PACKET SCHEDULING FOR HETEROGENEOUS MULTICAST TRANSMISSIONS

Vincent Roca
Université Pierre et Marie Curie
LIP6 - CNRS, thème Réseaux et Performances
8, rue capitaine Scott; 75015 Paris; FRANCE
phone: (+33) 1.44.27.75.14; fax: (+33) 1.44.27.87.32
vincent.roca@lip6.fr; http://www-rp.lip6.fr/~roca

Abstract

Multicast transmissions naturally raise the problem of the heterogeneity of receivers in terms of networking possibilities, host performances and user desires. Several solutions have been introduced but none suits all the situations. This paper describes a scheduling algorithm that unifies some of these solutions, in particular in case of continuous data flows, and which is suited to a broad range of applications. An implementation of this algorithm is used to illustrate its behavior.

1 Introduction: The Needs

Multicast transmissions naturally raise the problem of the heterogeneity of receivers. Several levels of heterogeneity can be identified: (1) the intrinsic performance of various parts of the network can be largely different; (2) the network performance can vary over the time because of congestion problems, of external factors (e.g. weather conditions in case of wireless communications), or a change of communication technology (e.g. switching from WaveLAN to GSM); (3) some privileged networks can take advantage of improved transmission services (e.g. using differentiated services); while other ones will be at the mercy of router congestions. (4) some hosts can be limited by their processing power (e.g. palmtop); and (5) some users can be content with a medium quality data flow, while other ones will insist to have the highest possible quality.

We believe that a large group will include several (all?) kinds of heterogeneity. Traditionally one restricts oneself to homogeneous multicast transmissions, sending data at a rate determined by the slowest receiver.

It implies defining a maximum rate either statically (compilation or application parameter) or dynamically [8]. Our proposal intends to fill this gap by providing a general purpose heterogeneous multicast scheme, i.e. the possibility to send *any data flow* to a group where each receiver chooses a reception rate according to its network quality, workstation power, and desires.

This paper is organized as follows: section 2 introduces several approaches to support heterogeneous multicast transmissions. Our proposal is detailed in section 3. Section 4 introduces some experimental results. Section 5 discusses additional properties, then we conclude.

2 Limitations of Current Solutions

Several directions have been proposed to address heterogeneity. Some of them rely on packet scheduling schemes (DSS/MMG/MCM), others on coding (e.g. the hierarchical coding of a video stream [5]), on transcoding (the information is tailored, e.g. an image definition is reduced, according to the needs of a particular client [9]), or on filtering [11] [19]. This section focuses on the scheduling schemes.

Destination Set Splitting (DSS)

This solution, also called Simulcast, achieves heterogeneity by using multiple uncoordinated streams at different rates on several multicast groups [1][2]. A receiver chooses the desired throughput and subscribes to the associated group. Here each receiver gets a copy of all the data.

This solution has several limitations:

- because data is sent on all the groups, it leads to a high load of the sender's network where traffic is concentrated.
- a receiver that wants to change of group (e.g. if the selected throughput turns out to be too high) may loose many packets if it unsubscribes to the old group and then subscribes to the new one. If it does the contrary (i.e. subscribing before unsubscribing), the presence of two data flows in the meanwhile can also lead to losses.

But this solution avoids any packet duplication at the receiver side (we'll see that it is not the case of our GMMG scheduling), and can be used with data flows generated on-the-fly.

Multiple Multicast Groups (MMG)

The MMG [3] scheme is based on a cumulative layered scheduling. The idea is to let each multicast group take advantage of the previous groups. Here also everyone receives everything at a pace that depends on its possibilities.

Let us consider a sender S that uses three multicast groups (figure 1). Data is first divided into four equal parts. On group 1, S sends parts 1 to 4 at a rate compatible with that of the slowest receiver. On group 2 S sends parts 2 and 4, which enables a receiver having subscribed to both groups to receive data twice as fast, etc.

```
G1:   part_1   part_3   part_2   part_4
G2:   part_2   part_4
G3:   part_3
      part_4
```

Figure 1: Scheduling in case of the MMG approach.

