A Multicast Transmission Schedule for Scalable Multi-Rate Distribution of Bulk Data using Non-Scalable Erasure-Correcting Codes

Yitzhak Birk and Diego Crupnicoff

Technion – Israel Institute of Technology
Haifa 32000, Israel
birk@ee.technion.ac.il, diego@mellanox.co.il

***Abstract*–This paper addresses the efficient multicast dissemination of bulk data from a single server to numerous clients. The challenge is complex: a client may commence reception at arbitrary times, should receive as little "extra" data as possible until it can reconstruct the entire content, and should have flexibility in choosing the data rate. From the network perspective, the data rate over any link should be as close as possible to the maximum single-downstream-client subscription rate. Also, the solution should scale to huge files and numerous subscribers, and should withstand changing network conditions and packet loss. Finally, it should be friendly to other traffic. For any base client-subscription rate and integer factors thereof, we jointly achieve all these goals in a near-optimal way while using standard ("any *k* of *N*") block erasure-correcting codes. Scalability in file size is attained by breaking the file into equisized groups of equisized blocks and separately encoding each group. The other properties are attained by a unique open-loop layered multicast transmission schedule. Each client merely subscribes to one or more standard multicast groups. The need to use special, non-standard and possibly proprietary codes that scale well is thus obviated.**

I. INTRODUCTION

A. Distribution of data

Distribution of a large amount of identical data to a large number of recipients is an important application of the Internet, both due to the importance of this function to users and because of the potentially enormous network-resource consumption. It is useful to distinguish among several categories of this service:

- Real-time ("live") streaming.
- On-demand streaming of stored data.
- Distribution of bulk data.

Examples include live multicast of video and audio presentations, on-demand or "near-on-demand" movies (VOD, NVOD), and distribution of software updates, respectively. Our interest in this paper is in the distribution of bulk data. Unlike streaming, in which the recipient cares about the latency until the first bit arrives along with a requirement for smoothness, distribution of bulk data is evaluated by the recipient based on the latency until the transfer is completed.

Distribution of bulk data can be executed in "push" mode, whereby the data is placed in the recipient's storage without being requested. Alternatively, it can be executed in "pull" mode, whereby no data is stored in the recipient's equipment until requested by him. The discussion in this paper is limited to "pull" distribution. Finally, distribution can occur in one step or in multiple steps. In the latter case, copies are first stored in multiple storage facilities, and are then "pulled" from those by the final recipients. This paper addresses the case wherein such mirroring, caching or buffering may not be used, and the network merely routes data from the origin to the destinations. We are thus concerned with the distribution of the same voluminous bulk data from one source to numerous recipients upon demand. (The work is also applicable to a single step of a multi-step approach.)

B. Problem statement

Given a large file located in a computer system, hereinafter called a server, and a very large number of client computers that desire to download the file and are connected to the server by means of a global interconnection network, we wish to make the file available to all clients in minimum time and with the least consumption of network resources. We next list the conditions under which any solution must operate and the requirements that it must satisfy, as well as the performance measures that are used to assess it.

Requirements and conditions:

- *Scalability:*
  - very large files;
  - numerous recipients.
- *Flexibility*:
  - arbitrary starting times by the clients;
  - different connection rates for different clients;
  - variable connection rate (for same client);
  - arbitrary client locations.
- *Robustness.* Very graceful (if any) degradation when packets are lost and/or network conditions change (congestion, connection dropping, etc.)
- *Friendliness* to other application and network traffic.

We elected to impose yet another requirement:

- *Pure Networking.* The solution should not require any data-storage capability (except for routing tables and control information) except at the origin server and at the clients; i.e., no mirroring, buffering or caching.

Performance measures:

- *Client perspective.* The elapsed time from the client's request until sufficient data has been received to permit reconstruction of the entire original file. Whenever a client receives data at the maximum possible rate for that client, a nearly equivalent measure is the excess amount of data (beyond the original file size) that a client must receive before it can reconstruct the original file. A solution is optimal if the amount of data that a client receives until it can reconstruct the original file is equal to the file size (ignoring header overhead and lost packets).
- *Server/Network perspective.* The amount of data that must be sent over any given link until all downstream clients are able to reconstruct the entire file. In any optimal solution, the data rate over any given link is equal to that of the connection of the fastest single client that is downstream from that link. (When combined with client-perspective optimality, this results in a truly optimal scheme. When evaluated in isolation, transmission of garbage could also qualify...)

The optimization of a solution for one performance measure may be at odds with its optimization for the other. The challenge is to find a solution that concurrently optimizes both, and does so under the conditions that were described while meeting the various requirements. This complex challenge is our goal in this paper.

C. Related work

The different types of information-dissemination services have been addressed in the past. For brevity, we restrict the discussion to distribution of bulk data. The most common approach to date entails the "pulling" of such data by each requesting client, either directly from the server, directly from mirror sites (the requesting user selects the site), or automatically through collaboration among the mirror sites and the origin server.

Although having each client pull the data directly using protocols such as FTP is perhaps still the most common solution, it obviously does not scale well to a large number of clients.

Akamai Inc., for example, is a provider of infrastructure and software for the latter type of service. Approaches that employ interim storage are viable and can be attractive. Nonetheless, they can be costly, add complexity, and can introduce new security problems. As stated, we restrict the discussion to "pure networking" solutions.

The most prominent "pure networking" solution to the above scalability problem is the use of a multicast distribution tree. Unlike its use for live streaming, a joining client must receive the entire file. Unlike near-on-demand streaming of stored data (NVOD), however, the order in which data is received by a client does not matter. Consequently, a client can subscribe to a multicast session at will, and remain connected until it has received the entire file. (We assume that packets carry serial numbers that identify the location of their content within the file.) With perfect communication (no packet losses) and equal subscription rates for all clients, such a scheme is optimal.

In practice, packets are lost. Moreover, different clients may fail to receive different packets. Various "Reliable Multicast" schemes have been proposed to address this problem [1][2][3]. One class of schemes entails the retransmission by the origin server of any missing packet to the requesting client(s). Such an approach does not scale very well to a large number of clients. Therefore, routers are sometimes used to assist in the consolidation of multiple requests for the same packets. However, even if this is done, any packet that failed to be received by some client must be retransmitted by the server. In [4], "coding on demand" is used to reduce the amount of data that must be retransmitted. An approach that is particularly attractive for a very large number of clients wishing to receive a large file over a time interval (e.g., downloading the new version of a common web browser during the first weeks of its availability) is for the server to transmit the file cyclically. Each client stays tuned until it has received the entire file or, after some deadline, requests a private transmission of few missing blocks. Such an approach can strike a sensible trade-off between latency from a client's perspective and network/server efficiency.

