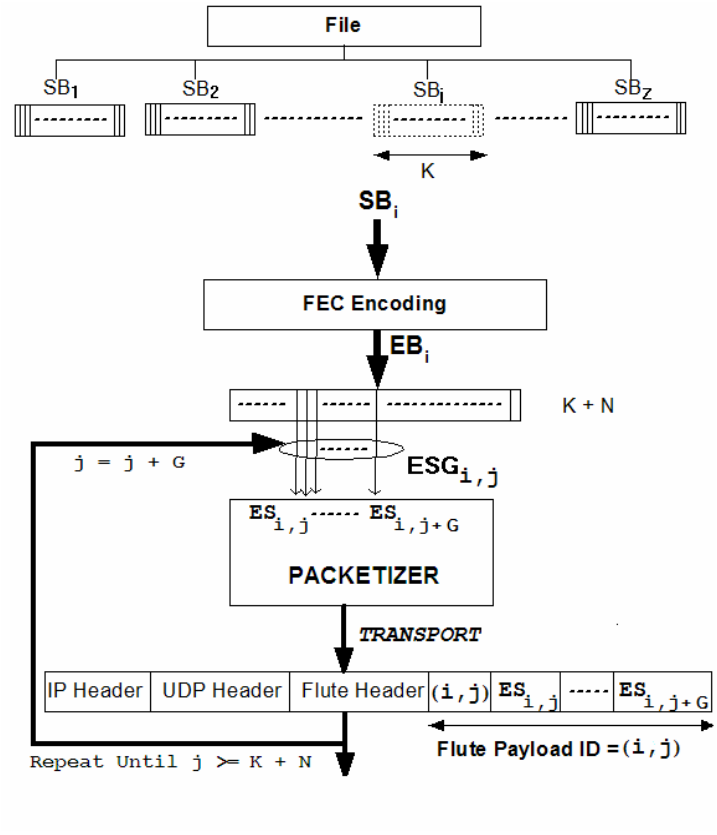


Figure 5.2 System model for the legacy download delivery.

In Figure 5.2, after each SB_i is delivered to FEC layer the result is EB_i that includes *N* encoding symbols (ES). Each encoding symbols is uniquely identified by the couple of its Source Block Number (SBN) and Encoding Symbol ID (ESI). A group of *G*

consecutive encoding symbols (ESG) starting from encoding symbol ID = j for SB_i is denoted as $ESG_{i,j}$ and identified by the couple (SBN,ESI) of the first encoding symbol, here (i , j). The ESGs are packed into FLUTE payload just after the place reserved for FLUTE Payload ID that is assigned to ESG ID, here (i, j) and transported until no more encoding symbols to send.

5.2 The Interleaved Download Delivery Model

The system model for the interleaved download delivery is shown in Figure 5.3. It shows the sender side flow of the download delivery with the SB Interleaving of block-size *b* that we considered in our work.

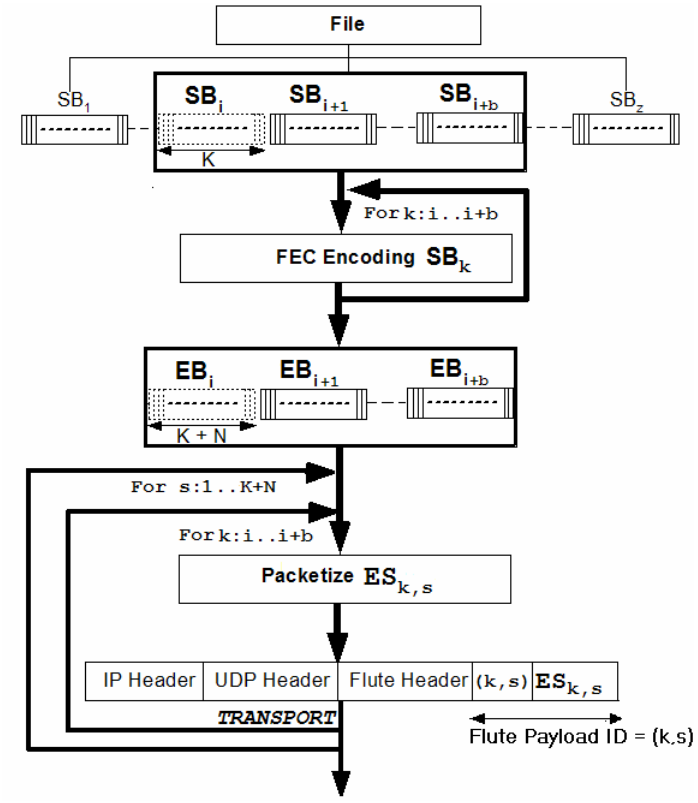


Figure 5.3 System model for the interleaved download delivery.

That is, *b* consecutive SBs constitute an interleaver-block and are sent in parallel in the order of ESIs. All encoding symbols in the interleaver-block with ESI=1 will be sent first, and so on. One more requirement of our interleaving strategy is that each FLUTE

packet must include one encoding symbol at a time. However, this process requires all the *b* EBs to be in memory before they are sent. So the parameter *b* and *SB* size can be used to adapt to different service conditions.

Minimally the interleaver-block size should be two otherwise interleaving cannot be applicable. However, increasing the interleaver-block size consumes much more memory and complicates the cost of the download process. Hopefully in our work we have caught a threshold point upon which increasing the interleaver-block size no longer gives benefit. In our work we have caught best results with *b=3* for MBMS link conditions and under our assumptions. Considering a fixed *b*, we have to adjust SB size accordingly so that the interleaver-block should be filled with always *b* SBs at a time. This is easily accomplished if the interleaver-block divides the number of SBs. So with the interleaving strategy that we discussed, partitioning of files into source blocks and determining the source block and symbol length will be affected by an extra parameter *b*. Since we have studied on small-scale file size we could not experimentally discover the overall aspect of the SB interleaving considering high file sizes.

5.3 Analytical Model

In this section we provide the formulization of the problem, hence 4 parameters: Waiting Time, Transmission Cost, Gain in waiting time and Gain in transmission cost for the proposed MBMS systems. To do so we define waiting time as “the minimal waiting time required to start playing the media on the terminal after the initialization of the MBMS download service”. So the term “waiting time” implies the initial startup time in progressive based downloads while it refers the downloading time in non-progressive downloads. In order to predict the initial startup delay, two types of receivers; Analyzing Receiver, and Actual Receiver are considered. Analyzing receivers estimate the initial startup delays for the target environment for the actual receiver. The initial startup delay is maintained on the sender side and sent to the receiver. The way that the sender has the estimated initial startup delays priori to the service and signaling of initial startup delays to receivers is out of scope of our work but it is studied by BenQ Mobile (2006).

We considered four environments corresponding to the system models studied: Time-Optimized and FEC-Optimized downloading environments, interleaved downloading environment and progressive downloading environment. For Progressive download types we implicitly assumed downloading time optimization. Since we have not Raptor code implementation, we are unable to work on the optimizations for Raptor FEC OTI parameters. So FEC overheads for Raptor are taken from the works by Siemens (March 2005) and 3GPP TSG SA WG#4 (June 2005) for different MBMS link conditions. From the same reason, we cannot study the effect of interleaved progressive download for Raptor. The gain functions allows us to identify how much savings in waiting time or in transmission cost can be gained from using the interleaved download delivery, progressive download delivery and interleaved progressive download delivery as well as from our optimizations. The file transmission is organized such that repair packets are transmitted after each source blocks to support progressive download.

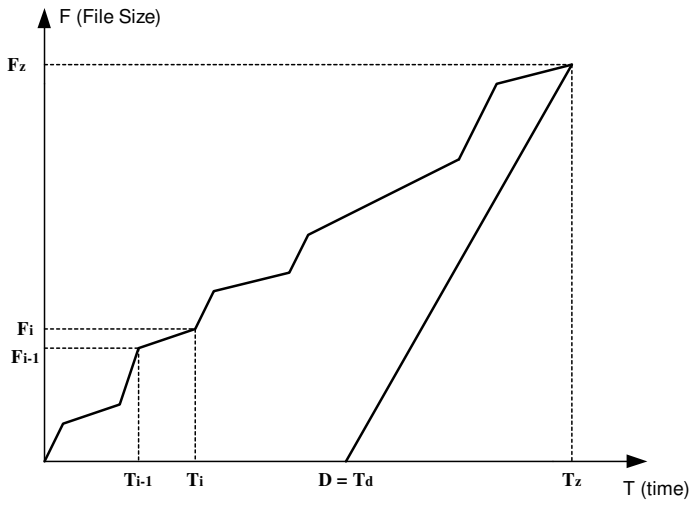


Figure 5.4 Download size as a function of time.

Figure 5.4 shows a worst case analyzing receiver obeying two assumptions: reliability and timely manner property where sending rate is less than media play rate and both playing and downloading ends at the same time. By considering the worst case scenario, analyzing receivers estimate the worst case waiting time for the actual receivers. Timely manner property implies that sender transmits symbols approximately at some constant rate and the receiver capability is sufficient enough to handle the

decoding of blocks without causing any overflow in the receiver buffer. Timely manner property is also explained in Figure 5.5. On receiver side, each time a source block is decoded, i is incremented and T_i is assigned to a timestamp, the time of writing the source block to the file. At time T_d media can be started to play progressively. With such a property and with a reliable delivery property, receivers can be assumed to play the media at some constant rate after waiting D time unit.

Figure 5.4 shows the linearity approximation between time and downloaded media. Congestion and buffering in the network decides the linearity approximation between time and downloaded media size. If the client suffers much from the buffering delays the linearity approximation is good otherwise it will be bad. However, for the receiver doing a progressive download, there is a complete linearity between time and played media size as long as 100% reliability is provided and suitable initial startup delay is given as indicated in the report by BenQ Mobile (2006). So we define partial receiving rate r_i as the size of source block decoded within a consecutive times T_{i-1} and T_i.

$$r_i = \frac{F_i - F_{i-1}}{T_i - T_{i-1}} = \frac{\Delta F_i}{\Delta T_i} \qquad (1)$$

where $F_0 = 0$, $T_0=0$; F_i denotes downloaded file size in KB and r_i denotes partial reception rate in Kbps at time T_i, $i: 1...z$, z is the total number of the source blocks.

R is defined as Expected Average Receiving Rate (EARR) computed at time T_z by the analyzing receiver that predicts the average receiving rate of the actual receiver. The analyzing receiver should use all r_i sequences up to T_z to find a single rate R that best estimates the actual average receiving rate of the receiver. That is the focus is the expected average receiving rate in near future, let's say after time T_z. In order to predict the initial startup delay D as well as the download duration for the actual receiver, an EARR function is defined. Choosing a good EARR function is not critical regarding to its effect on waiting time for small file sizes. So our choice is as follows:

$$R_i = \frac{r_i + R_{i\text{-}1}}{2} \tag{2}$$

where R_i is our choice for the EARR at T_i , $R_0 = r_1$; $i: 1...z$.

The reason behind our choice for such an EARR function, consider another EARR function that averages all r_i sequences. Let r_1=50, r_2=60, r_3=70 kbps where z=3. Then Eq.2 results in R_3 = 62.5 kbps however the averaging EARR results in R_3 = 60 kbps. Our choice more reacts to the recent changes in r_i sequences. That is, it is more reactive to the network conditions such as network congestion and buffering delays.

From Figure 5.4, we can construct the following approximation for the downloading time;

$$T_z = \frac{F}{R_z} \tag{3}$$

The environments that we worked under are denoted as superscript words: “time” and “FEC” to mean downloading time optimization and transmission cost optimization. The word “INT” is used as subscript to denote that the system is interleaved while the word “PROG” is used to indicate progressive download. Other subscripts “RS” and “Raptor” are used to indicate which FEC methods are used. So we define gain G^{time} to denote the gain in downloading time for downloading time optimization while G^{FEC} denotes the gain in FEC overhead for transmission cost optimization. G^{time} can be computed using Eq.2 and Eq.3 as follows:

$$T_{RS}^{time} = \frac{F}{R_{RS}^{time}} \tag{4}$$

$$T_{RS}^{FEC} = \frac{F}{R_{RS}^{FEC}} \tag{5}$$

$$T_{RS,INT}^{time} = \frac{F}{R_{RS,INT}^{time}} \tag{6}$$

$$G_{RS}^{time} = 100 * \left(1 - \frac{T_{RS}^{time}}{T_{RS}^{FEC}} \right) \tag{7}$$

$$G_{RS,INT}^{time} = 100 * \left(1 - \frac{T_{RS,INT}^{time}}{T_{RS}^{time}} \right) \tag{8}$$

where T_{RS}^{time} and $T_{RS,INT}^{time}$ denote the downloading times (in sec.) of the legacy and the interleaved downloading deliveries respectively, R_{RS}^{time} and $R_{RS,INT}^{time}$ denote the Expected Average Receiving Rate (in Kbps) of the legacy and the interleaved downloading deliveries respectively, G_{RS}^{time} indicates the gain in downloading time from downloading time optimization while $G_{RS,INT}^{time}$ denotes the gain in downloading time from the interleaved download all of which is for Reed Solomon coding.

The gain in FEC overhead G^{FEC} can be computed as follows:

$$G_{RS}^{FEC} = 100 * \left(1 - \frac{C_{RS}^{FEC}}{C_{RS}^{time}} \right) \tag{9}$$

$$G_{RS,INT}^{FEC} = 100 * \left(1 - \frac{C_{RS,INT}^{FEC}}{C_{RS}^{FEC}} \right) \tag{10}$$

Where C_{RS}^{FEC} and $C_{RS,INT}^{FEC}$ denote necessary FEC overheads (in percent) for reliability of the legacy and the interleaved downloading deliveries under transmission cost optimization respectively while C_{RS}^{time} indicates the necessary FEC overhead for reliability of the legacy download under downloading time optimization. Gain in FEC

overhead from the transmission cost optimization of the legacy download and the interleaved download using Reed Solomon coding, are denoted as G_{RS}^{FEC} and $G_{RS,INT}^{FEC}$ respectively.

For Reed Solomon all C, R and T values are experimentally discovered during emulations. Using the FEC OTI parameters, FEC overhead is calculates as follows;

$$C = 100* (max\_nb\_encoding\_symbols - max\_sb\_len) \ / \ max\_sb\_len \quad (11)$$

where *max_nb_encoding_symbols* indicates maximum number of encoding symbols in an encoded block and *max_sb_len* indicates maximum number of source symbols in a source block.

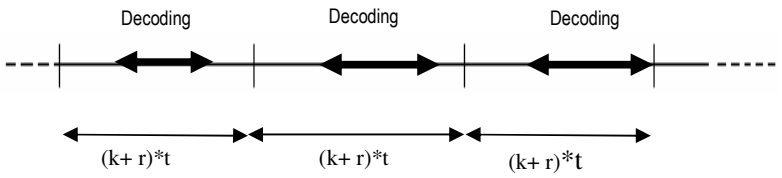


Figure 5.5 Receiving rate approximation.

Under the same network, same link conditions and same FEC partitioning with Reed Solomon we can produce analytical approximations for C, R and T values for Raptor.

For Raptor, we assumed that there is single FEC overhead selection for both optimizations which means the amount of FEC overhead is always selected to be minimum to provide 100% reliability. That is, transmission cost optimization and downloading time optimization uses the same FEC overheads for Raptor which are taken from the works by Siemens (March 2005) and 3GPP TSG SA WG#4 (June 2005) for different MBMS link conditions. So for Raptor the results are approximated using our analytical model here.

To formulize the problem for Raptor coding we need to consider few other details. Since transmission of a source block and its repair symbols follow each other, decoding

process of the block must start within the time interval *(k + r)* t* as shown in Figure 5.5, where *k* is the number of source symbols in the block, *r* is the number of repair symbols for the block and *t* denotes symbol period under reliability assumption; the average time between receiving of two consecutive symbols for an object. So symbol period depends on symbol size and transmission rate. However, decoding process may not be expected to be completed in this interval. The reason is that while decoding process continues the receiver can collect the symbols belonging to the next source block in parallel. That is, computation and communication can be overlapped during decoding.

If the receiver is overloaded or incapable of handling decoding processes in timely manner, such overlapping may occur. If the overlapping causes the decoding of the next source block to be failed we say that the timely manner property is not satisfied. So throughout the formulizations for Raptor as well as to support progressive downloading we assume that sender transmit symbols approximately at constant rate, hence a constant symbol period and the receiver satisfies the timely manner property.

Since each source block means extra repair symbols total time needed to download the media is total symbols including repair symbols multiplied by *t*:

$$T = \frac{F}{k * s}(k + r) * t = F * (\frac{k}{k * s} + \frac{C}{100}) * t \qquad (12)$$

where *s* is the symbol size in bytes, *k* is the number of source symbols, *r* is the number of repair symbols, C is FEC overhead in percent and *t* is the symbol period.

In order to have downloading time and gain approximations for Raptor we have assumed that for a source block size of *k*s*, while Reed Solomon requires C_{RS} percent transmission overhead, Raptor requires C_{Raptor} percent transmission overhead and symbol period $t = t_{Raptor} = t_{RS}$ since symbol sizes and transmission rates are same. Based on such an assumption and using Eq.12, if we subtract T_{Raptor} from T_{RS} we can arrive at following for Raptor:

$$T_{Raptor}^{X} = T_{RS}^{X} * \frac{F}{s}\left(\frac{C_{RS}^{X} - C_{Raptor}^{X}}{100}\right) * t_{Raptor}^{X} \quad (13)$$

where $X \in \{FEC, time\}$.

From Eq.12,

$$t_{RS}^{X} = \frac{T_{RS}^{X}}{F * \left(\frac{k}{k * s} + \frac{C_{RS}^{X}}{100}\right)} = t_{Raptor}^{X} \quad (14)$$

where $X \in \{FEC, time\}$.

As shown in Figure 5.5, for every time interval of $(k+r)*t$ exactly one source block is decoded. So the Expected Average Receiving Rate is computed as follows:

$$R_{Raptor}^{X} = \frac{k * s}{(k + r) * t_{Raptor}^{X}} = \frac{1}{\left(\frac{k}{k * s} + \frac{C_{Raptor}^{X}}{100}\right) * t_{Raptor}^{X}} \quad (15)$$

where $X \in \{FEC, time\}$.

While FEC overheads and gain from interleaving for Reed Solomon are experimentally explored in this study, Raptor FEC overheads are taken from the work by Siemens (March 2005) and 3GPP TSG SA WG#4 (June 2005). Since interleaving deals with packet loss patters, under the same network conditions with the same interleaving technique we can get the same interleaving gain in transmission cost for Raptor FEC protected download delivery system.

To formulize the problem for Progressive Download Delivery we considered Downloading time optimization. From Eq.15, we can derive the Expected Average Receiving Rate for progressive download using Raptor FEC as follows:

$$R_{Raptor,PROG}^{time} = \frac{1}{(\frac{k}{k*s} + \frac{C_{Raptor}^{time}}{100}) * t_{Raptor}^{time}} \tag{16}$$

where C_{Raptor} denotes the necessary FEC overheads for reliability, which is derived from the work by Siemens (March 2005) and 3GPP TSG SA WG#4 (June 2005).

From Figure 5.4, we can construct the following approximation for the download duration:

$$D + \frac{F}{R_{media}} = T_z = \frac{F}{R_z} \tag{17}$$

Using Eq.17 the Expected Average Receiving Rate and initial startup delay approximated at the end of file download using Reed Solomon is as follows;

$$D_{RS,PROG}^{time} = F * \left(\frac{1}{R_{RS,PROG}^{time}} - \frac{1}{R_{media}} \right) \tag{18}$$

where $D_{RS,PROG}^{time}$ denotes initial startup delay, R_{media} denotes media play rate in Kbps; F is media file size in KB; $R_{RS,PROG}^{time} = R_z$ is the Expected Average Receiving Rate computed using emulation results of Reed Solomon FEC protected Progressive Download under downloading time optimization. Similarly for Raptor, the initial startup time is derived as follows:

$$D^{time}_{Raptor,PROG} = F * \left(\frac{1}{R^{time}_{Raptor,PROG}} - \frac{1}{R_{media}} \right) \quad (19)$$

where $D^{time}_{Raptor,PROG}$ denotes initial startup delay, R_{media} denotes media play rate in Kbps; F is media file size in KB; $R^{time}_{Raptor,PROG} = R_z$ is the Expected Average Receiving Rate computed using the analytical model here.

We define gain $G^{time}_{RS,PROG}$ as the time gain of using progressive download compared to legacy download in terms of initial startup delay. Gain can be computed using R and T as follows:

$$G^{time}_{RS,PROG} = 100 * \left(1 - \frac{D^{time}_{RS,PROG}}{T^{time}_{RS}} \right) \quad (20)$$

where T^{time}_{RS} (Eq.4) denotes the duration of the legacy download of the media in seconds and G is the gain with progressive download. Similarly following is derived for Raptor:

$$G^{time}_{Raptor,PROG} = 100 * \left(1 - \frac{D^{time}_{Raptor,PROG}}{T^{time}_{Raptor}} \right) \quad (21)$$

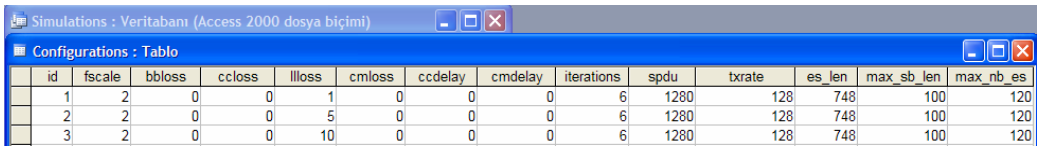
where T^{time}_{Raptor} (Eq.13) denotes the duration of the legacy download of the media in seconds. Similarly we can compute the Gain from Interleaved Progressive Download Delivery compared to legacy download in terms of initial startup delay as follows:

$$G^{time}_{RS,INT,PROG} = 100 * \left(1 - \frac{D^{time}_{RS,INT,PROG}}{T^{time}_{RS}} \right) \quad (22)$$

where T_{RS}^{time} (Eq.4) denotes the duration of the legacy download of the media in seconds and $D_{RS,INT,PROG}^{time}$ is the initial startup time observed during the emulations of Interleaved Progressive Download Delivery.

5.4 Simulation Environment

The prototype in Figure 5.1 is extended to simulate MBMS network conditions. Hence a database is designed including two tables: "Configuration" and "Simulation_Results". Simulation is configured mostly by setting the parameters in configuration table. Actually configuration table maintain the sizing parameters in application layer, FEC layer, IP layer, core network and RLC link layer.


