mgpm – a multicast-enabled presentation tool

Luigi Rizzo

Dip. di Ingegneria dell'Informazione, Università di Pisa
via Diotisalvi 2 – 56126 Pisa (Italy)
email: l.rizzo@iet.unipi.it
http://www.iet.unipi.it/~luigi/

Abstract. We present a multicast-enabled presentation tool designed to deliver slide-based presentations on the MBone. Adaptation and network-efficiency are primary objectives of this work: we want to deliver high quality slides with a moderate and adaptive use of network resources. To this purpose, we use a congestion-controlled, FEC-based multicast data delivery protocol for the bulk data transfer, while real-time data, which generally involve only a moderate amount of traffic, are transmitted using a specialized protocol with RTP-encapsulated packets. This approach lets high quality presentations be delivered using only a very low-rate data stream, a feature which is not present in similar applications to date.
In this paper we describe the overall architecture of mgpm and its components, discussing design choices and commenting our initial experience with the implementation. Full source code for this tool is available from the author.
Keywords: adaptive applications, reliable multicast, congestion control.

1 Introduction

In recent years, a number of tools have appeared for computer-supported co-operative work on the Internet. Such tools cover a wide range of applications, including audio and videoconferencing [3, 12], shared text editing [5], network whiteboards [11]. In this framework, a missing piece of software is a presentation tool for the delivery of a slide-based presentations to an audience distributed over the network. Existing tools such as nv, vic, wb (or their commercial equivalents) have been used for this purpose but with a number of limitations.

Presentations are rather peculiar in that the material is usually prepared beforehand, and it is often in the form of bulky data files (e.g. containing images or large postscript files). The size contrasts with the requirement that the speaker be able to switch rapidly from one slide to the other, and possibly annotate them with marks or drawings done while talking. Supporting such functionalities with a generic videoconferencing tool is hard because the high speed rendering of new slides requires high bandwidth, which may not be available for all receivers; on the other hand, switching to a lower quality (and bandwidth) might render slides and annotations poorly readable.

In this paper we present the approach we followed to create mgpm, a network-enabled presentation tool which is both scalable and network-conscious. mgpm is specifically designed to take advantage of the peculiarities of a slide-based presentation in order to i) deliver high-quality slides to a wide audience using a very low rate data stream, and ii) scale to large sets of receivers. The novelty of the application relies on the separation of bulk data and annotations, and the use of congestion-controlled protocols for the transfer of bulk data. In the rest of this section we present design goals and requirements for `mgpm`, and discuss some related work. In Section 2 we describe the architecture of the application and the communication protocols used for distributing data. Section 3 finally illustrates some implementation details, experience of use and suggests future enhancements for the application.

1.1 Goals and requirements

Our goal was to develop a tool that could be used to deliver slide-based presentations over the MBone (and possibly other multicast/broadcast networks or media) in a flexible and efficient way. By flexible we mean that the tool should support the typical actions performed by a speaker while showing slides, such as live annotations and quick navigation through the presentation. Additional services should also be supported such as let receivers record the presentation, produce a hardcopy, and browse through the slides independently from the actions of the speaker. Our metric for efficiency is the way network resources are used: the application should be able to work well over slow or lossy links, and should be adaptive to the network conditions in order to compete fairly with other network users.

As we have briefly commented in the previous section, a presentation tool is different from a whiteboard or a videoconferencing application for the following reasons:

- the tool is mainly unidirectional. Only the speaker is generally allowed to change or annotate slides (receivers can do that on their local copy, but their actions should not be propagated to the whole group).
  In contrast, a whiteboard is generally shared among many users, which are treated in a highly symmetric way; this influences the communication protocols used.
- the presentation material is prepared in advance, often including high-quality drawings, images or plots. The switch from one slide to the other must be quick, but afterwards slides tend to remain on-screen for a relatively long time.
  A video tool cannot usually exploit the advanced availability of the material, and it would need special buffering to smooth out the large degree of burstiness in the video stream associated with slide changes.
- annotations are dynamic, and it is important to preserve their timing, especially when recording a presentation for later replay.
  With a whiteboard tool, the shared workspace has often no notion of time, but it just contains objects which are drawn on the screen as soon as they are available at the receiver.

These differences prompted us in developing a separate tool for delivering presentations, which can overcome the limitations of network whiteboards and videoconferencing tools. In particular, we want to exploit the advanced availability of the presentation material to reduce the bandwidth requirements of the application, while preserving the quality of the service provided. Also, we want to improve receivers' ability to make use of the presentation material compared to the facilities offered by a generic videoconferencing tool.

1.2 Related work

Most of the times, slide-based presentations are transmitted over the MBone using a generic video tool such as `nv` [3] and `vic` [12], and by capturing real slides through cameras. This approach suffers from a significant quality degradation because of the capture process. Furthermore, video streams are not transmitted reliably, because the low persistence of live video requires frequent refresh of the image (or portions of it). Slides have a longer persistence, so a much lower refresh frequency is required, allowing considerable bandwidth savings provided the slide data is sent reliably.