We see several limitations to the MMG approach as proposed in [3]:

- the sender is supposed to know in advance the amount of data to send in order to partition it. This hypothesis is rather restrictive.
- the sender is supposed to know in advance the available throughput of each receiver in order to define the features of each group. This is usually impossible since group membership is by nature dynamic and the network features highly variable.
- our personal experience implementing our proposal suggests that the packet ordering within each group should be kept increasing.

Multiple-Channel Multicast (MCM)

This class of solutions is also based on a cumulative layered scheduling. It includes the Partition Organization channel scheduling (PO) [7] and the Session Organization (SO) channel scheduling [16]. Like MMG, these solutions require that the data stream is known in advance (in order to partition it) which restricts their use. On the other hand, an advantage of this hypothesis is that asynchronous starts are possible since packets are cyclically transmitted on each multicast group.

Discussion

The previous analysis has highlighted that many hypotheses on the data flow nature are made. For instance hierarchical coding is restricted to data flows that can be organized in cumulative layers. It typically concerns video streams. The MMG and MCM schemes require to know in advance the amount of data to send in order to find the optimal scheduling. A typical example is a file transfer application.

Unfortunately a broad range of applications that generate *on-the-fly data flows* are not satisfied. A white board (e.g. `wb` [18]) or an application sharing environment (e.g. doing X11 message multiplexing like `XTV` [20]) are two typical examples. The amount of data exchanged can be rather low during a certain span of time and then, because of a participant action (e.g. he launches a new X Windows application under `XTV`'s control), the tool generates "on the fly" large amounts of data. If the DSS approach can handle these applications, it has limitations as explained before.

3 The GMMG (Generalized MMG) Approach

This section introduces a hybrid solution that mixes some of the previous approaches. To be as general as possible, we consider that data is submitted by the application as a *collection of messages, called ADU* (Application Data Units) to the sending control layer. We make no hypothesis on the nature of the ADU and whether or not the application follows the ALF paradigm [4]. Unlike MGM, we do not consider the possibility of receivers starting reception asynchronously. This feature is handled at user level with the kind of applications considered.

The Case of an Isolated ADU

We first consider an isolated ADU. GMMG is then *very close to MMG*. The ADU of figure 2 is first segmented into sixteen 512 byte packets (i.e. small enough to avoid IP fragmentation). Each ADU is identified by its constituting packets, noted as a range of packet sequence numbers (e.g. [1; 16] identifies the first ADU which is segmented in 16 packets). A little vertical arrow indicates that the application submits a new ADU to the control layer. In figure 2 the sender S uses four transmission layers and schedules packets in a cumulative way, so that a receiver subscribing to an additional layer experiences a reception rate two times higher[1]. Therefore on layer i we transmit the packets of all the previous layers delimited by the upper half of the packets sent on layer i-1 (dashed areas of figure 2).

```
 [1;16]
L1    1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16
L2    9  10  11  12  13  14  15  16
L3    5   7  13  15
      6   8  14  16
L4    3  13
      4  14
     11  15
     12  16
```

Figure 2: GMMG scheduling of an isolated ADU.

Because transmissions on each layer occur at a regular rate determined by layer 1, we use this rate as a virtual time scale. As we neglect the transmission and propagation times, it is regarded either as the sending or receiving time scale. At time 1, S sends successively packet 1 on layer 1, packet 9 on layer 2, packets 5 and 6 on layer 3, etc. Then it waits. At time 2, S sends all the packets scheduled for time 2 and so on.