Simulations : Veritabanı (Access 2000 dosya biçimi)

Configurations : Tablo

id	fscale	bbloss	ccloss	llloss	cmloss	ccdelay	cmdelay	iterations	spdu	txrate	es_len	max_sb_len	max_nb_es
1	2	0	0	1	0	0	0	6	1280	128	748	100	120
2	2	0	0	5	0	0	0	6	1280	128	748	100	120
3	2	0	0	10	0	0	0	6	1280	128	748	100	120

Figure 5.6 Configurations of the simulation.

Figure 5.6 shows an example of three set of configuration parameters for the 512 KB (fscale=2) file distribution. Each set of configuration parameters is identified by the configuration ID. In the example, configuration ID 2 identifies that RLC link layer lost is 5% (lllost), PDU size is 1280 bytes and the others, which are as follows:

- "fscale" identifes the file to download. There are two files of 100 KB and 512 KB, which are identified by "fscale =1" and "fscale =2" respectively.
- "bbloss" is IP backbone loss ratio.
- "ccloss" is cell congestion loss ratio.
- "llloss" is link layer loss ratio (RLC PDU loss)
- "cmloss" is cell mobility loss ratio.
- "ccdelay" is maximum cell congestion delay in second. Once a cell is congested it stays congested for randomly changing duration up to "ccdelay" seconds.

- “cmdelay” is maximum cell mobility delay in second (max. cell change delay). Cell mobility takes a random duration upto “cmdelay” seconds.

- “max_sb_len”, “es_len” and “max_nb_es” are maximum source block length, encoding symbol length and maximum number of encoding symbols respectively, which are FEC OTI (Object Transmission information) parameters.

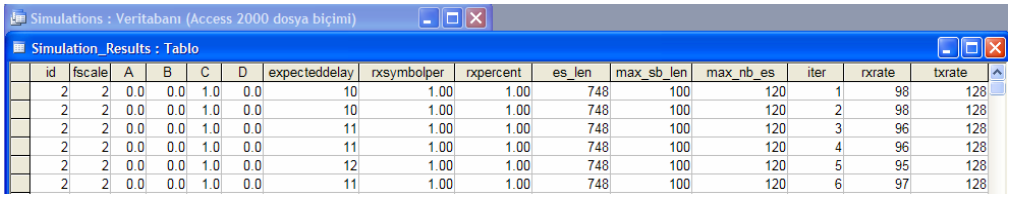
Burst errors are created using the algorithm for simulating the SDU loss pattern taken from the work by Digital Fountain, Ericsson & Others (May 2004), which is shown below. The function *transport_block_lost()* simulates the transmission/reception of a transport block, returning TRUE or FALSE according to whether the transport block is lost or received successfully respectively.

```
Initialise variable spare_octets to zero
Initialise variable last_block_lost to FALSE
Let block_length be the transport block length
FOR each SDU
      Let sdu_length be the length of the SDU
      sdu_lost := (spare_octets != 0) & last_block_lost
      IF (sdu_length <= spare_octets) THEN
            spare_octets := spare_octets - sdu_length
      ELSE
            remaining_octets := sdu_length - spare_octets
            blocks := Integer part of
            remaining_octets/block_length
            IF (blocks > 0)
                  FOR i := 1 TO blocks
                        sdu_lost := transport_block_lost() |
                  sdu_lost
                  END FOR
            ENDIF
            last_block_lost := transport_block_lost()
            sdu_lost := sdu_lost | last_block_lost
            spare_octets := block_length - (remainder of
            remaining_octets/block_length)
      ENDIF
      IF (sdu_lost) THEN
            Report SDU as lost
      END IF
END FOR
```

Other parameters such as SDU size, mapping of PDU losses to SDU losses that is given in above algortihm are implemented inside the code. After an iteration of the download, the simulation results are saved in Simulation_Result table shown in Figure 5.7. Figure 5.7 shows the simulation results of six iterations for the configuration ID =

2, which is set previously in configuration table. Each record shows a download result and identified by “id”, “fscale” and “iter”. “Id” intentifies the configuration parameters set in configuration table. “Iter” identifies the current iteration of the download. Total number of iterations is set by the “iterations” in configuration table. Other attributes are as follows:

- “A”,”B”,”C” and “D” identifies the lost distribution among IP backbone losses, cell congestion losses, RLC PDU losses and cell mobility losses, which are defined in the report by Digital Fountain, Ericsson, NEC, Nokia, Nortel, Siemens (May 2004). In the example in Figure 5.7, all the losses belong to “C”, which are RLC PDU losses.


Simulations : Veritabanı (Access 2000 dosya biçimi)

Simulation_Results : Tablo

id	fscale	A	B	C	D	expecteddelay	rxsymbolper	rxpercent	es_len	max_sb_len	max_nb_es	iter	rxrate	txrate
2	2	0.0	0.0	1.0	0.0	10	1.00	1.00	748	100	120	1	98	128
2	2	0.0	0.0	1.0	0.0	10	1.00	1.00	748	100	120	2	98	128
2	2	0.0	0.0	1.0	0.0	11	1.00	1.00	748	100	120	3	96	128
2	2	0.0	0.0	1.0	0.0	11	1.00	1.00	748	100	120	4	96	128
2	2	0.0	0.0	1.0	0.0	12	1.00	1.00	748	100	120	5	95	128
2	2	0.0	0.0	1.0	0.0	11	1.00	1.00	748	100	120	6	97	128

Figure 5.7 Entries in Simulation_Results Table.

- “expecteddelay” is the expected delay identifying the downloading duration in MBMS download or initial waiting time in MBMS progressive download.
- “rxsymbolper” is the percent of the received symbols without caring the successful or unsuccessfull decoding of source blocks.
- “rxpercent” is the the downloaded percent of the media with successful decoding of source blocks.
- “rxrate” is the average receiving rate under the configured network and link conditions.
- “txrate” is the transmission rate.

In the example in Figure 5.7, although transmission rate is 128 kbps, because of 5% RLC PDU losses, the average receiving rate is reduced to 98, 97 or 96 kbps under 20% FEC transmission.

The simulation flow at server side can be summarized as follows;

```
FOR each Item in Configurations_Table DO
    MBMS_Download = New Download ( Item)
    FOR i:=1 to Item.Iterations DO
          MBMS_Download.Send()
    END FOR
END FOR
```

The simulation flow at receiver side can be summarized as follows;

```
FOR each Item in Configurations_Table DO
    MBMS_Download = New Download ( Item)
    FOR i:=1 to Item.Iterations DO
          Result_Item = MBMS_Download.Receive (New
MBMS_Link_Conditions (Item));
                Add_To_Simulation_Results_Table (Result_Item);
    END FOR
END FOR
```

MBMS_Download.Receive and *MBMS_Download.Send* functions are FLUTE sender and FLUTE receiver that are based on Vidiator MBMS download prototype, which uses Reed Solomon FEC coding. The receive function takes the MBMS link conditions as parameters since the transmission rate and packet loss control module are implemented at the receiver side. After each download the receiver adds the simulation record, denoted as Result_Item in the flow, to the Simulation_Results table.

The configuration database is manually preconfigured before running the simulation. There can be many runs of the simulation. After each run of the simulation, Simulation_Results table are examined and new set of configuration parameters are produced. This process repeats until all sizing effects of the parameters are explored by repeating the pre-configuring and running sequence.

Experiments are done on a set of computers. Each computer runs the simulation for different configuration parameters which are given in Table 5.1. The MBMS download system consists of a single server and a single client. Considering server and client on the same machine allows the full control of our simulations. Both client and server maintain a database and create a record after each download. The record contains

evaluated target parameters for the selected set of configuration parameters shown in Table 5.1. Several sets of configuration parameters are scheduled to initiate several downloads with different characteristics. Each download experiment for one set of configuration parameters has 100 download iterations over which our target parameters such as waiting time and initial startup delay are computed using the average of the values. After performing all the combinations of the configuration parameters, local databases in each computer are merged into a single database. Experimental results are produced using that merged database.

CHAPTER SIX

EXPERIMENTAL RESULTS

This section is divided into four parts. First part shows the results of our experimental analyses for the legacy-download under two optimizations: Downloading time optimized versus transmission cost optimized reliability.

Table 6.1 Parameters studied accross layers.

Parameter	Unit	Layer	Experiment Set	System
File Size	Kilobyte	Application	{100,512}	All
Transmission cost optimized FEC overheads	Percent	Application	To be determined in this study	Legacy
Transmission cost optimized Waiting Time	Second	Application	To be determined in this study	Legacy
Downloading time optimized FEC Overheads	Percent	Application	To be determined in this study	Legacy
Downloading time optimized Waiting Time	{Second, Percent}	Application	To be determined in this study	Legacy
Gain in Waiting Time from Optimizations	{Second, Percent}	Application	To be determined in this study	Legacy
Initial Startup Delay in Progressive Download	Second	Application	To be determined in this study	Progressive
Gain in Initial Startup Delay from Progressive Download	{Second, Percent}	Application	To be determined in this study	Progressive
Gain in Downloading Time from Interleaving	{Second, Percent}	Application	To be determined in this study	Interleaved
Gain in FEC Overhead from Interleaving	Percent	Application	To be determined in this study	Interleaved
Initial Startup Delay in Interleaved-Progressive Download	Second	Application	To be determined in this study	Interleaved-Progressive
Gain in Initial Startup Delay from Interleaved-Progressive Download	{Second, Percent}	Application	To be determined in this study	Interleaved-Progressive
Symbol Length	Byte	FEC	{SDU – 48}	All
SB Size	Symbol	FEC	{10,50,80,100,120,150,180,200,230}	All
IP packet size	Byte	IP	{SDU}	All
SDU block size	Byte	Core Network	{400,600,800,1000,1400}	All
PDU block size (RLC block size)	Byte	RLC Radio L.	{640,1280}	All
PDU (RLC Link Layer) Losses	Percent	RLC Radio L.	{1,5,10}	All
Transmission Rates	Kbps	All layer	{64,128,256}	All
Interleaver block size	Source Block	FEC	{2,3,4}	Interleaved-(Progressive)
Media Play Rate	Kbps	Application	{128}	All

In second part gains obtained from the legacy download is further extended by Application Layer Interleaving mechanism. Third part shows progressive download approach. Hence third part shows our progressive download analyses to explore the gain in waiting time. In final section the gain obtained from the progressive download

is further improved by combining our application layer interleaving mechanism and progressive download. Hence the final section shows the results of analyses of Interleaved-Progressive download delivery. Here after, the term legacy download refers to the downloading time optimized legacy download unless otherwise stated.

6.1 Experimental Results for Legacy-Download Delivery

This section shows the results of our experimental analyses targeting two optimizations for legacy download delivery: Downloading time optimization and transmission cost optimization. Each optimization satisfies the reliability requirement. Finally we compare them by providing the gains in downloading time as well as in transmission cost for different network conditions described by the parameters. The list of parameters analyzed in this study across layers is summarized in Table 6.1.

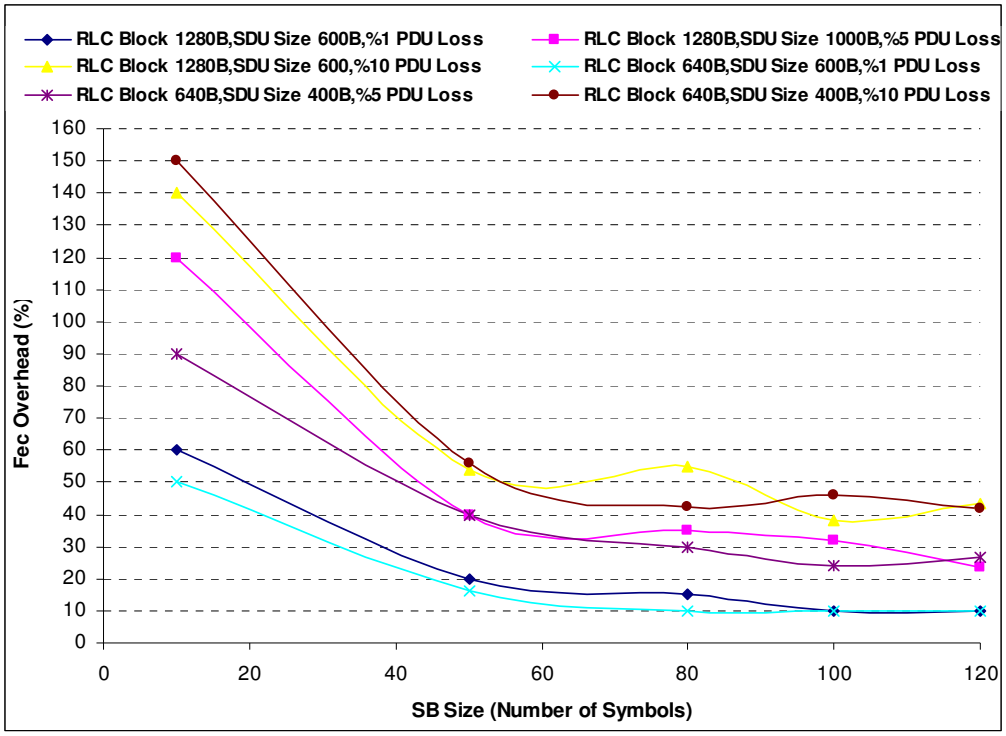


Figure 6.1 Transmission cost optimization for 100 KB file.

Reed Solomon FEC performance is very bad at small source block size. As shown in Figure 6.1, to Figure 6.4 increasing source block size will decrease the FEC overhead

required for reliability, hence causes an increase in FEC performance. However, the source block size exceeded a threshold value makes the downloading time increase in spite of still reducing the FEC overhead. This property provides trades off between downloading time and FEC overhead. Together with source block size, effects of other sizing parameters, such as symbol length, RLC block size (Radio Link Layer) and SDU (Service Data Unit) size all should be considered together.

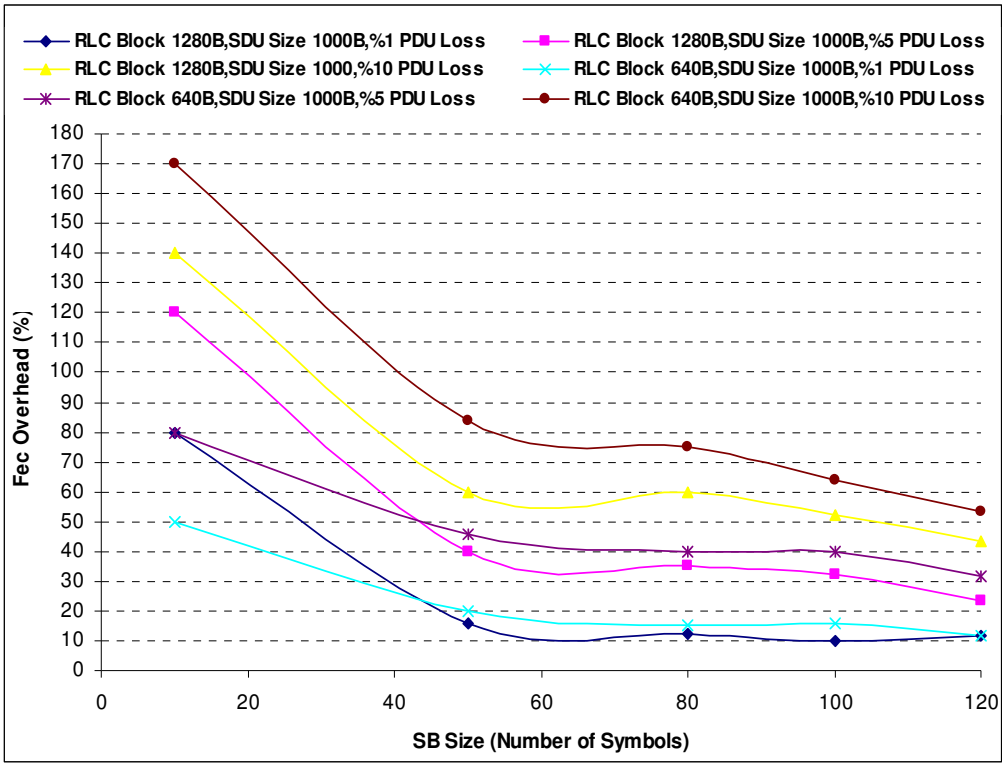


Figure 6.2 Download time optimization for 100 KB file.

By analyzing many possible combinations of the size choices we have been able to catch those cases that lead to download time optimized reliability. Download time optimized reliability aims the fastest download with reliability, which also provides savings in initial startup time for progressive download delivery.

We observed that as SB (Source Block) size increases FEC overheads dramatically decreases around up to SB size of 80 symbols thereafter the decrease slows. Additionally the small size file transport shows less deterministic behavior in terms of

FEC overhead cost required for reliability. These fluctuations can be seen in Figure 6.1 and Figure 6.2. With FEC overhead cost optimizations, shown in Figure 6.1 and Figure 6.3, FEC overhead is reduced with the decreased SDU sizes and increased SB size, hence increased number of symbols per block. This will increase the downloading time as shown in Table 6.2 and Table 6.3. With downloading time optimizations, shown in Figure 6.2 and Figure 6.4, downloading time is reduced with the increased SDU sizes and decreased SB sizes, hence decreased number of symbols per block with compared to FEC cost optimization. This will cause the FEC overhead to increase as shown in Table 6.2 and Table 6.3.

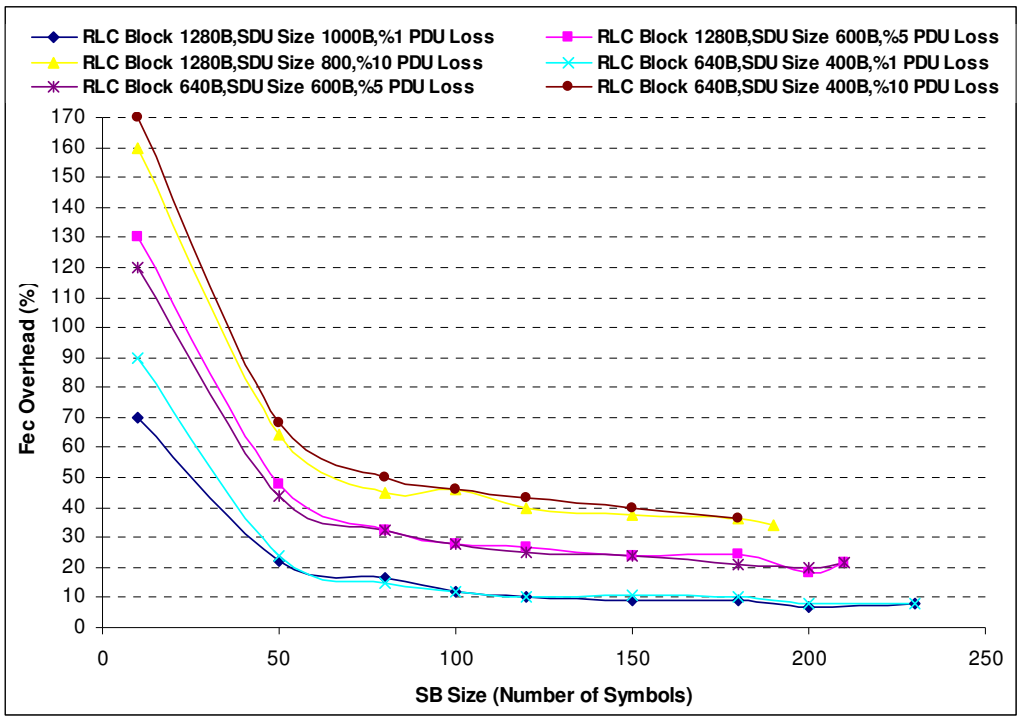


Figure 6.3 FEC overhead cost optimization for 512 KB file.

Comparing to both optimizations, FEC overhead optimized reliability is usually obtained at same or less SDU sizes and smaller symbol length but higher SB size. It is interesting while our recommended SDU size, among the set shown in Table 6.1, for FEC overhead optimized reliability is the one that is high but smaller than the RLC block size, for downloading time optimized reliability, it is the one that is small but

higher than the RLC block size. This gives us clue that in general, recommended SDU size should be equal to RLC block size. More detailed comparisons are given in Table 6.2 and Table 6.3, which also shows the trade off values between FEC and time costs, where they occur are shown as labels in the Figure 6.1 to Figure 6.4.

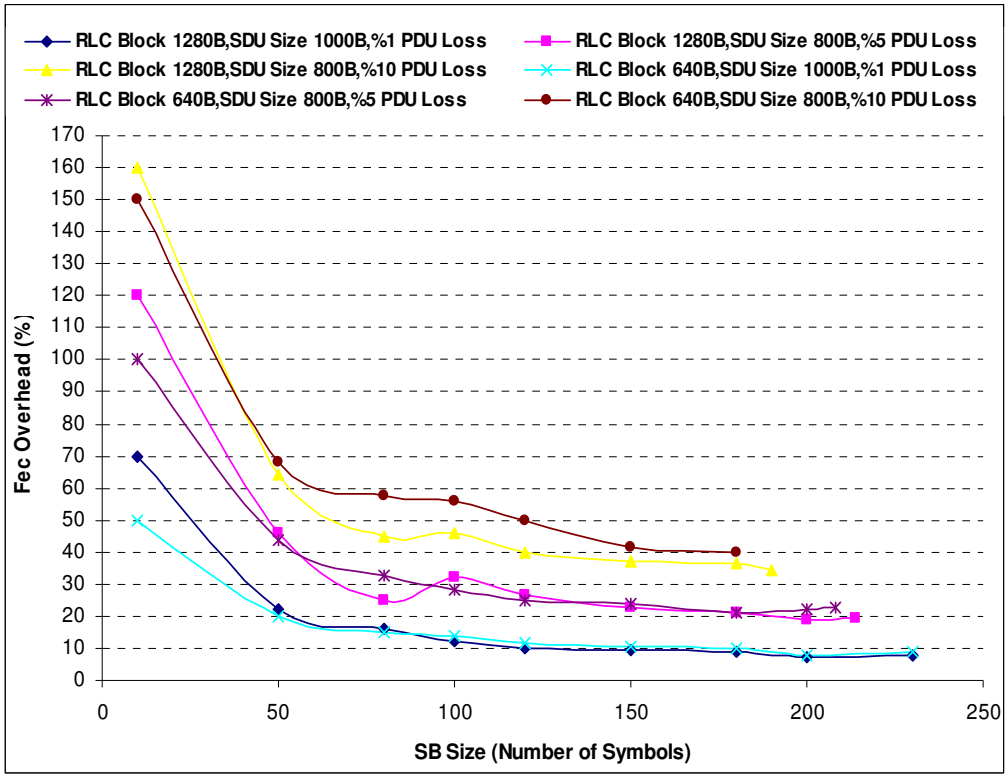


Figure 6.4 Download time optimization for 512 KB file.