Network whiteboards are a closer relative to presentation tools. The difficulty with them lies in their design as a shared workspace, which is different from the one-to-many model which is more natural for presentation tools. Furthermore, the use of pre-recorded material is not well supported by whiteboards.

Recently, a work by Gemmel[4] addressed this problem by implementing a multicast version of Power Point. Gemmel splits the input stream in different layers depending on the persistence of the data they carry, and uses three different protocols for transporting them. The multicast PowerPoint uses a very close derivative of our original, non congestion-controlled RMDP implementation [16] to transfer the bulk of the presentation; a separate protocol based on SRM [8] and FEC [14] to transfer the command stream; and a non-reliable protocol to transmit volatile data.

There are two main differences between mgpm and the multicast PowerPoint: first, `mgpm` uses a congestion-controlled protocol for the bulk data transfer, which makes our application adaptive to the network conditions. Second, `mgpm` restricts dynamic data to commands and annotations only. As a consequence, the real-time stream is much more compact and it can be transmitted without the use of feedback from the receivers (this also applies to the bulk transfer protocol we use). This is achieved through the use of a redundant protocol and the transmission of global state information in each packet. The no-feedback requirement of `mgpm` is particularly useful in that it permits the recording and playback of a session with ordinary RTP [17] tools.

2 mgpm architecture

mgpm is made of three components: a rendering tool, a bulk transfer protocol, and the command protocol. The task of the rendering tool is just to run the user interface: display slides and accept/process annotations and other user commands.

The rendering tool uses two data sources to drive its operations (Fig.1): one is a *source file* containing all the slides which make up the presentation itself; the other one is a *command stream* coming from the user, who selects which slides to show and can optionally annotate them live. These two sources have radically different features, and as such they are transferred using two different protocols.

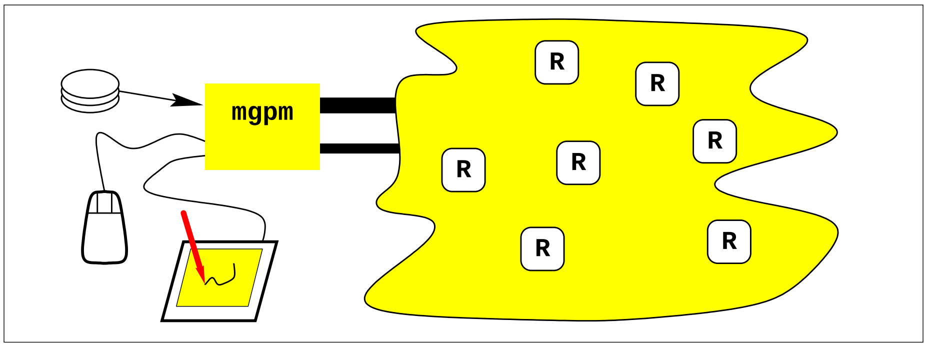

Fig. 1. The model of operation of `mgpm`, showing the different data sources used by the sender.

The source file is generally prepared in advance. Depending on the type of presentation, its size can range from less than 1KB to hundreds of KB and more. The source file needs a reliable communication protocol to be transferred to the receivers, but it does not have strict real time requirements: in fact, we can exploit the fact that it is prepared in advance to start its delivery early enough, so that receivers can complete reception even in presence of connectivity impairments. `mgpm` provides mechanisms for loading the source file using multicast or unicast communication, or from local storage (the latter is similar to listening to a lecture while looking at a hard-copy of the presentation material, e.g. a book.). Care has been taken to use network-friendly mechanisms (that is, congestion controlled) for loading the presentation. Such mechanisms – especially the reliable multicast one – will be described in more details in Section 2.2.

The command stream instead carries the actions performed by the speaker during the presentation, like selecting one or another slide, and possibly annotating them. Such actions are generally recorded as keyboard, mouse or pen input, and as such require a very low bandwidth to be transferred. Data in the command stream have stronger real-time requirements than those in the presentation file. Especially if the presentation is supported by other streams such as audio and video, some degree of inter-media synchronization is required so that annotations and page changes occur at the right time. Also, a (semi-) reliable transfer mechanism is required for such commands because page changes, annotations and drawings should not appear out of order or be truncated. On the other hand, the limited lifetime of commands such as page changes, which are effectively overridden by subsequent commands of the same kind, loosens the reliability requirements allowing some commands to be dropped if they cannot be retransmitted in a timely fashion. The protocol used to transfer the command stream is described in Section 2.3.

