# Large Scale Content Distribution Protocols

Christoph NEUMANN
INRIA Rhône-Alpes, Planète research team, France
christoph.neumann@inria.fr

Vincent ROCA
INRIA Rhône-Alpes, Planète research team, France
vincent.roca@inria.fr

Rod WALSH
Nokia Research Center, Tampere, FIN-33720, Finland
rod.walsh@nokia.com

## ABSTRACT

This paper introduces large scale content distribution protocols, which are capable of scaling to massive numbers of users and providing low delay end-to-end delivery. Delivery of files and static objects is described, with real-time content streaming being outside the scope of this paper. The focus is on solutions provided by the IETF Reliable Multicast Transport Working Group. More precisely, the paper explains FLUTE, ALC and the associated building blocks. Then it discusses how these components are used in the Multimedia Broadcast Multicast Service (MBMS) for 3G systems and in the IP Datacast (IPDC) service for Digital Video Broadcast for Handheld devices (DVB-H).

## Categories and Subject Descriptors

H.4.3 [**Information Systems Applications**]: Communications Applications

## General Terms

Reliability, Standardization, Experimentation, Theory, Design

## 1. INTRODUCTION

Large scale content distribution, where the same content is sent reliably to a large number of users, is a challenging and complex task. A point-to-point approach increases signaling and media-load overhead linearly with group size and is not efficient and generally infeasible for this purpose The only proven scalable solution is reliable multicast. This subject has been intensively studied, in particular at IETF (section 1.3).

However multicast routing deployment is far behind expectations (e.g. no ubiquitous multicast routing is available and commercial IPv6 multicast router support is minimal). Wide scale deployment of reliable multicast technologies in the public Internet has not yet taken off.

There are a few exceptions though, in particular in national/international research networks (FLUTE is already used within the M6BONE network), and within sites where multicast routing is easily deployed, and in very specific environments (e.g. in clusters).

Yet until recently a killer-application appealing to the commercial mass market was still missing. It may change with the advent of wireless radio networks that are by nature broadcast networks, and a renewed commercial interest in mobile TV. In particular, new generations of mobile devices (such as cellular phones) will undoubtedly dominate the future Internet [11], while taking advantage of technologies primarily designed for the Internet. More specifically, the IP-Datacast (IPDC) service for DVB-H and the Multimedia Broadcast Multicast Service (MBMS) for the 3G cellular systems are two such enabling systems, and both of them rely on the outcomes of the IETF working groups.

### 1.1 Challenges

Reliable multicast faces the following challenges:

- *Scalability:* Support for a high number of simultaneous receivers is essential (up to thousands or millions of receivers in some use cases). Ideally the number of receivers should not reduce system performance.
- *Channel and Client Heterogeneity:* The set of receivers and channels will be very heterogeneous, i.e. with very different bandwidths, processing capacities and loss patterns.
- *Content Heterogeneity:* Mass media content is diverse and variable - especially over time to fashion and commercial trends and so the transport solution needs to support any kind of content. Nevertheless this paper only focuses on files and other static contents.
- *Reliability:* The approach must be robust to each kind of packet losses and cope with intermittent connectivity, in order to provide a reliable distribution service.
- *Congestion Control:* Fair competition with other streams, on shared links, is essential to avoiding network congestion that would damage co-existing (e.g. unicast/email) services. This is not an issue for such fully-provisioned-multicast wireless networks as 3G MBMS or DVB-H systems, but is critical with content distribution over the public Internet.

Depending on the target use case, the above challenges may be met. Security is another challenging requirement in some situations, either to provide source authentication, message integrity, confidentiality and content access control, but it will not be discussed in this paper.

### 1.2 Delivery Service Models

We can distinguish three delivery models [12]:

The *streaming* service model is typically used for audio and video content produced in real-time, but other kinds of content (e.g. the data provided in real-time by a probe or the subtitle of a movie) can be streamed too. Here timeliness is often more important than full transmission reliability.

The *on-demand* service model is typically used for the distribution of popular content. The content, which is not necessarily static but may change over time, is continuously broadcast or multicast, and interested clients join the session, download the content and leave the session whenever they want. Transmissions can be performed cyclically using a file carousel, and are not necessarily sequential. Indeed packets may be sent in a random manner to provide loss-pattern and session-joining-time independent performance to receiver. The service is reliable, scalable, but is generally non real-time.

Finally, the *push* service model is a synchronous model, where all receivers are supposed to be ready before the transmission starts. It is a sender initiated model where the content is delivered from time $t_0$ for a finite duration. This model is typically used to deliver content to a selected set of receivers, and mechanisms using session announcements and receiver reception reports can be used to ensure a minimal synchronization between sender and receivers.

### 1.3 RMT IETF Solutions

A "one-size-fits-all" protocol cannot comply with all the challenges and delivery models presented before. Therefore the Reliable Multicast Transport (RMT) IETF Working Group has adopted a modular approach [31]:

- Building Blocks (BB) are the basic components, (re)-usable in different contexts and designed with flexibility in mind. They can be "plugged" or "unplugged" in order to enable or disable features according to the needs of a Protocol Instantiation (see below). The Forward Error Correction Building Block (FEC BB) is a particularly useful BBs.