Several evaluation criteria are considered:

total reception time: Time required to receive all the packets of all the ADUs, expressed in terms of layer 1 packet times ("virtual time"). For instance, in figure 2, subscribing to layers 1 and 2 allows the reception of all packets in 8 virtual time slots.

spreading of the ADU reception times: The regular arrival of ADUs (i.e. the reception of all the packets of each ADU) enables the receiver to process them regularly, as soon as possible, and limits buffer use. This spreading is represented by the list of reception times of each ADU.

total number of packets transmitted: Number of packets transmitted over all layers subscribed. In spite of the cumulative aspect of transmissions, certain packets are transmitted several times. Obviously, the fewer, the better. This metric is also expressed as the ratio of the number of packets sent for a given number of layers to the number of packets sent on layer 1 (i.e. non-duplicated packets).

[1] For simplicity we only consider ratios of 2. This is not compulsory as explained in section 5

cumulative throughput ratio: The total transmission rate on the sender's network increases with the number of transmission layers. The tables give the ratio of the maximum throughput when subscribing a given number of layers to the throughput of layer 1.

number of multicast groups: using a multicast group is costly, so the fewer the better. So far we have supposed that each layer corresponds to a multicast group. Section 5 discusses this assumption.

Table 1: Evaluation of the example of figure 2.

layers	end	spreading	nb of pkts/ratio	throughput ratio
L1 only	16	16	16 / 1.0 ratio	1.0 ratio
L1+2	8	8	24 / 1.5 ratio	2.0 ratio
L1+2+3	4	4	32 / 2.0 ratio	4.0 ratio
L1+2+3+4	2	2	40 / 2.5 ratio	8.0 ratio

Table 1 applies these criteria to figure 2. Compared to layer 1, the presence of layers 2 and 3 reduces by a factor 4 the end of reception, at the expense of a maximum total throughput 4 times higher, but with only twice as many packets sent. With layer 4, data is received 8 times faster with only 2.5 as many packets sent.

The Case of Close ADU Submissions

We now consider the case of close ADU submissions. The transmission control layer does not know in advance the amount of data to send. Therefore, it must continuously decide when to send each packet, and whether or not to reconsider the planned transmissions when a new ADU arrives. Three strategies are possible: the first one consists in sending ADUs independently to one another. Another one, more drastic, consists in reconsidering the whole transmission planning each time a new ADU is received. None of these solutions is recommended as explained in annex A. Therefore we focus on the third intermediate solution.

Description of the GMMG algorithm

This solution takes into account a new ADU as soon as it is available without modifying the transmissions already planned. When the application submits the new ADU, some packets from previous ADUs (possibly 0) wait to be sent on each level. Let call them rem_1, rem_2 and rem_3. We have: $rem_1 >= rem_2 >= rem_3 >= 0$. Let $part_1$, $part_2$ and $part_3$

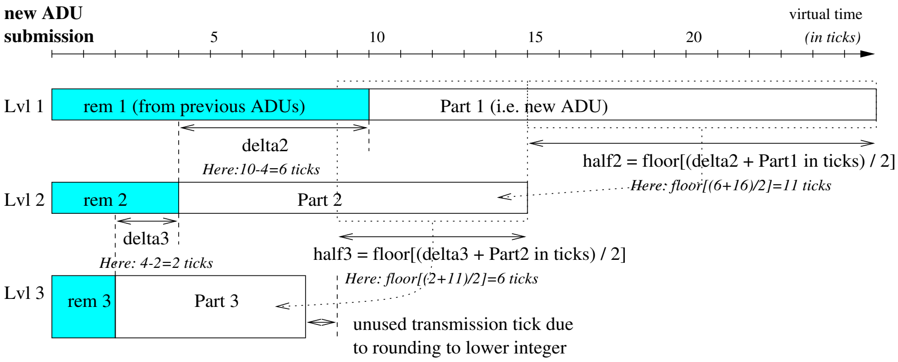

Figure 3: GMMG scheduling algorithm.