Decoupling the mechanisms used to transfer the two sources of data has enormous benefits on the network behaviour of the application. In fact, we can achieve quick rendering of high quality slides without the need for high-bandwidth connection to the source. Furthermore, because of the protocols used, `mgpm` can be run using a fully unidirectional communication channel, which is a very convenient feature to achieve scalability and recordable sessions.

2.1 The rendering tool

The rendering tool we used for `mgpm` is called "MagicPoint" or `mgp` [7]. `mgp` uses a simple markup language to prepare a presentation, and provides mechanisms to render most graphic formats. In order to keep the program reasonably small and simple, support for complex formatting or graphic formats, such as mathematical formulas or PostScript, is left to external programs (TEX, `ghostscript`) used to produce pixmaps with the desired resolution. MagicPoint provides simple navigation commands to switch slides, or annotate the presentation with simple hand-drawings. The basic presentation tool does not have any communication ability[1].

In order to produce `mgpm`, we have added communication capabilities to the original rendering tool, for transferring both the source file and the command stream using network-transparent binary formats. The two protocols used are described in the next sections.

2.2 The bulk transfer protocol

There are several options to transfer the source file to the receivers[2]. The simplest one is to load it from local storage; this is a very convenient method when the tool is used to deliver lectures using pre-recorded material distributed on CDs. In this case there is no network issue for the transfer of the presentation stream, and there are obviously no scalability problem with the use of this approach.

A second method permits the download of the presentation file using the HTTP protocol over TCP; this method is useful when the number of receivers is not expected to be too large, or they only have unicast connectivity. Because TCP implements congestion control, we are sure not to overload the network

[1] with the exception of a very simple synchronization mechanism: an external file can be `touch`'ed every time there is a slide change (this does not include annotations though).

[2] The download of the presentation is done in certain cases by forking an external program. So it is very simple to add more methods for transferring the presentation file

by transferring the presentation in this way. On the other hand, as the number of receivers grows, the same data might be transferred multiple times over the same paths from the source to the receivers, thus causing a waste of network resources. The use of Web caches can partially alleviate the problem, to a certain degree.

As the number of receivers grows, a much more sensible way to transfer data is to use multicast. `mgpm` uses RLC-RMDP, a congestion-controlled, reliable multicast file transfer application based on Forward Error Correction (FEC) techniques [14]. RLC-RMDP is really the composition of two separate mechanisms: RMDP [16], an FEC-based reliable transfer protocol; and RLC [18] a congestion control algorithm for multicast communication, designed to share bandwidth evenly with competing TCP streams.

RMDP

RMDP uses FEC to achieve reliability in the face of network losses, and scalability in presence of large sets of receivers. The principle of operation of RMDP is extremely simple: the file to transmit is split into packets – say k of them – that are in turn passed to a suitable *FEC encoder* capable to produce a larger number of *encoded packets*. The latter have the property that reconstruction of the original packets is possible as soon as any k encoded packets are available at a receiver (see Figure 2). As a consequence, what matters in the end is that a receiver collects a sufficient number of packets to reconstruct the original file, but it does not matter which ones[3]. So, the problem of achieving reliable data delivery changes from *qualitative* (we need *exactly* those packets) to *quantitative* (we need *any* k different packets). We characterize this feature by saying that encoded packets are *anonymous*.

When using FEC, depending on the application and the type of network, the sender can transmit encoded packets either on demand, or unconditionally, or with a mixed approach. In RMDP we use a mixed approach, by transmitting a moderate amount of redundant packets initially, and then adding more and more repairs as requests from receivers arrive.

The use of FEC gives both robustness to losses and scalability. The former is achieved when the sender uses some amount of unconditional redundancy in transmitting data: in this case, a moderate amount of lost packets can still be tolerated without the need for explicit feedback or retransmissions. Scalability comes from the fact that the any packet is good for all receivers for reconstruction purposes. Had we used a conventional recovery mechanism based on explicit retransmissions of specific packets, each retransmitted packet would have been useful only to a subset of receivers (those missing that specific one). The bandwidth savings deriving from the use of FEC-based repairs can be significant, especially in the case of uncorrelated losses [16].

[3] In practice, there are some practical constraints on the values used for k (the *FEC block size*) which slightly reduce the efficiency of the protocol for large files. More details on this issue are discussed in [14].