- A Protocol Instantiation (PI) is a set of BBs plus some PI-specific functions and headers (that are defined in a dedicated BB, for instance the Layered Coding Transport (LCT) BB for the Asynchronous Layered Coding (ALC) PI). Each PI aims to answer a very specific (or small set of) use case(s).

Thanks to this logical view, two PIs have been designed[1]:

- Asynchronous Layered Coding (ALC) [12]

- NACK Oriented Reliable Multicast (NORM) [3]

IP is a natural convergence layer on top of any physical layer, and so these RMT protocols rely on UDP/IP and are used either in a multipoint (their initial goal) or point-to-point scenario (which is sometimes useful). A general overview of these solutions is given in figure 1.

The remainder of this paper successively gives a short overview of FEC codes, introduces the ALC and NORM protocol instantiations, the fully-specified FLUTE file delivery transport, the datacasting services in 3G/DVB-H systems and highlights future challenges in conclusion.

[1]A third PI, router assisted protocol Tree-based Acknowledgment/Generic Router Assist (TRACK/GRA), has been considered for some time, but was later droped by the IETF RMT due to lack of progress.

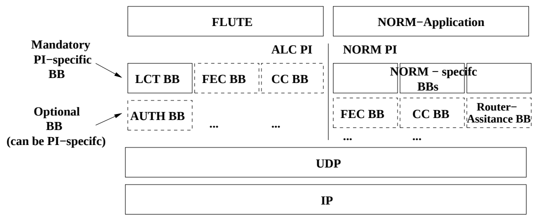


**Figure 1: The RMT IETF solutions.**

## 2. FEC CODES

Forward Error Correction (FEC) is a key component for reliability in scalable content delivery. Here FEC is used at transport/application layer and complements any FEC code that can be used in parallel to protect the bit streams, at a lower layer, over a physical link. Therefore, we only consider the "packet erasure channel" where packets either arrive perfectly or are lost.

With a $(n; k)$ FEC code, the $k$ *source symbols* of a source block are encoded into $n$ *symbols* (i.e. encoded symbols). Often one symbol is transported per data packet, though some FEC codes and use cases may lead the sender to pack several symbols per packet.

With "systematic codes", these $n$ encoding symbols are composed of the $k$ source symbols plus $n-k$ additional *parity symbols*, which are derived from the source symbols. A receiver can then reconstruct the $k$ source symbols provided it receives any $k$ symbols out of the total $n$ (or a few percents more than $k$ symbols with large and expandable block codes). The great advantage of using FEC is that the same parity symbol can be used to recover different lost symbols at different receivers.

### 2.1 Small Block FEC Codes

Reed-Solomon erasure codes (RSE) are extremely popular. But they are also intrinsically limited by the Galois Field (GF) they use [25], so the optimal $k$ and $n$ are relatively small. A typical example is GF($2^8$) where $k \leq n \leq 256$. With symbols of size one kilobyte, a FEC codec that produces as many parity symbols as data symbols ($n = 2k$) operates at most on 128 kilobyte blocks. All files lrager than 128 kilobyte must be segmented into several blocks, which reduces the erasure protection. Indeed, if $B$ blocks are needed, a symbol chosen randomly has a probability $1/B$ to recover a given erasure, and $B = 1$ is clearly the optimal solution. This phenomenon is known as the "Coupon Collector Problem" [6].

Another drawback is a huge encoding/decoding time with large $(k, n)$ values. Rizzo [25] reports encoding times in $O(k(n-k))$, and decoding times in $O(kl)$, where $l$ is the number of missing symbols, plus an additional matrix inversion time in $O(kl^2)$ (the inversion is done only once). This is the reason why GF($2^8$) is often preferred to GF($2^{16}$) in spite of block size limitations.

Yet small block codes are optimal because a receiver can perform FEC decoding as soon as it has received *exactly* $k$ packets out of the $n$. Such codes are called "Minimum Distance Separation" (MDS).

## 2.2 Large Block FEC Codes

Conversely, *large block codes* support large $k$ values (several hundreds of thousands symbols and more) while keeping high performance in encoding and decoding.

The "Coupon Collector Problem" is completely suppressed when a single block is used (which is often feasible), and it is largely reduced if several blocks must nonetheless be defined (e.g. because of practical block size limitations with a given FEC code). Even though not MDS, the reception overhead is usually smaller than that of RSE with large objects, while being an order of magnitude faster to process.

Most large block codes derive from the well known Low Density Parity Check (LDPC) codes [10]. LDPC codes rely on a parity check matrix which forms a system of linear equations between source and parity symbols. If the system is built appropriately, encoding is extremely fast. However, one cannot know in advance how many symbols must be received before decoding is successful (LDPC codes are not MDS), so decoding is performed progressively after each packet arrival. Due to the simplicity of LPDC, the decoding process is extremely fast too.

The LDPC-Staircase and LDPC-Triangle are two variants [27] for which a publicly available GNU/LGPL code implementations exist [21]. Like the LDPC codes, Tornado codes are sparse graph codes. However, Tornado codes are covered by patents [19, 18, 17, 16].