be the various packets belonging to the new ADU sent respectively on layer 1, 2 and 3. By definition, $part_1$ includes all the packets of the new ADU. $part_2$ is built as follows. We first calculate:

$$delta_2(in\ ticks) = tick_2(rem_2) - tick_1(rem_1)$$

where the $tick_i()$ function gives the number of ticks required to transmit the given number of packets on layer i[2]. $part_2$ includes all the packets of $part_1$ contained in the upper half, of duration $half_2$ ticks, of the set $delta_2 \cup part_1$. Therefore[3]:

$$half_2(in\ ticks) = floor[(delta_2 + tick_1(part_1))/2]$$

$$part_2 = \{upper\ half_2\ packets\ of\ part_1\}$$

Likewise $part_3$ includes the packets of $part_1 \cup part_2$ delimited by the upper half, of duration $half_3$, of the set $delta_3 \cup part_2$ (area within the dashed box of figure 3). $part_3$ is bounded by the times $start_3$ and end_3:

$$delta_3(in\ ticks) = tick_3(rem_3) - tick_2(rem_2)$$

$$half_3(in\ ticks) = floor[(delta_3 + tick_2(part_2))/2]$$

$$end_3\ (in\ ticks) = tick_2(rem_2 \cup part_2)$$

$$start_3\ (in\ ticks) = end_3 - half_3$$

$$part_3 = \{pkts\ of\ new\ ADU\ sent\ between\ start_3\ and\ end_3\ on\ L1 \cup L2\}$$

[2] $tick_i(l\ bytes) = l/number\ of\ bytes\ sent\ per\ tick\ on\ layer\ i$

[3] The $floor()$ function returns the largest integral value not greater than x

This algorithm can be generalized. Figure 4 illustrates it with a simple scenario: two ADUs of six packets each are submitted at times 0.5 and 2.5. At time 2.5 we have: $delta_2 = 4 - 1 = 3$ ticks. So the upper $half_2 = floor((3+6)/2) = 4$ packets of ADU 2 are sent on layer 2.

```
   [1;6]      [7;12]
     |          |
L1  V 1    2  V 3    4    5    6 |  7    8    9   10   11   12
L2     4    5    6 |  9   10   11   12        |
```

Figure 4: GMMG scheduling applied to a simple scenario.

Table 2: Evaluation of the example of figure 4.

layers	end	spreading	nb of pkts/ratio	throughput ratio
L1 only	12	6, 12	12 / 1.0 ratio	1.0 ratio
L1+2	8	3, 8	19 / 1.6 ratio	2.0 ratio

Compared to the two other solutions of annex A, this solution (1) improves the spreading of the end of ADU reception, and (2) reduces the total reception time for L1+L2. This is made possible by the precise knowledge of remaining packets on the various layers.

A Necessary Improvement: the Notion of Credit

This algorithm has limitations. Suppose we are building $part_i$. On the previous levels (i.e. 1 to $i-1$), between $start_i$ and end_i there can be packets that do not belong to the new ADU. These packets are ignored and the corresponding number of transmission slots of $part_i$ are lost. This is the *"lost cycle phenomenon"*. It happens in figure 3 when building $part_3$. A transmission slot is lost because the first packet of layer 1 within the dashed area (time 9) belongs to rem_1, not to $Part_1$.

If ADUs are submitted too rapidly, packets progressively accumulate on the lowest levels and "lost cycles" are more and more numerous. Higher layers are seriously affected and may even become useless. To avoid it we consider that each lost cycle creates *"credit"*. Let tot_credit be the sum of these credits for all layers. We calculate the possible anticipation:

$$anticipation\ (in\_ticks) = floor(tick_i(tot\_credit)/2);$$

that we subtract to the $start_i$ variable:

$$start_i = start_i - anticipation$$

By extending the area considered when building $part_i$, we make up for the lost cycles. Note that the implementation of GMMG described in [15] completely avoids this problem.

