Progressive Download for 3G Wireless Multicasting[1]

Zeki Yetgin
Dokuz Eylul University
zeki@cs.deu.edu.tr

Gamze Seckin
T-Mobile USA
Gamze.Seckin@t-mobile.com

Abstract

This study focuses on the gain of using progressive download instead of legacy download for streamable media files for wireless multicasting networks. With progressive download, the media content can be played after some waiting time, while the downloading still continues in the background. We focus on 3GPP's wireless multimedia broadcast multicast system (MBMS). First we provide a legacy MBMS download system optimized for waiting time to play the media. Then we provide a progressive MBMS download system to compare the gain in waiting time. Reliability is assumed to be provided with a FEC mechanism. We considered Reed Solomon and Raptor FEC and finally provided comparisons of both FEC methods with regards to savings in waiting time for progressive download. 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.

1. Introduction

Today both streaming and downloading services are offered over point-to-point wireless connections. Large scale media distribution of media makes this point-to-point approach inefficient especially for wireless networks. The recent development in multimedia applications with a parallel progress in wireless transport technologies has brought real-time and non-real time multimedia distribution in the form of multicasting and broadcasting. Such multimedia distribution mechanisms include streaming and downloading services for on-demand video, mobile TV, short clips for news, football results, software updates and more. Several technologies that provide broadcasting and multicasting for wireless networks are 3GPP MBMS (Multimedia Broadcast and Multicast System) [1] [2], 3GPP2 BCMCS (Broadcast and Multicast System), DVB-H (Digital Video Broadcast for Handhelds), and MediaFLO among others [3].

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

[1] *This project has been sponsored by DEU, TUBITAK and Vidiator between August 2006-December 2007.*

presentations. Download delivery mode is for delivering files where transmission reliability is important. Streaming mode is for real time delivery of continuous media where the quality is important. The downloading mode consumes less radio recourses despite its longer time of service consumption with respect to the streaming. However, progressive download combines the advantages of streaming and downloading in terms of time and bandwidth. With progressive download, the content of the download can be streamed sooner, after some initial startup delay, also called waiting time in this paper, in parallel with the ongoing download. So it can be referred to as “play while download”. By its nature, it can be considered a software overlay on top MBSM download mode. We believe that wireless multicasting 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 satisfaction can be increased if the media play starts earlier. The second reason is the optimization of radio resources. Download mode uses less bit rate compared to streaming. Since the progressive download inherits the benefits of download in terms of bit rate utilization and the benefits of streaming in terms of time utilization it allows the download at much lower bit rates than the streaming mode and much shorter waiting time than the download mode. The third reason, adding progressive download capability to any multicast delivery system will not require changes in the infrastructure since progressive download will utilize the existing download delivery mode.

The download reliability becomes more important if the service will be distributed over a non-reliable broadcast or multicast channel to several hundred users simultaneously. There are many protocols that provide reliable multicast transport. In [4] a given taxonomy breaks these protocols into four classes; one class of these protocols uses negative acknowledgements (NACK-only protocols) to request the retransmission of missing packets. A second class of protocols uses positive acknowledgments (Tree based ACK protocols) to indicate multicast data packets that are successfully received. A third class of protocols uses routers in the network to assist with retransmitting lost packets. The router assisted class adds network-centric requirements while the other classes require bi-directional connectivity between sender and receivers. Asynchronous Layered Coding (ALC) [5] class protocols are important in that senders provide forward error correction (FEC) to recover from packet losses with no messages from receivers or the routers of the network. IETF (The Internet Engineering Task Force) Reliable Multicast Transport working group states that, due to a variety of applications and the orthogonal requirements of these applications, a "one size fits all" protocol is not possible [6]. File Delivery over Unidirectional Transport (FLUTE) Protocol [7] is a protocol used to deliver files particularly over unidirectional systems from one sender to many receivers. Since FLUTE uses unreliable transport protocol, an application layer FEC is coupled with FLUTE to recover from packet losses. So with FLUTE, the reliability is assumed to be provided with a FEC mechanism. FLUTE is an ALC based IETF protocol. Hence FLUTE scales well and is preferred for unidirectional systems. This is the reason why FLUTE has gained popularity in both cellular and broadcast technologies such as MBMS and DVB-H.

This study investigates the gain of using progressive download instead of legacy download for streamable media files for wireless multicasting networks. Our work is based on 3GPP’s wireless multimedia broadcast multicast system. We base our work on the usage of FLUTE for unidirectional download. Experimental results of the work show that MBMS services can be enriched by adding progressive download service support on top of MBMS download mode. The work shows how much can be gained in terms of waiting time from having progressive download in MBMS for small size files. Hence the results encourage the usage of progressive download instead of legacy download mechanism where the data file streamable, for improved user experience for 3G wireless multicasting systems.

This paper is organized as follows; second section provides related work for reliable download methods. Third section provides the main problem addressed in this paper and discuss our solutions. Section four gives a brief overview of MBMS download delivery and its main components; FEC and FLUTE Protocol. Section five discusses the system models considered in the work. Section six provides experimental results and final section gives the conclusion.

2. Related Work

Recently progressive download of 3GP media files using FLUTE for broadcasting is studied in 3GPP working groups [8]. According to PSS specification [9], for point to point unicast connections a PSS client already supports progressive download of 3GP media files with HTTP connection over TCP/IP. By parsing a 3GP media file, a PSS client can determine whether the media can be played during the download. As indicated in [8], MBMS progressive download service support should be given via service description metadata as in session description protocol (SDP) where all receivers are supposed to receive the description before the service consumption.