For example, for RLC block size 640 B, SDU sizes are found mostly 400 and 600 B for the FEC optimized reliability, while it is mostly 800 and 1000 B for the download time optimized reliability. It seems mysterious that increased FEC overhead results in reduction in downloading time. Normally as FEC overhead increases downloading time increases too. The mystery lies in that fact that FEC performance increases as SB size increases with the cost of decoding time. In order to catch reliability at minimum FEC overhead, SB size should be properly selected as well as with the other sizing parameters. So higher FEC overhead but the smaller source block size possibly with the

other size choices for the sizing parameters changed can trade each other for downloading time and FEC overhead.

Comparisons under different optimizations are given in Table 6.2 for Reed Solomon and in Table 6.3 for Raptor. For small files the results shows not deterministic behavior and optimization effect is very small hence ignorable. Two type of gain is considered: Time and FEC overhead. Usually the gain in one causes a cost in the other.

Table 6.2 Comparison of optimizations for different transmission rates for Reed Solomon.

		Transmission Rate 256 Kbps						Transmission Rate 128 Kbps						Transmission Rate 64 Kbps					
		Transmission cost Optimization		Downloading time optimization				Transmission cost Optimization		Downloading time optimization				Transmission cost Optimization		Downloading time optimization			
File Size (KB)	PDU Loss (%)	FEC Ov. (%)	D. Time (sec)	FEC Ov. (%)	D. Time (sec)	Gain (%) Time	Gain (%) FEC	FEC Ov. (%)	D. Time (sec)	FEC Ov. (%)	D. Time (sec)	Gain (%) Time	Gain (%) FEC	FEC Ov. (%)	D. Time (sec)	FEC Ov. (%)	D. Time (sec)	Gain (%) Time	Gain (%) FEC
100	1	10	3.7	12	3.4	8	17	10	7.4	12	6.8	7	17	10	15.3	12	13.8	10	17
100	5	23	4.4	23	3.7	17	0	23	8.8	23	7.4	17	0	24	18.5	32	15.3	17	25
100	10	38	4.8	43	4.0	17	12	38	9.7	43	8.1	17	12	42	21.3	53	17.5	18	21
512	1	7	18.5	7	18.5	0	0	7	37.0	7	36.9	0	0	8	82.7	8	74.5	10	0
512	5	18	21.4	19	20.9	2	5	18	42.9	19	41.4	4	5	20	88.3	21	85.3	3	5
512	10	34	23.7	37	23.5	1	8	34	47.9	37	47.1	2	8	37	104.2	41	99.9	4	10

For example, in Table 6.2 for file 512 KB and 256 Kbps transmission rate the couple (Time,FEC)=(2,5) indicates that the downloading time optimization compared to the transmission cost optimization gives 2% gain in downloading time with the cost of increased FEC overhead. The reverse is also correct, the transmission cost optimization compared to the downloading time optimization provides 5% gain in FEC overhead with the cost of increased downloading time.

Table 6.3 shows the similar results for Raptor that are obtained under the same size selections of the sizing parameters for Reed Solomon, which characterize the same network, the same link conditions and the same source block partitioning. So Table 6.2 and Table 6.3 consider the same conditions for both Reed Solomon and Raptor. Since Raptor code implemantation is not free and available upon a huge cost, the analyses of the FEC OTI parameters for Raptor is not studied but the effect of network parameters are studied. So for Raptor we assumed that there is single FEC overhead selection for both optimizations where FEC overheads for different MBMS conditions are selected from the works by Siemens (March 2005) and 3GPP TSG SA WG#4 (June 2005). As a

consequence, the results for Raptor are approximated using our analytical model in which our assumptions for Raptor are given in detail.

Table 6.3 Comparison of optimizations for different transmission rates for Raptor.

		Transmission Rate 256 Kbps					Transmission Rate 128 Kbps					Transmission Rate 64 Kbps				
		Transmission cost Optimization		Downloading time optimization			Transmission cost Optimization		Downloading time optimization			Transmission cost Optimization		Downloading time optimization		
File Size KB	PDU Loss (%)	FEC Ov. (%)	D. Time (sec)	FEC Ov. (%)	D. Time (sec)	Gain (%) Time	FEC Ov. (%)	D. Time (sec)	FEC Ov. (%)	D. Time (sec)	Gain (%) Time	FEC Ov. (%)	D. Time (sec)	FEC Ov. (%)	D. Time (sec)	Gain (%) Time
100	1	8	3.6	8	3.3	8	8	7.2	8	6.6	8	8	15.0	8	13.3	11
100	5	21	4.4	21	3.6	18	21	8.7	21	7.2	17	22	18.2	22	14.1	22
100	10	35	4.7	35	3.8	19	35	9.5	35	7.6	20	39	20.9	39	15.9	24
512	1	4	18.0	4	18.0	0	4	35.9	4	35.9	0	4	79.7	4	71.7	10
512	5	12	20.3	12	19.7	3	12	40.7	12	38.9	4	14	83.9	14	80.4	4
512	10	22	21.6	22	21.0	3	22	43.6	22	41.9	4	26	95.9	26	89.3	7

The following is a summary of general observations for optimizations for legacy download in MBMS, where results for 100 KB file size is ignored:

- As source block size increases the legacy download system performance increases up to a point. Around that point a trade off between FEC overhead and downloading time can be provided for Reed Solomon by adjusting the size selections of the parameters described in different layers in Table 1.
- The sizing parameters describing the network that enable optimum downloading time for Reed Solomon also enable the optimization in downloading time for Raptor.
- Maximum savings in downloading time is around up to %10 with the downloading time optimization compared to the transmission cost optimization.
- Maximally 8.3 seconds gain in downloading time is provided by the downloading time optimization with regards to the transmission cost optimization.
- Maximum savings in FEC overhead is around up to %10 with the transmission cost optimization compared to the downloading time optimization for Reed Solomon.
- Maximum reduction in FEC overhead percent is around up to 4 units with the transmission cost optimization compared to the downloading time optimization for Reed Solomon.

6.2 Experimental Results for Interleaved Download Delivery

6.2.1 Experimental Results Under Transmission Cost Optimization

This section shows the gain in FEC overhead when we apply our application layer interleaving to the transmission cost optimized legacy download delivery. This means the legacy download and the interleaved download are under the transmission cost optimization.

Table 6.4 Gains from the interleaved download under FEC optimization for Reed Solomon.

		Transmission Rate 256 Kbps					Transmission Rate 128 Kbps					Transmission Rate 64 Kbps				
		Legacy Download		Interleaved Download			Legacy Download		Interleaved Download			Legacy Download		Interleaved Download		
File Size (KB)	PDU Loss (%)	FEC Ov. (%)	Down. Time (sec)	FEC Ov. (%)	Down. time (sec)	FEC Gain (%)	FEC Ov. (%)	Down. Time (sec)	FEC Ov. (%)	Down time (sec)	FEC Gain (%)	FEC O. (%)	Down. Ttime (sec)	FEC Ov. (%)	Down. Time (sec)	FEC Gain (%)
100	1	10	3.7	8	3.5	20	10	7.4	8	7.1	20	10	15.3	7	14.3	30
100	5	23	4.4	22	3.7	4	23	8.8	22	7.3	4	24	18.5	18	16.1	25
100	10	38	4.8	32	4.0	16	38	9.7	32	8.1	16	42	21.3	30	17.5	29
512	1	7	18.5	5	17.6	29	7	37.0	5	35.3	29	8	82.7	6	78.9	25
512	5	18	21.4	15	19.1	17	18	42.9	15	38.3	17	20	88.3	17	85.9	15
512	10	34	23.7	30	21.0	12	34	47.9	30	42.1	12	37	104.2	31	98.2	16

The results for Raptor are approximated using both the analytical model and the experimental results for Reed Solomon. The list of parameters analyzed in this section is summarized in Table 6.1. Savings in FEC overhead from the interleaved download delivery using Reed Solomon and Raptor are given in Table 6.4 and Table 6.5 respectively.

Table 6.5 Gains from the interleaved download under FEC optimization for Raptor.

		Transmission Rate 256 Kbps					Transmission Rate 128 Kbps					Transmission Rate 64 Kbps				
		Legacy Download		Interleaved Download			Legacy Download		Interleaved Download			Legacy Download		Interleaved Download		
File Size (KB)	PDU Loss (%)	FEC Ov. (%)	Down. Time (sec)	FEC Ov. (%)	Down. Time (sec)	FEC Gain (%)	FEC Ov. (%)	Down. Time (sec)	FEC Ov. (%)	Down Time (sec)	FEC Gain (%)	FEC Ov. (%)	Down. Time (sec)	FEC Ov. (%)	Down. Time (sec)	FEC Gain (%)
100	1	8	3.6	7	3.5	13	8	7.2	7	7.0	13	8	15.0	6	14.2	25
100	5	21	4.4	20	3.6	5	21	8.7	20	7.2	5	22	18.2	17	16.0	23
100	10	35	4.7	30	3.9	14	35	9.5	30	8.0	14	39	20.9	29	17.4	26
512	1	4	18.0	3	17.3	25	4	35.9	3	34.6	25	4	79.7	3	76.7	25
512	5	12	20.3	10	18.3	17	12	40.7	10	36.6	17	14	83.9	12	82.2	14
512	10	22	21.6	20	19.4	9	22	43.6	20	38.9	9	26	95.9	22	91.5	15

The following is a summary of general observations for the interleaved download under transmission cost optimization in MBMS, where results for 100 KB file size are ignored because of its small effect on both transmission size and downloading time:

- As file sizes increase, the necessary FEC transmission overhead decreases particularly for the network with RLC block = 1280 B. Generally RLC 640 B network requires more FEC overhead compared to the RLC 1280 B networks.
- For 100 KB file, results are not much deterministic where there are more fluctuations in gain with regards to 512 KB file.
- The gain from the transmission cost optimized interleaved download can save FEC overhead around up to 29% for Reed Solomon and 25% for Raptor with regards to transmission cost optimized legacy download.
- For Reed Solomon the gain in FEC overhead from the transmission cost optimized interleaved download is slightly higher then that of Raptor, where the gain difference is around up to 4%.
- The transmission cost optimized interleaved download provides up to 12% savings in downloading time for Reed Solomon and around up to 10% for Raptor with regards to the transmission cost optimized legacy download.

6.2.2 Experimental Results under Downloading Time Optimization

This section shows the gain in downloading time when we use the interleaved download delivery under downloading time optimization. This means the legacy download and the interleaved download are under the downloading time optimization. The results for Raptor are approximated using both the analytical model and the experimental results for Reed Solomon.

The list of parameters analyzed in this section is summarized in Table 6.1. Savings in downloading time from the interleaved download delivery using Reed Solomon and Raptor are given in Table 6.6 and Table 6.7 respectively.

Table 6.6 Gains from the interleaved download under downloading time optimization for Reed Solomon.

		Transmission Rate 256 Kbps					Transmission Rate 128 Kbps					Transmission Rate 64 Kbps				
		Legacy Download		Interleaved Download			Legacy Download		Interleaved Download			Legacy Download		Interleaved Download		
File Size (KB)	PDU Loss (%)	FEC O. (%)	D. Time (sec)	FEC Ov. (%)	D. Time (sec)	D. Time Gain (%)	FEC Ov. (%)	D. Time (sec)	FEC Ov. (%)	D. Ttime (sec)	D. Time Gain (%)	FEC Ov. (%)	D. Time (sec)	FEC Ov. (%)	D. Time (sec)	D. Time Gain (%)
100	1	12	3.4	12	3.4	0	12	6.8	12	6.8	0	12	13.8	10	13.9	0
100	5	23	3.7	22	3.7	0	23	7.4	22	7.3	1	32	15.3	20	15.5	0
100	10	43	4.0	36	4.0	0	43	8.1	36	8.0	1	53	17.5	38	17.1	2
512	1	7	18.5	5	17.6	5	7	36.9	5	35.3	4	8	75..3	7	71.7	5
512	5	19	20.9	16	19.1	9	19	41.4	16	38.4	7	21	85.3	19	79.8	7
512	10	37	23.5	32	21.0	11	37	47.1	32	42.1	11	41	99.4	37	90.2	9

Table 6.7 Gains from the interleaved download under downloading time optimization for Raptor.

		Transmission Rate 256 Kbps					Transmission Rate 128 Kbps					Transmission Rate 64 Kbps				
		Legacy Download		Interleaved Download			Legacy Download		Interleaved Download			Legacy Download		Interleaved Download		
File Size (KB)	PDU Loss (%)	FEC Ov. (%)	D. Time (sec)	FEC Ov. (%)	D. Time (sec.)	D. Time Gain (%)	FEC Ov. (%)	D. Time (sec)	FEC Ov. (%)	D. Time (sec)	D. Time Gain (%)	FEC Ov. (%)	D. Time (sec)	FEC Ov. (%)	D. Time (sec)	D. Time Gain (%)
100	1	8	3.3	8	3.3	0	8	6.6	8	6.6	0	8	13.3	7	13.4	0
100	5	21	3.6	20	3.6	0	21	7.2	20	7.2	0	22	14.1	14	14.7	0
100	10	35	3.8	29	3.8	0	35	7.6	29	7.6	0	39	15.9	28	15.9	0
512	1	4	18.0	3	17.3	4	4	35.9	3	34.6	4	4	71.7	4	69.4	3
512	5	12	19.7	10	18.2	8	12	38.9	10	36.4	7	14	80.4	13	75.6	6
512	10	22	21.0	19	18.9	10	22	41.9	19	37.9	10	26	89.3	23	81.3	9

The following is a summary of general observations for the interleaved download under downloading time optimization in MBMS, where results for 100 KB file size are ignored because of its small effect on both transmission size and downloading time:

- As file sizes increase, the downloading time increases as well particularly for the network with RLC block = 640 B since RLC 640 B network requires more FEC overhead compared to the RLC 1280 B networks.
- For 100 KB file, results are not much deterministic where there are more fluctuations in gain with regards to 512 KB file.
- The gain from the downloading time optimized interleaved download can save downloading time around up to 11% for Reed Solomon and 10% for Raptor with regards to the downloading time optimized legacy download.
- For Reed Solomon the gain in downloading time from the downloading time optimized interleaved download is slightly higher then that of Raptor, where the gain difference is around up to 2%.

- The downloading time optimized interleaved download provides up to 29% savings in FEC overhead for Reed Solomon and around up to 25% for Raptor with regards to the downloading time optimized legacy download.

6.2.3 General Comparisons

This section combines the gain in downloading time and the gain in FEC overhead from the interleaved download to make cross comparisons between transmission cost and downloading time optimizations for both Reed Solomon and Raptor FEC methods.

Table 6.8 General comparisons.

FEC	Reed Solomom				Raptor			
Optimization Type	Transmission Cost Optimized Legacy Download		Downloading Time Optimized Legacy Download		Transmission Cost Optimized Legacy Download		Downloading Time Optimized Legacy Download	
Gain Type	FEC Overhead (%)	Download Time (%)	FEC Overhead (%)	Download Time (%)	FEC Overhead (%)	Download Time (%)	FEC Overhead (%)	Download Time (%)
Gain from the Transmission Cost Optimized Interleaved Download	Avg = 19 Max = 29	Avg = 8 Max = 12	Avg = 23 Max = 29	Avg = 5 Max = 11	Avg = 17 Max = 25	Avg = 7 Max = 11	Avg = 17 Max = 25	Avg = 4 Max = 7
Gain from the Downloading Time Optimized Interleaved Download	Avg = 12 Max = 29	Avg = 10 Max = 13	Avg = 12 Max = 29	Avg = 7 Max = 11	Avg = 14 Max = 25	Avg = 10 Max = 15	Avg = 14 Max = 25	Avg = 7 Max = 10

Table 6.8 shows these comparisons using maximum and average gains in FEC overhead and in downloading time, where the results for 100 KB files are ignored because of its small effect on both transmission size and downloading time. Average values are computed using the results for 256, 128 and 64 Kbps transmission rates.

The following is a summary of general observations for the interleaved download considering both optimizations in MBMS, where results for 100 KB file size are ignored because of its small effect on both transmission size and downloading time:

- The transmission cost optimized interleaved download under Raptor FEC protection makes reduction in FEC overhead at the average 50% with regards to the downloading time optimized legacy download under Reed Solomon FEC protection.

- The downloading time optimized interleaved download under Raptor FEC protection makes reduction in downloading time at the average 15% with regards to the transmission cost optimized legacy download under Reed Solomon FEC protection.
- In general Reed Solomon has more benefits from interleaving with regards to Raptor.

6.3 Experimental Results for Progressive Download Delivery.

This section shows waiting time analyses for 100% reliability and provides the savings in waiting time from progressive download delivery in MBMS. This section provides further increase in the gain obtained from downloading time optimized legacy download by progressive downloading. Here after, the term legacy download refers to the downloading time optimized legacy download unless otherwise stated. Progressive download delivery is also considered to be under downloading time optimiziation. MBMS download implies Raptor FEC protected MBMS download unless specified otherwise. The list of parameters analyzed in this study across layers is summarized in Table 6.1.

First we observe the initial startup delay for 100% reliability for various Reed Solomon SB sizes. Figure 6.5 and Figure 6.6 show initial startup delay analyses observed for small and medium file sizes respectively, transmitted over 256 kbps, 128 kbps and 64 kbps, under 1%, 5% and 10% link loss ratios for Reed Solomon FEC protected download. The SDU size is selected to be the optimum for these conditions as well as the SB to provide 100% reliability. We observed that for as the SB size increases the initial startup delay decreases. The amount of FEC overhead increases as the PDU loss ratio increases. As the FEC overhead increases it is expected that the initial startup delay increases too.

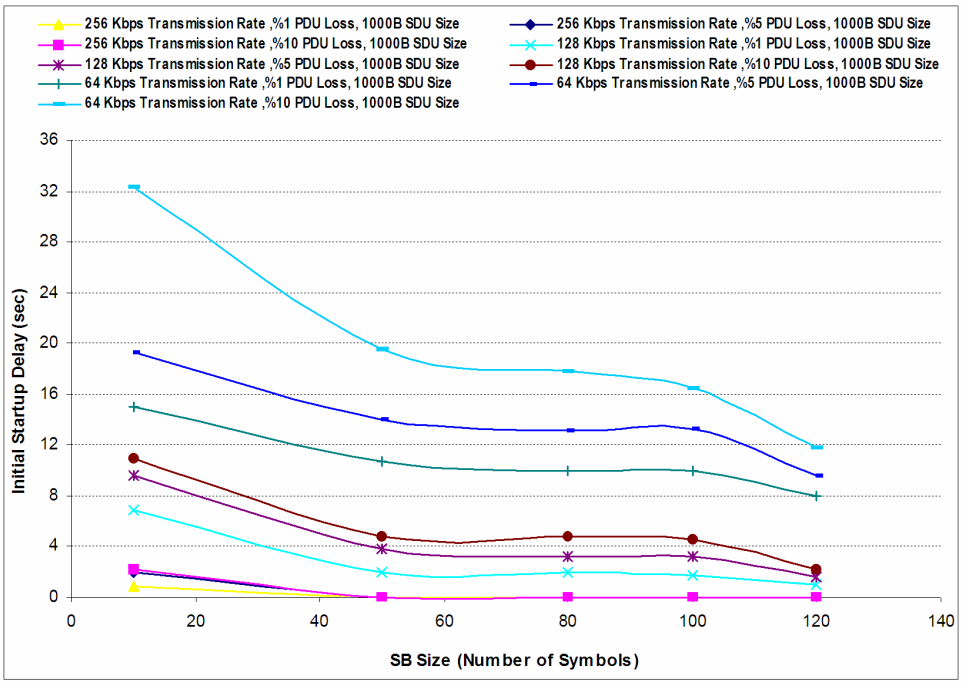


Figure 6.5 Initial startup delay analyses for 100 KB file.

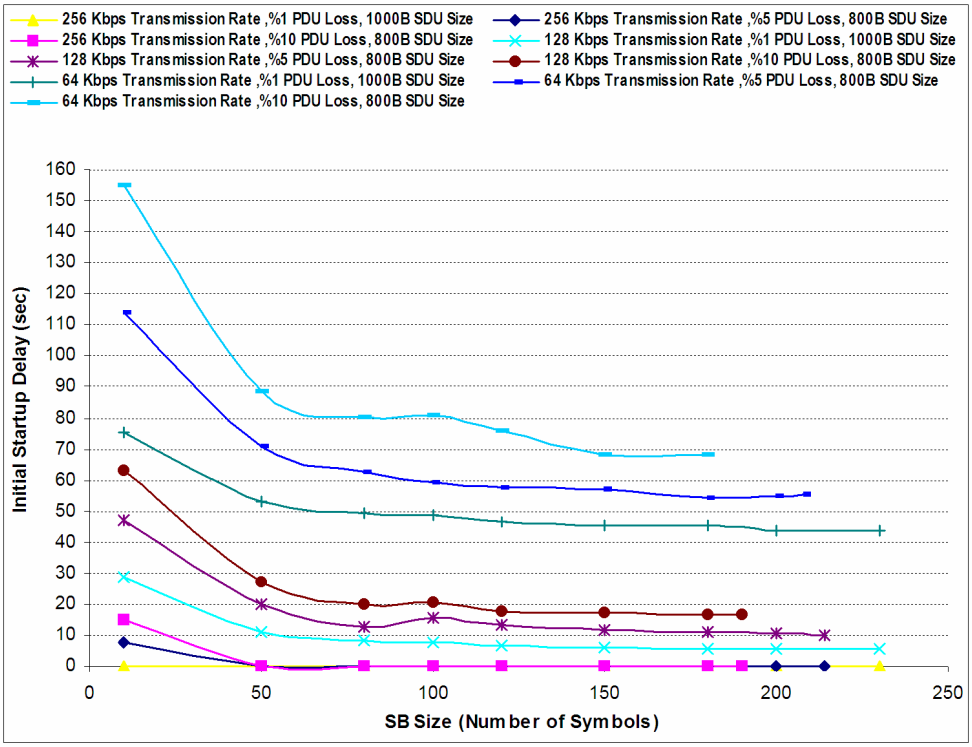


Figure 6.6 Initial startup delay analyses for 512 KB file.