Table 1 presentes a performance comparision and shows the encoding[2] and decoding speeds of the RSE, LDPC Staircase and LDPC Triangle codecs, on a Pentium 4 Xeon 3.06-GHz 2GB RAM for the case of symbols that are 1024 bytes long. We clearly see that the two LDPC codes largely outperform the RSE code.

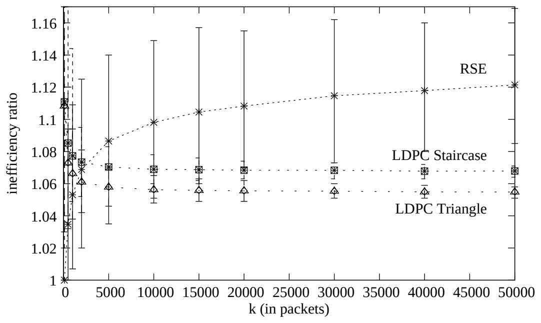
| *Code* | *Encoding* | *Decoding* |
|---|---|---|
| LDPC Staircase | 734.998 Mbps | 816.505 Mbps |
| LDPC Triangle | 443.813 Mbps | 434.866 Mbps |
| RSE | 21.471 Mbps | 52.255 Mbps |

**Table 1: Average encoding/decoding speeds on a Pentium 4 Xeon 3.06GHz, with object size = 20000 symbols and FEC expansion ratio n/k = 1.5.**

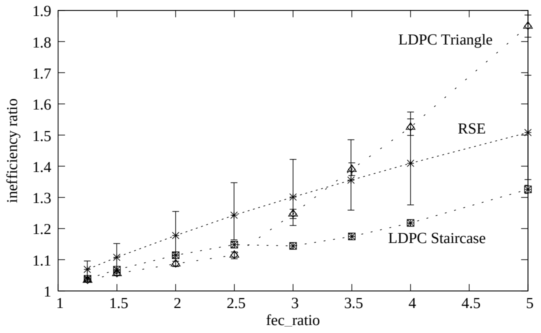
The reception overheads for these codes are depicted in figure 2. The metric is the global decoding inefficiency ratio, defined as the number of symbols required for decoding divided by the number of source symbols. This ratio is shown as a function of the object size (figure 2(a)) and as of the FEC expansion ratio $n/k$ (figure 2(b)). We clearly see that the two LDPC codes achieve good performance for object larger than 2000 symbols, and that for a given FEC expansion ratio one of the two large block FEC always outperforms RSE. Figure 2 also shows the minimum and maximum 99% confidence intervals which indicate that the two LDPC codes yield more stable results than RSE.

## 2.3 Expandable/Rateless FEC Codes

With large block codes, the value of $n$ must be defined *before* encoding is performed and cannot be changed afterwards. Moreover the FEC inefficiency ratio increases with the number of encoding symbols produced. Expandable



(a) Inefficiency ratio as a function of the object size; FEC expansion ratio = 1.5



(b) Inefficiency ratio as a function of the FEC expansion ratio; object size = 20000 symbols.

**Figure 2: Inefficiency ratios of RSE, LDPC Staircase and LDPC Triangle.**

codes do not have any of these limitations. They can operate on very large blocks ($k$) and an unlimited number of parity symbols ($n - k$) can be produced without affecting the erasure recovery capabilities. As the "code rate" (i.e. the ratio $k/n$) can be very small, they are also called "rateless codes".

Expandable codes are well suited to content broadcasting, since the sender often wants to produce new parity symbols on the fly, without limitations (e.g. in on-demand mode). With these codes, a receiver is guaranteed to receive never any duplicate parity symbols. Raptor [28] codes fall in this category.

## 3. ASYNCHRONOUS LAYERED CODING (ALC)

### 3.1 Introduction and Principles

ALC is a content delivery protocol that does not require any feedback from receivers. This makes it *massively scalable* (millions of receivers are easily supported). This feature also enables ALC to be used over *unidirectional links* where there is no (or a limited) return channel (e.g. satellite broadcast).

ALC has been designed to support the three delivery service models introduced before: *push*, *on-demand*, and to a lesser extent *streaming*.

[2]The encoding speed only takes into account the produced parity symbols.

In order to provide a fully or (partially) reliable service, *ALC largely relies on the use of FEC*. After a FEC encoding of the content source blocks, redundant data is transmitted (along with the original data in the systematic FEC case).

ALC also meets the channel and client heterogeneity challenges. Concerning the channel heterogeneity, a theoretically infinitely long (e.g. in on-demand mode) ALC session that repeats symbol transmissions guarantees that clients can decode the content successfully, independently of all other receivers and no matter the number of losses experienced. Therefore, ALC provides *high robustness*. ALC also supports receiver heterogeneity in terms of sustainable reception rate, by using (optional) *multi-layered transmission* (section 3.2.3).

The following terminology will be used: ALC transports *objects* (e.g. files). Each object is logically divided into (one or more) *blocks* of appropriate size upon which FEC codes calculate source and parity symbols.

## 3.2 ALC Building Blocks

### 3.2.1 Layered Coding Transport (LCT) Building Block

The LCT BB [14] defines the basic features and the LCT header that provides a number of fields to convey in-band session information to receivers.

