Layered Multicast Scheduling for the L_∞ Objective

Qingbo Cai* Vincenzo Liberatore†

Abstract

Layered multicast is a scalable solution to data dissemination over the Internet. The performance of layered multicast hinges upon the transmission schedule. In this paper, we study the scheduling problem in the layered multicast context. This work generalizes the extensively studied multicast scheduling problem (layered and non-layered) by introducing simultaneously the data popularity and the interaction among layers, and addresses the strictest L_∞ objective, *i.e.*, to find a schedule which is good for all individual layers. Compared to the previous work, this paper presents a polynomial-time approximation algorithm that uses a different approach and can address the general layered multicast scheduling problem with an arbitrary number of layers. This algorithm is 1.334-approximation for the two-layer case and 1.862-approximation for the general multi-layer cases.

1 Introduction.

Data dissemination scales to Internet numbers if servers use *multicast* channels [17, 5]. However, a scalable multicast server must accommodate a wide range of end-to-end bandwidth available to clients. Heterogenous bandwidth is addressed by *layered multicast* [27, 15, 16, 11, 25, 32], in which the multicast channel is divided into *layers*. Each client individually decides whether to receive data from each layer according to its available bandwidth. If a client receives from a layer, it also receives from all layers below it (*i.e.*, layers are *cumulative*) [27, 22, 16]. The transmission rate doubles from one layer to the layer above it, thereby guaranteeing that each client receives data at a rate of at least $\frac{1}{2}$ of its available bandwidth [27, 22]. Although layered multicast has occasionally been investigated with non-cumulative layers and different layer rates (e.g., [11]), the mainstream model assumes cumulative layers to enable overlapping multicast routes and adopts rate doubling to simplify multicast scheduling while maintaining high bandwidth utilization [27, 22, 15, 25, 26, 10, 33]. Applications of layered multicast include multimedia transmission [23, 14], Web contents dissemination [28], software upgrades [30], broadcast links (e.g., satellite, TV) [19], and layered multicast is especially crucial in overlay networks [29, 34].

In a multicast system, the critical performance metric is the average waiting time for clients to receive the desired data. In turn, the client waiting time depends on the frequency and order of transmission of the desired data, and so it is a function of the server transmission schedule. *Layered Multicast Scheduling* (LMS) is an extension of the well-known *Non-Layered Multicast Scheduling (NLMS)* problem [2, 3, 4, 6, 21], and it is significantly more complicated since LMS addresses the waiting times of heterogeneous clients. The LMS problem has been extensively studied (e.g., [15, 10]). However, most previous LMS work has ignored data popularity, which can be gathered by a middleware [9, 34, 1] and is a central factor in NLMS. This paper investigates the LMS problem with data access probabilities (data popularity).

The LMS problem generalizes the NLMS problem by introducing layers and their interaction. The NLMS problem is to find a schedule for broadcasting a given set of N data items over multiple channels, where each data item is associated with an *access probability* p_i. The objective is to minimize the average waiting time of a random request for a data item. The simpler NLMS problem intuitively reduces to the following problem: given a set of real numbers $\{\tau_1, \tau_2, \ldots, \tau_N\}$, find a "regular" infinite sequence of the integers $\{1, 2, \ldots, N\}$ where the integer i appears "approximately" every τ_i positions. The NLMS problem can be solved in polynomial time for the special case of two data items [8]. The golden ratio sequence algorithm (*GRS*) was developed in [18] and later proved to be $\frac{9}{8}$-approximation for NLMS [6]. GRS is based on the three distance theorem conjectured by H. Steinhaus and independently verified by Vera T. Sós and P. Erdös, *et al*. A polynomial-time approximation scheme (PTAS) was presented in [21]. Some NLMS variants were also studied in [31, 20, 24, 7].

The LMS problem [15, 10] is a stronger version of the NLMS problem, and intuitively corresponds to finding a "regular" infinite sequence that also contains regular subsequences. More specifically, if we extract a subsequence by taking one position out of 2 $(4, 8, \ldots)$,

*Division of Computer Science, Case Western Reserve University.