One Step Further: the Case of Continuous Data Flows

In the previous section we considered close ADU submissions. We now go further and consider the case of continuous data flows. This is (partially) illustrated in figure 5.

```
[1;12][13;16][17;28]
     |   |   |
  L1 V 1 V 2 V 3   4   5   6   7   8   9  10  11  12| 13  14  15  16| 17  18 ...
  L2   7   8   9  10  11  12| 13  14  15  16| 20  21  22  23  25  26  27  28|
  L3   4   6  11| 13  15| 17  19  21  23  25  27|
       5  10  12| 14  16| 18  20  22  24  26  28|
  L4   3| 13| 17  21  25|
       9| 14| 18  22  26|
      11| 15| 19  23  27|
      12| 16| 20  24  28|
```

Figure 5: Continuous data flows (extract).

If the ADU submission rate is superior to the packet transmission rate of level 1, then $delta_i$ and $half_i$ increase. It means that $part_i$ tends to be identical to $part_1$. If the end of reception time of the ADUs is still improved, this is to the detriment of the total number of packets sent, and GMMG tends to DSS where data is sent on each layer. *Our approach is thus a balance between the MMG (sparse ADU submission) and the simulcast (continuous ADU submission) flow controls.*

Because data cannot indefinitely accumulate at the sender waiting to be transmitted on the lower layers, the application will eventually be blocked. Lower layers will take advantage of this pause to keep up with higher layers. Therefore, a limitation not solved by this approach is that the *transmission of truly continuous data flows, on the long range, take place at the rhythm of the lowest transmission level* because of limited buffer capabilities. A partial solution is introduced in section 5.

Defining Several Transmission Subflows

Up to now we have supposed that ADUs had to be sent to all the receivers. In fact an application may identify several subflows and leave each receiver choose what to receive (as with a hierarchical coding).

```
     [1;16] subflow 1
     [a;l]  subflow 2
       |
L1   V 1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16
L2     9   10   11   12   13   14   15   16
L3     a    b    c    d    e    f    g    h    i    j    k    l
L4     g    h    i    j    k    l
```

Figure 6: Scheduling of two ADUs belonging to two subflows.

In figure 6 we consider the subflow 1 sent to everybody, and the subflow 2 providing optional details. For each subflow we offer the possibility to receive data at different rates. A receiver now has two complementary levels of choice: (1) *what detail* he needs (i.e. how many subflows), and (2) *how fast* data should be received (i.e. how many layers for each subflow). Depending on the application some combinations can be meaningless, but this is an application decision that does not concern the transmission control layer.

4 Preliminary Simulation Results

We have implemented this algorithm as a C library and linked it to a traffic generator. This section illustrates the algorithm behavior with a data stream that we believe is significant of that of a white-board (for a detailed performance evaluation see [15]). There is no packet loss. The scenario consists of three bursts separated by inactivity periods:

step 1: send two ADUs of ten packets each ([1;10] and [11;20]), at one time tick interval (i.e. at times 1 and 2)

step 2: wait 10 ticks

step 3: send again two more ADUs ([21;30] and [31;40]), at one tick interval (i.e. at times 12 and 13)

step 4: wait 28 ticks (so that level 1 finishes to send all the packets left)

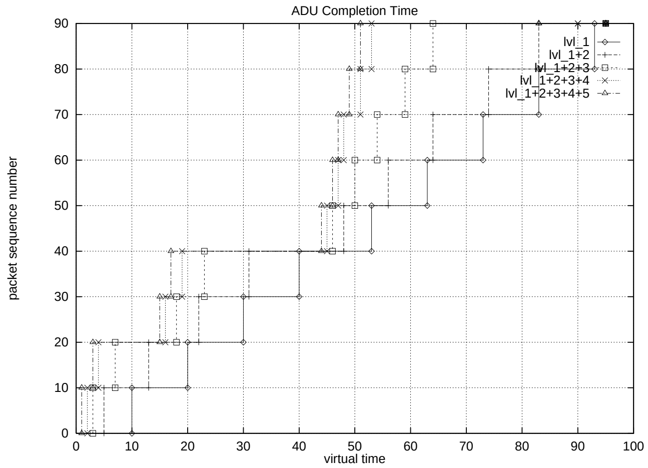

Figure 7: End of ADU reception while varying the number of layers.

step 5: send five ADUs ([41;50] to [81;90]) at one time tick interval (i.e. at times 41 to 46). Wait until the sender has received all of them, even if the ADU has already been completed[4].