The cyclic transmission approach, however, has a substantial drawback: a missing packet can only be replaced by the same packet in the next transmission round. Consequently, a client must usually receive much more data than the file size before it can reconstruct the entire file. This consumes the client's own bandwidth resources, and also causes some increase in traffic over upstream links of the multicast tree. (The latter effect may not be pronounced because, as long as even one client residing downstream from a single link is still in its first round of receiving the file, the transmission over that link will take place regardless of the client that is waiting for some missing packet.)

A network path with packet losses can be regarded as an erasure channel. Consequently, erasure-correcting codes such as the well-known Reed-Solomon codes can be used to obviate the need for the client to wait for the retransmission of a specific packet [5]. With these, a file is broken into k equisized fragments. $(N-k)$ additional fragments are derived from those, such that any k fragments suffice for the reconstruction of the original k fragments and thus the entire file. If N is sufficiently large so that a client can be assumed to have received at least k fragments by the time it should have received N, cyclic transmission of the N fragments by the server would result in an optimal solution for every client, which also makes optimal use of network and server resources. Studies of the use of such codes for various types of data distribution include [6][7][8][9].

Remark. In cyclic multicast, the server transmits at a given rate and a client can reconstruct the file from any k different packets. Therefore, unlike in storage or other communication scenarios, the coding does not represent any communication overhead. k and N may thus be chosen at will, as long as the "any k of N suffice" property holds. (This does preclude the simplest of codes, namely replication.)

Unfortunately, the decoding of Reed-Solomon erasure-correcting codes becomes extremely complex as the sizes of k and N increase, precluding their use as just described. Recently, however, similar codes that do scale well have been invented [10]. These are slightly sub-optimal, requiring some expansion of the block size, but have been shown to be computationally efficient even for fairly large files, and have been put to use for distribution of bulk data [11]. These codes, however, are non-standard and are proprietary. In contrast, Reed-Solomon codes are in the public domain, with numerous efficient implementations (for sufficiently small k and N) available in hardware, software and firmware [9]. This gave rise to the question whether standard codes, applied to moderately sized groups of blocks, can be employed in an optimal (or near-optimal) solution for the bulk-data distribution problem as defined earlier.

In [12], it is proposed to break a file into equisized groups of equisized blocks (independent of file size), encode each group separately and transmit blocks from the different groups in a round-robin fashion. This is referred to as ***"group interleaving"***. Simulation results are provided for a single transmission rate, and cross-channel scheduling is alluded to for layered multicast. In [13], group interleaving is proven to be the optimal transmission schedule for separately-encoded groups. This is moreover shown to be true for any packet-loss rate. For independent packet losses, the mean overhead from a client perspective (relative to the encoding of the entire file as a single group) is shown to be under 20% across a very broad range of loss rate and file size. This overhead is shown to be even smaller when errors occur in bursts. Finally, the overhead with grouping and a random packet-transmission schedule is some 60%.

In practical networks, different data rates are available to different clients, be it due to their last-mile connection, a non-uniform structure of the network, other traffic, or even due to clients' own allocation of data rate to different applications. To address such situations effectively, various "layered multicast" schemes have been proposed, with the rate usually determined by the subscription of each client to subsets of the layers (or channels) [11][14][15][16][17]. (Some of these references discuss streaming rather than bulk data.) Some schemes permit a router to drop packets as an explicit rate-control mechanism based on its knowledge of downstream clients' data rates [10]. In view of the importance of group interleaving in the reduction of overhead, it is important to preserve the group-interleaving property when extending grouping to layered multicast.

In [18], a multilayer schedule is proposed for a set of channels, wherein the transmission rate over any given channel equals the sum of those over the lower channels, and a client may subscribe to any contiguous subset of channels that includes the slowest one. This schedule is claimed without proof to retain the group interleaving property for any client. We refer to this organization of channel rates and subscription rule as ***"cumulative exponential"***. This subscription rule permits limited flexibility in the client subscription rate. Further discussion will appear in later sections. Another multi-rate approach, which requires coding of the entire file, appears in [19]. Yet another multilayer scheme [20] claims to improve upon [18] by certain measures, but appears to pay less attention to the efficient utilization of network links (client-subscription is not cumulative). Also, some of the claimed benefits may be sensitive to packet-loss rate.

The main contribution of the current paper is a novel layered multicast schedule for use with grouping. This schedule jointly addresses all aspects of efficient distribution of bulk data as defined earlier, including scalability, from both a client's perspective and that of the network, while permitting the efficient use of standard (non-scalable) erasure correcting codes. Another important contribution is evaluation of this schedule, which includes formal proofs of many of its important properties, including the preservation of group interleaving from a client's perspective regardless of its starting time and for a broad range of subscription rates. The proposed schedule would be most useful in conjunction with a client-subscription policy and/or packet dropping by the routers as means of flow- and congestion control. For specific policies, which are beyond the scope of this paper and are not specific to our schedule, see [14][15].

The remainder of the paper is organized as follows. In Section 2, we present our transmission schedule. In Section 3, we prove its optimality for cumulative exponential subscriptions. In Section 4 we evaluate this schedule for finer rate-selection granularity and under dynamically changing network conditions. Section 5 contains concluding remarks.

II. TRANSMISSION SCHEDULE

Given a file of size S, it is partitioned into $G=S/k$ groups, each comprising k packets worth of data. The k packets constituting each group are encoded using an "any k of N" erasure-correcting code to produce N packets, and these are transmitted cyclically according to the schedule that will be developed shortly.

A. Requirements

Group interleaving has been shown to be the best way of overcoming the need to limit the value of k. Since N is also limited (for computational efficiency), we wish to guarantee that a sufficiently-larger-than-k number of consecutive packets received by any client from any single group, are all distinct. Our challenge in designing the schedule is to guarantee the following properties in the packet stream seen by every client that subscribes to any permissible subset of the multicast channels:

- **Group Interleaving.** Any G consecutive packets received by a client belong to distinct groups.
- **Packet Interleaving.** A sufficiently large number (ideally N, but this is not a must) of any consecutive same-group packets arriving at a client are all distinct.
- **Invariance to starting time**: The aforementioned properties hold for any client starting time.

Also, the schedule should be optimal from a network perspective. To this end, we require a client to only subscribe

to a contiguous set of channels beginning with the lowest-rate channel. We refer to this as ***cumulative subscription***.

B. The schedule

Exponential Channel Rates

Given a base rate *B,* we set the transmission rate R_j for channel *j (j=0,1,...)* as follows:

$$R_j = \begin{cases} B & j = 0 \\ \sum_{i=0}^{j-1} R_i & j > 0 \end{cases} \quad (1)$$

Note that, for $j>1$, $R_j=2R_{j-1}$.