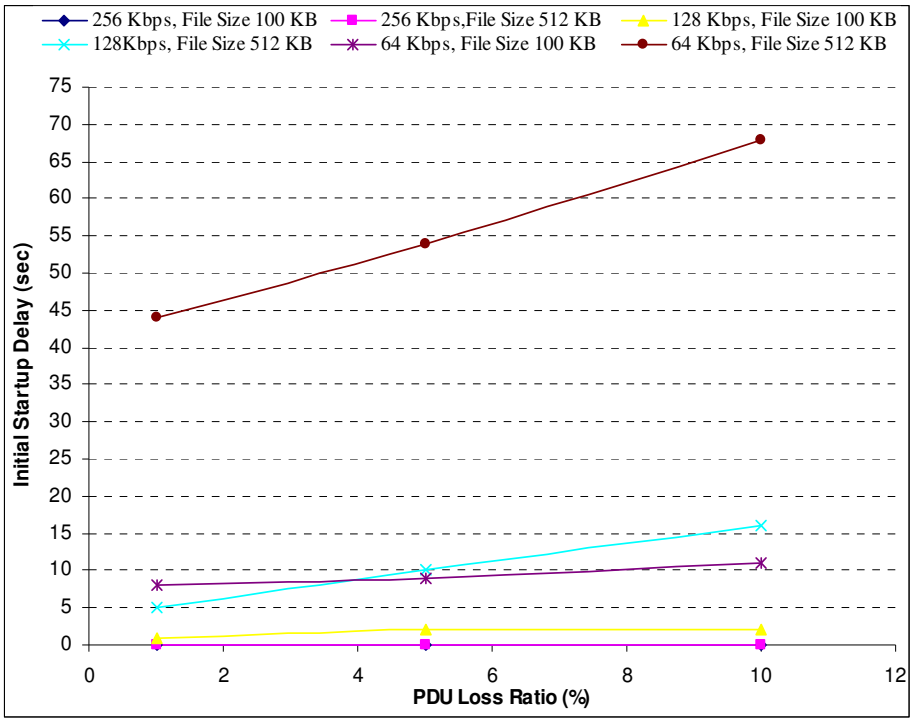


Figure 6.7 Initial startup delay for various PDU loss ratios.

For very small files the difference in initial startup delay between Reed Solomon and Raptor is around a few seconds which is negligible due to the small file sizes. Using the analytical model and experimental results here we provide comparative observations between Raptor and Reed Solomon FEC protected progressive downloads at different transmission rates that are listed in Table 6.9, Table 6.10 and Table 6.11.

Table 6.9 Progressive download gain comparisions for MBMS at 256 Kbps.

File Size (KB)	PDU Loss (%)	Reed Solomon FEC Overhead (%)	Reed Solomon Waiting time in Progressive D.(sec)	Reed Solomon Waiting time in Legacy D.(sec.)	Reed Solomon Progressive Download Gain (%)	Raptor FEC Overhead (%)	Raptor Waiting time in Progressive D.(sec)	Raptor Waiting time in Legacy D.(sec.)	Raptor Progressive Download Gain (%)	Waiting time difference in Progressive D. (Reed Solomon - Raptor) (sec.)
100	1	12	0	3.4	100	8	0	3.3	100	0
100	5	23	0	3.7	100	21	0	3.6	100	0
100	10	43	0	4.0	100	35	0	3.8	100	0
512	1	7	0	18.5	100	4	0	18.0	100	0
512	5	19	0	20.9	100	12	0	19.7	100	0
512	10	37	0	23.5	100	22	0	21.0	100	0

Table 6.10 Progressive download gain comparisions for MBMS at 128 Kbps.

File Size (KB)	PDU Loss (%)	Reed Solomon FEC Overhead (%)	Reed Solomon Waiting time in Progressive D.(sec)	Reed Solomon Waiting time in Legacy D.(sec.)	Reed Solomon Progressive Download Gain (%)	Raptor FEC Overhead (%)	Raptor Waiting time in Progressive D.(sec)	Raptor Waiting time in Legacy D.(sec.)	Raptor Progressive Download Gain (%)	Waiting time difference in Progressive D. (Reed Solomon - Raptor) (sec.)
100	1	12	0.6	6.8	92	8	0.3	6.6	95	0.2
100	5	23	1.1	7.4	85	21	1.0	7.2	86	0.1
100	10	43	1.8	8.1	77	35	1.4	7.6	82	0.5
512	1	7	4.9	36.9	87	4	3.9	35.9	89	1.0
512	5	19	9.4	41.4	77	12	6.9	38.9	82	2.4
512	10	37	15.1	47.1	68	22	9.9	41.9	76	5.2

Table 6.11 Progressive download gain comparisions for MBMS at 64 Kbps.

File Size (KB)	PDU Loss (%)	Reed Solomon FEC Overhead (%)	Reed Solomon Waiting time in Progressive D.(sec)	Reed Solomon Waiting time in Legacy D.(sec.)	Reed Solomon Progressive Download Gain (%)	Raptor FEC Overhead (%)	Raptor Waiting time in Progressive D.(sec)	Raptor Waiting time in Legacy D.(sec.)	Raptor Progressive Download Gain (%)	Waiting time difference in Progressive D. (Reed Solomon - Raptor) (sec.)
100	1	12	7.5	13.8	45	8	7.1	13.3	47	0.5
100	5	32	9.0	15.3	41	22	7.9	14.1	44	1.2
100	10	53	11.3	17.5	36	39	9.7	15.9	39	1.6
512	1	8	42.5	74.5	43	4	39.7	71.7	45	2.8
512	5	21	53.3	85.3	38	14	48.4	80.4	40	4.9
512	10	41	67.9	99.9	32	26	57.3	89.3	36	10.6

The following is a summary of our observations for MBMS Progressive download discarding results for the 100 KB file:

- For higher transmission rates, higher gain from progressive download is obtained. Gain obtained for 256, 128 and 64 Kbps transmission rates are around 100%, 80% and 40% on the average respectively.
- In general, progressive download provides savings in waiting time from 36% up to 100% gain, up to 32 seconds with compared to legacy download.
- In general, progressive download using Reed Solomon provides savings in waiting time from 32% up to 100% gain, up to 32 seconds with compared to legacy download.
- Progressive download using Reed Solomon provides savings in waiting time from % 45 up to % 100 gain, up to 40 seconds, with compared to FEC optimized legacy download.

- Progressive download with Raptor has savings in initial startup delay from 7% up to 33%, which is up to 11 seconds with regards to progressive download with Reed Solomon.

6.4 Experimental Results for Interleaved Progressive Download Delivery.

This section shows the gain in initial startup delay as well as in FEC overhead when we use interleaving in progressive download delivery using Reed Solomon FEC protection. We did not study interleaving effect in progressive download for Raptor. The interleaved progressive download delivery is considered to be under downloading time optimization where service and network parameters are selected to minimize the download duration hence initial startup delay.

We tried to find the optimum interleaver-block size and hence made experiments for the interleaver-block size of 2, 3 and 4 source blocks. As a result we have caught best results with an interleaver-block size of 3 under both MBMS link conditions. We explored the MBMS network from the interleaving point of view by jointly analyzing the interleaving effect of the parameters listed in Table 6.1 on the minimum waiting time as well as its effect on the FEC overhead for 100% reliability.

The results provided us the gain in FEC overhead as well as waiting time from having application layer interleaving in MBMS network conditions. So using the analytical model and the experimental results we obtained comparative observations for Reed Solomon FEC protected interleaved download delivery that are provided in Table 6.12, Table 6.13, and Table 5.14. These tables also provide the overall results and gains for our four proposed MBMS download systems.

The following is a summary of our general observations for Reed Solomon FEC protected MBMS downloads discarding results for the 100 KB file:

- Gain from the interleaving can save FEC overhead up to 29% and save initial sartup delay up to 40% in progressive download delivery.

Table 6.12 Summary for Reed Solomon protected MBMS at 256 Kbps.

		Non-Interleaved			Interleaved					
File Size (KB)	PDU Loss (%)	FEC overhead (%)	Waiting time in Progressive D.(sec)	Waiting time in Legacy D.(sec)	FEC Overhead (%)	Waiting time in Interleaved Progressive D.(sec)	Waiting time in Interleaved Legacy D. (sec)	Gain in waiting time from Inter. Prog. D. w.r.t Legacy D. (%)	Gain in FEC Overhead from Interleaving (%)	Gain in initial startup delay from Interleaving (%)
100	1	12	0	3.4	12	0	3.4	100	0	0
100	5	23	0	3.7	22	0	3.7	100	4	0
100	10	43	0	4.0	36	0	4.0	100	16	0
512	1	7	0	18.5	5	0	17.6	100	29	0
512	5	19	0	20.9	16	0	19.1	100	16	0
512	10	37	0	23.5	32	0	21.0	100	14	0

Table 6.13 Summary for Reed Solomon protected MBMS at 128 Kbps.

		Non-Interleaved			Interleaved					
File Size (KB)	PDU Loss (%)	FEC overhead (%)	Waiting time in Progressive D.(sec)	Waiting time in Legacy D.(sec)	FEC Overhead (%)	Waiting time in Interleaved Progressive D.(sec)	Waiting time in Interleaved Legacy D. (sec)	Gain in waiting time from Inter. Prog. D. w.r.t Legacy D. (%)	Gain in FEC Overhead from Interleaving (%)	Gain in initial startup delay from Interleaving (%)
100	1	12	0.6	6.8	12	0.6	6.8	92	0	0
100	5	23	1.1	7.4	22	1.1	7.3	85	4	1
100	10	43	1.8	8.1	36	1.8	8.0	78	16	4
512	1	7	4.9	36.9	5	3.3	35.3	91	29	32
512	5	19	9.4	41.4	16	6.4	38.4	85	16	32
512	10	37	15.1	47.1	32	10.1	42.1	79	14	33

Table 6.14 Summary for Reed Solomon protected MBMS at 64Kbps.

		Non-Interleaved			Interleaved					
File Size (KB)	PDU Loss (%)	FEC overhead (%)	Waiting time in Progressive D.(sec)	Waiting time in Legacy D.(sec)	FEC Overhead (%)	Waiting time in Interleaved Progressive D.(sec)	Waiting time in Interleaved Legacy D. (sec)	Gain in waiting time from Inter. Prog. D. w.r.t Legacy D. (%)	Gain in FEC Overhead from Interleaving (%)	Gain in initial startup delay from Interleaving (%)
100	1	12	7.5	13.8	10	7.6	13.9	45	17	0
100	5	32	9.0	15.3	20	9.2	15.5	40	38	0
100	10	53	11.3	17.5	38	10.9	17.1	38	28	4
512	1	8	42.5	74.5	7	39.7	71.7	47	13	6
512	5	21	53.3	85.3	19	47.8	79.8	44	10	10
512	10	41	67.9	99.9	37	58.2	90.2	42	10	14

- Gain in waiting time from interleaved progressive download obtained for 256, 128 and 64 Kbps transmission rates are around 100%, 85% and 45% on the average respectively, when compared to legacy download.
- In general, interleaved progressive provides savings in waiting time up to 42 seconds, from % 42 up to % 100 gain with compared to legacy download.

- Interleaved Progressive download provides savings in waiting time up to 46 seconds, from % 49 up to % 100 gain, when compared to FEC optimized legacy download.

CHAPTER SEVEN

CONCLUSIONS

In this thesis we focused on 3GPP's MBMS download delivery with new novel methodologies applied on it. We provided a survey of the competing wireless multicast technologies with a focus on reliable download. We studied application layer solutions to increase the performance of the MBMS download delivery. Hence, we provided mechanisms to increase the user satisfaction of the download service such as waiting time. The mechanisms studied in this thesis to reduce the waiting time are the application layer interleaving mechanism and progressive download mechanism as well as optimizations. To the best of our knowledge these topics have not been studied in the literature and our work is providing a leading path for future research.

We analyzed and compared four MBMS download systems: a legacy download delivery, an interleaved download delivery, progressive download delivery and interleaved progressive download delivery under our optimizations. We considered downloading time and transmission cost optimization for both Reed Solomon and Raptor FEC protected MBMS systems. While our results for Reed Solomon are experimentally exact the results for Raptor are approximated using our analytical model more or less to see Raptor behavior against interleaving and progressive downloading mechanisms.

First we explored what benefits can be obtained from both optimizations in legacy download. We have explored MBMS from the download point of view by analyzing many combinations of service parameters as well as network parameters for an efficient download service under various MBMS link conditions. Then based on these optimizations, we provide experimental analyses to show the gain in using application layer interleaving in MBMS. Finally in order to further decrease the waiting time, progressive downloading is used with the interleaved download systems. Finally we provided performance comparisons of the legacy and the enhanced download deliveries

such as progressive download and interleaved progressive download under our optimizations.

We consider small 3G mobile media with 128 kbps constant media play rate. With our results, the optimizations provides around up to %10 gain in both FEC overhead and waiting time with compared to MBMS legacy download. In general, interleaving provides savings in FEC transmission cost up to 29% and provides savings in downloading time up to 10% in MBMS legacy download delivery and savings in initial startup delay up to 40% in MBMS progressive download delivery. We see that Reed Solomon FEC protected download system has more benefits from application layer interleaving. While MBMS progressive download provides 36% up to 100% gain and saves up to 32 seconds in waiting time, the MBMS interleaved progressive download delivery provides 45% to 100% gain and saves up to 40 seconds in waiting time with compared to MBMS legacy download.

We expect that progressive download support should be provided in MBMS. This work will pioneer to support progressive download in MBMS. The results of this study will provide guidelines to designers to fine-tune MBMS download service parameters for reliability and encourage new works on progressive download and application layer interleaving in MBMS. Our work will also be beneficial for 3G wireless multicast download and streaming service providers for identifying various issues, resolving them and optimizing the system performance.

Future Work

Our progressive download mechanism is a direct extension of the MBMS downloading mechanism. However, new mechanims can be used to increase the power of the progressive download in MBMS. Since we have no Raptor FEC codes available, our results for Raptor are approximated and might be far from the actual values. So we recommend using the Raptor FEC code implementation to get exact values. In our work, we used small file sizes and 128 Kbps media play rates, same work can be extended to the cases for high file sizes and different media play rates. The thesis

encourages novel approaches to support progressive download in MBMS, which will be dominant download method in future.

REFERENCES

About CrownCastle (n.d). Retrieved September 18, 2007, from http://www.crowncastle.com/about-us/index.aspx

Adamson B., Borrman C., Handley M., Macher J.(2004). *Negative-acknowledgment (NACK)-Oriented Reliable Multicast (NORM) Protocol*. Tech. Rep. RFC 3940. Retrieved September 19, 2007, from http://www.ietf.org/rfc/rfc3040.txt

Almeroth K.C. (2000). The Evolution of Multicast: From the Mbone to Interdomain Multicast to Internet2 Deployment, *IEEE Network*, Vol. 14, No. 1, pp:10-20.

BenQ Mobile (2006). *Support of Progressive Download in MBMS*. 3GPP TSG SA Meeting#39. S4-060267. Dallas, TX, USA.

Berrou C., Glavieux A., and Thitimajshaima P. (May 1993), Near Shannon limit error-correcting coding and decoding: turbo-codes. *In Proc. IEEE International Conference on Communiacation*., Vol. 2, pp. 1064–1070.

CNET Asia (n.d). *CNET's quick guide to 3G cell phone services* . Retrieved September 18, 2007, from http://asia.cnet.com/buyingguides/smartphones/0,39061031,61953931-5,00.htm

Deutsch P. (May 1996). *DEFLATE Compressed Data Format Specification version 1.3.* Tech. Rep. RFC 1951. Retrieved September 21, 2007, from http://www.ietf.org/rfc/rfc1951.txt

Deutsch P. (May 1996). *ZLIB Compressed Data Format Specification version 3.3.* Tech. Rep. RFC 1950. Retrieved September 21, 2007, from http://www.ietf.org/rfc/rfc1950.txt

Digital Fountain, Ericsson, NEC, Nokia, Nortel, Siemens (May 2004). *Permanent document on: simulation guidelines for the evaluation of FEC methods for MBMS download and streaming services*. 3GPP TSG SA Meeting#31. S4-060267. Canada. Retrieved September 21, 2007, from ftp://ftp.3gpp.org/tsg_sa/WG4_CODEC/TSGS4_31/Docs/S4-040348.zip

Diot C., Levine B. N.,Lyles B., Kassem H., Balensiefen D. (2000). Deployment Issues for the IP Multicast Service and Architecture. *IEEE Network*, Vol. 14, No. 1, pp: 78-88.

Dutta A. (2003). Multicasting Streaming Media to Mobile Users. *IEEE Communications Magazine*, Vol. 41, No. 10, pp: 81-89.

DVB-H (n.d). Retrieved September 19, 2007, from http://www.dvb-h.org/

Engebretsen L., Sudan M. (2002). Harmonic Broadcasting is Bandwidth-Optimal Assuming Constant Bit Rate, in *Proc. Annual ACM-SIAM Symposium on Discrete Algorithms*. San Francisco, CA, USA.

ETSI TS (2006). *IP Datacast over DVB-H: Content Delivery Protocol*. Retrieved September 19, 2007, from http://www.dvb-h.org/PDF/ts_102472v010201p.pdf

Gasiba T., Stockhammer T., Afzal J., Xu W. (2006). System Design and Advanced Receiver Techniques for MBMS Broadcast Services. *IEEE International Conference on Communications*. Vol. 12, pp: 5444-5450. Istanbul, Turkey.

Gossain H. (2002). *Multicast: Wired to Wireless*. Retrieved September 18, 2007, from http://citeseer.ist.psu.edu/gossain02multicast.html

Hu A. (2001), Video-on-Demand Broadcasting Protocols: A Comprehensive Study. *In Proc. IEEE Infocom*. Anchorage, Alaska.

IETF Reliable Multicast Transport Working Group (n.d), *Description of Working Group*. Retrieved September 19, 2007, from http://www.ietf.org/html.charters/rmt-charter.html

ITU-T Recommendation H.264 (2005). *Advanced video coding for generic audiovisual services*. ISO/IEC 14496.

Jenkac H., Stockhammer T., Xu W. (March 2006). Asynchronous and Reliable On-Demand Media Broadcast. *IEEE Network Magazine, Special Issue on Multimedia over Wireless Broadband Networks*, Vol. 20, No. 2, pp. 14-20.

Jenkac H., Stockhammer T., Xu W., Abdel Samad W. (May 2006). Efficient Video-on-Demand Services over Mobile Datacast Channel. *Journal of Zhejiang University Science, Special Issue for Selected Papers (Part I) of 15th International Packet Video Workshop (PV2006),* Vol. 7, No. 5, pp. 873-884.

Li X., Ammar M. and Paul S. (1999). *Video Multicast over Internet*. Retrieved September 18, 2007, from http://citeseer.ist.psu.edu/li99video.html

Liu J., Li. B & Zhang Y. (2003). *Adaptive Video Multicast over the Internet*. IEEE Multimedia. Vol. 10, No. 1, pp. 22-33.

Luby M. (2005). *Application Layer Fec Erasure Codes and Cellular Multicast/Broadcast Standards*. Digital Fountain, Algo+lma Internal Seminars, Retrieved September 19, 2007, from http://algo.epfl.ch/contents/group/seminar/Luby1005.pdf

Luby M., and Goyal V.(April 2004). *Wave and Equation Based Rate Control (WEBRC) Building Block*. Tech. Rep. RFC 3738. Retrieved September 21, 2007, from http://www.ietf.org/rfc/rfc3738.txt

Luby M., Gemmell J., Vicisano L., Rizzo L., Handley M., and Crowcroft J.(2002). *Layered Coding Transport (LCT) Building Block*. Tech. Rep. RFC 3451. Retrieved September 19, 2007, from http://www.ietf.org/rfc/rfc3451.txt

Luby M., Gemmell J., Vicisano L., Rizzo L., Handley M., and Crowcroft J.(2002). *Forward Error Correction (FEC) Building Block*. Tech. Rep. RFC 3452. Retrieved September 19, 2007, from http://www.ietf.org/rfc/rfc3452.txt

Luby M., Gemmell J., Vicisano L., Rizzo L., Handley M., and Crowcroft J.(2002). *Asynchronuous Layered Coding (ALC) Protocol Instantiation*. Tech. Rep. RFC 3450. Retrieved September 19, 2007, from http://www.ietf.org/rfc/rfc3450.txt

Luby M., Vicisano L., Gemmell J., Rizzo L., Handley M., and Crowcroft J.(2002). *The Use of Forward Error Correction (FEC) in Reliable Multicast.* Tech. Rep. RFC 3453. Retrieved September 21, 2007, from http://www.ietf.org/rfc/rfc3453.txt

Luby M., Watson M., Gasiba T., Stockhammer T. (October 2006), Mobile Data Broadcasting over MBMS Tradeoffs in Forward Error Correction. *In Proc. ACM ; Proceedings of the 5th international conference on Mobile and ubiquitous multimedia* Vol. 193, Article No. 10, Stanford, California.

Luby M., Watson M., Gasiba T., Stockhammer T., and Xu W. (January 2006). Raptor Codes for Reliable Download Delivery in Wireless Broadcast Systems. *CCNC 2006. IEEE* Vol. 1, pp. 192-197.

Mark Watson and Thomas Stockhammer (n.d). *The MBMS Mobile Multimedia Standard and Application-Layer Forward Error Correction.* Retrieved September 21, 2007, from http://www.wirelessdesignmag.com/ShowPR.aspx?PUBCODE=055&ACCT=0031786&ISSUE=0610&RELTYPE=PR&Cat=13&SubCat=1&PRODCODE=M0210&PRODLETT=A&CALLFROM=PR&CommonCount=0

Microsoft (2003). *The Reliable Multicast Protocol Component of Windows Server 2003* Retrieved September 19, 2007, from http://www.microsoft.com/technet/community/columns/cableguy/cg0603.mspx

Mobile TV WG (2006). *Mobile TV: The Groundbreaking Dimension*. UMTSF/GSMA Joint Mobile TV Work Group Report Version 2.21.

Mukhtar R. G. (2003). Efficient Internet Traffic Delivery over Wireless Networks. *IEEE Communications Magazine*, Vol. 41, No. 12, pp. 46-53.

Nortel Networks (April 2004). *Mapping of SDUs to Radio Blocks for FEC simulations*. S4-AHP120, 3GPP TSG SA WG#4, Lund, Sweden. Retrieved September 23, 2007, from http://www.3gpp.org/ftp/tsg_sa/WG4_CODEC/Ad-hoc_PSM/Docs/S4-AHP120.zip

Obraczka K. (1998). Multicast Transport Protocols: A Survey and Taxonomy. *IEEE Communications Magazine*, Vol. 36, No. 1, pp: 94-102.

OMA BCAST (n,d). Retrieved September 15, 2007, from http://www.openmobilealliance.org/tech/wg_committees/bcast.html

OMA BCAST (2004). *OMA Mobile Broadcast Services Architecture (OMA-AD_BCAST-V1_0_0-20041101-D)* Draft Version 1.0. Retrieved September 19, 2007, from http://member.openmobilealliance.org/ftp/Public_documents/bcast/Permanent_documents/OMA-AD_BCAST-V1_0_0-20041101-D.zip

Paila T., Lubby M., Lehtonen R., Roca V., Walsh R. (2004). *FLUTE, File Delivery over Unidirectional Transport*. RFC 3926.

Peltotalo J., Peltotalo S., Harju J. (July 2005). Analysis of the FLUTE Data Carousel. *In Proc. 10th EUNICE Open European Summer School*, Colmenarejo, Spain, pp. 138 – 142.