An ALC/LCT session is fully identified by the $\{source\ address; TSI\}$ tuple, where TSI is the *Transport Session Identifier*. The destination IP address and the destination UDP port do not identify an ALC session, but rather, they are only used in transmission for distinguishing between ALC channels (layers) of a certain ALC session.

Each object is uniquely identified by its *Transport Object ID* (TOI). Nothing is specified on how to generate the TOI: it may be incremented for each new object, or be the hash of the object. For that reason, the size of the TOI field is not predefined ([14] defines a $[16; 112]$ bit range).

LCT defines a *Congestion Control Information* (CCI) field to associate a congestion control (CC) building block. Yet LCT does not specify the CCI field format which must be specified by the congestion control BB itself. The field is set to null if no congestion control is used (e.g. in 3G and DVB-H environments - which are not part of the public Internet).

Finally, LCT defines *Header-Extensions* to carry additional information. Optional header fields that are not always used or have a variable size can be added this way. For instance the FEC BB and the authentication BB heavily rely on these extension headers.

### 3.2.2 FEC Building Block

The FEC BB [13, 30] is a particularly important BB in achieving scalability and reliability with ALC (and NORM), even though it is not mandatory. However, a FEC code alone is not usable, since the sender and receivers must: (1) identify the codec, (2) identify each symbol, and (3) specify the various parameters used by the code for a given object. To that purpose the FEC BB defines:

1. FEC Encoding ID and FEC Instance ID: These two values fully identify the FEC code. The FEC Encoding ID is the primary identifier, and identifies either specific FEC codes with *fully-specified* schemes, or a set of codes with *under-specified* schemes. In the latter case, the FEC Instance ID must be used in order to completely identify a code.
2. FEC Payload ID (FPI): Each FEC Encoding ID is associated with a FEC Payload ID, which uniquely identifies each symbol of an object after FEC encoding. For instance with the under-specified FEC Encoding ID 128, each encoding symbol of a given object is identified by its Source Block Number and Encoding Symbol ID.
3. FEC Object Transmission Information (FEC OTI): To each FEC Encoding ID, a FEC OTI must be specified to synchronize the FEC codes implementations at both ends with respect to the various parameters of an object (e.g. maximum source block length, object size, etc.).

### 3.2.3 Congestion Control Building Block

When used over the public Internet, ALC must include a congestion control BB. In order to preserve the massive scalability, only feedback free schemes are defined. Therefore congestion control is multi-layered and completely receiver-driven: the content is multicast on several layers, each layer being associated to a distinct multicast group. Then each receiver chooses dynamically how many layers to receive, depending on the sustainable transmission rate along the path to this receiver.

Several (more or less) TCP-friendly protocols have been defined: RLC [29], FLID-SL/DL [5], and WEBRC [15].

The client behavior will differ according to the protocol: transmissions will take place either at some fixed predefined rates on each layer (RLC, FLID-SL), or using a cyclic, dynamically changing bit-rate (FLID-DL, WEBRC); the protocol can either be equation-based (WEBRC) or event based (RLC, FLID). Naturally, these fundamental differences impact upon their performance and TCP-friendliness.

When used over provisioned, pre-allocated, channels (e.g. within 3G MBMS and DVB-H environments), no transport layer congestion control is needed.

### 3.2.4 Authentication Building Block

The goal of this BB is to perform both a packet source authentication and packet integrity check. The TESLA authentication approach [24] is a good solution that is well suited to the case of high rate transmissions over lossy channels. [9] gives more details on how to use it in an ALC (or NORM) context.

## 4. NACK ORIENTED RELIABLE MULTICAST (NORM)

## 4.1 Introduction and Principles

The NORM approach essentially relies on retransmission requests (Negative Acknowledgments or NACKs), and optionally on positive acknowledgments (ACKs) after an explicit request from the source. Due to of this feedback mechanism, NORM scalability is limited to *small or medium* sized groups [4]. Large groups are a problem because the timer-based feedback suppression mechanism (used to increase NORM scalability) is still not sufficient to prevent feedback implosion in this case.

Receivers should be *relatively homogeneous* and have comparable and sustainable reception rates. Indeed the associated congestion control protocol adapts the transmission rate to the slowest receiver. Consequently, a small number of receivers can easily slow down an entire NORM session. This is not scalable even under the best assumptions, since the throughput goes down with $1/\sqrt{number\ of\ receivers}$ [7].

NORM is a more *complex protocol* than ALC. For instance there are 13 different packet types! We will not detail NORM as we did for ALC because this protocol is less interesting in our context. Interested readers can refer to [3]. We only give some general ideas.

## 4.2 NORM Building Blocks

[2] defines several BBs, in particular:

- NORM Sender Transmission strategies
- NORM Repair Process with timer-based feedback suppression
- NORM Receiver Join Policies
- The FEC BB [13]: With FEC Encoding ID 129, the default scheme, the block size can be adapted dynamically. In case of a high loss rate, the sender can choose to reduce the block size in order to generate a higher number of parity packets and vice-versa.
- The Round-Trip Timing Collection BB and the Group Size Determination/Estimation BB
- The congestion control BB: One of two BBs may be used: PGMCC [26] and TFMCC [32]
- The authentication BB: Here also the TESLA authentication scheme can be used, as explained in [9].
- The Router/Intermediate System Assistance BB: This BB turns NORM into a router assisted protocol (similarly to TRACK/GRA)

Other BBs are specified in [2], but we will not detail them. Nevertheless the number BBs defined once again emphasizes the complexity of the protocol.