Cumulative subscription

This is not a property of the schedule, but will be assumed in its design and analysis throughout this paper. Performance under finer-grain cumulative subscription will be discussed later. Non-cumulative subscription with the same schedule is discussed elsewhere [13]. A client's ***subscription level*** is the number of the highest channel to which it subscribes. Fig. 1 depicts the relationship between subscription level, data rate and channels.

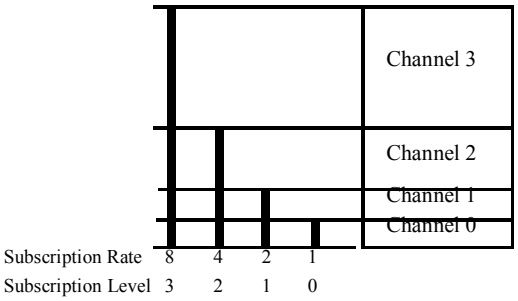

Figure 1. Cumulative exponential channels

Schedule construction

We begin with the lowest-rate channel (j=0), and incrementally introduce additional ones while taking care to keep the interleaving properties of the cumulative stream.

Initially, let us only consider the group to which a packet belongs. Since receivers at subscription level *0* receive only the packets sent on channel *0*, it is clear that packets on channel *0* must be scheduled so that any *G* consecutive packets belong to distinct groups. Without loss of generality, we send packets on channel *0* cyclically from groups *0* through *G-1*. Considering next the clients at subscription level 1, we must schedule channel *1* so that, when combined with channel *0*, the aggregate packet stream still exhibits group interleaving.

At this point, we introduce the general idea used for preserving group interleaving at all levels. Let us assume that the schedule achieves the required interleaving properties through level *l*-1. This means that any *G* consecutive packets transmitted jointly over channels *0* through *l-1* belong to *G* different groups. By setting the rate of channel *l* to the sum of the rates of channels 0,1…,*l*-1, the time for transmitting *G* packets at level *l* becomes exactly half the time for transmitting *G* packets at level *l-1*.

Referring to Fig. 2, consider the two halves of a time interval during which *G* packets (from different groups) are transmitted at level *l-1*. In order to maintain the group interleaving property at level *l*, we schedule the transmission on channel *l* as follows: during the first half, we transmit packets from those groups whose packets are transmitted at level *l-1* during the second half, and during the second half we transmit packets of the groups whose packets are transmitted at level *l-1* during the first half. Clearly, the rate doubling facilitates the construction.

Remark. Note that this principle does not dictate the order in which the groups are organized within each "half interval" on channel *l*.

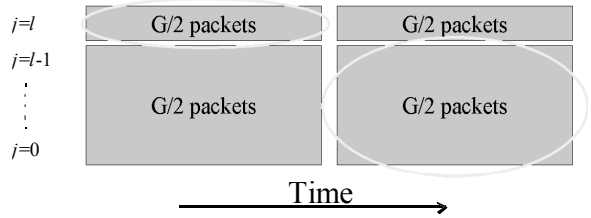

Figure 2. Motivation for exponential channels

Fig. 3 depicts the group index assignment for the first two channels for *G=8*. The number within the blocks denotes the group to which the packet belongs. In this case, the group indexing has a period of *G*, and is the same in both channels to within a cyclical shift.

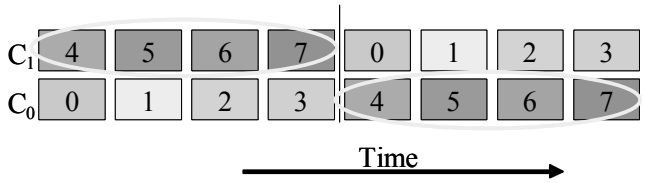

Figure 3. Group-index assignments: channels 0,1.

The scheduling of subsequent channels is based on the same principle. However, as depicted in Fig. 4, the group ordering within each "half interval" becomes trickier. Contrary to what may have been expected, consecutive intervals of *G* packets on channels higher than *1* do not look the same. (Note the order change from *(2,6,3,7)* to *(6,2,7,3)* in channel *2*.) In fact, the schedule is constructed such that the group-index period on channel *j* is $G \cdot 2^{j-1}$ rather than *G*. The analysis of the schedule will prove this to be optimal.

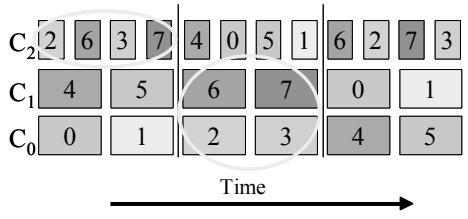

Figure 4. Group-index assignment: channel 2.

The determination of the packet index within each group follows a similar process. The goal is to assign packet indexes so that N consecutive packets from the same group and transmitted at any subscription level are distinct. Fig. 5 highlights consecutive group-*4* packets at subscription level *2*. For the schedule to attain the desired results, all these packets must have distinct packet indexes.

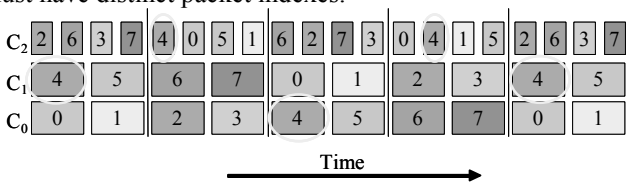

Figure 5. Packet-index assignment.

Packets belonging to different groups are obviously distinct. Therefore, without loss of generality, we consider the packet-index assignment for a single group, and use this assignment for all other groups. Further insights into the construction of the schedule will be offered during the analysis of its properties.

The Packet Schedule Formulas

The formulas expressing the transmission schedule presented below are the main design contribution of this paper.

A multi-channel transmission schedule should specify, for each time slot, the channel that should use the slot as well as the group from which the packet should be taken and that packet's index within the group. However, because of the relationship among the channel rates, their interleaving is trivial and intuitive. For facility of exposition, we therefore treat the channels as if they are active concurrently unless stated otherwise. We use q to denote the qth time slot allocated to a channel. $g(j,q)$ and $p(j,q)$ denote the group number of the packet that is transmitted over channel j in slot q and that packet's index within the group, respectively (See Fig. 6.)

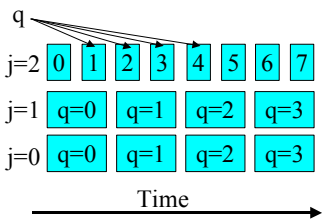

Figure 6. Packet schedule parameters.