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 a 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, in [10] an intermediate delivery mode called “Preload Delivery Mode” based on MBMS download delivery mode is presented. The proposed system constitutes an actual bridge between streaming and downloading 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.

In [11] an asynchronous and reliable solution for streaming, conceptually more suitable for progressive download, is proposed. 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 [12] [13], 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 in [11] [14] 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. Reed-Solomon codes with and without interleaving [15] and Raptor codes, also investigated in [16] for MBMS, are used in these analyses. 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, in [17] minimum waiting time with minimum FEC overheads required for reliability is discovered among a large set of sizing parameters. Second, according to MBMS specification [2], the FEC repair symbols are sent after the all source blocks are sent to the clients. However, this approach is not suitable for progressive download application. Random transmission strategy [15] of FEC repair symbols is not suitable for progressive download too. Reliable download analyses supporting progressive download is recently studied in [8] and [18].

Apart from FEC reliability, some of the works use data carousal technique over FLUTE [19] 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 [2], 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.

Progressive download in MBMS is still in debate in 3GPP working groups and further discussions are postponed to future MBMS releases. One of the important reason, among many others, is there is no clear work that show our gains from having progressive download in MBMS. The main problem addressed in this paper is how much gain in waiting time will be possible from having progressive download in MBMS.

We will study two MBMS download deliveries: (i) Legacy-Download Delivery (ii) Progressive-Download Delivery. Our aim is to show that considering progressive download in MBMS can be quite reasonable with acceptable initial startup delay and considerable gain. In order to have an acceptable initial startup delay we investigated how to reduce waiting time by preserving 100% reliability. In our analyses in this paper, 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 is done to find the best gain in waiting time for a large set of system parameters under various MBMS network conditions.

3. MBMS download delivery

This section gives brief overview of MBMS Download delivery. MBMS download delivery is based on FLUTE protocol, which is particularly suitable for unidirectional wireless multicast networks. FLUTE is based on Asynchronous Layered Coding (ALC) protocol [5] that makes the FLUTE well scalable and hence preferable for unidirectional systems.

ALC is an instantiation of Layered Coding Transport (LCT) [20] 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, FEC mechanism 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. A source block is a fragment of the original object (media). A Block of "k" source symbols constitute a source block.

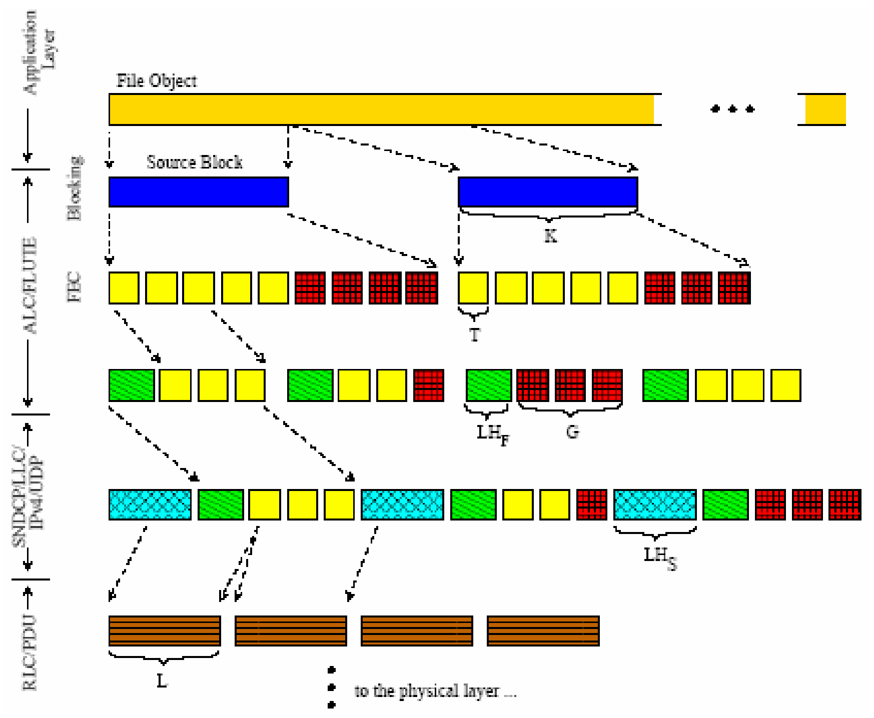

Figure 1. MBMS Download Protocol Stack

Decoding algorithms allow the recovery of the "k" source symbols from any set of the "n" received symbols. Detailed information for FEC can be found in [2] and [21]. Most popular FEC codes are Raptor codes as initially introduced by Shokrollahi in [22] and Reed Solomon codes [23]. For the MBMS system, Raptor codes have been selected due to their high performance among others. So 3GPP has mandated the support of Raptor codes [2] for their terminals that uses the MBMS service.

Figure 1, described in [24], 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 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), and among others as specified in [7]. 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.

3. System Model

In this section we provide the formulization of the problem, the system model and its parameters. We also provide our assumptions for both legacy and progressive download. To do so we define a gain function which allows us to identify how much time can be gained in waiting time from using progressive download as opposed to legacy download. We define the initial startup delay for progressive download and legacy download 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 delay in progressive download while it refers the downloading time in legacy downloads. 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 in [8].