Rekh S., Rani S.S., Shanmugam A. (2005). Optimal Choice of Interleaver for Turbo Codes. *Academic Open Internet Journal*. Vol. 15. Retrieved September 21, 2007, from http://www.acadjournal.com/2005/v15/part6/p7/

Rizzo L. (1998). *Reed-Solomon FEC codec (revised version of July 2nd, 1998)*, Retrieved September 15, 2007, from http://info.iet.unipi.it/~luigi/vdm98/vdm980702.tgz

QualComm (n.d). Retrieved September 19, 2007, from www.qualcomm.com

Shokrollahi A.(2003). *Raptor codes, Digital Fountain*, Tech. Rep. DR2003-06-001.

Siemens (March 2005). *Efficient FEC code for reliable MBMS user services*. 3GPP TSG SA Meeting#27. SP-050164. Tokyo, Japan.

Siemens AG (August 2006). *Content Preloading for MBMS User Services*. 3GPP TSG SA Meeting#40. S4-060385. Sophia Antipolis, France. Retrieved September 21, 2007, from http://www.3gpp.org/ftp/tsg_sa/WG4_CODEC/TSGS4_40/Docs/S4-060385.zip

UMTS Forum. (2005). *3G/UMTS Towards mobile broadband and personal Internet*, Retrieved September 15, 2007, from http://www.3gpp.org/ftp/PCG/PCG_14/DOCS/PDF/PCG14_16.pdf

Varshley U. (1999). Architectural Issues to Support Multicasting over Wireless and Mobile Networks. *IEEE Wireless Communications and Networking Conference* (WCNC), Vol. 1, pp. 41-45.

Varshley U. (2002). Multicast over Wireless Networks. *Communications of the ACM*, Vol: 45, No:12, pp:31-37.

Vidiator Technology US (n.d), www.vidiator.com

Walsh R., Peltotalo J., Peltotalo S., Mehta H., Curcio I., (January 2006). *SDP Descriptors for FLUTE.* IETF, draft-mehta-rmt-flute-sdp-05.txt. Retrieved September 21, 2007, from http://www.ietf.org/internet-drafts/draft-mehta-rmt-flute-sdp-05.txt

Wan T., Subramanian R. K. (2004). Implementing QoS Support for Wireless Multicast Nodes. *Proceedings International Conference on Communications (ICC)*. Vol. 7, pp. 3974- 3978.

Wang J.,Sinnarajah R., Chen T.,Wei Y., and Tiedemann E. (2004). Broadcast and Multicast Services in cdma2000. *IEEE Communications Magazine*. Vol. 42, No. 2, pp: 76-82.

Welzl M., and Eddy V.(March 2007). *Congestion Control in the RFC Series.* IETF, draft-irtf-iccrg-cc-rfcs-01. Retrieved September 21, 2007, from http://www3.tools.ietf.org/html/draft-irtf-iccrg-cc-rfcs-01

Yetgin Z., Seckin G. (2007). Progressive Download for 3G Wireless Multicasting. *IEEE CS 2007 International Conference on Future Generation Communication and Networking (FGCN 2007)* in Jeju Island, Korea.

Yetgin Z., Seckin G. (April 2008), Progressive Download for 3G Wireless Multicasting, *International Journal of Hybrid Information Technology (IJHIT) Journal* Vol.1 No.2.

Yetgin Z., Seckin G. (March 2008), Progressive Download for Multimedia Broadcast Multicast Services, accepted to be published in *IEEE Multimedia Journal*.

Yetgin Z., Seckin G. (October 2007). Reliable Download Analyses for Multimedia Broadcast Multicast Services, *IAENG The World Congress on Engineering and Computer Science*, San Francisco, USA.

3GPP (n.d). Retrieved September 15, 2007, from http://www.3gpp.org/About/about.htm

3GPP2 (n.d). Retrieved September 15, 2007, from http://www.3gpp2.org/Public_html/Misc/AboutHome.cfm

3GPP2 BCMCS (2005). *Broadcast and Multicast Service in cdma2000 Wireless IP Network*. X.S0022, Retrieved September 15, 2007, from http://www.3gpp2.org/Public_html/specs/tsgx.cfm

3GPP TS 26.234 (2007*). Transparent end-to-end Packet-switched Streaming Service (PSS); Protocols and codecs*. Retrieved September 22, 2007, from http://www.3gpp.org/ftp/Specs/2007-06/Rel-7/26_series/26234-731.zip

3GPP TSG SA WG#4 (June 2005). *Report of MBMS FEC Status*. SP-050246, 3GPP TSG SA Meeting #28, Quebec, Canada. Retrieved September 23, 2007, from http://www.3gpp.org/ftp/TSG_SA/TSG_SA/TSGS_28/Docs/PDF/SP-050246.pdf

3GPP TSG 26.346 (2007). *Multimedia Broadcast/Multicast Service, Protocols and Codecs.* Retrieved September 19, 2007, from http://www.3gpp.org/ftp/Specs/2007-06/Rel-7/26_series/26346-740.zip

3GPP TSG 26.946 (2007). *Multimedia Broadcast/Multicast Service (MBMS) user service guidelines.* Retrieved September 19, 2007, from http://www.3gpp.org/ftp/Specs/2007-06/Rel-7/26_series/26946-700.zip

APPENDIX A

TEMPORAL ANALYSES

A.1 Detailed Transmission Cost Analyses

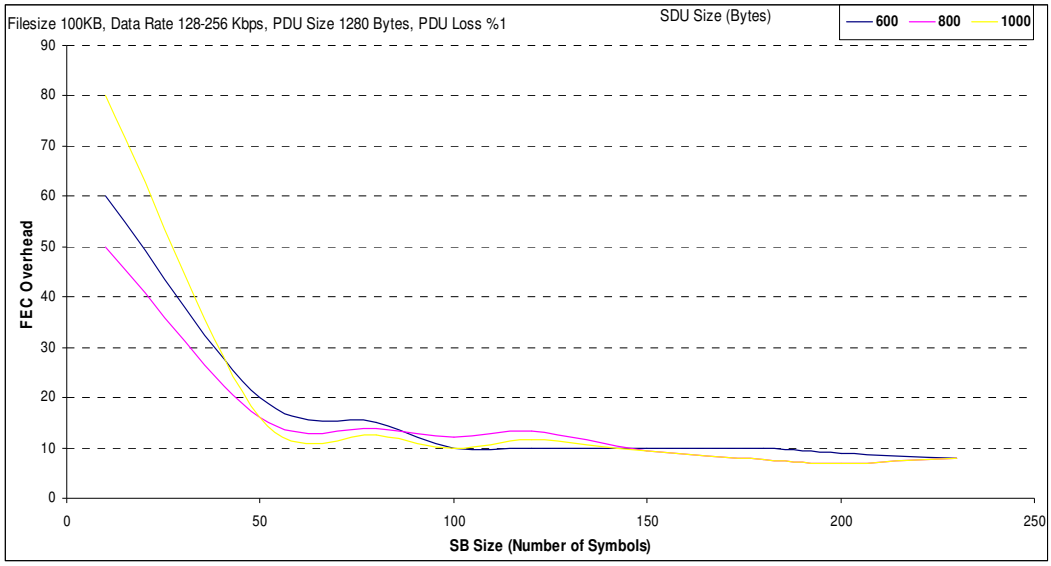


Figure A.1 Transmission cost analyses for Filesize 100KB, Data Rate 128-256 Kbps, PDU Size 1280 Bytes, PDU Loss %1.

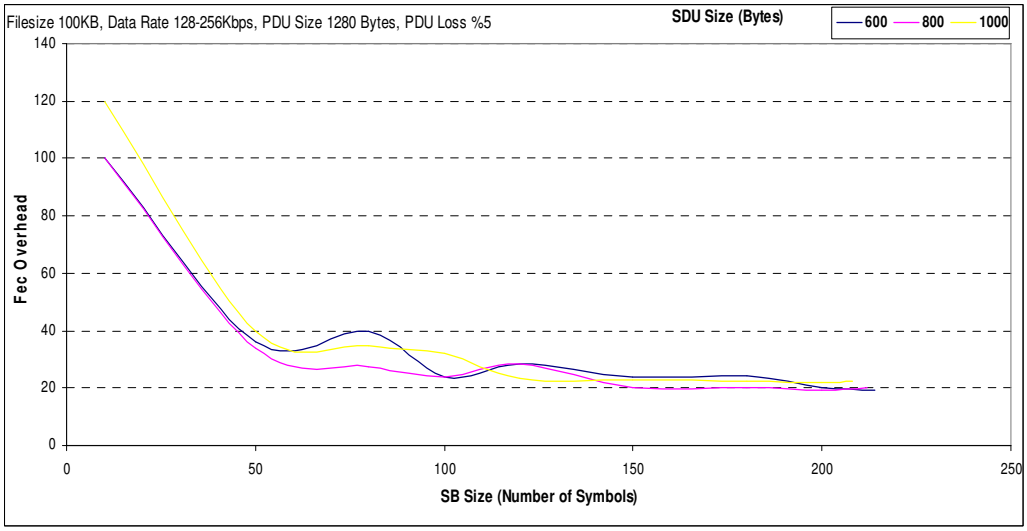


Figure A.2 Transmission cost analyses for Filesize 100KB, Data Rate 128-256 Kbps, PDU Size 1280 Bytes, PDU Loss %5

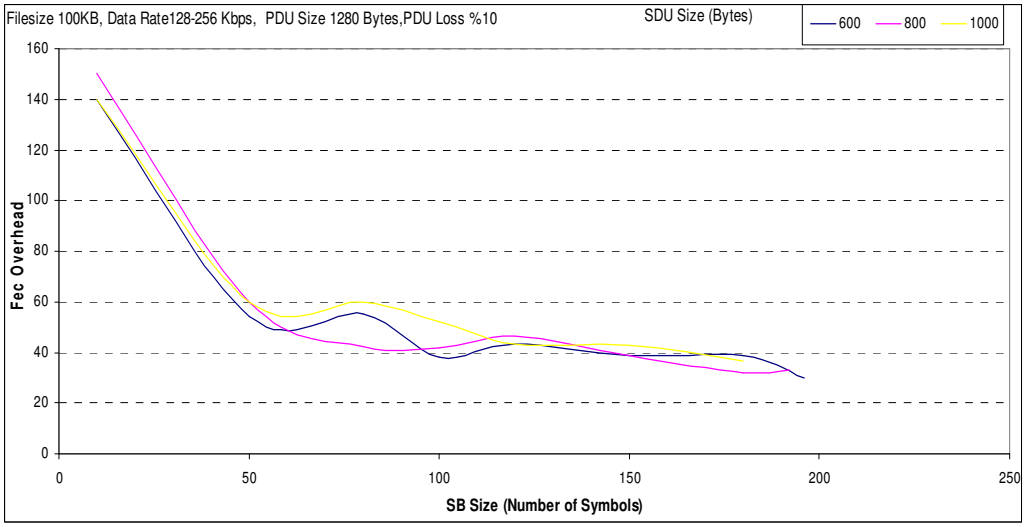


Figure A.3 Transmission cost analyses for Filesize 100KB, Data Rate 128-256 Kbps, PDU Size 1280 Bytes, PDU Loss %10.

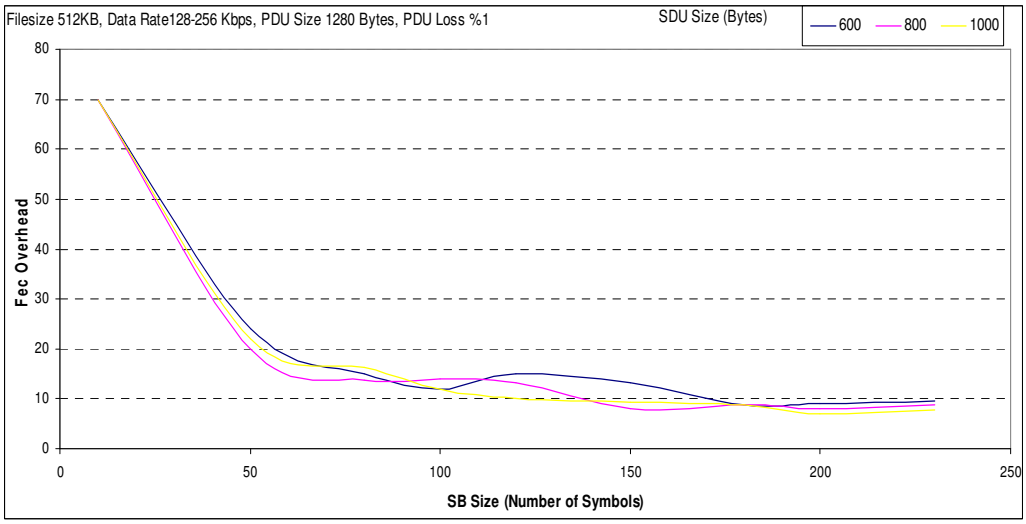


Figure A.4 Transmission cost analyses for Filesize 512KB, Data Rate 128-256 Kbps, PDU Size 1280 Bytes, PDU Loss %1.

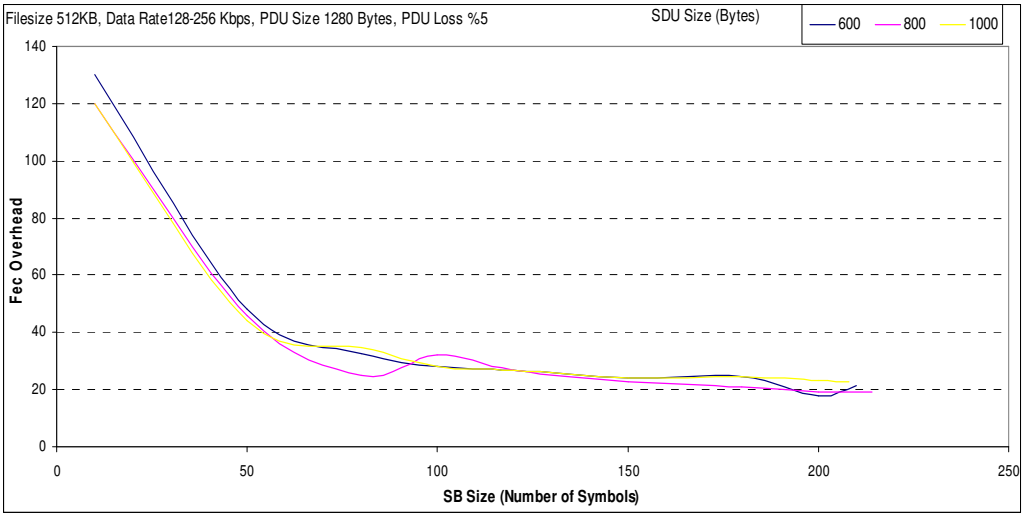


Figure A.5 Transmission cost analyses for Filesize 512KB, Data Rate 128-256 Kbps, PDU Size 1280 Bytes, PDU Loss %5.

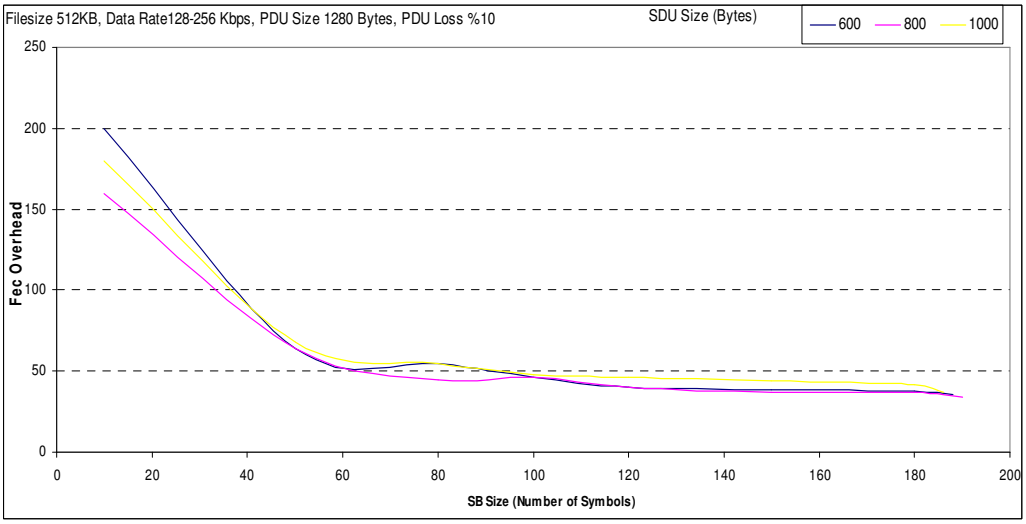


Figure A.6 Transmission cost analyses for Filesize 512KB, Data Rate 128-256 Kbps, PDU Size 1280 Bytes, PDU Loss %10.

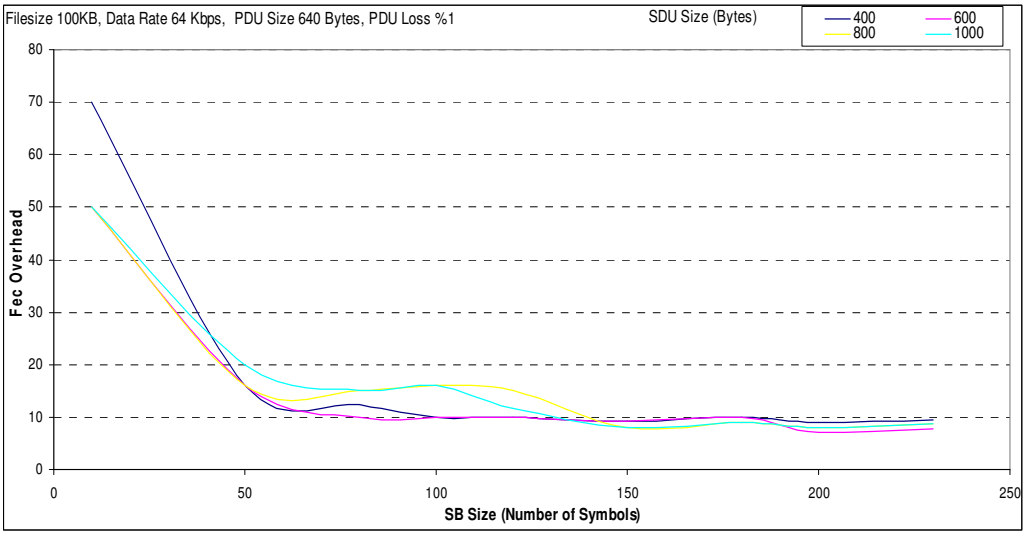


Figure A.7 Transmission cost analyses for Filesize 100KB, Data Rate 64 Kbps, PDU Size 640 Bytes, PDU Loss %1.

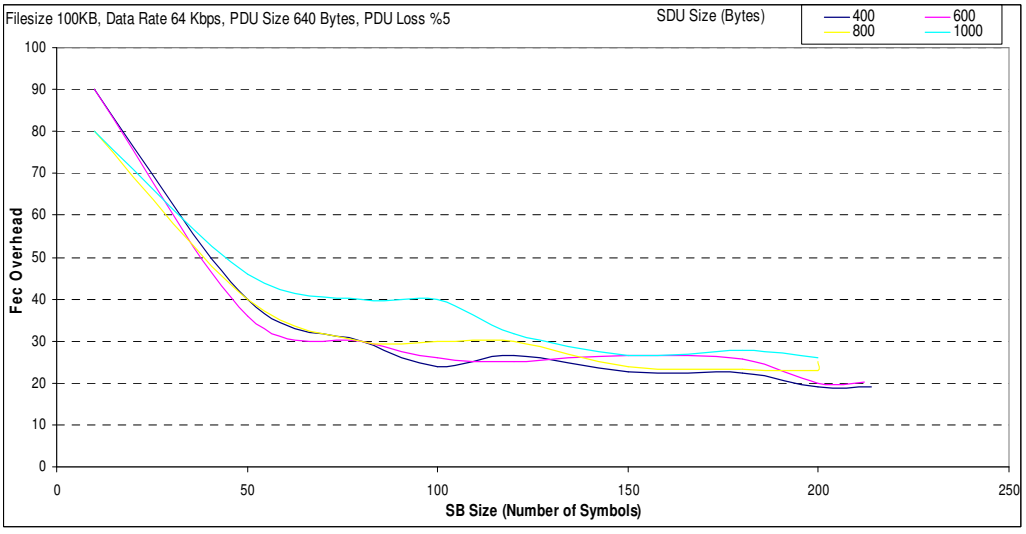


Figure A.8 Transmission cost analyses for Filesize 100KB, Data Rate 64 Kbps, PDU Size 640 Bytes, PDU Loss %5.

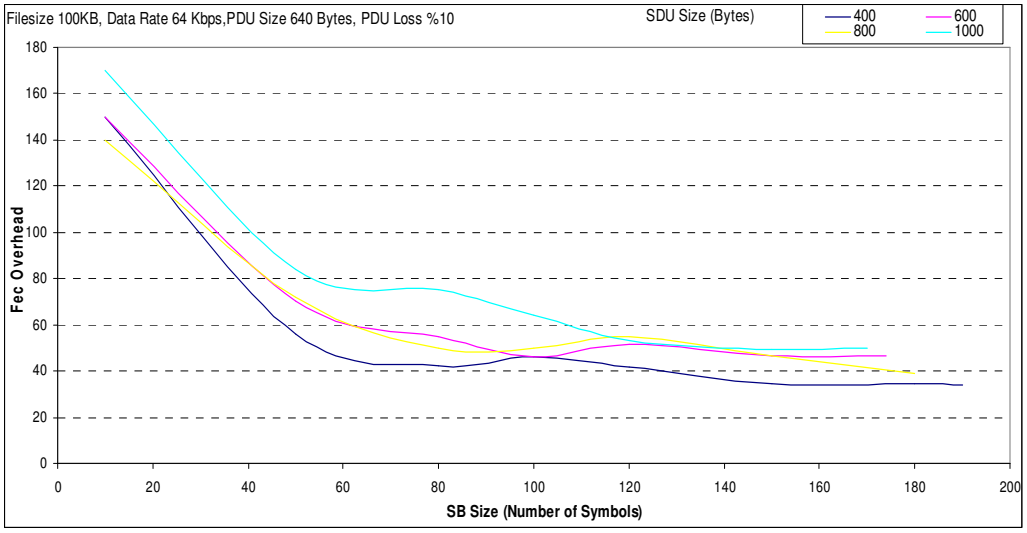


Figure A.9 Transmission cost analyses for Filesize 100KB, Data Rate 64 Kbps, PDU Size 640 Bytes, PDU Loss %10.