The transmission schedule for channel *j* is:

$$g(j,q)=\left\lfloor \frac{q}{2^{\max(0,j-1)}} \right\rfloor + \left\lfloor \frac{G}{2^{j}} \right\rfloor + \max\left(1, \left\lfloor \frac{G}{2^{\max(0,j-1)}} \right\rfloor\right) \cdot \left| q \right|_{2^{\max(0,j-1)}} \Bigg|_{G} \quad (2)$$

$$p(j,q)=\left\lfloor \frac{q}{G\cdot 2^{\max(0,j-1)}} \right\rfloor + \left\lfloor \frac{N}{2^{j}} \right\rfloor + \max\left(1, \left\lfloor \frac{N}{2^{\max(0,j-1)}} \right\rfloor\right) \cdot \left\| \frac{q}{G} \right\|_{2^{\max(0,j-1)}} \Bigg|_{N}$$

In the next section, we prove that this schedule indeed meets its design goals.

III. Properties of the Schedule

In this section, we provide formal proofs of the schedule's properties. These properties, in turn, guarantee the benefits of group interleaving that have been demonstrated elsewhere for a single channel. Due to space limitations, however, some of the actual proofs are omitted altogether or only sketched. The interested reader is referred to [13] for complete proofs and, when relevant, simulations.

A. Network-perspective optimality

The cumulative subscription constraint guarantees that the schedule is optimal from a network perspective, because for any link there is at least one downstream client that subscribes to all the channels carried by that link. We next consider the properties required for client-perspective optimality.

B. Group-Interleaving

Consider $G=2^J$ groups. (This restriction on the values of G will be relaxed later.) We define a ***slot*** to be the transmission time of a single packet over the slowest channel. We number the slots starting from 0, and denote the slot number with s. Similarly, a ***j-mini-slot*** is the transmission time of a packet on channel j. There are thus 2^{j-1} j-mini-slots per slot in channel j $(0<j\leq<J)$, and a single j-mini-slot in channel *0*. Fig. 7 illustrates the terminology.

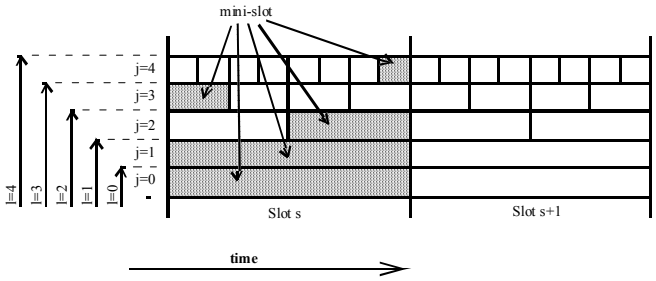

Figure 7. Slot and minislot definitions.

We number the 2^{j-1} j-mini-slots within a slot in channel j from 0, and denote by t the mini-slot number. As shown in Fig. 8, q (the packet position within a channel j as defined above) can be expressed as:

$$q=\begin{cases} s, & j=0 \\ 2^{j-1}\cdot s+t, & j>0 \end{cases} \quad (3)$$

Using the definitions of s and t, we can now rewrite the expression for the group index as:

$$g(j,q)=\begin{cases} \left| s \right|_{G} & j=0 \\ \left| s+2^{J-j}+2^{J-j+1}\cdot t \right|_{G} & 0<j\leq l \end{cases} \quad (4)$$

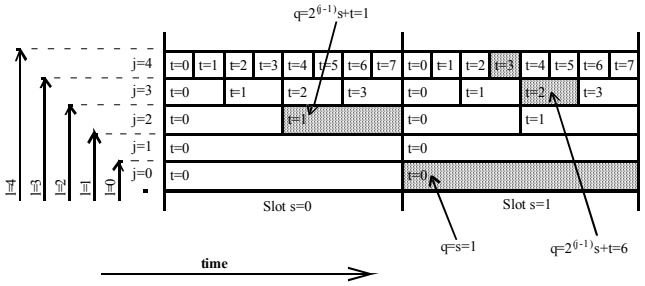

Figure 8. Packet position (q) as a function of slot (s) and minislot (t).

Referring to Fig. 9 as an illustrative example, we wish to prove that the G packets in the shaded region include exactly one packet per group; i.e., that a client that begins receiving packets at a slot boundary will not see a packet of the same group again before having received packets from all other groups. Note that, based on the channel-rate formula, 2^{J-l} slots are required for the transmission of G packets at subscription level l (not channel l).

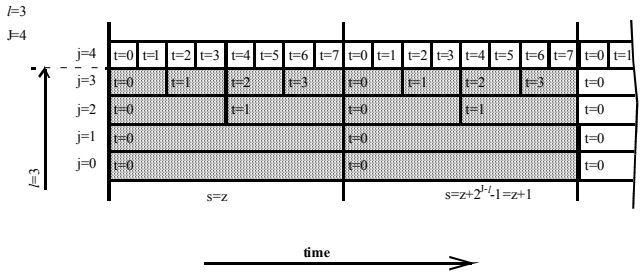

Figure 9. Slots for $G=16$ packets at level $l=3$.

Let z denote the number of the first slot in our observation window. We first prove this property for $z=0$ in Theorem 3, and extend it to any z in Theorem 4.

Lemma 1. *When considering G consecutive packets starting at z=0, the modulo in the group index formula (4) can be removed.*

Proof. Trivial for j=0. Considering the first $G=2^J$ packets transmitted at level $l\in\{1,\ldots,J\}$ and recalling the channel rates, it follows that the required time interval comprises 2^{J-l} slots. (This is equal to the number of those packets transmitted on channel *0*.) Consequently, $s\in\{0,\ldots, 2^{J-l}-1\}$. Combining this with the given range of j and the range of t for any given channel (from the definition of a j-mini-slot), and substituting in (4) while ignoring the modulo sign, it can readily be seen that $g(j,q)<G$, so the modulo sign may be dropped. ■

We next consider the binary representations of the group index of the packets within our observation window, and show that they are unique. Specifically, we show that if the binary representations of two packets are equal then these packets are transmitted on the same channel at the same time, i.e., they are the same packet.

Let us express an integer $a\in(0,1,\ldots, 2^J-1)$ in binary form as $a=\sum_{i=0}^{J-1} b(a,i)\cdot 2^i$, and recall that g is in this range. Next, using the expression for $g(j,q)$ in (4) and focusing on the relevant ranges of s,t,q for the time interval under consideration, we obtain a breakdown of the binary representation of g.