Figure 7 shows the ADU reception times according to the number of layers subscribed, and figure 8 the packets sent on each layer. The horizontal steps in figure 7 at sequence numbers 20, 40 and 90 indicate that all the ADUs of steps 1, 3, and 5 respectively have been received. Subscribing to more layers leads to a faster reception of ADUs. Yet the gains are not always a power of two. This is due to the delay experienced on lower levels. For instance, during step 5, packet scheduling on layer 2 can only take advantage of layer 1 transmissions for the ADU [41; 50]. Then ADUs are received at the same rhythm no matter whether one has subscribed to layer 1 only or to both layers. This is different with layers 3 to 5. Their higher transmission rate is sufficient to process the packet burst of step 5 and each level takes advantage of transmissions on (some of) the previous levels.

The final duplication ratio is given in table 3. If this ratio is 3.6 when receiving all the layers, the five ADUs of step 5 have been received in 8 time ticks instead of 47, namely 5.8 times faster! These experiments show that our cumulative transmission organization can handle bursty

[4]It means that extra duplicated packets will be received. Another possibility would be for the receiver to close the socket once all the ADUs have been received. In that case the duplication ratio would be largely improved.

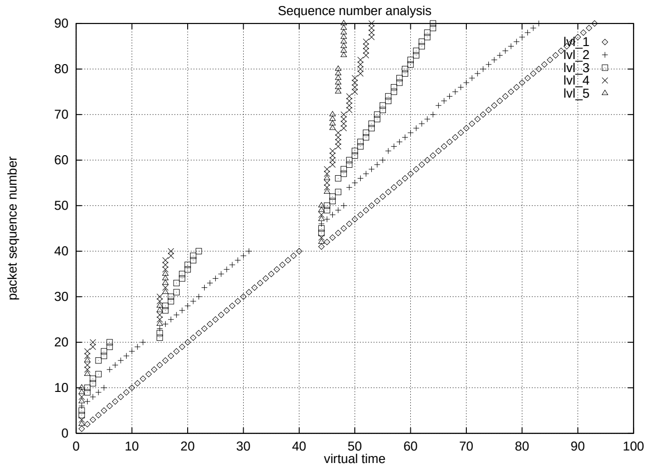

Figure 8: Packet sent on each layer when subscribing all of them.

Table 3: Duplication ratio while varying the number of levels.

Level 1	Levels 1+2	Levels 1+2+3	Levels 1+2+3+4	Levels 1+2+3+4+5
1.0	1.7	2.5	3.2	3.6

data flows efficiently while minimizing the number of packets sent.

5 Additional Aspects And Related Work

We now discuss some additional aspects related to the use of our scheduling algorithm. Many items (e.g. congestion control, reliability, how many layers to use, etc.) are not covered by this paper even if they are required.

Storing ADUs on the Disk to Limit the Continuous Data Flow Problem

The problem with (pseudo) continuous data flows is not so serious because our transmission scheme is implemented as a user-level library [15]. The process image (code+data) is naturally swappable and using a large transmission buffer will only result in a large area in the swapping partition. The sender can cope with "reasonably long" continuous data flows since data is in fact stored on the disk rather than in physical memory[5].

[5] A common rule of thumb is to define a swapping area 2 to 3 times larger than the available physical memory.

If not sufficient, it is possible to go further and to *explicitly store data on the disk* rather than in memory buffers. The scheduling module only keeps a small memory buffer which is used as a cache to store data read from the disk archive. Because transmissions are planned in advance, the disk access latency can be easily hidden. This technique enables very large transmission archives without competing with other processes.