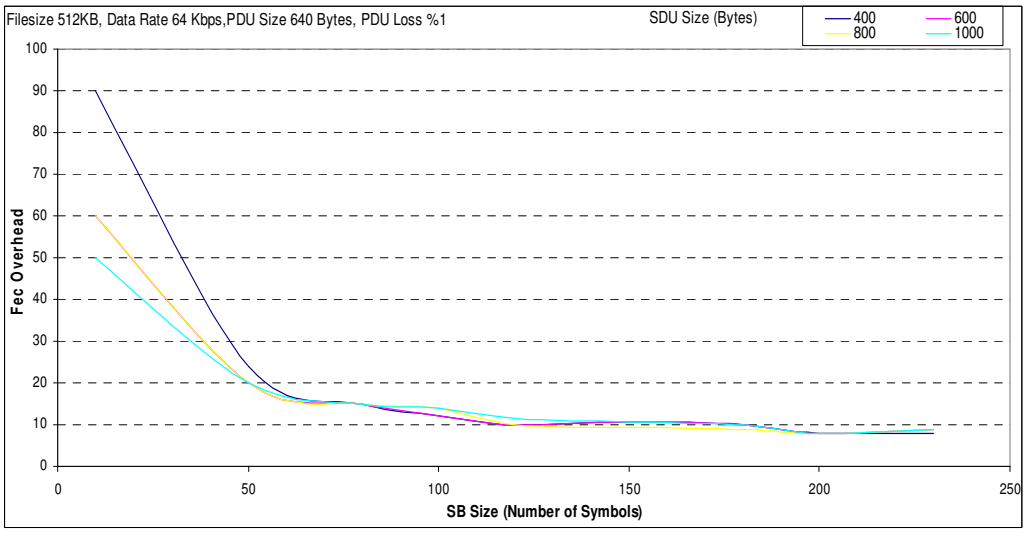


Figure A.10 Transmission cost analyses for Filesize 512KB, Data Rate 64 Kbps, PDU Size 640 Bytes, PDU Loss %1.

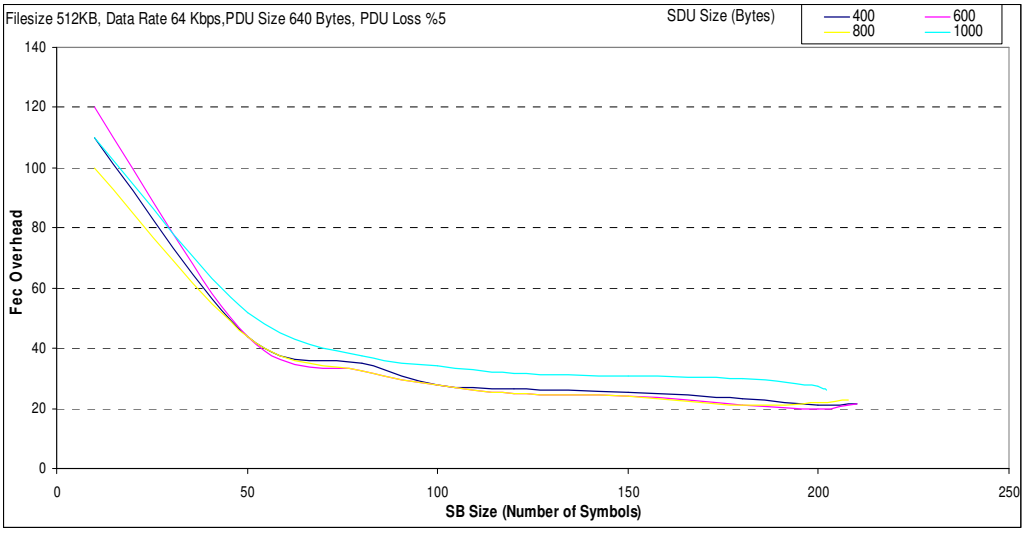


Figure A.11 Transmission cost analyses for Filesize 512KB, Data Rate 64 Kbps, PDU Size 640 Bytes, PDU Loss %5.

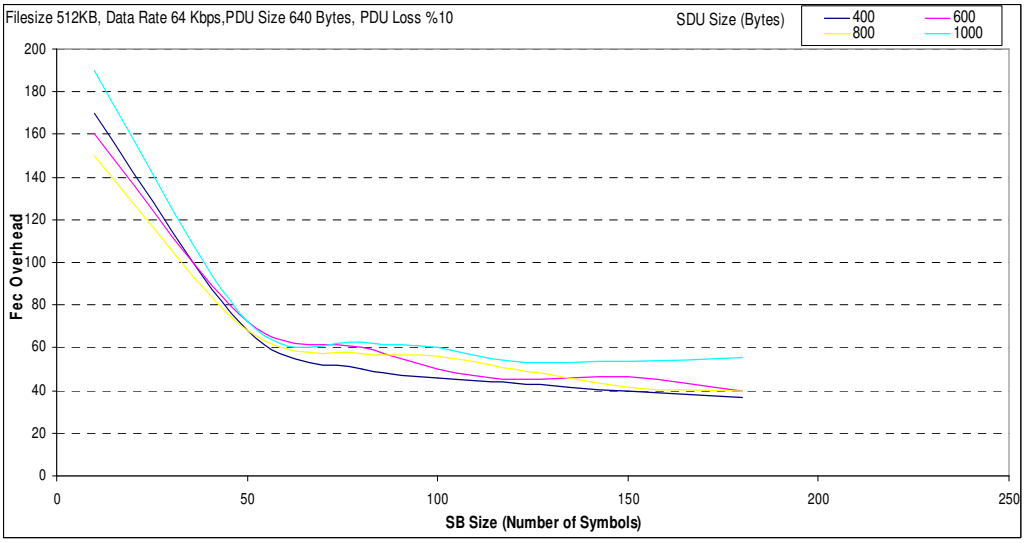


Figure A.12 Transmission cost analyses for Filesize 512KB, Data Rate 64 Kbps, PDU Size 640 Bytes, PDU Loss %10.

A.2 Detailed Initial Startup Time Analyses

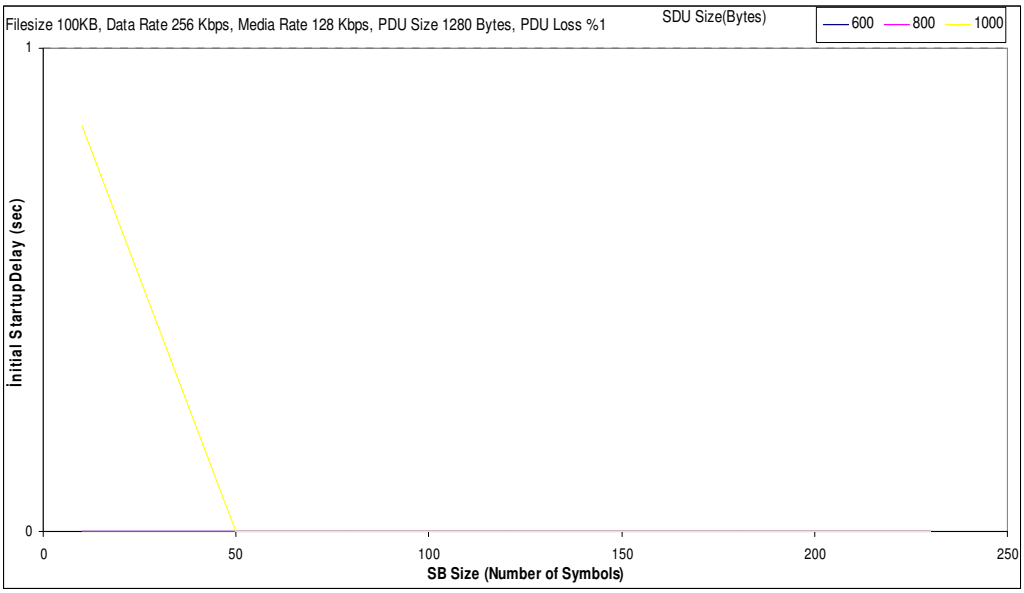


Figure A.13 Initial startup time analyses for Filesize 100KB, Data Rate 256 Kbps, PDU Size 1280 Bytes, PDU Loss %1.

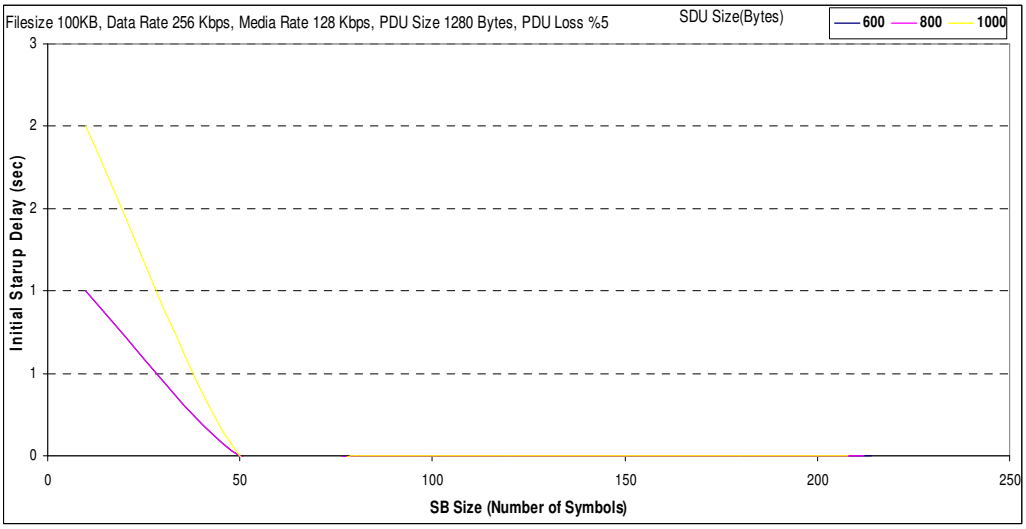


Figure A.14 Initial startup time analyses for Filesize 100KB, Data Rate 256 Kbps, PDU Size 1280 Bytes, PDU Loss %5.

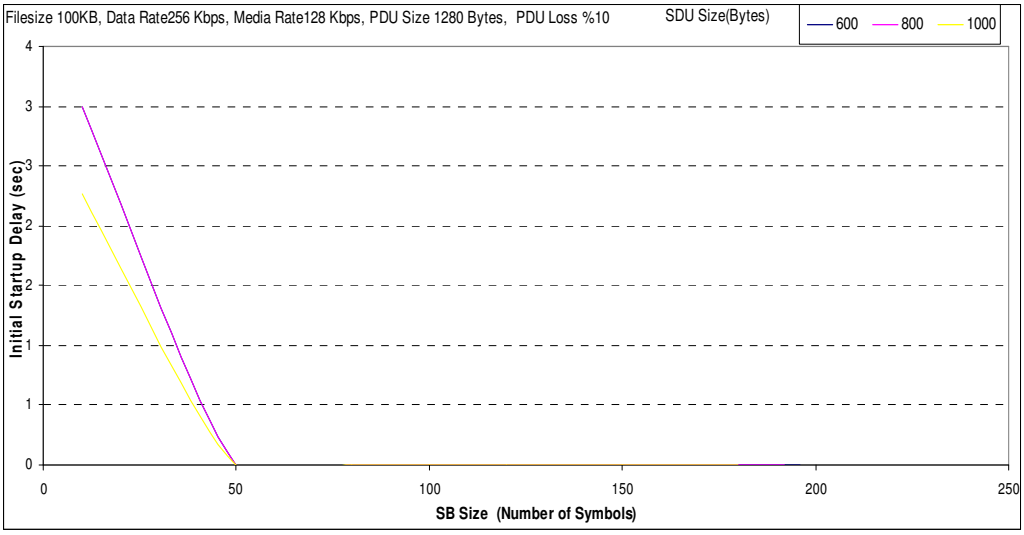


Figure A.15 Initial startup time analyses for Filesize 100KB, Data Rate 256 Kbps, PDU Size 1280 Bytes, PDU Loss %10.

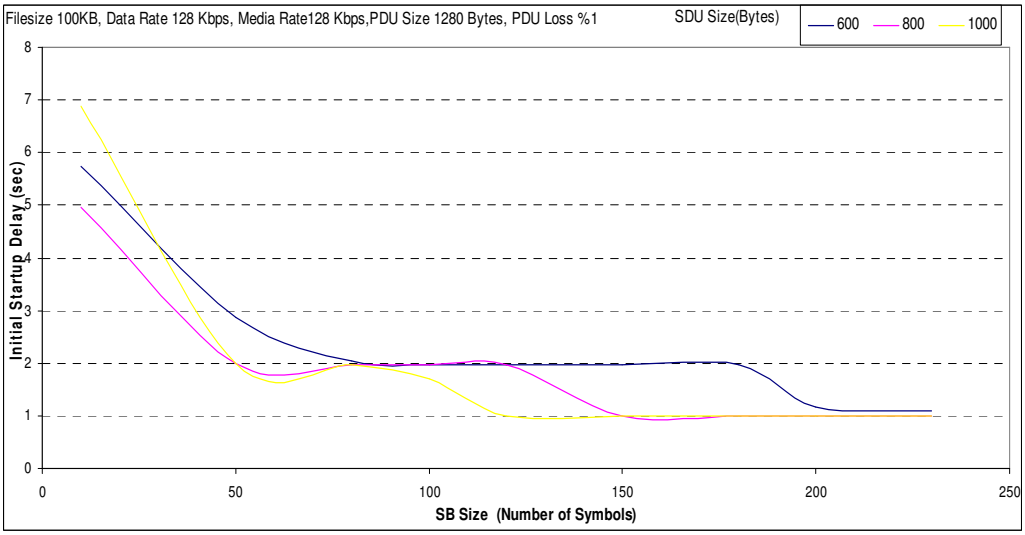


Figure A.16 Initial startup time analyses for Filesize 100KB, Data Rate 128 Kbps, PDU Size 1280 Bytes, PDU Loss %1.

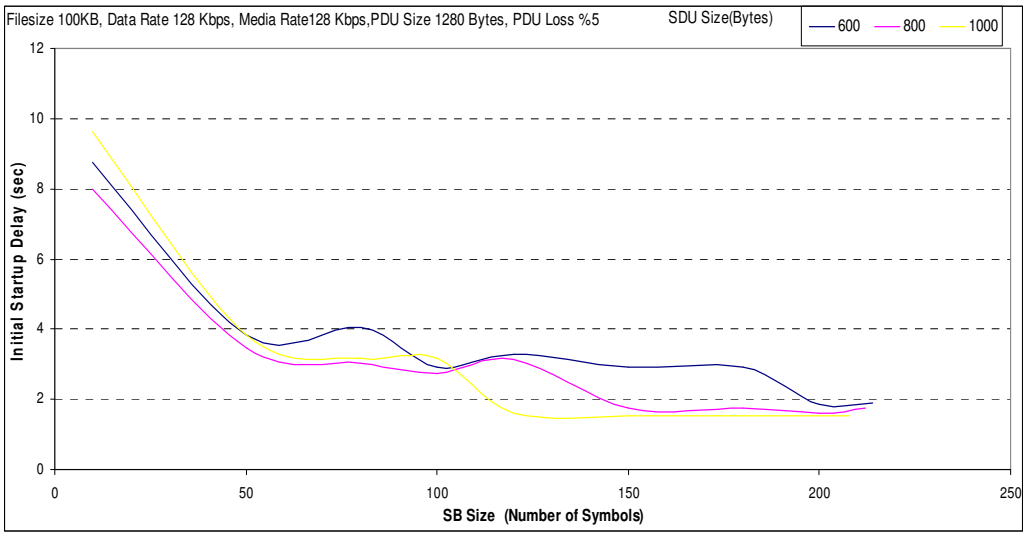


Figure A.17 Initial startup time analyses for Filesize 100KB, Data Rate 128 Kbps, PDU Size 1280 Bytes, PDU Loss %5.

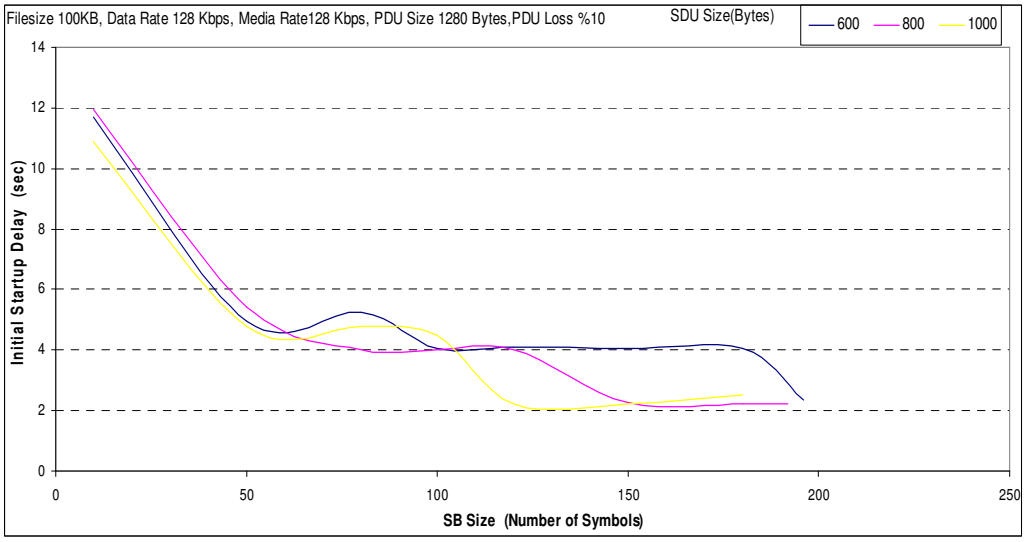


Figure A.18 Initial startup time analyses for Filesize 100KB, Data Rate 128 Kbps, PDU Size 1280 Bytes, PDU Loss %10.

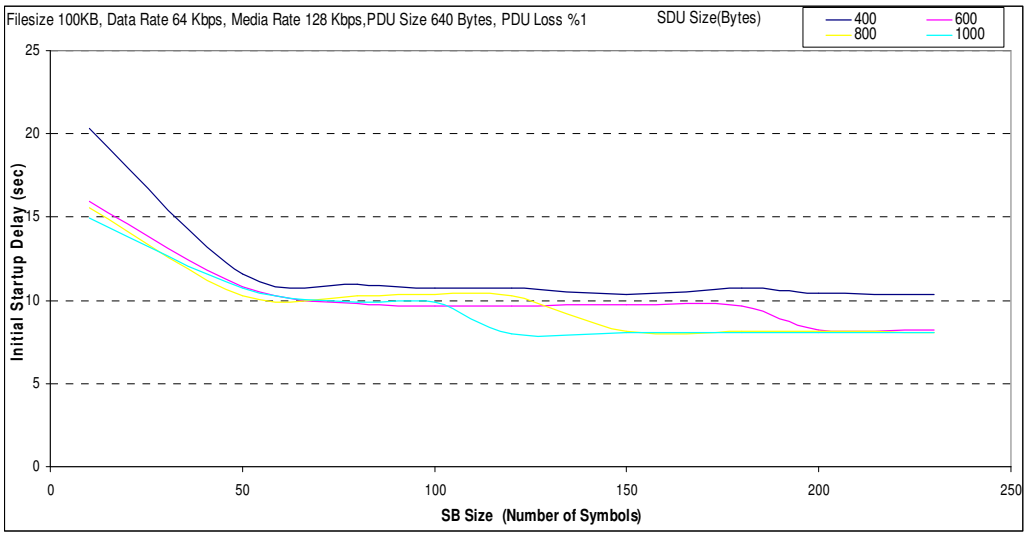


Figure A.19 Initial startup time analyses for Filesize 100KB, Data Rate 64 Kbps, PDU Size 640 Bytes, PDU Loss %1.

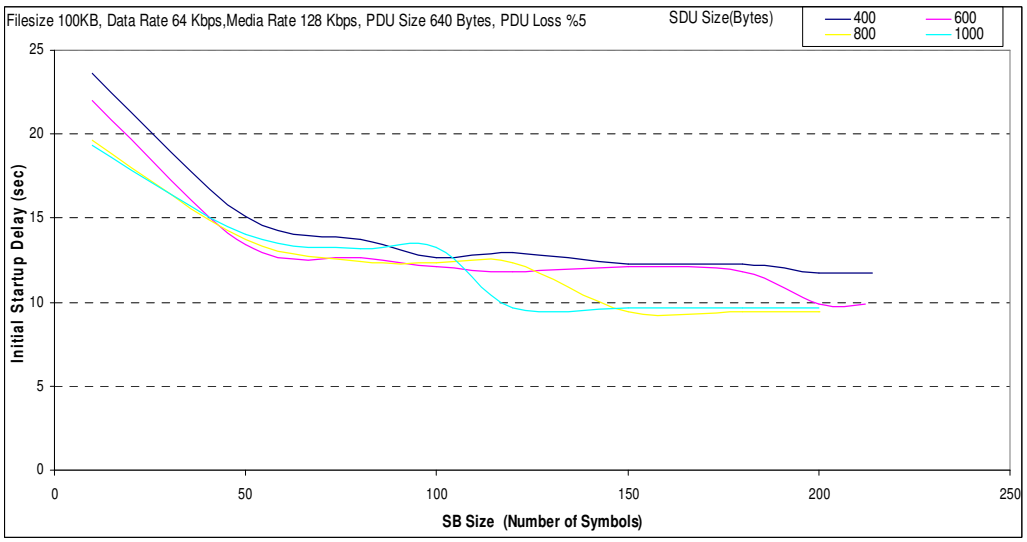


Figure A.20 Initial startup time analyses for Filesize 100KB, Data Rate 64 Kbps, PDU Size 640 Bytes, PDU Loss %5.

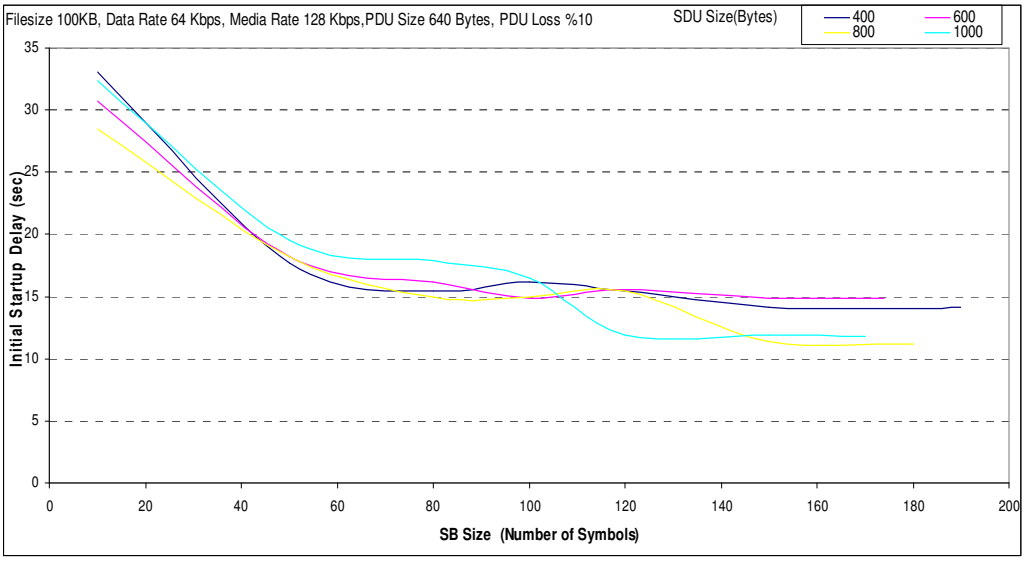


Figure A.21Initial startup time analyses for Filesize 100KB, Data Rate 64 Kbps, PDU Size 640 Bytes, PDU Loss %10.