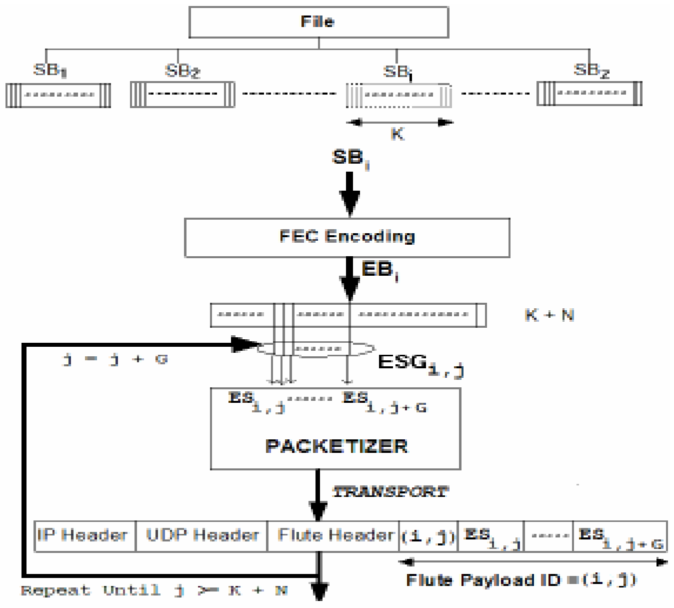

Figure 2. System Models for Proposed Download Delivery

The system model shown in Figure 2 is valid for both the download delivery and the progressive download delivery since the progressive download is based on the download mode of the MBMS. In order to support progressive downloading we assumed that repair symbols are sent just after each source block. Figure 2 also shows the flow of the FLUTE packetization. Each SB_i is delivered to FEC layer for encoding where 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, shown as repeat until loop in Figure 2.

In order to support progressive downloading the file transmission is organized such that repair packets are transmitted after each source blocks. The minimal FEC overhead to provide 100% reliability is computed for different FEC source block sizes, protocol data unit (PDU) sizes, service data unit (SDU) sizes and different PDU packet loss ratios.

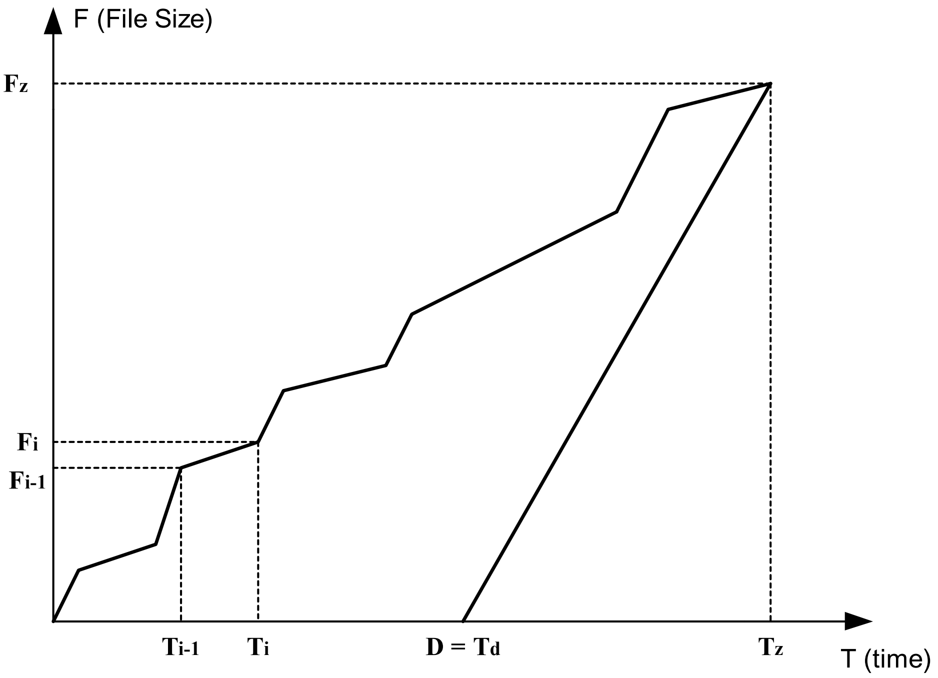

Figure 3. Download size as a function of time

Figure 3 shows an example for a receiver doing progressive download with the sending rate less than media play rate and the receiver obeys two assumptions: reliability and timely manner property. 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. 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.

Congestion and buffering in the network decides the linearity approximation between time and downloaded media size in Figure 3. 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 [8]. So we define partial reception rate r_i as the size of source block decoded within a consecutive times T_{i-1} and T_i.

$$r_i = \frac{F_i - F_{i-1}}{T_i - T_{i-1}} = \frac{\Delta F_i}{\Delta T_i} \tag{1}$$

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.

In order to find a fundamental R that characterizes the all r_i instances, simple way is to take an average of the preceding R and r_i at time T_i. During the simulations we notice that choosing the way of finding a good approximation of R is not critical regarding to its effect on initial startup delay for small file sizes. So our choice is as follows:

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

where R_i is approximation for the average reception rate up to time T_i ,$R_0 = r_1$; $i: 1...z$.

From Figure 3, we can construct the following approximation for the download duration;

$$D + \frac{F}{R_{media}} = T_z = \frac{F}{R_z} \tag{3}$$

Using Eq.3 the initial startup delay approximated at the end of file download is as follows;

$$D = T_d = F * \left(\frac{1}{R_z} - \frac{1}{R_{media}} \right) \tag{4}$$

where T_d or D denotes initial startup delay, R_{media} denotes media play rate in Kbps; $F=F_z$ is media file size in KB. For progressive download using Reed Solomon FEC, average reception rate can be denoted as R_{RS}, which is computed using Eq.2. Initial startup delay using Reed Solomon is denoted as follows;

$$D_{RS} = F * \left(\frac{1}{R_{RS}} - \frac{1}{R_{media}} \right) \tag{5}$$

We define gain G as the gain in waiting time from using progressive download compared to legacy download. Gain can be computed using R and T as follows:

$$T_{RS} = \frac{F}{R_{RS}} \tag{6}$$

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

where T_{RS} denotes the legacy download duration in seconds and G_{RS} is the gain percent in waiting time with progressive download compared to the legacy download using Reed Solomon coding.