†Division of Computer Science, Case Western Reserve University.

the subsequences should also be regular in the sense that the integer i appears approximately every τ_i positions. In the layered multicast settings, subsequences correspond to layers, and the LMS problem is fundamentally to find a schedule with good subschedules. A similar sequencing problem found applications in declustering multidimensional data [13].

The LMS problem is a multi-objective optimization problem since the costs should be minimized on all layers. If the *subscription profiles*, *i.e.*, the distribution of users subscribing to each layer, is known, the multiple per-layer objectives can be reconciled into the L_1 form, which is the weighted sum of the per-layer costs using the subscription profiles as weighting coefficients. The L_∞ objective refers to the maximum of the per-layer approximation ratios. It requires no knowledge of subscription profiles and thus is more stringent. A 2-approximation randomized algorithm is presented for both the L_1 and the L_∞ objectives in [12]. However, the derandomization only works for the L_1 objective due to the multi-objective nature of LMS. For the L_∞ objective, a combinatorial construction was given with a performance guarantee of 1.6875 for two layers only. This construction is inherently inextensible to the general LMS problem with an arbitrary number of layers.

In this paper, we study the general LMS problem for the L_∞ objective. We present a polynomial-time approximation algorithm for LMS, which is fundamentally different from the previous combinatorial construction and can address LMS with an arbitrary number of layers. In brief, our approach constructs a periodic schedule by concatenating several schedule blocks. Each schedule block is carefully generated so that some layers (*targets*) have good costs and others (*non-targets*) have moderate costs. Since targets are rotated among the blocks, when averaging the per-layer cost across the blocks in a period, the L_∞ objective is minimized. Our algorithm achieves a better approximation ratio (1.334) than the combinatorial construction (1.6875) in the two-layer case, and is at most 1.862-approximation in the general multi-layer case.

This paper is organized as follows. In Section 2, we define the LMS problem. Then, Section 3 summarizes our approach. In Section 4, we provide a subschedule extraction algorithm which will be used as a building block in the main algorithms. In Section 5, we present the algorithms and performance analyses for LMS with 2 layers, 3 layers, and more than 3 layers, respectively.

2 Preliminaries.

The Schedule. A multicast server continuously and repeatedly transmits N equal-size *pages* denoted as $1, \ldots, N$ on L layers numbered $0, \ldots, L-1$ ($L \geq 2$). The time axis is divided into unit length *time-slot*s, where each time-slot is the duration to transmit one page on layer 0. During each time-slot, the transmission rate (*i.e.*, the number of pages transmitted) on layer l is defined as:

$$r_l = \begin{cases} 1, & l = 0; \\ 2^{l-1}, & 1 \leq l \leq L-1. \end{cases} \tag{2.1}$$

Let $R_l = \sum_{j=0}^{l} r_j$ denote the cumulative rate for layer l, then $R_l = 2^l$. W.l.o.g., we assume that $R_{L-1} = 2^{L-1} \leq N$. A *schedule* is a sequence $S = \{A_{l,t} : 0 \leq l \leq L-1, t \geq 1\}$, where $A_{l,t}$ denotes the set of pages transmitted on layer l during time-slot t (see Figure 1 for an example of a 3-layer schedule). We also define a *subschedule* S_l for layer l as a sequence $\{A_{l,t} : t \geq 1\}$, and define a *combined subschedule* $S_{i,\ldots,l}$ for layer $i, \ldots, l$ as $\{A_{j,t} : i \leq j \leq l, t \geq 1\}$. A *(sub)schedule block* of S with length T is the (sub)schedule S in T consecutive time-slots.

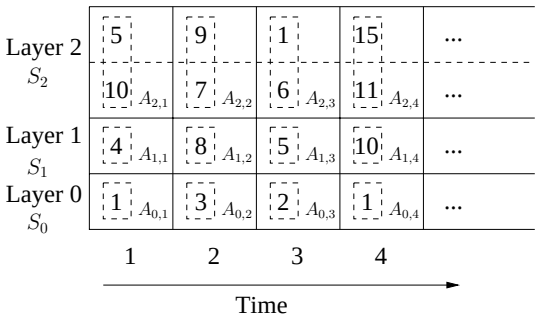

Figure 1: An example of a 3-layer schedule.