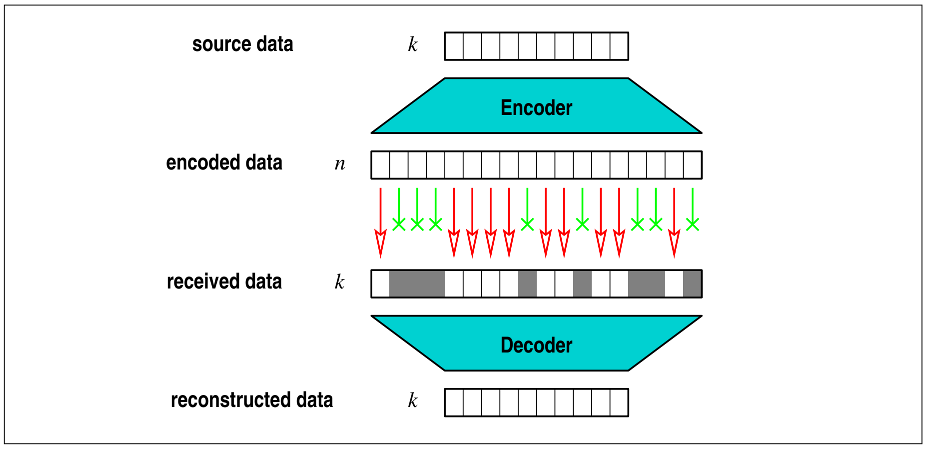

Fig. 2. The principle of operation of FEC encoding/decoding.

The basic version of RMDP as documented in [16] does not use congestion control; it can be used to serve receivers with different bandwidths to the source only thanks to its loss-tolerant behaviour, and can be made relatively network friendly by choosing a suitably low transmit rate. However, when designing large-scale applications, congestion control is a primary goal. To this purpose, we extended RMDP with the RLC congestion control algorithm.

RLC

RLC [18] implements congestion control by exploiting the ability of multicast routers to quickly start/stop forwarding a multicast group depending on the requests coming from the receivers. RLC is based on the assumption that a data stream can be partitioned in a number of component (*layers*) which can be used in a hierarchical fashion. The assumption holds for video and audio streams, where each set of layers can be mapped to a different quality.

RLC works by transmitting each layer over a different multicast group. Receivers select which groups to receive from (we call this the *subscription level*), hence choosing their total receive bandwidth, depending on the experienced packet losses. If multicast groups are used in a hierarchical fashion, and receivers behind the same bottleneck are sufficiently synchronized among themselves, then the set of receivers can effectively modulate the traffic flowing through the bottleneck link in a TCP-friendly way. Figure 3 shows an example of this process, with different receivers using different subscription levels. The total traffic on each link is determined by the maximum subscription level of the receivers downstream, which in turns dynamically adapts to the current network conditions.

The response of RLC to congestion signals (i.e. packet losses) mimics the "simple" TCP equilibrium equation (see e.g. [10]):

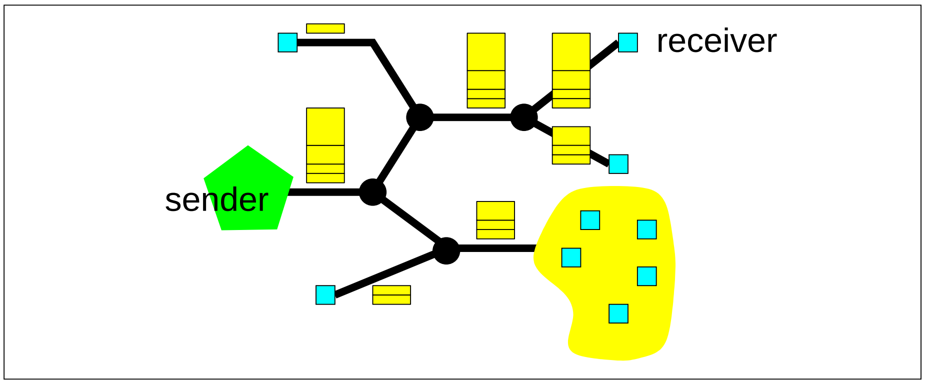

Fig. 3. RLC operation in a network with different paths.

$$\text{throughput} = \frac{cs}{\text{RTT}\sqrt{\text{loss}}}$$

where c is approximately constant, s is the packet size, and RTT is a scaling parameter which resembles the effect of the round trip-time on TCP connections.

The idea of a Layered Multicast scheme, which RLC is based upon, has been initially suggested in RLM [13] as a method for multicasting video at different quality levels. However, in order to work in practice, a Layered Multicast scheme requires that receivers behind the same bottleneck be synchronized in their changes of subscription level; otherwise, the independent attempts would result in persistent congestion. RLC solves the problem very simply by imposing that an increase of the subscription level can only occur upon reception of specially-marked packets. This effectively synchronizes receivers without the need for explicit communication among them. Furthermore, the sender periodically sends a short burst of traffic on each group to probe the network without requiring the receiver to increase the subscription level and suffer the (relatively) long-lasting effects in terms of congestion.

RLC-RMDP

RLC can be used for reliable transfer using one feature of the FEC encoding – namely, the fact that packets are anonymous. In fact, the receiver can make use of received packets irrespective of the layer they come from, and this is what makes it possible for receivers to move up and down in the number of layers they subscribe to, adapting to the available bandwidth of the network. Figure 4 shows this process for a set of receivers which start together, but on different network paths. Receivers climb up the bandwidth curve up to the point where R3 first, and R2 later, experience losses, at which points they either stop increasing the subscription level or even return to the previous (lower) one.