## 4.3 Comparison of NORM and ALC

If we compare NORM and ALC with respect to the challenges introduced in section 1 we see that: (1) only ALC supports massive scalability; (2) ALC supports client heterogeneity; (3) both protocols support file transmission, but NORM is not suited to data having real time constraints; (4) for long reception blackouts, ALC offers more robustness than NORM; (5) both protocols can use congestion control.

Therefore, in the following we focus on ALC since it better addresses the challenges introduced earlier, offers a higher flexibility, can be adapted to very different contexts, and is less complex.

## 5. FLUTE FOR FILE DELIVERY

## 5.1 Introduction and Principles

File Delivery over Unidirectional Transport (FLUTE) [23] is a fully-specified file delivery protocol/application built on top of ALC. It inherits all the major assets of ALC: an unlimited scalability, the ability to use almost any IP transmission channel (unidirectional or not), and a high robustness. FLUTE transmits meta-information for each file delivered in a file delivery session. The meta-information contains such information as the file identifier (as a URI which may include the file name), file size, content type (MIME type) and content encoding (such as compression). It is capable of delivering, for instance, the required video codec and textual description along with the video media file itself. A receiver can then choose to decode all or a subset of files based on this information. For example, a receiver without a certain video codec my chose to drop all ALC packets associated with files using that video codec (to avoid processing data which could finally be unusable).

### *5.1.1 File Delivery Table (FDT)*

The meta-information of *all files* of a FLUTE session is contained in the session's abstract File Delivery Table (FDT). To each file is associated at least a *TOI* (used by ALC to identify the object) and a file URI (Uniform Resource Identifier) (*Content-Location* attribute). Additional information may be specified, including: the file size (*Content-Length*), the MIME-type (*Content-Type*), and a MD5 hash (*Content-MD5*). The FDT information is not generally static, in particular an attribute may change (e.g. a file is renamed), and files may be added or dropped dynamically.

The FDT is delivered via *FDT-Instances*, which are discrete XML representations of a subset of the abstract FDT. To represent its dynamic nature, each FDT-Instance includes an expiry timestamp (*Expires*), which applies to all data contained in that FDT Instance. Additionally an FDT-Instance may contain the *Complete* attribute to indicate that the meta-information transmitted so far will not change anymore. Once used, there can be no file addition or attribute change and no additional files will be added to the session - making it simpler for a receiver to determine when it has received all interesting files for a particular session.

It is the sender's role to decide when and how to transmit FDT-Instances (i.e. which subset of the known valid FDT information to include in FDT-Instance). It is, however, recommended to transmit a file description before the file itself, so that the receivers know how to handle incoming ALC packets (e.g. by dropping them, or directly writing on disk, or keeping them in memory). There are several ways to schedule the FDT-Instances. In on-demand mode, they must be periodically transmitted to support mid-session receiver-joins - enabling receiver selection and download of content they are interested in, at any time.

FDT-Instances are transmitted with the *dedicated TOI value 0*. By searching the TOI of a received packet in its cached FDT database, a receiver easily finds the associated file and can decide to process it or drop it.

Figure 3 shows an example FDT-Instance. Two files are described, of type html and mp3 (*Content-Type*), and mapped onto TOIs 1 and 2 respectively. The html file has been content encoded before transmission, i.e. the FLUTE sender has compressed it using "gzip" (*Content-Encoding*). The size of the uncompressed file is "6100" bytes (*Content-Length*), but the actual transport size is smaller. The MD5 checksum (*Content-MD5*) of the uncompressed content allows receivers to check the integrity of the received file. The FDT-Instance has a limited validity period (*Expires*), which represents the time in seconds relative to January 1st 1900, 0h when it will time out.

```
<?xml version="1.0" encoding="UTF-8"?>
<FDT-Instance
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:fl="http://www.example.com/flute"
xsi:schemaLocation="http://www.example.com/flute-fdt.xsd"
Expires="2890842807">
  <File  Content-Location=
             "http://www.example.com/menu/tracklist.html"
         TOI="1"
         Content-Length="6100"
         Content-Type="text/html"
         Content-Encoding="gzip"
         Content-MD5="+VP5IrWploFkZWc11iLDdA=="/>
  <File  Content-Location=
             "http://www.example.com/tracks/track1.mp3"
         TOI="2"
         Content-Length="392103"
         Content-Type="audio/mp3"/>
</FDT-Instance>
```

**Figure 3: FDT-Instance example.**

## 5.2 FLUTE in Practice

### 5.2.1 Delivery Models

FLUTE can be used differently according to packet and file scheduling. In the basic use case, the content is only sent once, each file one after another, as depicted in figure 4. The content is static (i.e. does not change during transmission) and the FDT-Instance is delivered before the first file. Each receiver can then choose and download the content it is interested in. This corresponds to a *push* delivery model (section 1.2) where all receivers need to be ready before transmission starts. With this model, loss recovery capabilities are limited, since the session may not last long enough for some receivers to entirely recover from losses. Therefore an additional loss recovery mechanism may be required, (e.g. one such file repair mechanism is provided for 3G MBMS, section 6.2).