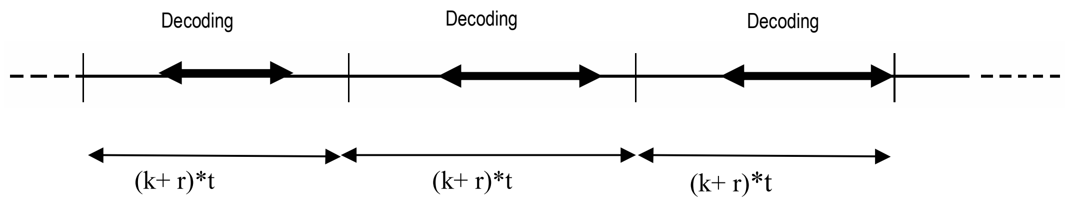

Figure 4. Reception rate approximation for Raptor

To compute the gain for Raptor coding we need to consider few other details. Since transmission of source symbols and repair symbols follows each other, decoding process of blocks must start within the time interval *(k + r)* t* as shown in Figure 4, where *k* is the number of source symbols, *r* is the number of repair symbols and *t* denotes symbol period; the average time between receiving of two consecutive symbols. However, decoding process is not expected to be completed in this interval. The reason is that while decoding process continues the receiver can collect symbols belonging to next source block in parallel. That is, computation and communication can be overlapped during decoding. Since each source block means extra repair symbols total time needed to download the media is total symbols including repair symbols multiplied by *t*:

$$T = \frac{F}{k * s}(k + r) * t \tag{8}$$

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

In order to have initial startup delay 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. Since symbol period mostly depends on the sending rate and symbol size it is approximately same for both transmissions. Based on such an assumption and subtracting T_{Raptor} from T_{RS} we can arrive at following approximations for Raptor:

$$T_{Raptor} = T_{RS} - \frac{F}{s}\left(\frac{C_{RS} - C_{Raptor}}{100}\right) * t \tag{9}$$

where *C* denotes FEC overhead in percent. From Eq.8,

$$t_{Raptor} = \frac{k * s * T_{Raptor}}{F * (k + r)} \tag{10}$$

As shown in Figure 4, for every time interval of *(k+r)*t*, exactly one source block is decoded. So average reception rate is computed as follows:

$$R_{Raptor} = \frac{k * s}{(k + r) * t_{Raptor}} \tag{11}$$

Once finding average reception rate, initial startup delay and the corresponding gain for raptor can be calculated as follows:

$$D_{Raptor} = F * \left(\frac{1}{R_{Raptor}} - \frac{1}{R_{media}} \right) \tag{12}$$

$$G_{Raptor} = 100 * \left(1 - \frac{D_{Raptor}}{T_{Raptor}} \right) \tag{13}$$

We used Vidiator MBMS design based on [2], [25] and [26]. The system architecture is summarized in Figure 5. We focus only on the download module in Figure 5. To emulate MBMS link conditions we implemented a transmission rate and packet loss control module. The MBMS link conditions are aligned with [21], [27] and [28]. Files are sent in a single loop and played progressively at 128 Kbps. Considered UTRAN bearers are 64, 128 and 256 kbps. Each simulation uses the combinations of all above parameters and is repeated at least 100 times for different PDU loss patterns and different SB size, different symbol length and different transmission rates for files 100K (small) and 512K (medium). A client is assumed to process SBs in any size for decoding in a timely manner. We assumed that there is a mapping of one IP packet to one SDU block and that each IP packet contains only one symbol of varying length.

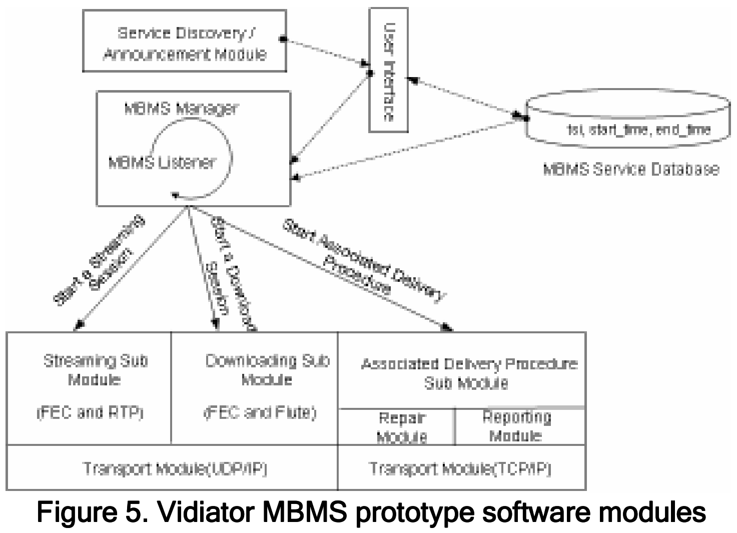

Figure 5. Vidiator MBMS prototype software modules

The symbol length for FEC is calculated as follows:

Symbol Length = S – 48, where S is the SDU size and IP\UPD\FLUTE header (44) + FEC payload ID (4) = 48 Bytes.