R1 does not see any loss, and thus can terminate reception earlier using the maximum sender's bandwidth. R2 needs a longer time to complete. R3 is on a path which is temporarily subject to more traffic, so it will settle to a lower bandwidth until the time when the network shows additional capacity. At this point, the receiver will increase its subscription level again. The area under all receive profiles is approximately the same in all cases.

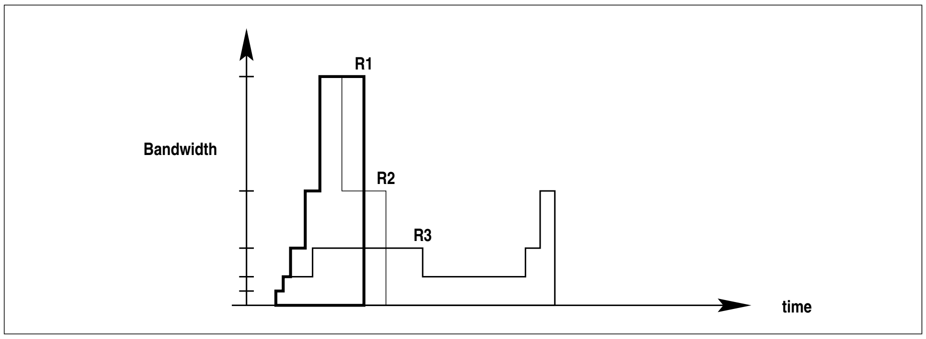

Fig. 4. The receive bandwidth profiles for receivers on different paths.

2.3 The command protocol

User commands in `mgpm` correspond to events recorded through simple input devices such as keyboard, mice, pens or remote controls. Such commands constitute a very low rate stream, with only rare and short bursts peaking at hundreds of bits per second or even less. In `mgpm`, we enforce the low bandwidth requirements by explicitly denying the generation of high-bandwidth data in the command stream: only those slides/images present in the presentation file can be displayed, together with annotations. This is not a severe limitation because it is unlikely that the user has a chance to produce complex data while giving a talk. If necessary, such data (e.g. a live video, the output of a program running at the sender's site, etc.) can be displayed using other, more appropriate tools.

The command stream has two requirements: real time transfer (because input should be loosely synchronized with other streams such as audio and video), and (semi)-reliable delivery, because a truncated data stream could result in displaying the wrong slide or drawing only part of an annotation.

Both requirements are not terribly strong: a short delay (a few seconds) and jitter in the data delivery are perfectly tolerable, considering that page changes and annotations are persistent (unlike audio or video) and the presence of other information (e.g. a parallel audio stream, or the content of the current/previous slide) can fill up the gap. As for the reliability requirements, we can make use of the low bandwidth requirements in two ways:

– we apply the *Application Level Framing* principle [1] by making each packet processable independently of others. We achieve this by making sure that commands (e.g. a single drawing action) are not split across packets, and additionally by sending in each packet important status information such as the indication of the current page, pen color, position and font, etc.. This allows the receiver to process any packet irrespective of the presence or absence of previous ones – missing packets may cause incomplete but never wrong actions.
– We send redundant data over the control stream. Because user commands are small and generated at low speed, at most a few tens of bytes per second are generated even when the user is extremely active. By sending a packet every few seconds with the most recent commands, we make sure that each command is retransmitted several times thus overcoming even severe packet losses. This is similar to the approach used in `nte` [5].
  A low bandwidth, background task is used to retransmit commands that do not fit into the last packet, so that long network outages can be recovered.

The above techniques effectively remove the need for receivers' feedback. Even with the use of redundancy, the overall traffic generated by the sender remains within a few hundreds of bits per second – a negligible amount of traffic even over low speed connections.

When using an unidirectional protocol like this, there is a tradeoff between the need of quickly recovering from losses (which the sender does not know of) and keeping network usage low. While user I/O activity generates traffic, the data rate is so low (because of input device limitations such as keyboard, mice and tablets) that it is not even necessary to artificially enforce bandwidth limitations on the output stream. When the user is idle, replicas of the last set of commands are sent with an exponentially decaying rate (similar to the one used by TCP to handle retransmission timeouts), settling at a "keepalive" rate of a few packets per minute. This way each command is retransmitted several times (until it slides out of the current window) and over a large interval of time, so that short congestion situations can be easily recovered from.

3 Implementation and performance

The current implementation of `mgpm` is based on the publicly available implementations of `mgp` and `RLC-RMDP`, and is distributed in source format from the author. The code compiles cleanly on FreeBSD and other Unix platforms, and we also have a Windows port using an X11 compatibility library (i.e. the Windows version of mgpm does not require and X11 Server running on the host).