The Input. The number of pages N, the access probabilities p_i associated with each page i where $p_i > 0$ and $\sum_{i=1}^{N} p_i = 1$.

The Objective. If a client receives pages from layer l, it simultaneously receives pages from layers $0, \ldots, l-1$, *i.e.*, layers are *cumulative*. The *subscription level* of a client is the highest layer that it can receive pages from. A client must eventually receive every page independently of the subscription level (*reliability requirement*).

Page requests are generated at the beginning of each time-slot, and are independent and identically distributed over time. Each page i is requested with probability p_i. Suppose that, at the beginning of time-slot t, a request for page i is raised by a client with subscription level l. This request will be satisfied at the end of time-slot t' ($t' \geq t$) where page i is transmitted on at least one of layers $0, 1, \ldots, l$, *i.e.*, $i \in \bigcup_{h=0}^{l} A_{h,t'}$

and $i \notin \bigcup_{j=t}^{t'-1} \bigcup_{h=0}^{l} A_{h,j}$. Then, the client has to wait $w_{l,t}(i) = t' - t + 1$ time-slots before page i can be accessed. The cost $Cost(S, l)$ of a schedule S on layer l is defined as the average waiting time for a request with subscription level l (where the average is taken over the time and the page requested), *i.e.*,

$$Cost(S, l) = \limsup_{n\to\infty} \frac{1}{n} \sum_{t=1}^{n} \sum_{i=1}^{N} p_i w_{l,t}(i) \ . \tag{2.2}$$

Let $OPT_l = \inf_S Cost(S, l)$ denote the optimal value of $Cost(S, l)$. Our goal is to minimize the maximum of per-layer approximation ratios (the L_∞ norm) ignoring an additive constant, *i.e.*, to find a schedule S that minimizes

$$\lambda_L = \max_{0\le l\le L-1} \left\{ \frac{Cost(S, l) - c}{OPT_l} \right\} \tag{2.3}$$

for some constant c that does not depend on any problem parameter.

We next define the state vector, and give an equivalent definition of the cost in terms of the state variables in Lemma 2.1. At time-slot $t \ge 1$, the *state vector* is defined as $\overrightarrow{s^t} = \left(\overrightarrow{s_0^t}, \ldots, \overrightarrow{s_{L-1}^t}\right)$, where $\overrightarrow{s_l^t} = \left(s_{l,1}^t, \ldots, s_{l,N}^t\right)$ and $s_{l,i}^t \ge 0$ denotes the number of time-slots elapsed from the most recent transmission of page i on any of the layers $0, \ldots, l$. By definition, $s_{l,i}^t = 0$ if $i \in \bigcup_{h=0}^{l} A_{h,t}$, and $s_{l,i}^t = s_{l,i}^{t-1} + 1$ otherwise. Assume that the 0th transmission of a page i is at some time-slot before 0, then $s_{l,i}^0$ is arbitrary but fixed.

LEMMA 2.1. (TIME REVERSIBILITY) *For any schedule S, the following equality holds:*

$$Cost(S, l) = \limsup_{n\to\infty} \frac{1}{n} \sum_{t=1}^{n} \sum_{i=1}^{N} p_i \left(s_{l,i}^t + 1\right) \ . \tag{2.4}$$

Proof. [Sketch] The proof is standard and similar to that in [20].

Hereafter, we will use the equivalent definition of $Cost(S, l)$ given in Lemma 2.1.

LEMMA 2.2. *If S is a periodic schedule with period T, then:*

$$Cost(S, l) = \frac{1}{T} \sum_{t=1}^{T} \sum_{i=1}^{N} p_i \left(s_{l,i}^t + 1\right) \ . \tag{2.5}$$

Proof. This lemma holds due to Lemma 2.1 and the properties of lim sup.