This approach is more general than just its application to our scheduling algorithm. In particular it could be of benefits for reliable multicast transmissions. Data buffering is a problem because the sender must be able to cope with any delayed retransmission request, (even in case of a receiver-initiated approach [13]) which requires to keep data in memory for long (infinite?) periods.

Non Bijective "Transmission Level" to "Multicast Group" Mapping

GMMG (like other layered schemes) uses several transmission levels (or levels), and for simplicity we considered so far that they were mapped to several multicast groups. But several problems arise:

- resources are kept by multicast routers for each group,
- DVMRP [14] leads to periodical flooding and pruning stages for each group, and finally
- several multicast addresses must be allocated and announced.

A first way to limit the problems consists in mapping several transmission levels on a multicast group. For instance group 1 can be used for level 1 which provides an elementary transmission service. Group 2 can be used both for levels 2 and 3 for an intermediate transmission service. Finally group 3 can be used for levels 4 and 5, providing a high throughput service.

Another approach to provide a large range of throughputs with a limited number of multicast groups is to define a throughput scale that is not necessarily a power of 2 [3]. This is also possible with GMMG using the efficient implementation described in [15]. This latter accommodates *any scale of throughputs* since the number of packets sent on each level at each time tick can be freely set.

6 Conclusions

We have introduced a new scheduling algorithm (GMMG) for heterogeneous multicast transmissions unifying the MMG and DSS approaches. This algorithm can be used with a broad range of applications like CSCW ("Computer Supported Collaborative Work") tools that generate on-the-fly data flows. It has also been implemented in a general purpose library which is freely distributed (with source code) in the author's home page. A companion paper [15] discusses the design of this library and compares various scheduling algorithms including GMMG.

Acknowledgments

The author thanks S. Fdida, his colleagues from LIP6 and the anonymous reviewers who contributed to improve the quality of this paper.

References

[1] M. Ammar, L. Wu, "Improving the Performance of Point to Multi-Point ARQ Protocols through Destination Set Splitting", *Proceedings of IEEE INFOCOM'92, May 1992.*

[2] M. Ammar, D. Towsley, "Flow control for multicast applications", *Multicast tutorial during SIGCOMM'97, September 1997.*

[3] S. Bhattacharyya, J. Kurose, D. Towsley, R. Nagarajan, "Efficient multicast flow control using multiple multicast groups", *Proceedings of IEEE INFOCOM'98, February 1998.*

[4] D. Clark, D. Tennenhouse, "Architectural considerations for a new generation of protocols", *Proc. ACM SIGCOMM'90, September 1990.*

[5] J. Bolot, T. Turletti, "Scalable feedback control for multicast video distribution in the Internet", *Proceedings SIGCOMM'94, September 1994.*

[6] C. Diot, W. Dabbous, J. Crowcroft, "Multipoint communication: a survey of protocols, functions, and mechanisms", *IEEE Journal on Selected Areas in Communications, Vol. 15, No. 3, April 1997.*

[7] M. Donahoo, M. Ammar, E. Zegura, "Multiple-Channel Multicast scheduling for scalable bulk-data transport", *Proceedings of IEEE INFOCOM'99, March 1999.*

[8] S. Floyd, V. Jacobson, S. McCanne, C. Liu, L. Zhang, "A reliable multicast framework for light-weight sessions and application level framing", *Proceedings ACM SIGCOMM'95, 1995.*

[9] A. Fox, S. Gribble, Y. Chawathe, E. Brewer, "Adapting to network and client variation using infrastructural proxies: lessons and perspective", *IEEE Personal Communications, Vol. 5 No. 4, August 1998.*

[10] B. Levine, JJ. Garcia-Luna-Aceves, "Improving Internet multicast with routing labels", *Proceeding of IEEE ICNP'97, October 1997.*

[11] M. Luby, L. Vicisano, T. Speakman, "Heterogeneous multicast congestion control based on router packet filtering", *Work in progress, presented at RMRG meeting, June 1999.*