`mgpm` preserves most features of the standalone tool (MagicPoint), including the ability to control the presentation (e.g. annotate it, or switch to a different page) at the receiver's site. Figure 5 shows a screen dump of the sender (left) and one receiver who made some personal annotation on the current slide. Because the full presentation is available locally, each receiver can independently

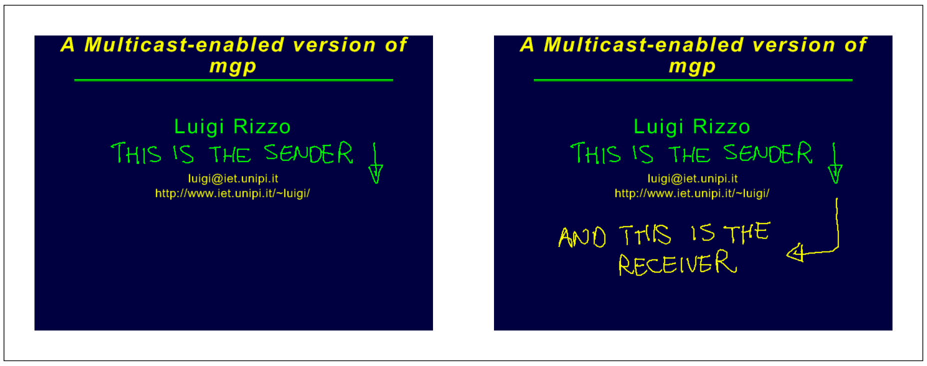

Fig. 5. A screen dump of a sender and a receiver.

browse through the slides, something that is generally not possible with a generic videoconferencing tool.

Starting the tool is extremely easy. On the sender side, a typical command line is something like

`mgpm -M` *group/port/ttl/rate file*

which transmits both the command stream and RLC-RMDP layers on a set of multicast groups starting with the specified one. A modified syntax is available to control separately the generation of the command stream and bulk data – e.g. in some cases the background transmission of the slides might be done by a third party, or not necessary because receivers are expected to fetch the presentation by other means.

On the receive side, the tool can be started by hand (or using an `sdr` plugin) as

`mgpm rmdp://group:port/`

where the URL-like syntax specifies the address to be used to acquire the command stream and the presentation file. Also in this case, more command line options are present to specify separately the source of commands and the location of the presentation file – the latter can also be acquired via `http` or from a local file. Rendering features (e.g. screen size, color depth) are independent at the sender and the receivers – all commands are scaled and transformed according to the settings of the receiver.

3.1 RLC-RMDP issues

In order to use RLC-RMDP with `mgpm` we applied two minor but significant changes: RTP encapsulation of data packets, and removal of feedback from the receivers.

The first modification allows us to record and replay the data stream as well as the command stream with common media tools as discussed in Section 3.3. The second one permits the use of the protocol over fully unidirectional networks.

As a result of the lack of feedback, the sender transmits continuously throughout the presentation, and each receiver simply leaves multicast groups when the presentation has been completely downloaded. From the network point of view (assuming a properly working pruning in the multicast routers) this is equivalent to turning off the sender when reception is complete. There is some overhead in the sender who has to encode and transmit unnecessary packets (blocked at the local interface if there are no receivers, anyways), but because of the low rate in use this overhead is not excessive. The bandwidths used for bulk transfer are configurable from the command line, and by default range from 8 to 128 Kbit/s, comparable to those used for low quality video streams. We remark that this stream only needs to be received during the file download. We have chosen the lower rate for the bulk transfer so that a typical modem connection can easily carry it together with GSM-compressed voice (e.g. a `rat/vat` session [6, 9]), and still leave a fraction of usable bandwidth.

It is worthwhile to remark that, in order for RLC to work as desired, the multicast infrastructure must correctly implement pruning [2]. Some performance improvements can also come if the multicast routers implement the fast leave mechanism described in [15]. Some multicast routers or hosts do not implement the pruning mechanism. In such cases, RLC will not work properly because groups will never be dropped if not after a few minutes. When this happens, and assuming that there is not sufficient bandwidth for all layers, a receiver will experiment persistent congestion and fall back to the use of just the lowest layer. This does not prevent the completion of the reception, but certainly reduces the effectiveness of use of the protocol.

3.2 The command stream

The protocol used to transmit the command stream follows the guidelines of Section 2.3. There is no explicit enforcement on the rate used by the command stream as this is inherently limited by the input peripherals. Further, reception of the command stream is generally mutually exclusive with the bulk transfer stream.