3 Summary of Approach.

Our approach to construct a periodic schedule S for L layers is as follows. A period of S consists of j schedule blocks $B_1, \ldots, B_j$ of length $T_1, \ldots, T_j$ respectively, where the number of blocks j and the block lengths $T_1, \ldots, T_j$ depend on L. During each schedule block B_{j_1} $(1 \le j_1 \le j)$, we carefully select a certain subset of layers $\{l_1, l_2, \ldots, l_i\}$ (w.l.o.g., $l_1 < l_2 < \cdots < l_i$) to be *target layers* and attempt to minimize the cost of S incurred in B_{j_1} on these layers. The target layers are rotated across blocks so that each layer becomes a target in some of the j blocks. Moreover, each block B_{j_1} is generated so that each target layer l_k has a good cost C_{l_k,j_1} in block B_{j_1} and, simultaneously, each non-target layer $l (l \ne l_1, l_2, \ldots, l_i)$ has a moderate cost C_{l,j_1} in B_{j_1}. Then, by setting the block lengths $T_1, \ldots, T_j$ appropriately, rotating the targets across blocks, and averaging the cost during a period (*i.e.*, $Cost(S, l) = (C_{l,1}T_1 + C_{l,2}T_2 + \cdots + C_{l,j}T_j)/(T_1 + T_2 + \cdots + T_j)$), the per-layer approximation ratios are close to each other and the maximum is minimized.

The core element of our construction is the target selection and the schedule block generation. Generally, we set the distance between adjacent targets to be 3. The reason is two-fold. First, the subschedule for the layer immediately below a target layer can be obtained by *subschedule extraction* with a performance guarantee as in Algorithm 1 (Section 4), which greedily generates a one-level subschedule for lower layers from a schedule or a combined subschedule without degrading the performance on higher layers. Second, the layer immediately above a target layer can be helped by lower layers with a moderate cost due to the cumulative structure of layers. However, exceptions of target selection exist when $L = 2$ or 3 and will be revisited in Section 5.1 and 5.2. The schedule block generation has two steps: the target optimization and the non-target optimization. During the target optimization, given a target set $\{l_1, l_2, \ldots, l_i\}$, we independently generate the combined subschedules $S_{0,\ldots,l_1}$, $S_{l_1+1,\ldots,l_2}, \ldots, S_{l_{i-1}+1,\ldots,l_i}$ using the PTAS for NLMS [21]. Notice that, for a target layer $l_k > l_1$, only a fraction of the cumulative rate R_{l_k} is dedicated to the combined subschedule $S_{l_{k-1}+1,\ldots,l_k}$. However, we will prove that the approximation ratios for the target layers are constant even with a fractional use of their cumulative rates, which will be good enough for our construction. As for the non-target layers, the subschedules S_l $(0 \le l < l_i, l \ne l_1, \ldots, l_i)$ are extracted from the combined subschedules obtained in the previous step. The extraction is done either arbitrarily or by Algorithm 1. Arbitrary extraction is suitable for a non-target layer which is immediately above a target layer, since due to the cumulative structure, a non-target can rely on the

lower target layer subschedule, which is good. However, the performance of arbitrary extraction deteriorates if applied to a non-target which is two layers above a target layer. Therefore, we devise an algorithm that greedily extracts a good subschedule. For the remaining non-targets l ($l_i < l \leq L-1$), subschedules are generated by the PTAS or by arbitrary extraction. Hence, we can generate schedule blocks with good costs for target layers and moderate costs for non-target layers simultaneously.

Another issue in the construction is the inter-block dependency. The schedule in a block B_{j_1} has an impact on the schedule cost in the next block B_{j_1+1} in the form of the state variables. This is because the cost in block B_{j_1+1} depends on the initial state variables of B_{j_1+1}, which in turn depend on the state variables at the end of the previous block B_{j_1} by the definition of state variables (Section 2). Our strategy is to bound the state variables by applying Algorithm 1 and the PTAS for NLMS (Proposition 5.1). Then, by setting the block lengths $T_1, \ldots, T_j$ to be large enough, the impact of this inter-block dependency is averaged in blocks and is only an additive constant.

4 Subschedule Extraction.

We next present an algorithm which extracts a good subschedule block from a periodic NLMS schedule. This algorithm works by greedily selecting a subset of pages from the set of pages transmitted in an NLMS schedule at each time-slot. It is not directly applicable to the general LMS problem, but will be exploited as a building block in the LMS construction in Section 5. Since it relies on a *good* algorithm for NLMS (Definition 4.1), we next briefly recall the NLMS definitions [20].