For $j=0$:

$$g = s = \sum_{i=0}^{J-l-1} b(s,i)\cdot 2^i + \sum_{i=J-l}^{J-1} 0\cdot 2^i, \qquad (5)$$

so

$$b(g,i)_{j=0} = \begin{cases} b(s,i) & 0\le i < J-l \\ 0 & J-l \le i < J \end{cases}. \qquad (6)$$

For $j>0$:

$$g = s + 2^{J-j} + 2^{J-j+1}\cdot t, \qquad (7)$$

so

$$b(g,i)_{0<j\le l} = \begin{cases} b(s,i), & 0\le i < J-l \\ 0, & J-l \le i < J-l \\ 1, & i = J-j \\ b(t, i-J+j-1), & J-j < i < J \end{cases} \qquad (8)$$

Fig. 10 depicts schematic views of the binary representation of g.

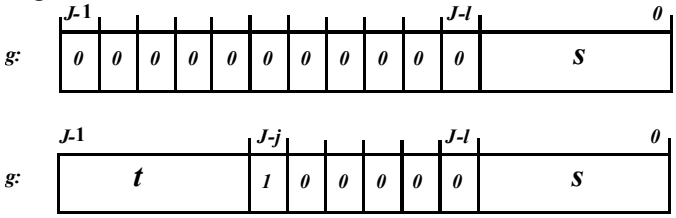

Figure 10. Binary representation of g: a) $j=0$; b) $j>0$.

Using this binary representation of g, we now prove the following lemma by comparing the binary coefficients of two packets with the same group index within the calculated range of s, j and t.

Lemma 2. *Consider the first G packets transmitted at level l (i.e., jointly over channels 0..l-1) during a time interval starting at zero. The binary representations of the group indexes of these packets are distinct. Consequently, if two such packets belong to the same group then they are actually the same packet.*

Proof. The expressions for $b(g,i)$ depend on j and are thus channel-specific. Yet, as illustrated in Fig. 10, it can readily be observed that for any given **level** l, and regardless of channel, the $J-l$ least significant bits of g are determined only by s. The next $l-j-1$ bits are $(0,0,\ldots,0,1)$, which depends only on j, and the remaining $j-1$ bits only depend on t. Based on this, it can be shown that two binary representations of the group indexes of two of the first G packets can only be equal if the respective values of s,t and j are all equal. ■

Theorem 3. *For $G=2^J$ groups and for any subscription level $0<l<J$, the first G packets transmitted starting at slot zero (in terms of the transmission schedule formulas) belong to G different groups.* ■

The extension to any starting slot z now follows.

Theorem 4. *For $G=2^J$ groups and for any subscription level $0<l<J$, G consecutive packets transmitted starting at any slot boundary (in terms of the transmission schedule formulas) belong to G different groups.*

Proof. When starting at a time slot z other than *0,* the range of s in the representation of g becomes $s \in (z, z+1, ..., z+2^{J-l}-1)$. The ranges of j and t remain unchanged. Let $g' \equiv g-z$. Obviously, the theorem holds for g'. However, if a set of values of g' are all distinct modulo G, then adding a constant value z to each and taking the result modulo G would again produce a set of distinct values. ■

Generalized G (not a power of 2)

We have proved that group interleaving is preserved by our schedule whenever G is a power of two ($G=2^J$). This requirement, however, may be somewhat restrictive in practical cases. The file size in bytes is equal to k times G times the packet size in bytes. In actual implementations, we expect k (the number of data packets per FEC group) to be fixed (hard-coded in the implementation). As for the packet size, altering it may require intervention at lower layers of the transmission protocol, which is highly undesired. In view of this, G is determined by the file size, so restricting it to powers of *2* poses a severe limitation.

Fortunately, our schedule exhibits optimal group interleaving properties for any $G=W\cdot 2^J$ and for any subscription level l up to (and including) J, where J is an arbitrary non-negative integer and W is an arbitrary odd positive integer. The proof, which is omitted for brevity, follows a similar approach to those for $G=2^J$. See [13] for details.

Our schedule thus preserves the group interleaving property for any $G=W\cdot 2^J$ (any file size), any subscription level $l \le J$, and any starting time.

C. Packet-index interleaving

Packet-interleaving is an intra-group issue. A group comprises N distinct packets that are transmitted in some cyclical order in (channel, time-slot) pairs that are allocated to the group. Of these, a client must receive any k distinct packets. The true requirement is for a client to receive k distinct packets of any given group before any packet repeats. The difficulty of satisfying this requirement depends on the packet loss probability and on the value of k, as well as on the probability with which it must be satisfied. (Clearly, it is impossible to give an absolute guarantee.) Given a value of N, which is dictated by computational complexity considerations (as was explained earlier, code-rate overhead is not an issue here), the best one can do is to guarantee that, for any subscription level, any N consecutive same-group packets are distinct. In this section, we show that our schedule is optimal in this respect under certain restrictions, and very good in other situations.

In the remainder of this section we prove that, for any $G=W\cdot 2^J$ and $N=2^M$, and for any subscription level $0 \le l \le \min(J,M)$, any $N\cdot G$ consecutively-scheduled packets are distinct.

We define a **superslot** as the time it takes to transmit G packets on the slowest channel. We number the **superslots** starting from *0* and denote the superslot number with *ss*.

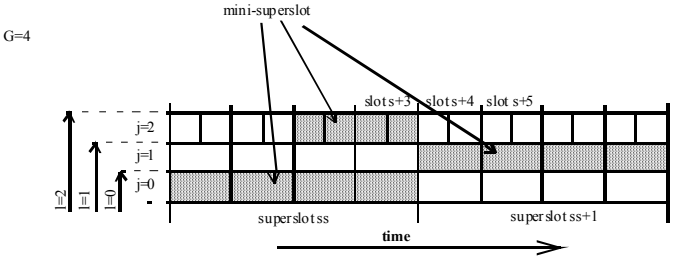

Figure 11. Superslot and *j*-mini-superslot definitions.

We define a ***j*-mini-superslot** as the time it takes to transmit G packets on channel j (see Fig. 11). There are thus 2^{j-1} j-mini-superslots per superslot in channel j $(0<j\le l)$, and a single j-mini-superslot in channel *0*. We number the j-mini-superslots within a superslot in channel j from *0,* and denote the mini-superslot number with: $tt \in \{0,1,...,2^{j-1}-1\}$. We use $gg \in \{0,1,...,G-1\}$ to denote the G packets transmitted in a j-mini-superslot. As depicted in Figure 12, q (the packet position within a channel) can be written as:

$$q = \begin{cases} G\cdot ss + gg & j = 0 \\ G\left(2^{j-1}\cdot ss + tt\right) + gg & j > 0 \end{cases} \qquad (9)$$

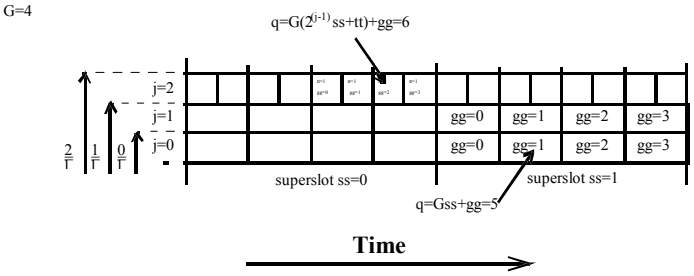

Figure 12. Packet position (q) as a function of *ss, tt* and *gg*.