PDU (RLC) block sizes of 1280 B and 640 B are considered for two UTRAN bearers, 128-256 kbps and 64 kbps transmission rates respectively. The list of parameters analyzed in this study across layers is summarized in Table 1.

Table 1. Parameters Studied Across Layers

Parameter	Unit	Layer	Experiment Set
File Size	Kilobyte	Application	{100,512}
Initial Startup Delay	Seconds	Application Layer	To be determined in this study
Gain	Percent	Application Layer	To be determined in this study
Symbol Length	Byte	FEC	{SDU – 48}
SB Size	Symbol	FEC	{10,50,80,100,120,150,180,200,230}
IP packet size	Byte	IP	SDU
SDU block size	Byte	Core Network	{400,600,800,1000,1400}
PDU block size (RLC block size)	Byte	RLC Radio Link layer	{640,1280}
PDU (RLC Link Layer) Losses	Percent	RLC Radio Link layer	{1,5,10}

An algorithm to map PDU losses to SDU losses is provided in [27]. Error bursts in UMTS are generated by using the algorithm such that a single SDU loss may cause one or more PDU packet losses. We considered 1% (low), 5% (medium) and 10% PDU losses (high). Loss due to mobility such as cell handovers is not considered.

4. Experimental Results

This section shows the results of our experimental analyses for the legacy versus progressive download systems. Experiments are done on a set of computers. Each computer runs the simulation for different configuration parameters which are given in Table 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. The MBMS download system is actually the FLUTE download that is based on [25], which uses Reed Solomon FEC coding. Both client and server maintain a database and create a simulation record after each download. The record contains evaluated target parameters for the selected set of configuration parameters shown in Table 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. 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.

First we observe the initial startup delay for 100% reliability for various Reed Solomon SB sizes. Figure 6 and Figure 7 show the initial startup delay observed for small and medium file sizes respectively, transmitted over 256 kbps, 128 kbps and 64 kbps, with 1%, 5% and 10% loss ratios. The SDU size is selected to be the optimum for these conditions as well as the SB to provide 100% reliability. These optimum values are shown as labels on top of each figure. We observed that for as the SB size increases the initial startup delay decreases since Reed Solomon FEC performance is very bad at small source block size. Increasing the source block size will decrease the FEC overhead required for reliability.