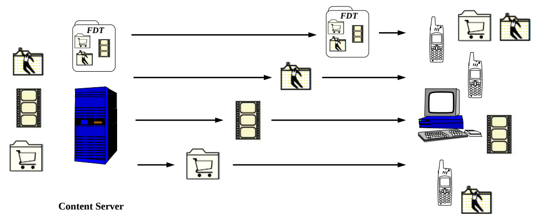


**Figure 4: One shot delivery (push model).**

In *on-demand* delivery the files are typically transmitted in a carousel, cyclically over a longer time period, as depicted in figure 5. If the content does not change in this use case, receivers can join and download the content at any time, and all losses will finally be recovered just by listening - providing the session is sufficiently long. In order to make the download performance independent of the loss model, the packets of all files may be transmitted in a random order.

However, if a file changes dynamically, it must be described in a new FDT-Instance and the file must be changed in the carousel. The current cycle can then be stopped in order to make the new file available rapidly, or the sender may choose to wait for the next carousel cycle. This is depicted in figure 6.

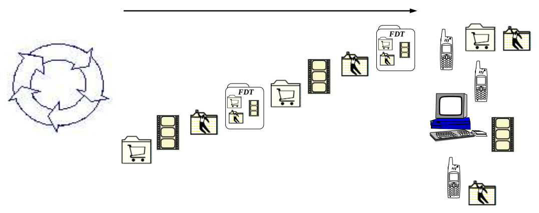


**Figure 5: Static carousel (on-demand model).**

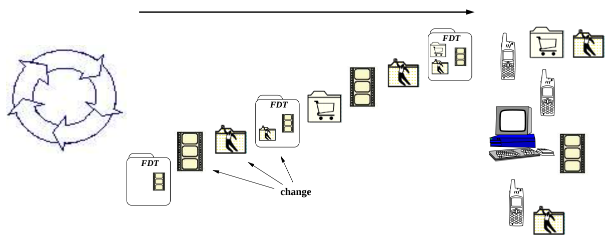


**Figure 6: Dynamic carousel.**

### 5.2.2 Session Description

To inform potential receivers of an ongoing FLUTE sessions and in order to communicate all FLUTE and ALC session parameters to the receivers, [20] explains how to use the Session Description Protocol (SDP) syntax to describe the parameters required to begin, join, receive data from, and end FLUTE sessions.

### 5.2.3 Service Announcement

Service announcement consists of the delivery of service descriptions (metadata) from server to receiver. Since this delivery must be both reliable and scalable, it lends itself very well to the concepts of massive content distribution of discrete objects. Recent work related to "Internet Media Guides" (IMGs) [22] has proposed the use of FLUTE as the primary means to announce metadata over multicast/broadcast capable networks.

## 6. LARGE SCALE CONTENT DELIVERY FOR MOBILE DEVICES

## 6.1 An Example: Olympic Games

During a big event like the Olympic Games, a lot of digital content is produced and made available to the public. The *content is extremely heterogeneous*: live video and audio streams, sport event results (archived and in real-time), recorded interviews and sport events, timetables of upcoming events, programs/contents guides, etc. Due to the popularity of the Olympic Games, there are potentially *thousands or millions of clients accessing the same content at the same time.* Bringing all this content to the clients is a challenging task, and in this example, we assume that clients are equipped with UMTS/MBMS and/or DVB-H mobile terminals.

*UMTS* features a bidirectional channel which makes it possible to explicitly request and receive a content over a point-to-point connection. However this is not necessarily the most effective solution. In particular, it raises *scalability issues:* with a high number of clients, (1) the server must support a large number of simultaneous downloads, handle potentially a lot of retransmissions requests (the wireless channel can be a very lossy one), and (2) the current WCDMA wireless channels, where finite bandwidth is shared among all clients, do not yet handle a very high numbers of simultaneous high speed downloads. It also raises *power consumption issues:* for a mobile device, using the uplink channel consumes significant power as the device acts as the radio transmitter (other reverse applies for reception), even if only a very small number of feedback messages are sent. Since mobile devices continue to have limited battery life as new mobile applications eat additional power at a comparable rate to the gains battery storage capacity, this issue has to be taken very seriously.

Due to these considerations, a popular content (e.g. the results of the 100m finals), requested simultaneously by a high number of clients, should be broadcast rather unicast to each client independently to counter the problems raised above. Moreover this perfectly matches the broadcast nature of the radio channel.

### 6.2 MBMS and IP Datacast Services

The Multimedia Broadcast Multicast Service (MBMS) is an enhancement to the current 3G/UMTS cellular systems providing mass media services over IP Multicast and multipoint radio to distributed mobile user groups. It specifies two primary delivery services, download and streaming, upon which user services can be built. [1] standardized the use of FLUTE for both discrete object (or file) download and the announcement of metadata fragments, and likewise, RTP (Real-time Transport Protocol) for multicast delivery of continuous audio/visual streams (figure 7). Additionally MBMS defines out-of-band repair mechanisms that allow recovery of partially received files of a FLUTE session. This is done via explicit requests to the source and by defining a reception report mechanism.