Lemma 5. *The number of superslots required for transmitting $G\cdot N$ packets at level l is 2^{M-l}.* ■

With the newly defined variables, we can rewrite the packet-index formula of the original schedule (2) as:

$$p(j,q) = \left\| \left\lfloor \frac{G\cdot ss + gg}{G} \right\rfloor \right\|_N \qquad j = 0 \qquad (10)$$

$$p(j,q) = \left\| \left\lfloor \frac{G\left(2^{j-1}ss + tt\right) + gg}{G\cdot 2^{j-1}} \right\rfloor + 2^{M-j} + \right.$$

$$\left. + 2^{M-j+1} \cdot \left\| \left\lfloor \frac{G\left(2^{j-1}ss + tt\right) + gg}{G} \right\rfloor \right\|_{2^{j-1}} \right\|_N \qquad j > 0$$

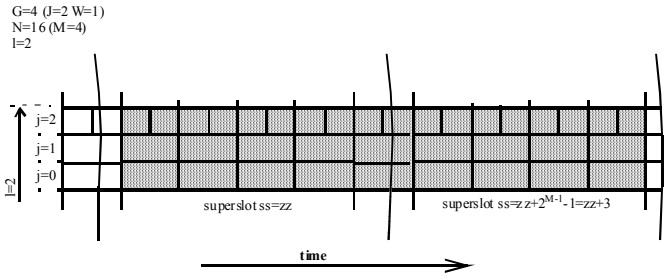

Figure 13. Superslots for *NG* packets at level *l*.

Referring to Fig. 13, we wish to prove that the $G \cdot N$ packets in the shaded region include exactly one instance of each packet index per group. In other words, we prove that if we start receiving packets at a superslot boundary, then we will not see any packet twice before having received all the possible different packets once. We prove this property for every subscription level $0 \le l \le min(J,M)$. Below is an outline of the proof. See [13] for the complete proof.

The proof is based on considering a *j*-mini-superslot (which contains *G* consecutive packets in channel *j*) as a block. We begin by proving that all these packets receive the same packet index. Having done this, we can think of each *j*-mini-superslot as if it were a "packet" to which a packet index is assigned using the same technique as was used to cause the group-interleaving property to hold. Next, we carry out a parameter transformation. This transformation enables us to use the group-interleaving theorem to show that the packet index assigned to each *j*-mini-superslot is such that each of *N* consecutive *j*-mini-superslots is assigned a different packet index. Finally, we show that the *G* packets within each *j*-mini-superslot (that received the same packet index) belong to *G* distinct groups.

Theorem 6. *Let $G=W \cdot 2^J$ (the number of FEC groups into which the file was divided for encoding) and $N=2^M$ (the number of code packets per FEC group). Then, for any subscription level $0 \le l \le min(J,M)$, any $N \cdot G$ consecutively-scheduled packets starting at a superslot boundary are all different. In other words, the packet index that accompanies the group index for every packet is such that within $N \cdot G$ consecutive packets starting at a superslot boundary at any subscription level in the allowed range, there is no packet that has been scheduled twice* [13]. ■

Packet interleaving when starting reception at any slot boundary

For very large files, *G* becomes large and the distance between superslot boundaries grows proportionally. In some practical cases, the superslot boundary requirement may therefore be somewhat restrictive. It is one of our goals to provide a scheme that allows individual receivers to join the transmission at any point in time without loss of optimality. In this section, we explore the packet index interleaving properties of our schedule when a receiver joins the transmission between superslot boundaries.

Theorem 7. *When starting at **any** slot boundary (not necessarily a superslot boundary), for any subscription level $0 \le l \le min(J,M)$, at least N/2 different packets from any group are transmitted before any packet is repeated.*

Proof. From Theorem 6 we know that, when starting at a superslot boundary, *N* different packets per group will be seen before a repetition occurs. We also know that within a superslot there are 2^l packet indexes. As illustrated in Figure 14, the total number of superslots until complete reception of $N \cdot G$ packets at subscription level *l* is 2^{M-l}. In the remaining complete superslots (excluding the one within which we started) there are then $2^l(2^{M-l}-1)=2^M-2^l$ different packet indexes. Clearly, the worst case (minimum different packet indexes) is attained when *l* is highest. Since $l<M$, the smallest number of packet indexes is $2^{M-l}=N/2$. ■

The fact that a receiver gets (barring losses) at least *N*/2 different packets per group before seeing a repetition is a very satisfactory result. Since *N* is typically one order of magnitude larger than *k* (recall that increasing *N* represents no communication overhead in our case), it follows from results of the previous section that a receiver is extremely likely to have completed its reception long before the *N*/2 different packets per group were transmitted. For all practical purposes, the behavior when starting at slot boundaries is thus equivalent to that with infinite *N*.

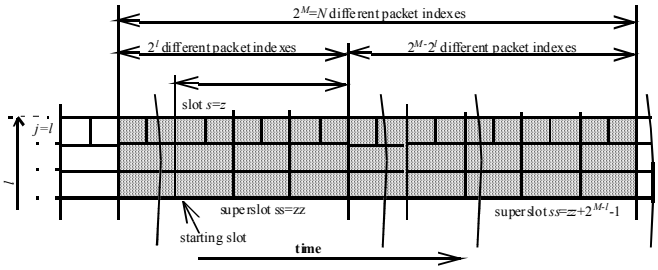

Figure 14. Starting at a regular slot boundary.

IV. EXTENSIONS

In the previous section, we proved the optimality of our schedule for cumulative subscription to channels with exponentially increasing data rates and for subscription levels up to *J*.

In high-speed networks, higher subscription rates may be desired, regardless of coding parameters. Also, while permitting a broad range of subscription rates, doubling the rate when going to the next level may be too coarse in many practical situations. Finally, the analysis was carried out under an assumption of a constant (in time) subscription rate by any given client, whereas in practice this rate may change, be it due to the client's own considerations pertaining to the utilization of its bandwidth or to changing network conditions. In this section, we evaluate our schedule under relaxed operational constraints.