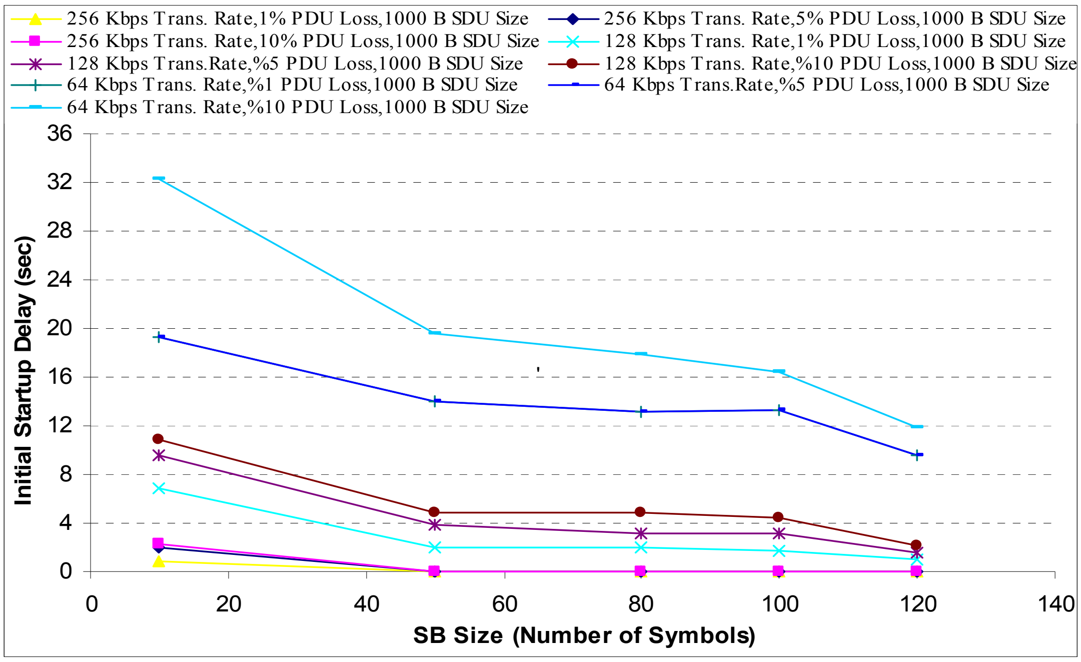

Figure 6. Initial startup delay for 100 KB file

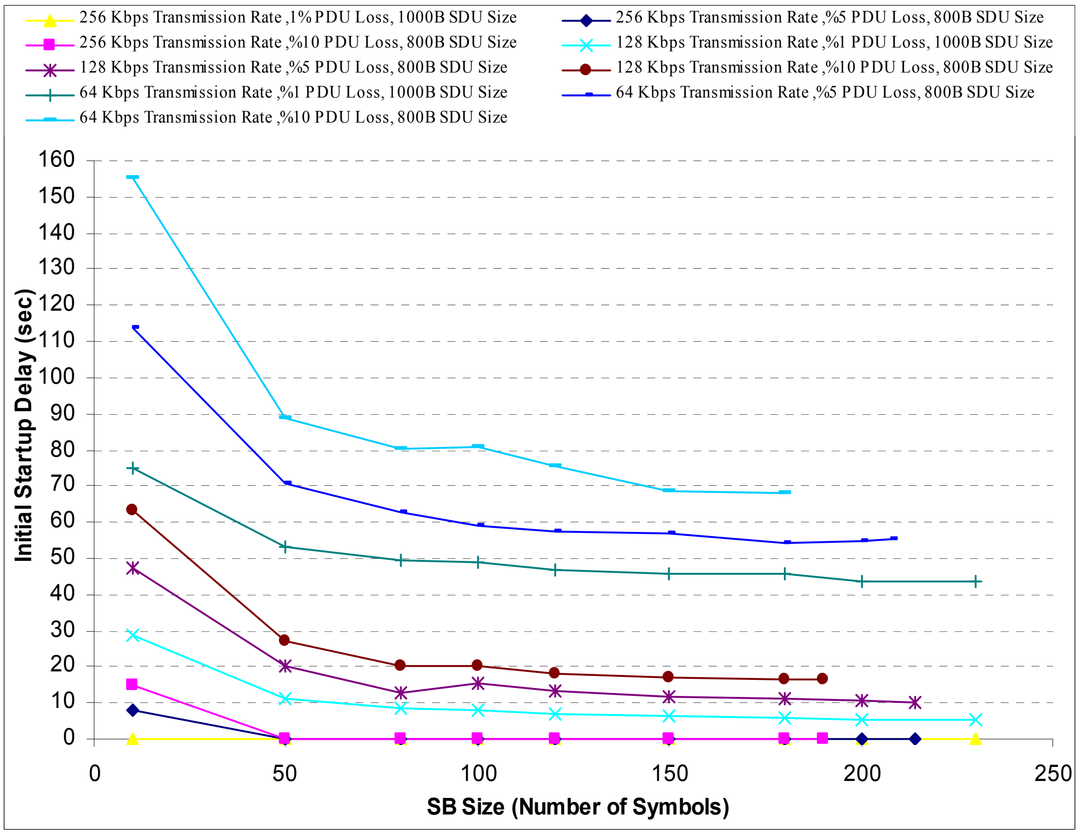

Figure 7. Initial startup delay for 512 KB file

As the FEC overhead increases it is expected that the initial startup delay increases due to increased repair symbols. Similarly increased packet losses require an increase in required FEC overheads for reliability. These facts are observed in Table 2, 3 and Table 4. So as packet losses increase initial startup time particularly for small transmission rates increase too as shown in Figure 8.

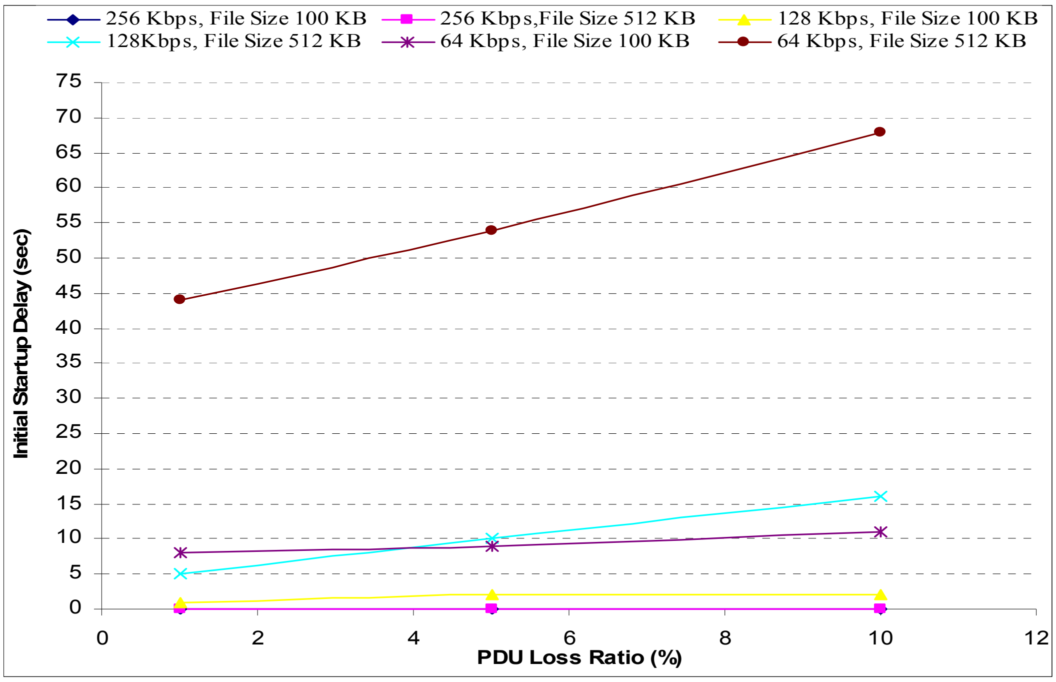

Figure 8. Initial startup delay for various PDU loss ratios

For very small files the difference in FEC overhead between Reed Solomon and Raptor is in the range of 1 to 2% which is negligible due to the small file sizes. So we assumed that for 100 KB media, Raptor codes have the same required overhead with Reed Solomon. For 512 KB media, the required overheads are obtained from [29].