`mgpm`, as well as `wb`, does not implement a "telepointer", i.e. remote control of the receiver's cursor. This is because controlling a telepointer is much more challenging in terms of frequency of updates (to track the movements of the pointer), delay and synchronization with other media. In particular, reproducing gestures like circling around a location on the screen, or "underlining" a word, require a fine-grained timestamping of pointer movements (currently we use a very coarse timestamp mechanism, using the timestamp in the RTP header for the whole packet) and a much more timely delivery of data, which would be badly affected by network delays and jitter. While it is well known how to deal with delay and jitter e.g. in audio tools, the command stream does not need jitter compensation and as a consequence we do not implement such mechanisms. Furthermore, the implementation of a telepointer would result in an increased network load (at least a few Kbits/s, considering that typical mice/tablets communicate at 1200..9600 bps and we have to add the overhead for timestamps and protocol headers) which we do not believe necessary, given that this feature can be effectively replaced by annotations.

3.3 Gateways and Media Recording

A useful feature of many Internet cooperative tools is the ability to tunnel sources through media gateways (e.g. to reach receivers without multicast connectivity), or to record sources for later replay. `mgpm` fully supports this feature because all multicast traffic (both command stream and the RLC-RMDP traffic) it generates is encapsulated into RTP packets. Thus, standard RTP tools can be used for tunneling and recording.

It is particularly interesting that the playback of a recorded RLC-RMDP session will still make full use of the RLC congestion control scheme. In fact, the sender-side of RLC essentially only specifies a schedule for packet transmissions, and this information is completely captured in the RTP timestamps and automatically reproduced by the media players. The timestamps in RTP headers will also let the command stream to be replayed synchronously with other media such as audio.

3.4 Performance

`mgpm` is mostly built by integrating components developed by the author in previous works, so we can borrow most performance analysis results from these components.

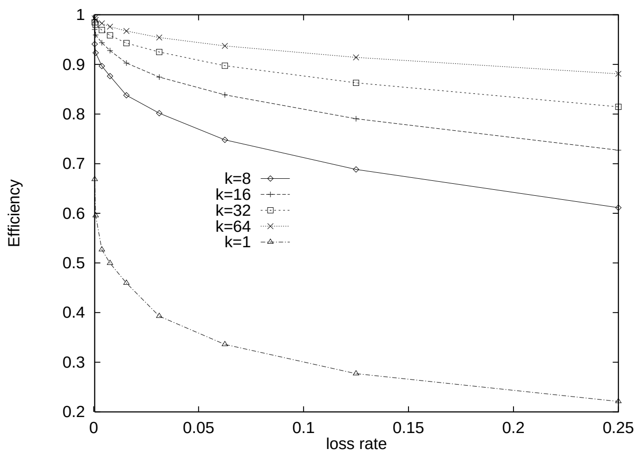

Fig. 6. The encoding (in)efficiency due to interleaving with different block sizes for a file of 2048 packets and uniform random losses. Efficiency is much higher with bursty losses such as the ones experienced with RLC. Also, the use of a congestion-controlled protocol makes the receiver operate in an area with very low losses.

The basic version of the FEC-based bulk communication protocol is analysed in [16], which discusses the main performance indexes, namely coding overhead and coding efficiency. The behaviour of RLC congestion control algorithm is reported in [18]. We briefly summarize the results here.

- **coding overhead**.
  FEC-based protocol require some work to produce the encoded data to be transmitted, and a similar amount of work at the receiver for the decoding process. The encoder we have used has a coding/decoding speed which can be measured as c/k where k is the FEC blocksize, and c is a constant which depends on the CPU and memory speed of the system doing the encoding, and ranging around 60 MBytes/s at the time of this writing for entry-level PC hardware, and in any case above 10 MBytes/s even for old Pentium-class system.
  In mgpm we used a FEC blocksize $k = 16$. This means that we can do the coding/decoding at above 5 Mbit/s on any system of interest, which is about 40 times higher than the peak data rate we used (as a consequence, only a small fraction of the CPU is used for the actual encoding process).
- **coding efficiency**.
  Ideally, using a single FEC block that spans the entire file, all packets received are useful for reconstruction. Because on deterministic encoders the encoding costs increase with the blocksize, a common approach is to split the file in blocks of fixed size and send the packets from each block interleaved. By doing this, and without using a selective retransmission scheme, the decoding performance may suffer because we might receive packets for already completed blocks while accumulating the k packets for the block which suffered the most losses.
  The graphs in Fig 6 show the phenomenon for different block sizes and uniformly distributed losses, for a file of 2048 packets. The efficiency decreases as the file size and loss rate increases, but the phenomenton is much less relevant for bursty losses (which are comparable to the effect of moving to a lower subscription level in RLC-RMDP) as most blocks then will miss a similar number of packets.

The above considerations of course only apply to the case when RLC-RMDP is used to transfer data. When the presentation file is transferred using HTTP, we do not suffer any overhead.

Concering the real-time protocol, most common actions are encoded with a handful of bytes (e.g. 3 bytes for skipping to a new page, 2 or 4 bytes per coordinate when drawing), so bandwidth usage is not really a concern.