A. Higher-than-J subscription levels

The definition of our schedule can generate higher rates than $B2^{J-1}$, and thus permits additional (higher) subscription

levels, yet our optimality proofs in the previous section only apply up to level J. Simulations have nonetheless shown that the deviation from perfection when using these higher rates is only by a few percents [13]. This is in contrast with [18], wherein the layer construction is tied closely to the file size and coding parameters, and higher subscription layers are simply impossible. It is also worth noting that clients subscribing to levels up to J do not incur any penalty due to the non-optimality of higher levels.

B. Fine-grain rate selection

Consider channel $j>1$ in the original schedule. Its data rate is $B2^{j-1}$, where B is the base rate. Let us think of this channel as comprising 2^{j-1} time-interleaved sub-channels, each with rate B. The new cumulative subscription rule is the same as the original one except that, when it comes to the highest channel to which a client subscribes, it may subscribe to any number of contiguous subchannels (starting with the first one). With this "linear" subscription rule, the subscription granularity becomes B while retaining the same range of available rates. We refer to this technique as ***Channel Sampling***.

Schedule formulas

Let us refer to sub-channel t of channel j ($t=0,1,..2^{j-1}$) as channel $j.t$. The use of t, which was used earlier to denote mini-slots, is intentional, as each subchannel of any given original channel is allocated its mini-slots in a round-robin fashion. By appropriate substitutions in the original schedule formulas, we arrive at those for the subchannel.

For channel $j.t$, the group-index and packet-index of the packet in position q are given, respectively, by:

$$g(j.t,q)=\left|q+\left\lfloor\frac{G}{2^{j}}\right\rfloor+\max\left(1,\left\lfloor\frac{G}{2^{\max(0,j-1)}}\right\rfloor\right)\cdot t\right|_{G} \tag{11}$$

$$p(j.t,q)=\left\|\left\lfloor\frac{q\cdot 2^{\max(0,j-1)}+t}{G\cdot 2^{\max(0,j-1)}}\right\rfloor+\left\lfloor\frac{N}{2^{j}}\right\rfloor+\right. \tag{12}$$

$$\left.+\max\left(1,\left\lfloor\frac{N}{2^{\max(0,j-1)}}\right\rfloor\right)\cdot\left\|\left\lfloor\frac{q\cdot 2^{\max(0,j-1)}+t}{G}\right\rfloor\right\|_{2^{\max(0,j-1)}}\right|_{N}$$

In the remainder of this section, we assess the performance of the schedule for this subscription rule.

Whenever the subscription level does not include entire ("exponential") channels, packet interleaving is retained to an extent that makes any imperfection unnoticeable. The interesting question pertains to the impact of the imperfection in group interleaving. While our schedule was carefully crafted under an assumption of cumulative exponential subscription, we next show that it is also well suited for channel sampling.

Theorem 8. *Any G consecutive packets transmitted on a sub-channel j.t belong to G different groups.*

Proof. For any given sub-channel, i.e., given values of j and t, only q changes in the group-index formula. Therefore, to within an additive constant, $g(j.t,q)=|q|_G$. This obviously represents group interleaving. ■

Remark. This feature of the schedule is closely related to the fact that the original schedule was designed with a group-assignment period of $G\cdot 2^{j-1}$ rather than the more intuitive value of G.

Sampled channels - simulation results

While the per-subchannel group interleaving is good, the overall group interleaving is not preserved. In this section, we provide some simulation results. The simulations compare the mean required time for completing the reception of a file at a given subscription rate with the corresponding time when using group interleaving with a single channel. (Specifically, we plot the mean time for receiving 32 packets. The transmission time of a single packet at the base rate is taken as the unit of time.)

This isolates the effect of imperfect interleaving. (Recall that even perfect interleaving is somewhat inferior to treating the entire file as a single error-correction code group.)

Fig. 15 depicts the comparison for $k=16,N=32$ and a packet loss rate of 0.05. As can readily be appreciated, the results are indistinguishable from those with packet interleaving. Simulations with other loss rates and coding rates show the same relative results. The only case in which the difference can be noticeable is with no erasure-correction and no packet loss. There, the slight imperfection in the interleaving can cause a difference by up to a factor of two. However, this is an unrealistic and uninteresting case, and even slight coding or some non-zero packet-loss probability eliminate the difference.

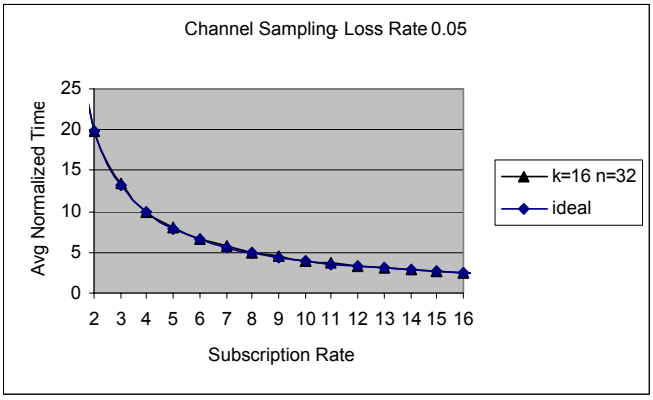

Figure 15. Channel sampling; loss rate = 0.05.

C. Dynamic network conditions

In this section, we evaluate our schedule under dynamic network conditions. The evaluation focuses on the client perspective, because the cumulative subscription rule, rather than the schedule, determines the network perspective.

Network Dynamics Simulations

We used a Markovian subscription-rate generator for the client, causing the rate to usually vary around some initial rate. Fig. 16 depicts a typical scenario.

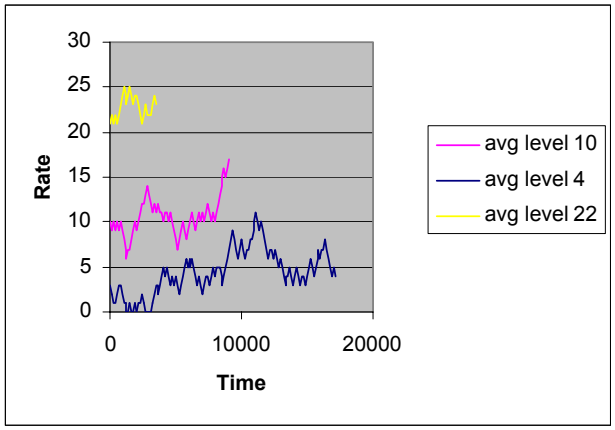

Figure 16. Subscription-rate changes.