Table 2. Progressive Download Gain Comparison for Reed Solomon and Raptor FEC for MBMS at 256 Kbps

File Size (KB)	PDU Loss (%)	Reed Solomon FEC Overhead (%)	Reed Solomon Startup Delay in Progressive D.(sec)	Reed Solomon Startup Delay in Legacy D.(sec.)	Reed Solomon Progressive Download Gain (%)	Raptor FEC Overhead (%)	Raptor Startup Delay in Progressive D.(sec)	Raptor Startup Delay in Legacy D.(sec.)	Raptor Progressive Download Gain (%)	Difference in Startup Delay (Reed Solomon - Raptor) (sec.)
100	1	12	0	3.4	100	12	0	3.4	100	0
100	5	23	0	3.7	100	23	0	3.7	100	0
100	10	43	0	4.0	100	43	0	4.0	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 3. Progressive Download Gain Comparison for Reed Solomon and Raptor FEC for MBMS at 128 Kbps

File Size (KB)	PDU Loss (%)	Reed Solomon FEC Overhead (%)	Reed Solomon Startup Delay in Progressive D.(sec)	Reed Solomon Startup Delay in Legacy D.(sec.)	Reed Solomon Progressive Download Gain (%)	Raptor FEC Overhead (%)	Raptor Startup Delay in Progressive D.(sec)	Raptor Startup Delay in Legacy D.(sec.)	Raptor Progressive Download Gain (%)	Difference in Startup Delay (Reed Solomon - Raptor) (sec.)
100	1	12	0.6	6.8	92	12	0.6	6.8	92	0
100	5	23	1.1	7.4	85	23	1.1	7.4	85	0
100	10	43	1.8	8.1	77	43	1.8	8.1	77	0
512	1	7	4.9	36.9	87	4	3.9	35.9	89	1
512	5	19	9.4	41.4	77	12	6.9	38.9	82	2
512	10	37	15.1	47.1	68	22	9.9	41.9	76	5

Table 4. Progressive Download Gain Comparison for Reed Solomon and Raptor FEC for MBMS at 64 Kbps

File Size (KB)	PDU Loss (%)	Reed Solomon FEC Overhead (%)	Reed Solomon Startup Delay in Progressive D.(sec)	Reed Solomon Startup Delay in Legacy D.(sec.)	Reed Solomon Progressive Download Gain (%)	Raptor FEC Overhead (%)	Raptor Startup Delay in Progressive D.(sec)	Raptor Startup Delay in Legacy D.(sec.)	Raptor Progressive Download Gain (%)	Difference in Startup Delay (Reed Solomon - Raptor) (sec.)
100	1	12	7.5	13.8	45	12	7.5	13.8	45	0
100	5	32	9.0	15.3	41	32	9.0	15.3	41	0
100	10	53	11.3	17.5	36	53	11.3	17.5	36	0
512	1	8	42.5	74.5	43	4	39.7	71.7	45	3
512	5	21	53.3	85.3	38	14	48.4	80.4	40	5
512	10	41	67.9	99.9	32	26	57.3	89.3	36	11

In Table 2, Table 3 and Table 4 we provided experimental results to compare the gain we obtain from progressive download as opposed to legacy download, both for Reed Solomon FEC and Raptor FEC. The following is a summary of our observations for MBMS progressive download:

1. 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.
2. For higher media sizes, we observe less gain from progressive download. 512 KB media reduces gain percentage up to 3% to 7% with compared to 100 KB media.
3. In general, progressive download provides savings in initial startup delay 3 to 32 seconds with compared to legacy (non-progressive) download.
4. Progressive download with Raptor has savings in initial startup delay starting from 7% up to 33%, which is about 1 to 11 seconds with regards to progressive download with Reed Solomon.

5. Conclusion

In this study we focus on the gain of using progressive download instead of legacy download for streamable media files for wireless multicasting networks.

Our results show that progressive download provides satisfactory gain with regards to initial startup delay for 3G wireless multicasting. We observed 40% to 100% gain and saved up 3 to 32 seconds in initial startup delay with compared to legacy download for different file sizes, bitrates and FEC overheads and under different loss ratios. The impact of reliability on progressive download with Raptor FEC and Reed Solomon FEC was comparable. Raptor provided savings in initial startup delay starting from 7% up to 33% which is about 1 to 11 seconds compared to Reed Solomon. We believe that our results will encourage more work on progressive download for wireless multicasting.