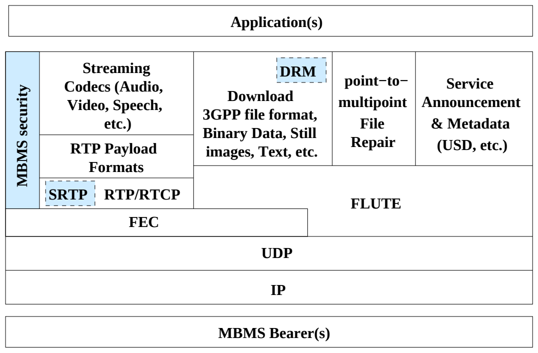

Application(s)
MBMS security
Streaming Codecs (Audio, Video, Speech, etc.)
RTP Payload Formats
SRTP
RTP/RTCP
DRM
Download 3GPP file format, Binary Data, Still images, Text, etc.
point−to−multipoint File Repair
Service Announcement & Metadata (USD, etc.)
FLUTE
FEC
UDP
IP
MBMS Bearer(s)

**Figure 7: The MBMS protocol stack in the downlink.**

*DVB-H* is an unidirectional, high bitrate technology, designed for any kind of content broadcasting to handheld devices. It is an excellent solution to broadcast live video or audio streams, such as in the Olympic games example. Additionally, DVB-H can also be used to broadcast popular content, for instance to make sporting results rapidly available to all DVB-H mobile devices. When a bidirectional channel is needed, for instance to access pay-per-view content or to add further reliability services, then UMTS can be used in parallel to DVB-H. The IP Datacast (IPDC) specifications [8] define the download and streaming services for DVB-H, and are based on FLUTE and RTP respectively, just like MBMS.

The potential of multicast to provide a low cost delivery channel to an existing massive base of mobile users, specifically for mass media, is very exciting. Mobile mass media, including MBMS and IP Datacast using DVB-H is likely to offer the ultimate showcase to demonstrate the benefit of large scale immediate content distribution protocols.

Practical network implementations of MBMS are expected by the end of 2007, and the first functional mobile terminals supporting MBMS are estimated to be available by the end of 2008. For IPDC/DVB-H trials and pilots have already started in many countries. Therefore, the first generation of DVB-H services will reach the market much sooner than MBMS.

## 7. CONCLUSIONS

This paper introduces the principles and the use of multicast content delivery protocols and applications. It focuses on the FLUTE application and the underlying ALC transport protocol because of their increasing importance, in particular after their adoption in the 3GPP MBMS and DVB-H IP Datacasting standards. The paper also explains the FEC building block, which is a key component for any scalable content delivery solution. The NORM transport protocol provides several useful mass media features and is briefly described, although its field of application to large scale delivery is much more limited than that of FLUTE and ALC.

However, FLUTE/ALC does not offer a fully reliable distribution service when transmissions have finite duration (e.g. in *push* mode). Therefore a complementary reliability mechanism, built on top of FLUTE or ALC, may be required in some cases. This has been done in MBMS with a dedicated file repair mechanism, whose scalability in case of widespread packet loss remains to be proven. Future work to establish the optimal combinations of "application level" reliability techniques across a wide range of use cases would contribute substantially to further improve large scale content distribution services. Ideally, any such scheme or optimization will improve scalability and independence from the underlying transport layer (i.e. ALC/FLUTE or NORM).

IPDC and MBMS provide security through a hybrid of DRM (Digital Rights Management) and specialized transport security protocols. The full advantage of IETF security mechanisms (such as IPsec and MSEC - Multicast Security) is yet to be explored and exploited. Especially in open networks (e.g. Wifi hot spots) and the public Internet, security is another area from which to expect future development. Providing source authentication and message integrity service is only the first step to that goal.

## 8. REFERENCES

[1] 3GPP. *3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Multimedia Broadcast/Multicast Setvice; Protocols and Codecs (Release 6)*, Nov. 2004. 3GPP TS 26.346 v1.5.0.

[2] B. Adamson, C. Bormann, M. Handley, and J. Macker. *NACK-Oriented Reliable Multicast (NORM) Building Blocks*, Nov. 2004. Request For Comments 3941.

[3] B. Adamson, C. Bormann, M. Handley, and J. Macker. *Negative-acknowledgment (NACK)-Oriented Reliable Multicast (NORM) Protocol*, Nov. 2004. Request For Comments 3940.

[4] R. B. Adamson and J. Macker. Quantitative prediction of nack oriented reliable multicast (norm) feedback. In *Proc. IEEE MILCOM 2002*, Oct. 2002.

[5] J. Byers, M. Frumin, G. Horn, M. Luby, M. Mitzenmacher, A. Roetter, and W. Shaver. Flid-dl: Congestion control for layered multicast. In *NGC2000*, Nov. 2000.

[6] J. Byers, M. Luby, M. Mitzenmacher, and A. Rege. A digital fountain approach to reliable distribution of bulk data. In *ACM SIGCOMM'98*, Aug. 1998.