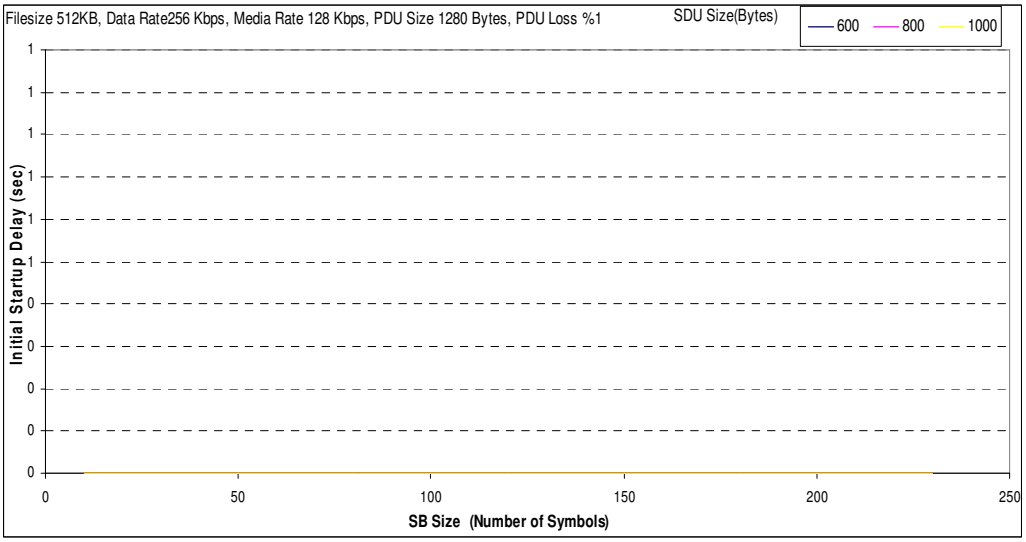


Figure A.22 Initial startup time analyses for Filesize 512KB, Data Rate 256 Kbps, PDU Size 1280 Bytes, PDU Loss %1.

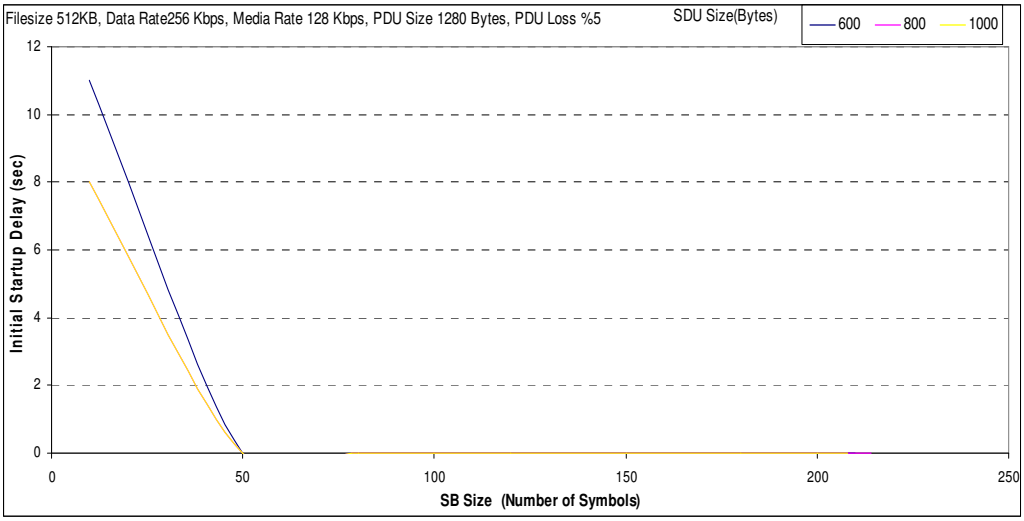


Figure A.23 Initial startup time analyses for Filesize 512KB, Data Rate 256 Kbps, PDU Size 1280 Bytes, PDU Loss %5.

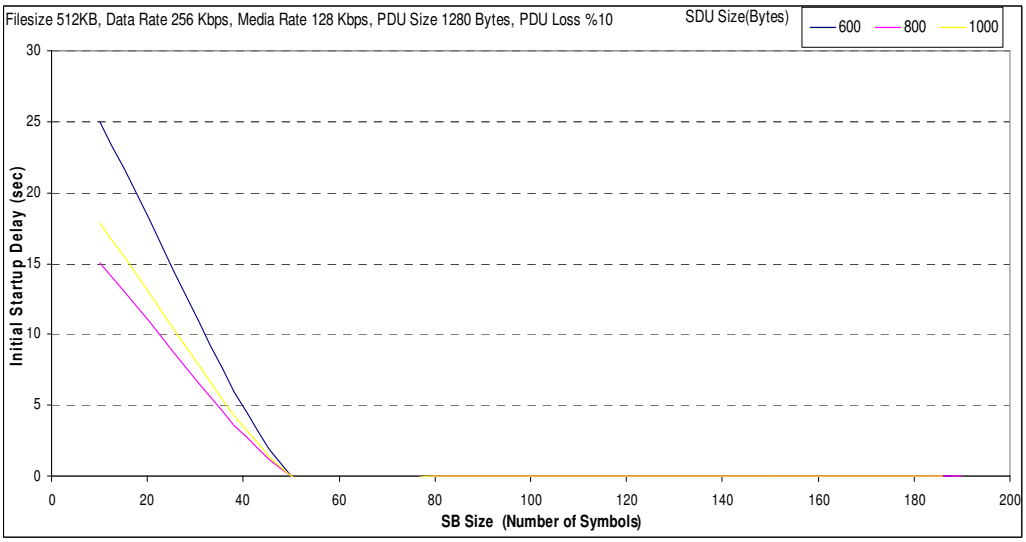


Figure A.24 Initial startup time analyses for Filesize 512KB, Data Rate 256 Kbps, PDU Size 1280 Bytes, PDU Loss %10.

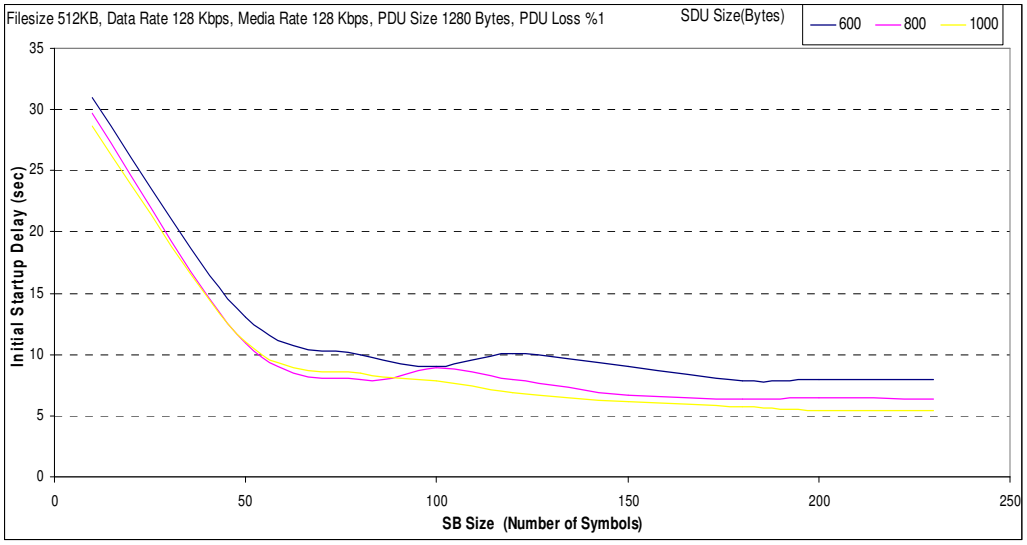


Figure A.25 Initial startup time analyses for Filesize 512KB, Data Rate 128 Kbps, PDU Size 1280 Bytes, PDU Loss %1.

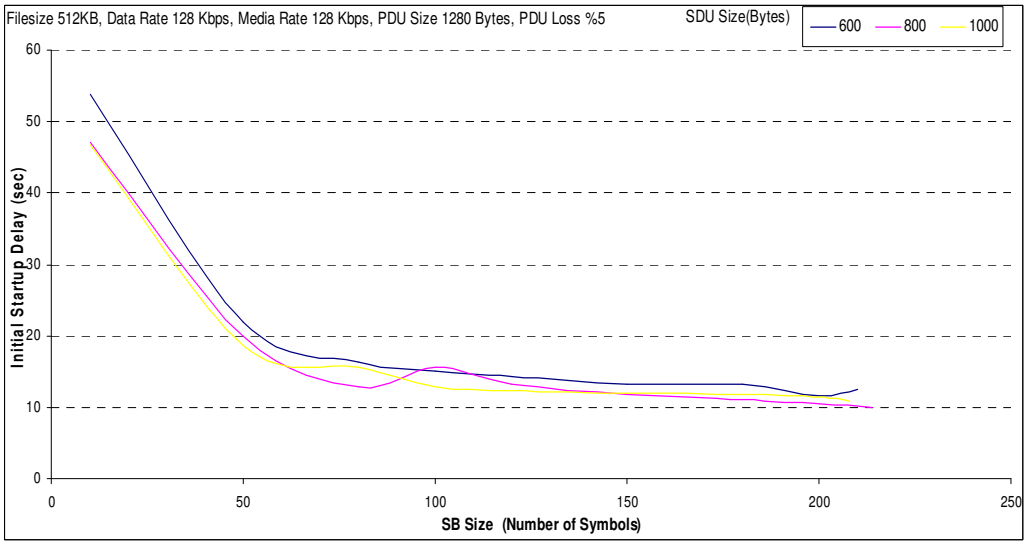


Figure A.26 Initial startup time analyses for Filesize 512KB, Data Rate 128 Kbps, PDU Size 1280 Bytes, PDU Loss %5.

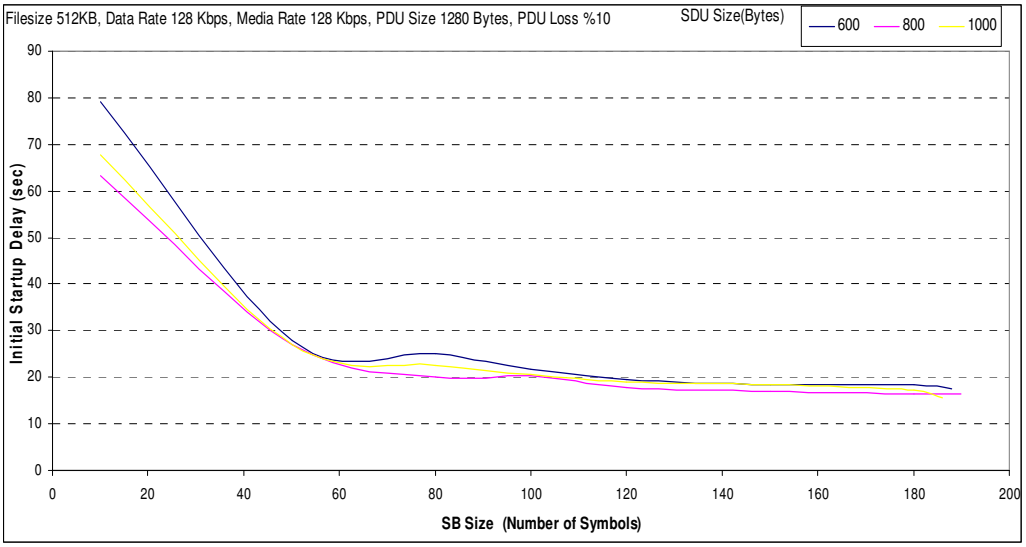


Figure A.27 Initial startup time analyses for Filesize 512KB, Data Rate 128 Kbps, PDU Size 1280 Bytes, PDU Loss %10.

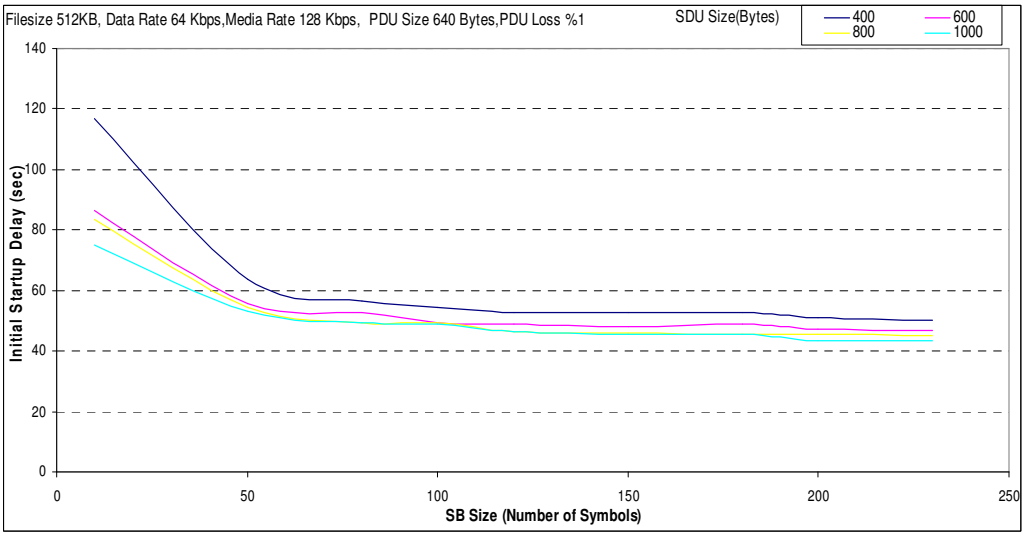


Figure A.28 Initial startup time analyses for Filesize 512KB, Data Rate 64 Kbps, PDU Size 640 Bytes, PDU Loss %1.

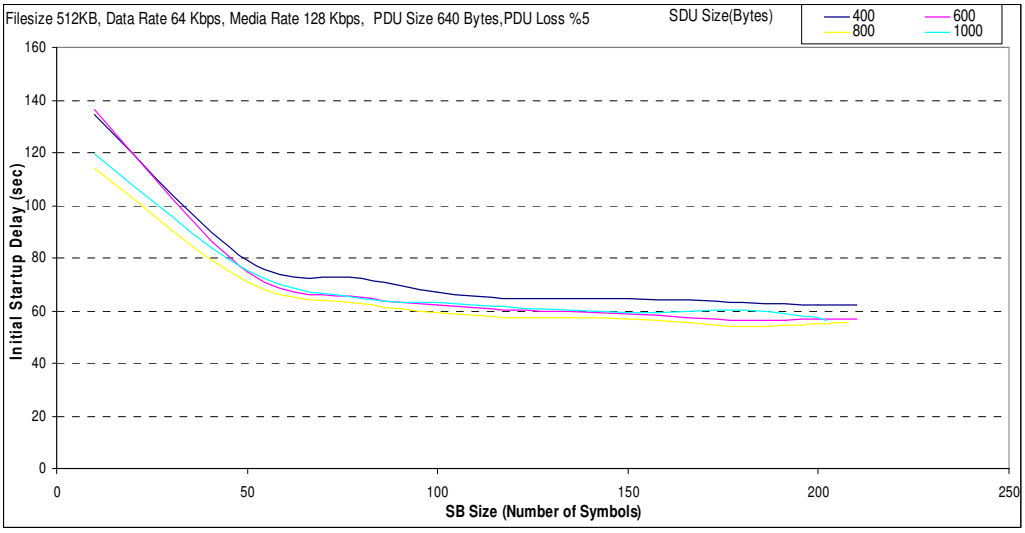


Figure A.29 Initial startup time analyses for Filesize 512KB, Data Rate 64 Kbps, PDU Size 640 Bytes, PDU Loss %5.

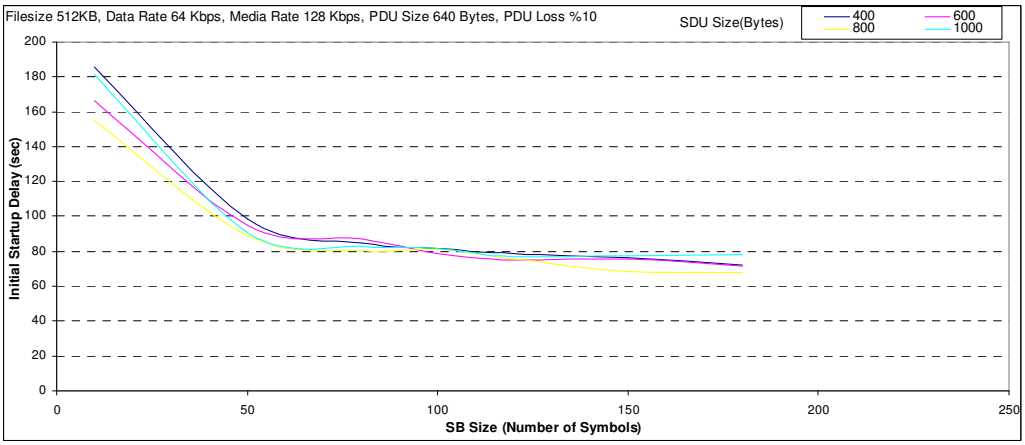


Figure A.30 Initial startup time analyses for Filesize 512KB, Data Rate 64 Kbps, PDU Size 640 Bytes, PDU Loss %10.

APPENDIX B

PROTOTYPE

B.1 Prototype Service Modules

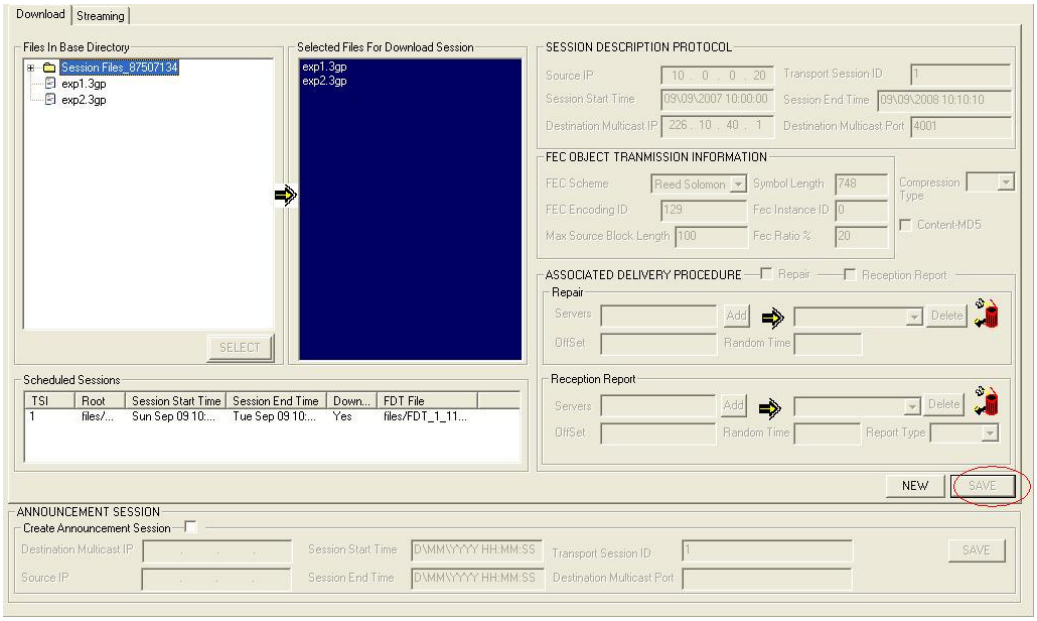


Figure B.1 User interface of the download service of the prototype when a new download service is scheduled.

Figure B.1 shows the creation of a download service that will distribute two multimedia files “exp1.3gp” and “exp2.3gp” to a multicast group identified by Destination Multicast IP and Transport Session Identifier.

Figure B.2 shows creation of an announcement service for the download service newly scheduled. This announcement service will distribute the service description metadata of the actual download service to the multicast group identified by Destination Multicast IP and Transport Session ID in announcement section of Figure B.2.

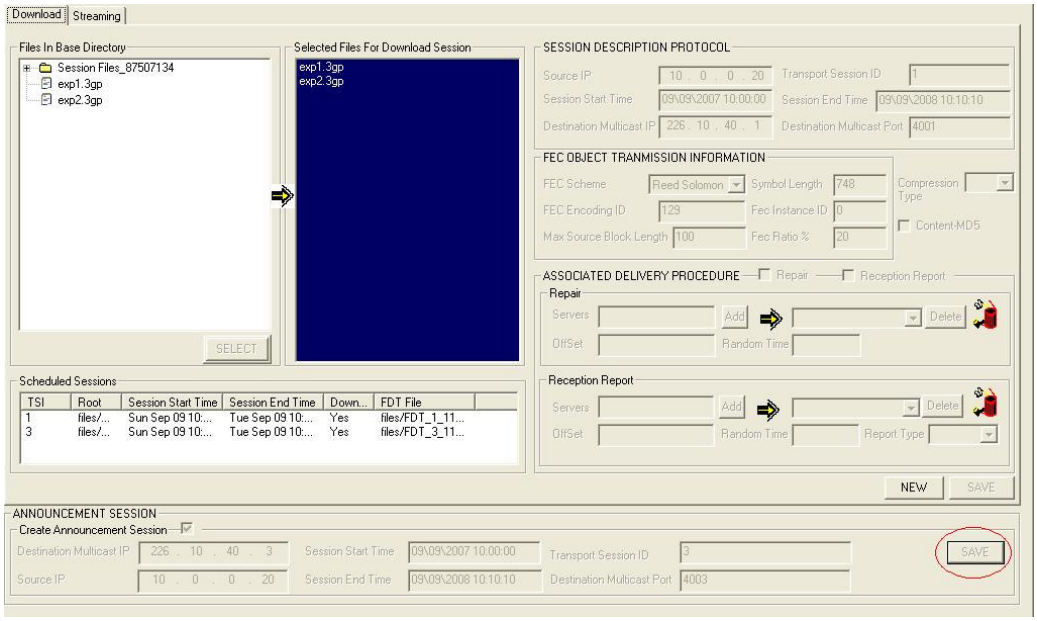


Figure B.2 User interface of the download service of the prototype when an announcement service is scheduled.