References

[1] http://www.3gpp.org/
[2] 3GPP TS 26.346, "Multimedia Broadcast/Multicast Service (MBMS); Protocols and Codecs", version 6.4.0, March 2006.
[1] [3] UMTSF/GSMA Joint Mobile TV Work Group Report,"Mobile TV: The Groundbreaking Dimension" v2.21, 2006.
[4] B. Whetten, L. Vicisano, R. Kermode, M. Handley, S. Floyd, M. Luby, "Reliable Multicast Transport Building Blocks for One-to-Many Bulk-Data Transfer", Tech. Rep., IETF RFC 3948, January 2001.
[5] M. Luby, J. Gemmell, L. Vicisano, L. Rizzo, M. Handley, and J. Crowcroft, "Asynchronuous Layered Coding (ALC) Protocol Instantiation", Tech. Rep. , IETF RFC 3450, Dec. 2002.
[6] http://www.ietf.org/html.charters/rmt-charter.html
[7] T. Paila et al., "FLUTE, File Delivery over Unidirectional Transport", Tech. Rep., IETF RFC 3926, October 2004.
[8] S4-060267, 3GPP TSG System Aspects Meeting#39," Support of Progressive Download in MBMS", Dallas, TX, USA, 15 – 19 May 2006.
[9] 3GPP TS 26.234, "Transparent end-to-end Packet-switched Streaming Service (PSS); Protocols and codecs", version 6.9.0, September 2006.
[10] S4-060385, 3GPP TSG System Aspects Meeting#40, "Content Preloading for MBMS User Services", Sophia Antipolis, France, 28 August – 1 September 2006.
[11] H. Jenkac, T. Stockhammer, W. Xu, " Asynchronous and Reliable On-Demand Media Broadcast", IEEE Network Magazine, Special Issue on Multimedia over Wireless Broadband Networks, Vol. 20, No. 2, March 2006, pp. 14-20.
[12] A. Hu, "Video-on-Demand Broadcasting Protocols: A Comprehensive Study", in Proc. IEEE Infocom. Anchorage, Alaska, 2001.
[13] L. Engebretsen, M. Sudan, "Harmonic Broadcasting is Bandwidth-Optimal Assuming Constant Bit Rate", in Proc. Annual ACM-SIAM Symposium on Discrete Algorithms. San Francisco, CA, USA, 2002.
[2] [14] H. Jenkac; T. Stockhammer, W. Xu,W. Abdel Samad, "Efficient Video-on-Demand Services over Mobile Datacast Channels", Journal of Zhejiang University Science, Special Issue for Selected Papers (Part I) of 15th International Packet Video Workshop (PV2006), Vol. 7, No. 5, May 2006, pp. 873-884.
[3] [15] SP-050164, 3GPP TSG System Aspects Meeting#27, "Efficient FEC code for reliable MBMS user services", Tokyo, Japan, 14-17 March 2005.
[4] [16] M. Luby, M. Watson, T. Gasiba, T. Stockhammer, W. Xu, "Raptor Codes for Reliable Download Delivery in Wireless Broadcast Systems". CCNC 2006, Las Vegas, NV, January 2006.
[5] [17] Z. Yetgin, G. Seckin, "Progressive Download for 3G Wireless Multicasting", in Proc. IEEE CS International Conference on Future Generation Communication and Networking (FGCN 2007), Jeju Island, Korea, 6 - 8 December 2007.
[18] Z. Yetgin, G. Seckin," Reliable Download Analyses for Multimedia Broadcast Multicast Services", in Proc. IAENG, The World Congress on Engineering and Computer Science San Francisco, USA, 24-26 October 2007.
[19] J. Peltotalo, S. Peltotalo, J. Harju:,"Analysis of the FLUTE Data Carousel", in Proc. 10th EUNICE Open European Summer School, Colmenarejo, Spain, 6 – 8 July 2005, pp. 138 – 142.
[6] [20] M. Luby, J. Gemmell, L. Vicisano, L. Rizzo, M. Handley, J. Crowcroft , "Layered Coding Transport (LCT) Building Block", Tech. Rep. , IETF RFC 3451, December 2002.
[21] S4-040348, 3GPP TSG System Aspects Meeting #31," Permanent document on: simulation guidelines for the evaluation of FEC methods for MBMS download and streaming services", Montreal, Canada, May 2004.
[22] A. Shokrollahi, "Raptor codes", Tech. Rep., Digital Fountain, DR2003-06-001, June 2003.
[7] [23] J. Lacan, V. Roca, J. Peltotalo, S. Peltotalo, "Reed Solomon Forward Error Correction", draft-ietf-rmt-bb-fec-rs-05.txt (work in progress) May 2008.

[8] [24] T. Gasiba, T. Stockhammer, J. Afzal, W. Xu, "System Design and Advanced Receiver Techniques for MBMS Broadcast Services", IEEE International Conference on Communications, vol. 12, Istanbul, Turkey, June 2006, pp. 5444-5450.
[25] http://atm.tut.fi/mad/
[9] [26] www.vidiator.com
[10] [27] S4-AHP120, 3GPP TSG System Aspects," Mapping of SDUs to Radio Blocks for FEC simulations", Lund, Sweden, April 2004.
[28] 3GPP TS 26.946, "Multimedia Broadcast/Multicast Service (MBMS); User service guidelines", version 6.1.0, September 2006.
[11] [29] SP-050246, 3GPP TSG System Aspects Meeting #28, "Report of MBMS FEC Status", Quebec, Canada, June 6-9, 2005.
[12]

Authors

[13] [14]

Zeki Yetgin (zeki@cs.deu.edu.tr) studied mathematics in Science Faculty in Gazi University in Ankara. He received a B.Sc. degree in 1999 and an M.Sc. degree in 2002 from Eastern Mediterrenean University, Computer Engineering Department, Famagusta, Cyprus, Turkey. Since 2003 he has been doing Ph.D. in Dokuz Eylul University, Department of Computer Engineering, Izmir, Turkey. He is actively involved in a *TUBITAK (The Turkish National Science Foundation)* project where 3G wireless multicasting in UMTS is studied.
His research interests are mainly in the area of reliable multimedia transmission over wireless (broadcast) channels. He is particularly involved in progressive download in MBMS. His other interests are researches including Parallel and Distributed Computing, Learning and Reasoning in Fuzzy environments and Image/Speech processing.

Gamze Seckin (gamze.seckin@t-mobile.com) received her Ph.D. in Arizona State University in the Department of Computer Science and Engineering in 2000, after the completion of her B.S. and M.S. in the Department of Computer Engineering in Ege University, in 1992 and 1994 respectively. She is a former Fulbright Research Scholar, a member of IEEE Communications Society and IEEE Women in Engineering Society. She has awards, publications, pending patent applications, international standardization contributions, and a book chapter on research topics specializing ATM, IP, mobile and wireless networks. She worked at Vidiator Technology (US) Inc. (formerly Luxxon Corporation), a Hutchison Whampoa Company, for more than 7 years, first as Research Engineer then as the Director of Technology. Currently Gamze works as the Principal Engineer in the Network Evolution and Strategy Team in T-Mobile USA.