[7] A. Chaintreau, F. Baccelli, and C. Diot. Impact of network delay variation on multicast sessions performance with tcp-like congestion control. *IEEE Trans. on Networking (TON)*, pages 500–512, 2002.

[8] ETSI. *IP Datacast Baseline Specification: Specification of Interface I MT (a080)*, Apr. 2004. DVB Interim Specification.

[9] S. Faurite, A. Francillon, and V. Roca. *TESLA source authentication in the ALC and NORM protocols*, July 2005. Work in Progress: <draft-faurite-rmt-tesla-for-alc-norm-00.txt>.

[10] R. G. Gallager. Low density parity check codes. *IEEE Transactions on Information Theory*, 8(1), Jan. 1962.

[11] S. Kesha. Why cell phones will dominate the future internet. *ACM Computer Communication Review (CCR)*, 2005.

[12] M. Luby, J. Gemmell, L. Vicisano, L. Rizzo, and J. Crowcroft. *Asynchronous Layered Coding (ALC) protocol instantiation*, Dec. 2002. Request For Comments 3450.

[13] M. Luby, J. Gemmell, L. Vicisano, L. Rizzo, M. Handley, and J. Crowcroft. *Forward Error Correction Building Block*, Dec. 2002. Request For Comments 3452.

[14] M. Luby, J. Gemmell, L. Vicisano, L. Rizzo, M. Handley, and J. Crowcroft. *Layered Coding Transport (LCT) building block*, Dec. 2002. Request For Comments 3451.

[15] M. Luby, V. Goyal, S. Skaria, and G. Horn. Wave and equation based rate control using multicast round trip time. In *ACM SIGCOMM'02*, Aug. 2002.

[16] M. Luby, A. Shokrollahi, V. Stemann, M. Mitzenmacher, and D. Spielman. *Irregularly graphed encoding technique*, June 2000. U.S. Patent No. 6,081,909.

[17] M. Luby, A. Shokrollahi, V. Stemann, M. Mitzenmacher, and D. Spielman. *Loss resilient decoding technique*, June 2000. U.S. Patent No. 6,073,250.

[18] M. Luby, A. Shokrollahi, V. Stemann, M. Mitzenmacher, and D. Spielman. *Message encoding with irregular graphing*, Dec. 2000. U.S. Patent No. 6,163,870.

[19] M. Luby, A. Shokrollahi, V. Stemann, M. Mitzenmacher, and D. Spielman. *Loss resilient code with double heavy tailed series of redundant layers*, Feb. 2001. U.S. Patent No. 6,195,777.

[20] H. Mehta, R. Walsh, I. Curcio, J. Peltotalo, and S. Peltotalo. *SDP Descriptors for FLUTE*, May 2005. Work in Progress: <draft-ietf-rmt-flute-sdp-02.txt>.

[21] C. Neumann, V. Roca, and J. Labouré. *An Open-Source Implementation of Low Density Parity Check (LDPC) Large Block FEC Codes.* URL: http://www.inrialpes.fr/planete/people/roca/mcl/.

[22] Y. Nomura, R. Walsh, J.-P. Luoma, H. Aseada, and H. Schulzrinne. *A Framework for the Usage of Internet Media Guides*, July 2005. Work in Progress: <draft-ietf-mmusic-img-framework-08.txt>.

[23] T. Paila, M. Luby, R. Lehtonen, V. Roca, and R. Walsh. *FLUTE - File Delivery over Unidirectional Transport*, Oct. 2004. Request For Comments 3926.

[24] A. Perrig, R. Canetti, J. D. Tygar, and D. Song. The tesla broadcast authentication protocol. *RSA CryptoBytes*, (Summer), 2002.

[25] L. Rizzo. Effective erasure codes for reliable computer communication protocols. *ACM Computer Communication Review*, 27(2), Apr. 1997.

[26] L. Rizzo. pgmcc: a tcp-friendly single-rate multicast congestion control scheme. In *ACM SIGCOMM '00*, 2000.

[27] V. Roca and C. Neumann. Design, evaluation and comparision of four large block fec codes, ldpc, ldgm, ldgm staircase and ldgm triangle, plus a reed-solomon small block fec codec. Research Report 5225, INRIA, June 2004.

[28] A. Shokrollahi. Raptor codes. Research Report DR2003-06-001, Digital Fountain, 2003.

[29] L. Vicisano, L. Rizzo, and J. Crowcroft. Tcp-like congestion control for layered multicast data transfer. In *IEEE INFOCOM'98*, Feb. 1998.

[30] B. Watson, M. Luby, and L. Vicisano. *Forward Error Correction (FEC) Building Block*, Apr. 2005. Work in Progress: <draft-ietf-rmt-fec-bb-revised-00.txt>.

[31] T. Whetten, L. Vicisano, R. Kermode, M. Handley, S. Floyd, and M. Luby. *Reliable Multicast Transport Building Blocks for One-to-Many Bulk-Data Transfer*, Jan. 2001. Request For Comments 3048.

[32] J. Widmer and M. Handley. Extending equation-based congestion control to multicast applications. In *ACM SIGCOMM '01*, 2001.