4 Conclusions and future work

We have presented the architecture of mgpm, a network enabled presentation tool designed to deliver high-quality slide-based presentations using a one-to-many communication model. In its current implementation `mgpm` supports the functionalities which are peculiar to a presentation tool, namely a user-interface, a bulk transfer protocol for transferring the presentation data, and a real-time protocol used to transfer commands and annotations. The whole system has been designed to provide a high-quality service in a network-adaptive manner, and to be suitable for use also on highly asymmetric (or unidirectional) networks.

The tool has been designed and implemented as a research vehicle, mainly intended to study protocol and communication issues for this kind of applications. Despite that, the initial prototype has resulted in a quite usable tool, and our initial experience with it has been highly positive, also because of the ability to record a presentation with standard tools for archival or reuse purposes.

The communication aspects of the prototype are also fully satisfactory, in terms of performance, scalability and flexibility. We recognize that `mgpm` would benefit from a better integration with audio and video tools, and support for bidirectional interaction between the speaker and the receiver (together with a floor control mechanism). Future work will involve investigation in these areas.

Acknowledgements

This work has been partly supported by the Ministero dell' Università e della Ricerca Scientifica e Tecnologica (MURST) in the framework of the Project "Design Methodologies and Tools of High performance Systems for Distributed Applications".

We would like to thank Lorenzo Vicisano and Jon Crowcroft for their work on RLC-RMDP, and Davide Devoti for his help in the implementation of `mgpm`.

References

1. D.D.Clark, D.L.Tennenhouse, "Architectural considerations for a new generation of protocols", in Proc. of SIGCOMM'90, Philadelphia, PA, Sept.1990, ACM.
2. W.Fenner, RFC2236, Internet Group Management Protocol, Version 2, Nov. 1997
3. R.Frederich, "Network Video (nv)", Xerox Palo Alto Research Center, software on-line
(`ftp://ftp.parc.xerox.com/net-research/`)
4. J.Gemmel, E.Schooler, R.Kermode, "An Architecture for Multicast Telepresentations", Journal of Computing and Information Technology (to appear).
See `http://www.research.microsoft.com/barc/JGemmell/`
5. M.Handley, J.Crowcroft, "Network Text Editor (NTE): A scalable shared text editor for the MBone", Proc. of ACM SIGCOMM'97, Cannes, France 1997
6. V.Hardman, I.Kouvelas, M.A.Sasse, A.Watson: "A packet loss Robust Audio Tool for use over the Mbone", Research Note RN/96/8, Dept. of Computer Science, University College London, 1996.
7. J.Itoh, Y.Nisida, K.Yamamoto, "MagicPoint: a UNIX-based presentation tool", Internet Initiative Japan Research Report, available as
`ftp://ftp.itojun.org/pub/paper/itojun-freeenix99.txt`

8. S.Floyd, V.Jacobson, C.Liu, S.McCanne, L. Zhang, "A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing, Scalable Reliable Multicast (SRM)", ACM SIGCOMM 95, available as ftp://ftp.ee.lbl.gov/papers/srm_sigcomm.ps.Z
9. V.Jacobson, S.McCanne: "The LBL audio tool vat", Manual page (ftp://ftp.ee.lbl.gov/conferencing/vat/)
10. J.Madhavi, S.Floyd, "TCP-Friendly Unicast Rate-Based Flow Control", Tech.note to the e2e mailing list, 1997 (http://www.psc.edu/networking/papers/)
11. S.McCanne, "A distributed whiteboard for network conferencing", May 1992, UC Berkeley, CS268 Computer Networks term project.
12. S.McCanne, V.Jacobson, "vic: a flexible framework for packet video", in Proc. of ACM Multimedia'95, San Francisco, CA, Nov.1995, ACM, pp.511-522.
13. S. McCanne, V. Jacobson, and M. Vetterli, "Receiver-driven Layered Multicast", SIGCOMM'96, August 1996, Stanford, CA, pp.1-14.
14. L.Rizzo, "Effective erasure codes for reliable computer communication protocols", ACM Computer Communication Review, Vol.27, n.2, April 1997, pp.24-36.
15. L.Rizzo, "Fast Group Management in IGMP", Proc. of the 1998 Hipparch Workshop, London, June 1998
16. L.Rizzo, L.Vicisano, "RMDP: an FEC-based Reliable Multicast protocol for wireless environments", ACM Mobile Computing and Communications Review, Vol.2, n.2, April 1998
17. H. Schulzrinne, S.Casner, R.Frederich, V.Jacobson, "RTP: A Transport Protocol for Real-Time Applications", IETF AV Transport WG, Jan.1996, RFC-1889.
18. L.Vicisano, L.Rizzo, J.Crowcroft: "TCP-like congestion control for layered multicast data transfer", Infocom98, S.Francisco, March 1998