[12] S. McCanne, V. Jacobson, M. Vetterli, "Receiver-driven layered multicast", *Proceedings ACM SIGCOMM'96, October 1996.*

[13] S. Pingali, D. Towsley, J.F. Kurose, "A comparison of sender-initiated and receiver-initiated reliable multicast protocol", *IEEE Journal on Selected Areas in Communications, April 1997.*

[14] T. Pusateri, "Distance Vector Multicast Routing Protocol", *Internet Draft <draft-ietf-idmr-dvmrp-v3-06.txt>, March 1998.*

[15] V. Roca, "Design of a library for the heterogeneous multicast distribution of data flows generated on the fly", *Submitted, July 1999.*

[16] L. Vicisano, "Notes on a cumulative layered organisation of data packets across multiple streams with different rates", *Work in progress, May 1998.*

[17] L. Vicisano, L. Rizzo, J. Crowcroft, "TCP-like congestion control for layered multicast data transfer", *Proceedings of IEEE INFOCOM'98, February 1998.*

[18] "wb - LBNL Whiteboard Tool", *available at URL http://www-nrg.ee.lbl.gov/wb/*

[19] R. Wittmann, M. Zitterbart, "Active multicasting for heterogeneous groups", *Proceeding of 4th IFIP Broadband Communications, BC'98, April 1998.*

[20] Hussein Abdel-Wahab, Mark Feit, "XTV: A Framework for Sharing X Window Clients in Remote Synchronous Collaboration", *Proceedings, IEEE TriComm '91: Communications for Distributed Applications and Systems, April 1991.*

A Two Suboptimal Algorithms

This section explains why the first two algorithms mentioned in section 3 have not be selected.

Send ADUs Independently to One Another

```
   [1;6]      [7;12]
     |          |
L1  V 1    2  V 3    4    5    6 |  7    8    9   10   11   12
L2    4    5    6 | 10   11   12 |
```

Figure 9: Scheduling with ADUs sent independently to one another.

Figure 9 illustrates the first strategy using a simple scenario: two ADUs, each of them segmented into six packets, are submitted at times 0.5 and 2.5. Here, S sends the packets 1 to 6 (first ADU) without taking into account the availability of packets 7 to 12 (second ADU). Then S send the upper half of the packets of the second ADU on level 2 as if it was an isolated ADU. Because transmissions are performed blindly, reception only finishes at time 9 with both layers.

Table 4: Evaluation of solution 1.

layers	end	spreading	nb of pkts/ratio	throughput ratio
L1 only	12	6, 12	12 / 1.0 ratio	1.0 ratio
L1+2	9	3, 9	18 / 1.5 ratio	2.0 ratio

Renegotiate Packet Transmission at Each ADU Submission

```
     [1;6]      [7;12]
       |          |
L1   V 1    2   V 3    4    5    6 |  7    8    9   10   11   12
L2     4    5     8    9   10   11 | 12 |
```

Figure 10: Scheduling when reconsidering transmission at each ADU.

An opposite solution is possible, taking into account the availability of the second ADU immediately (figure 10). Here, the whole transmission planning is reconsidered at time 3 and the transmission of packet 6 on layer 2 is given up. As packets 3 to 12 remain to be sent, S continues with the transmission of the upper half of the remaining packets on layer 2. It is packet: $3 + floor((12-2)/2) = 8$. The reception is faster than with solution 1, but the first ADU is only available at time 6. Besides, the total number of packets is higher because packets 4 and 5 have been sent pointlessly on layer 2. The problem is that this solution does not take the ADU boundaries into account.

Table 5: Evaluation of solution 2.

layers	end	spreading	nb of pkts/ratio	throughput ratio
L1 only	12	6, 12	12 / 1.0 ratio	1.0 ratio
L1+2	7	6,7	19 / 1.6 ratio	2.0 ratio