Figure B.3 Scheduled services in services database prior to sessions start.

Figure B.3 shows the content of the service database just after the creation of the download and its announcement services while Figure B.4 shows the content just after the services are started.

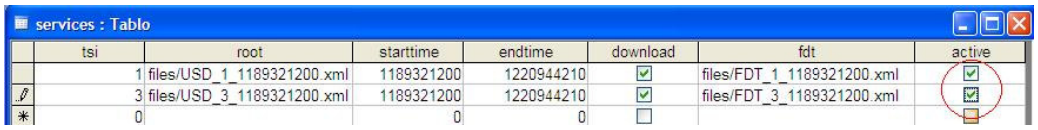


Figure B.4 Scheduled services in services database just after sessions start.

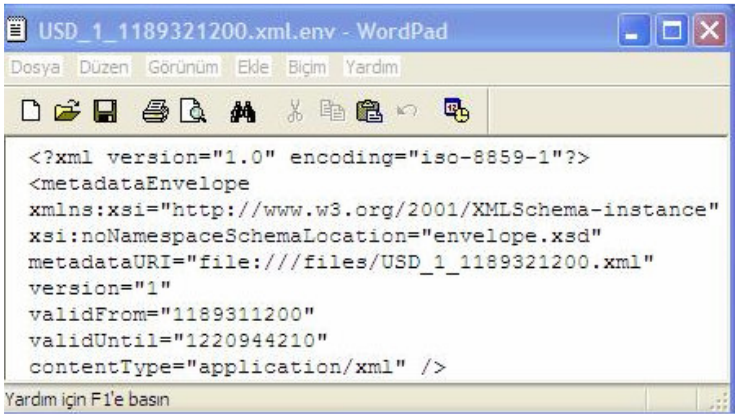


Figure B.5 User service description envelope file created for the download service.

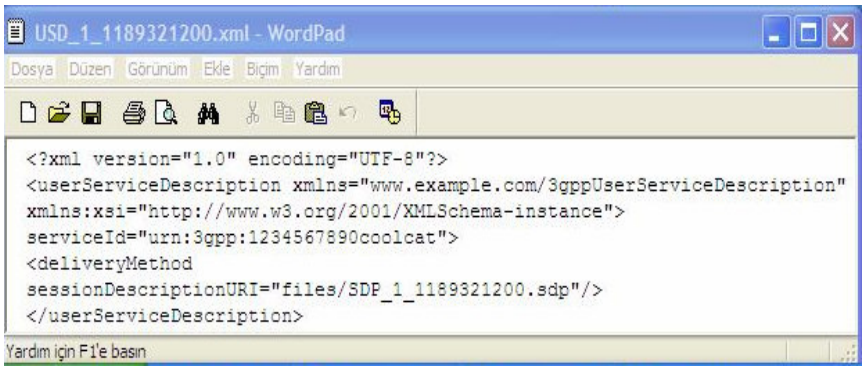


Figure B.6 User service description file created for the download service.

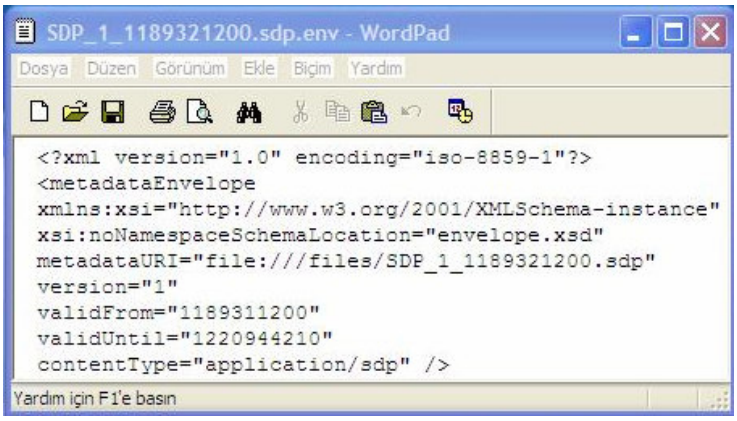


Figure B.7 Session description envelope file created for the download service.

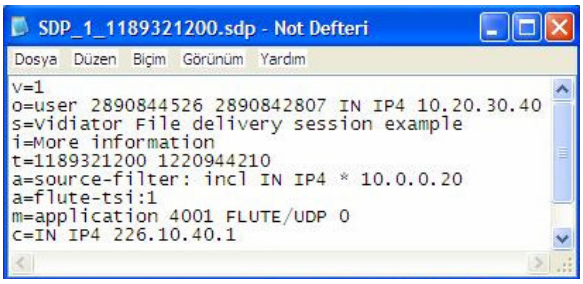


Figure B.8 Session description file created for the download service.

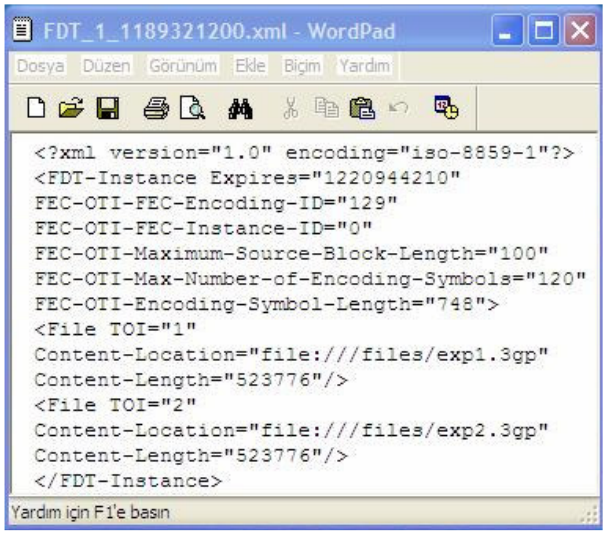


Figure B.9 File description table file created for the download service.

Figure B.5 and Figure B.6 show envelope and metadata fragment association of the user service description metadata fragment. They are created at the phase of scheduling of the download service shown in Figure B.1

Figure B.7 and Figure B.8 show envelope and metadata fragment association of the session description metadata fragment. They are created at the phase of scheduling of the download service shown in Figure B.1

Figure B.9 shows file description table created for the download service. It is also created at the phase of scheduling of the download service shown in Figure B.1.

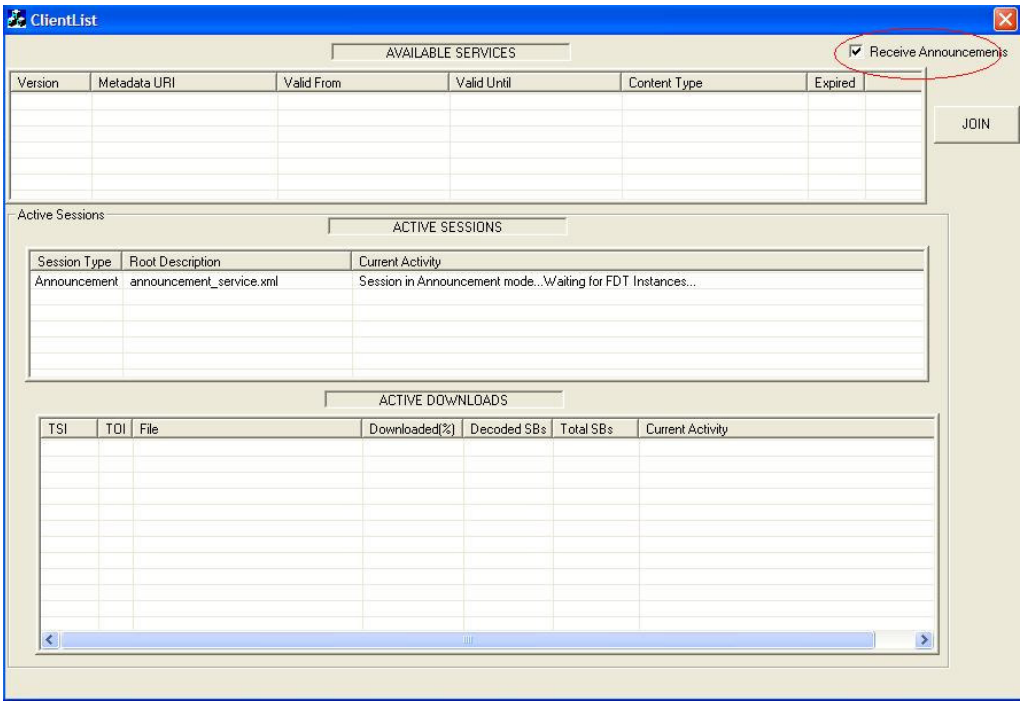


Figure B.10 Receiver user interface when announcement session is activated.

Now services are ready to start when their session start time comes. Server is running. Now Figure B.10 shows the receiver side user interface that is newly join to the announcement channel. There are no available services known by the user. In order to automatically receive the announcement, the user has to join to the announcement channel. As soon as the announcement channel is joined, corresponding information becomes available to the user in active session section of the user interface.

Figure B.11 shows the situation when a new service is available from the announcement channel and the user leaves the announcement channel. There are four service description files downloaded from the announcement channel. These files are metadata files and they update the receiver side service database, the content of which is shown in Figure B.12.

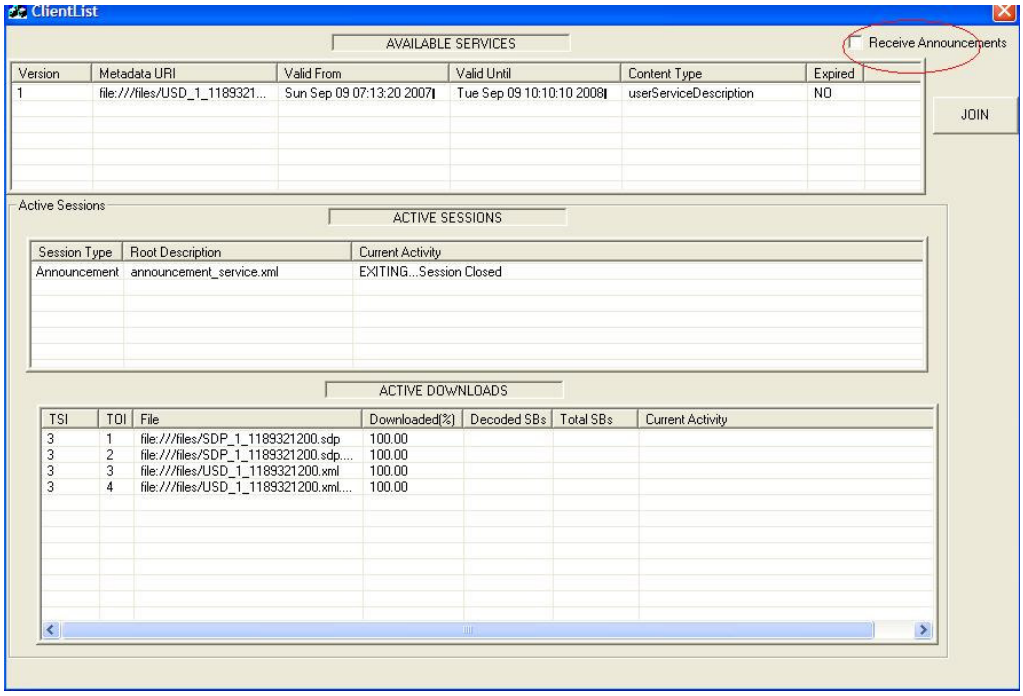


Figure B.11 Receiver user interface when a new service is available.

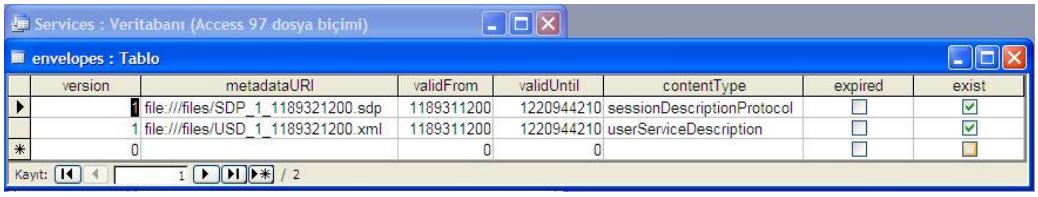


Figure B.12 Receiver service announcement database after a new service is available.

Figure B.12 indicates service announcement descriptions. As new envelopes or metada fragments come from the announcement channel, it updates the corresponding entries in database.

Figure B.13 shows the situation when the user is joined to the actual download service available and used that service.

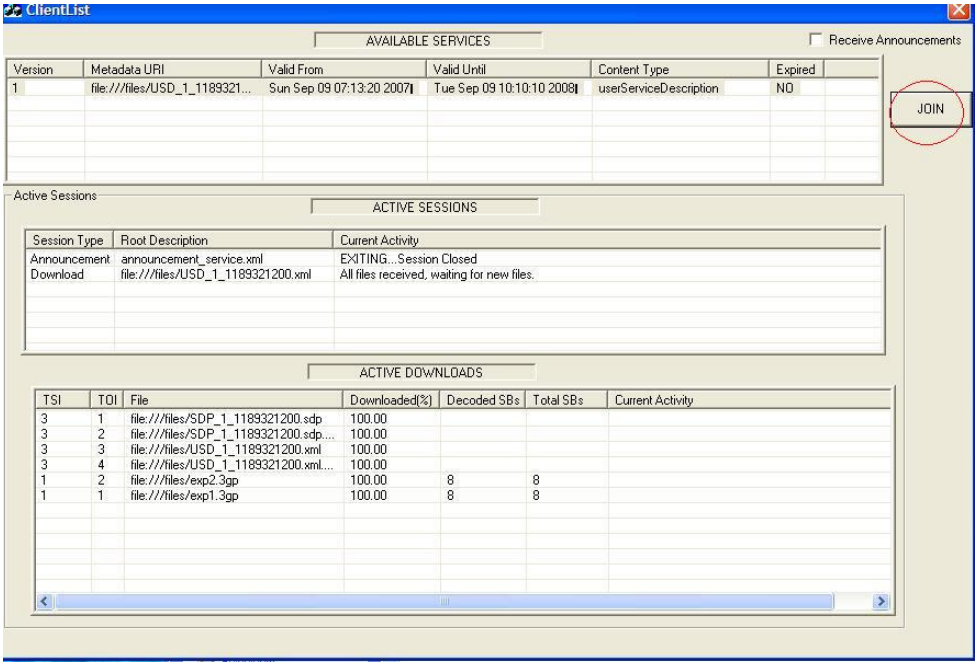


Figure B.13 Receiver user interface after the download service is used.

B.2 Prototype Enhancement for MBMS Emulations

The prototype is extended to emulate progressive download as well as interleaved download or interleaved progressive download under MBMS network conditions. Hence a database is designed including two tables: “configuration” and “entries” (girisler). Emulation is configured mostly by setting the attributes in configuration table. However, some of the attributes are controlled inside the code.

Figure B.14 shows the current state of the emulation for the example of a download service distributing two 3gp media files. In active downloads section “Iter” provides iteration related information. In the example there are 100 iterations, now the emulation just finished iteration 3. It means 6 downloads are done. MBMS link and MBMS network conditions are identified by the configuration ID, which is shown in Figure B.15 in detail. In the example, configuration ID 2 means RLC link layer lost is 5% (lllost), PDU size is 1280 bytes. Other informations are as follows:

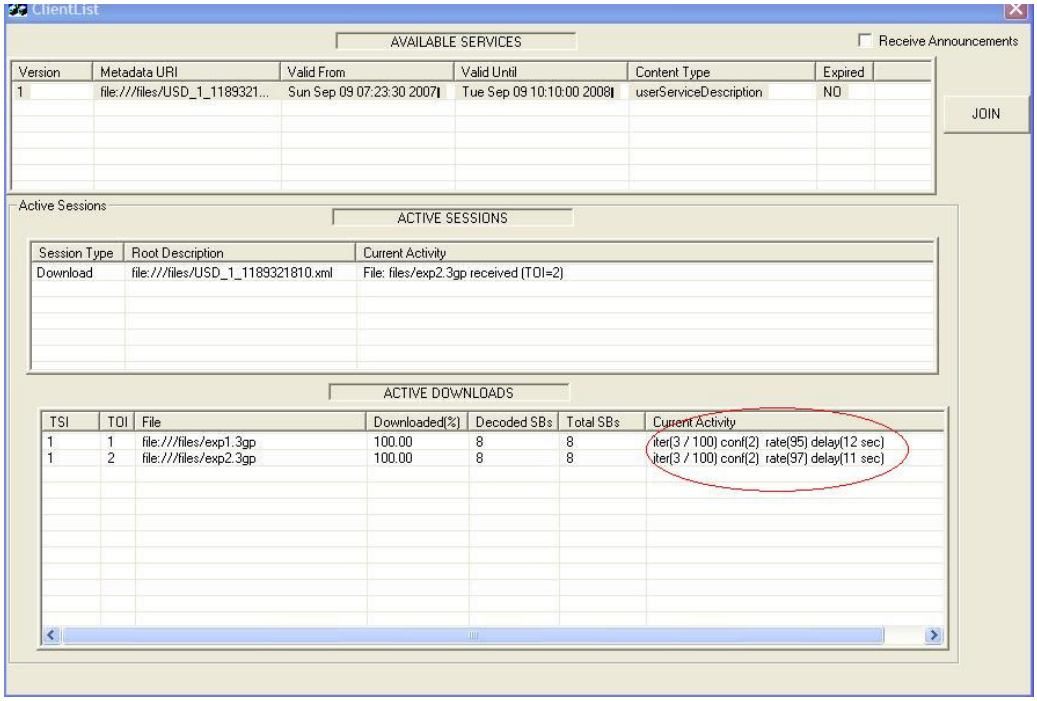


Figure B.14 Emulation user interface of the receiver side media download.

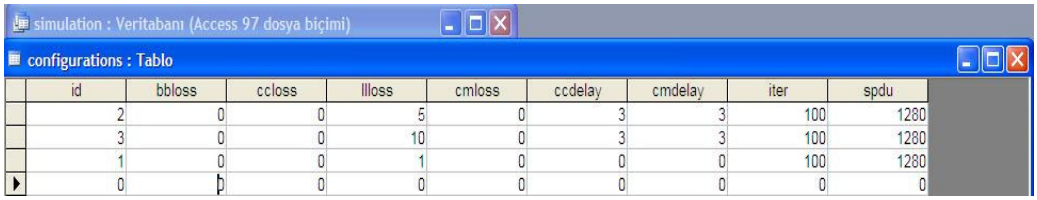
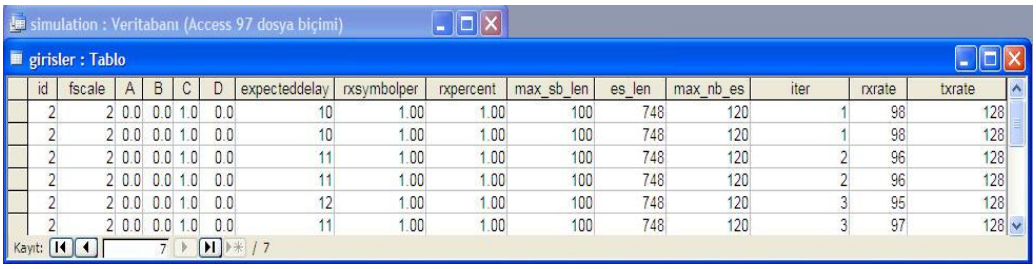


Figure B.15 Configuration of the emulation in emulation database.

- "bbloss" is IP backbone loss ratio.
- "ccloss" is cell congestion loss ratio.
- "llloss" is link layer loss ratio (RLC PDU loss)
- "cmloss" is cell mobility loss ratio.
- "ccdelay" is maximum cell congestion delay in second. Once a cell is congested it stays congested for randomly changing duration up to "ccdelay" seconds.
- "cmdelay" is maximum cell mobility delay in second (max. cell change delay). Cell mobility takes a random duration upto "cmdelay" seconds.

Other parameters such as SDU size, Transmission rate and mapping of PDU lossess to SDU losses are implemented inside the code. With each iteration our emulation calculates and saves following information shown in B.16. Each record (iteration information) is identified by “id”, “fscale” and “iter”.


simulation : Veritabanı (Access 97 dosya biçimi)

girisler : Tablo

id	fscale	A	B	C	D	expecteddelay	rxsymbolper	rxpercent	max_sb_len	es_len	max_nb_es	iter	rxrate	txrate
2	2	0.0	0.0	1.0	0.0	10	1.00	1.00	100	748	120	1	98	128
2	2	0.0	0.0	1.0	0.0	10	1.00	1.00	100	748	120	1	98	128
2	2	0.0	0.0	1.0	0.0	11	1.00	1.00	100	748	120	2	96	128
2	2	0.0	0.0	1.0	0.0	11	1.00	1.00	100	748	120	2	96	128
2	2	0.0	0.0	1.0	0.0	12	1.00	1.00	100	748	120	3	95	128
2	2	0.0	0.0	1.0	0.0	11	1.00	1.00	100	748	120	3	97	128

Kayıt: 7 / 7

Figure B.16 Entries in emulation database.

“Id” intentifies the configuration parameters set for MBMS network conditions in configuration table while “fscale” identifes the media file downloaded. There are two files of small and medium sizes, which are identified by “fscale =1” and “fscale =2” respectively. “Iter” identifies the current iteration of the download. Total number of iterations is set by the “iter” in configuration table. Other informations are as follows:

“A”,”B”,”C” and “D” identifies the lost distribution among IP backbone losses, cell congestion losses, RLC PDU losses and cell mobility losses, which are defined in the report by Digital Fountain, Ericsson, NEC, Nokia, Nortel, Siemens (May 2004). In the exaple in Figure B.16 all the losses belong to “C”, which are RLC PDU losses.

- “expecteddelay” is the expected delay identifying the waiting time or initial startup time in MBMS download.
- “rxsymbolper” is the received symbols percent without caring the successful or unsuccessfull decoding of source blocks.
- “rxpercent” is the the downloaded percent of the media with successful decoding of source blocks.

- “max_sb_len”, “es_len” and “max_nb_es” are maximum source block length, encoding symbol length and maximum number of encoding symbols respectively, which are FEC OTI informations.
- “rxrate” is the average receiving rate under the configured network and link conditions.
- “txrate” is the transmission rate.

In the example in Figure B.16, although transmission rate is 128 kbps, because of 5% RLC PDU losses, the average receiving rate is reduced to 98, 97 or 96 kbps under 20% FEC transmission cost with other FEC OTIs shown in the table, 1280 bytes RLC block size, 800 bytes SDU size as well as other parameters.