We then "computed" the integral of the client's subscription rate over time, starting at the commencement of subscription and ending when the client received the required packets for file reconstruction. The resulting number of transmitted packets was divided by the number obtained when using a single "average" subscription rate for the entire reception. The results show an overhead of approximately 2% across an extremely broad range of file sizes and packet-loss probabilities. The results tend to be slightly better with higher subscription rates due to the finer granularity of rate changes.

V. Conclusions

In this paper, we presented a novel, open-loop multi-channel packet transmission schedule for use in layered multicast of bulk data. The scheme is scalable in both file size and the number of clients, flexible in terms of subscription rates, and exhibits near-optimal performance even with dynamically changing network conditions and subscription rates.

Group interleaving is the best way of utilizing erasure correction codes for multicast subject to a constraint of moderate code-group sizes. Underlying the good properties of our schedule is the fact that it successfully extends these optimal properties to a broad range of concurrent subscription rates, and nearly does so for many additional rates. We also note in passing that the group interleaving results in high resilience to bursty errors, which are common.

When compared with other layered schedules that use moderate-size coding groups, the schedule presented in this paper appears to dominate, as it jointly addresses all relevant aspects from both the network's perspective and that of a client in an optimal or near-optimal way. Also, unlike some, it can be used to generate virtually unlimited subscription levels, and remains nearly optimal even when "linear" subscription steps are permitted (fixed steps rather than exponential). Lastly, its main properties have been proven formally.

When compared with the use of a single code group for the entire file, the increase in the amount of data that a client must receive until it can reconstruct the file is below 20% across a broad range of file sizes and packet-loss rates. This penalty, which is unavoidable when using moderate-size code groups, is tolerable in all but very few practical situations. As a result, the standard, readily available and free erasure correcting codes can be used, and the need for proprietary, non-standard codes that can scale to large file sizes is obviated.

The actual use of our scheme by a client, similarly to other layered schemes, entails subscription to subsets of multicast channels, as well as the use of embedded packet serial numbers in order to reorder received packets and reconstruct the original file.

One drawback of the otherwise near-optimal channel sampling scheme, which permits fine-grain control of the subscription rate, is the need for a potentially very large number of multicast channels. When this is a problem, we note that it is possible to partition only some of the channels into smaller ones. An alternative approach, relaxing the cumulative subscription constraint in conjunction with other measures, will be reported elsewhere.

References

[1] S. Floyd, V. Jacobson, C. Liu, S. McCanne, and L. Zhang, "A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing", IEEE/ACM Trans. Net., Dec. 1997, vol. 5(6), pp. 784-803

[2] D. Towsley, J. Kurose, S. Pingali, "A Comparison of Sender-Initiated and Receiver-Initiated Reliable Multicast Protocols", IEEE J. Sel. Areas in Commun., vol. 15(3), April 1997, pp 398-406.

[3] C. Diot,, W. Dabbous,, J. Crowcroft,, "Multipoint Communication: A Survey of Protocols, Functions and Mechanisms", IEEE J. Sel. Areas in Commun., vol. 15(3), April 1997, pp 277-290

[4] Y. Birk and T. Kol: "Informed-Source Coding-on-Demand (ISCOD) over Broadcast Channels", Proc. INFOCOM 1998, vol. 3, pp. 1257-1264.

[5] R.E.Blahut, "Theory and Practice of Error Control Codes" Addison Wesley, MA, 1984.

[6] D. Rubenstein, J. Kurose, and D. Towsley, "Real-Time Reliable Multicast Using Proactive Forward Error Correction", TR 98-19, CS Dept., Univ. of Massachusetts, Amherst, March 1998

[7] J. Nonnenmacher, E. Biersack, D. Towsley, "Parity-Based Loss Recovery for Reliable Multicast Transmission", ACM SIGCOMM '97, Sep. 1997, pp. 289-300. (also: TR 97-17, CS Dept., Univ. of Mass., Amherst, Mar. 1997).

[8] J. Nonnenmacher, E.W.Biersack, "Reliable Multicast: Where to use Forward Error Correction", Proc. IFIP 5th Int'l Workshop on Protocols for High Speed Networks, pp.134-148, Sophia Antipolis, France, Oct.1996.

[9] L. Rizzo, "Effective Erasure Codes for Reliable Computer Communication Protocols", ACM Comp. Commun. Rev., vol. 27(2), Apr 97, pp 24-36.

[10] M. Luby, L. Vicisano, and T. Speakman, "Heterogeneous multicast congestion control based on router packet filtering", RMT meeting, Pisa, March 1999.

[11] J. Byers, M. Luby, M. Mitzenmacher, A. Rege, "A Digital Fountain Approach to Reliable Distribution of Bulk Data", Proc. ACM SIGCOM '98, Vancouver, B.C., Sep. 1998. Also: TR-98-013 (UC Berkley ICSI).

[12] L. Rizzo, L. Vicisano, “A Reliable Multicast data Distribution Protocol based on software FEC techniques”, Proc. 4th IEEE Workshop on the Architecture and Implementation of High Perf. Commun. Sys., HPCS'97, Greece, June 1997.

[13] D. Crupnicoff and Y. Birk, “Scalable Reliable Multi-Rate Point-to-Multipoint Distribution of Bulk Data”, CCIT report #407, Ctr. For Commun. and Info. Tech., Electr. Engr. Dept., Technion, Dec. 2002.

[14] J. Byers, M. Frumin, G. Horn, M. Luby, M. Mitzenmacher, A. Roetter, W. Shaver, “FLID-DL: Congestion Control for Layered Multicast”, Proc. 2nd Int’l Workshop on Networked Group Commun. (NGC 2000), Stanford, CA, Nov. 2000, pp. 71-81

[15] L. Vicisano, L. Rizzo, and J. Crowcroft. “TCP-like congestion control for layered multicast data transfer.” Proc. INFOCOM '98, San Francisco, April 1998.

[16] S. McCanne, V. Jacobson, M. Vetterli. “Receiver- riven Layered Multicast”, Proc. ACM Sigcomm, 1996.

[17] S. Bhattacharyya, J. F. Kurose, D. Towsley, R. Nagarajan, ``Efficient Rate- Controlled Bulk Data Transfer using Multiple Multicast Groups'', In Proc. of INFOCOM '98, San Francisco, April 1998.

[18] L.Vicisano, “Notes on a cumulative layered organisation of data packets across multiple streams with variable-rate”, unpublished notes

[19] J. Byers, M. Luby, M. Mitzenmacher, “Fine-Grained Layered Multicast”, Proc. IEEE INFOCOM 2001, Anchorage, April 2001.

[20] M.J. Donahoo, M. H. Ammar, and E. W. Zegura, “Multiple-channel Multicast Scheduling for Scalable Bulk-data Transport,” INFOCOM'99, March, 1999.