4.1 NLMS Definitions. The cost $Cost(S)$ of an NLMS schedule S is defined similarly to the LMS cost in Equation 2.2, and the time reversibility also holds for the NLMS cost [6, 20]. At time-slot t ($t > 0$), let s_i^t denote the number of time-slots elapsed from the most recent transmission of page i in S. The cost of S incurred at time-slot t is defined as $C(S,t) = \sum_{i=1}^{N} p_i(s_i^t + 1)$. Let $Cost(S,n)$ be the average cost of S incurred over time-slots $1, \ldots, n$, *i.e.*, $Cost(S,n) = \frac{1}{n}\sum_{t=1}^{n} C(S,t)$. Then, we can express the cost of S as $Cost(S) = \limsup_{n\to\infty} Cost(S,n)$. Notice that $Cost(S) = Cost(S,T')$ if T' is a period of S.

4.2 Subschedule Extraction. Let $S^{(W)}$ ($W > 0$) be an NLMS schedule when W pages can be scheduled during each time-slot. Given a schedule $S^{(W)}$ produced by some algorithm A, Algorithm 1 will extract a subschedule block $S^{(V)}$, where at each time-slot, V pages from the W pages in $S^{(W)}$ are scheduled ($V = \beta \cdot W \in \mathbb{N},\ 0 < \beta < 1$). Moreover, we will establish an upper bound on the performance of Algorithm 1 as a function of the approximation ratio α of A. In the LMS settings, $S^{(W)}$ will be a combined subschedule for layers $i, \ldots, l+1$ ($i \leq l \leq L-2$), and Algorithm 1 will be exploited to divide the W pages at each time-slot in $S^{(W)}$ between the subschedule S_{l+1} on layer $l+1$ and the combined subschedule $S_{i,\ldots,l} = S^{(V)}$ for layers $i, \ldots, l$. This page assignment does not change the cost of layer $l+1$ since layers are cumulative, but it affects the cost of layer l, which will be bounded by the performance guarantee of Algorithm 1. Algorithm 1 relies on a good algorithm A for NLMS that is defined as follows.

Definition 4.1. *An algorithm A is said to be* good *for NLMS if and only if it satisfies:*
(i) A runs in polynomial time;
(ii) A has an approximation ratio of α;
(iii) A generates periodic schedules whose period T' is bounded from above by a polynomial function of $\frac{1}{\alpha-1}$ and N and is independent of the transmission rate.

An example for such an algorithm A is the PTAS in [21] where $\alpha = 1+\epsilon$ and $T' = \frac{N^2+N}{\epsilon}$.

Given $S^{(W)}$, let $\tau_i^{(j)}$ ($j = 1, 2, \ldots$) be the number of time-slots between the $(j-1)$st and the jth transmission of page i in $S^{(W)}$. W.l.o.g., we can assume that no pages are scheduled more than once at any time-slot, and that the 0th transmission of a page i is at some time-slot before 0. Then, $\tau_i^{(1)}$ is arbitrary but fixed. Let $N(i,t)$ be the number of transmissions of page i in the first t time-slots in $S^{(W)}$. In particular, $N(i,0) = 0$. Let $g_{i,t}$ be the number of time-slots between time-slot t and the next transmission (*i.e.*, the $(N(i,t)+1)$st transmission) of page i in $S^{(W)}$. At time-slot t, we define $s_{V,i}^t$ as the number of time-slots elapsed from the most recent transmission of page i in $S^{(V)}$. Therefore, $\tau_i^{(j)}, N(i,t)$, and $g_{i,t}$ depend on $S^{(W)}$, and $s_{V,i}^t$ depends on $S^{(V)}$.

For example, in Figure 2, $W = 2$, $V = 1$, and the $(j-1)$st and jth transmission of page $i = 6$ are in time-slot $(t-3)$ and $(t+2)$ respectively. Then, by the definition of $s_{V,i}^t$, $N(i,t)$, $\tau_i^{(j)}$, and $g_{i,t}$, we obtain $s_{1,6}^t = 3$, $N(6,t) = j-1$, $\tau_6^{(j)} = 5$, and $g_{6,t} = 2$.

If U_t denotes the set of W pages transmitted at time-slot t in $S^{(W)}$, then, by definition, we have

$$s_{W,i}^t = \begin{cases} 0, & i \in U_t; \\ s_{W,i}^{t-1} + 1, & i \notin U_t; \end{cases} \tag{4.6}$$

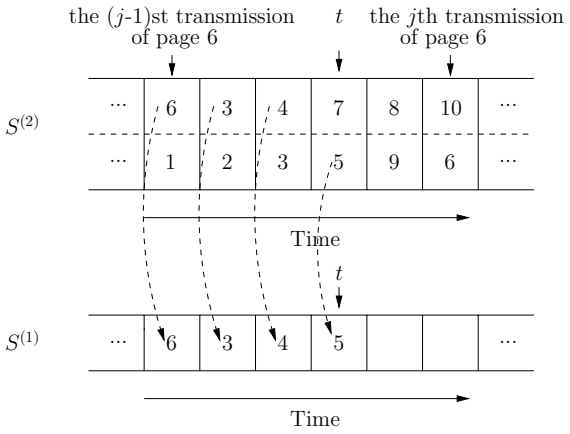

Figure 2: An example of subschedule extraction.

$$N(i,t) = \begin{cases} N(i,t-1)+1, & i \in U_t; \\ N(i,t-1), & i \notin U_t; \end{cases} \tag{4.7}$$

$$g_{i,t} = \begin{cases} \tau_i^{(N(i,t)+1)}, & i \in U_t; \\ g_{i,t-1}-1, & i \notin U_t; \end{cases} \tag{4.8}$$

and

$$g_{i,t-1} = 1, \qquad i \in U_t\ . \tag{4.9}$$

Algorithm 1 Subschedule extraction

Input: A schedule $S^{(W)}$ generated by a good algorithm A for NLMS, its period T', a constant β ($0 < \beta < 1$, $\beta W \in \mathbb{N}$), and a positive integer T_1';
Output: A subschedule block $S^{(V)}$ ($V = \beta W$) with length T_1';

1: **for** $t = 1$ to T_1' **do**
2: $J := \emptyset$;
3: **for** $i := 1$ to V **do**
4: find page $j \in (U_t \setminus J)$ that maximizes $p_j \left(s_{V,j}^{t-1}+1\right) \sum_{x=1}^{\infty} (1-\beta)^x \tau_j^{(N(j,t)+x)}$;
5: $J := J \cup \{j\}$;
6: **end for**
7: schedule the page set J;
8: **end for**

Algorithm 1 is a greedy algorithm to extract a subschedule $S^{(V)}$ from $S^{(W)}$. The main intuition of Algorithm 1 is to minimize the amortized cost of $S^{(V)}$ by a greedy selection of the page set J. Note that the infinite sum in step 4 can be calculated in $O(WT')$ time, since $S^{(W)}$ is periodic with period T'.

For the performance analysis, we define the potential of $S^{(V)}$ at time-slot t as

$$\Phi\left(S^{(V)}, t\right) = \sum_{i=1}^{N} p_i s_{V,i}^t \left(g_{i,t} + \sum_{x=1}^{\infty} (1-\beta)^x \tau_i^{(N(i,t)+1+x)}\right)\ . \tag{4.10}$$

This potential function incorporates the properties of both the schedule $S^{(W)}$ (in the form of $g_{i,t}$, $N(i,t)$, and τ_i^j) and the subschedule $S^{(V)}$ (in the form of $s_{V,i}^t$). Obviously, $\Phi(S^{(V)}, t) \geq 0$ ($t \geq 0$). In particular, observe that $\Phi\left(S^{(V)}, 0\right)$ is a function of (i) $g_{i,0}$ and $\tau_i^{(j)}$ ($j > 1$), which depend on $S^{(W)}$ by definitions; and (ii) $s_{V,i}^0$, which is arbitrary but fixed and non-negative. Therefore, $\Phi\left(S^{(V)}, 0\right)$ is arbitrary but fixed for any given $S^{(W)}$. If a schedule S is constructed by concatenating schedule blocks (as later in Section 5), then, for a block $S^{(V)}$ obtained by Algorithm 1 in the middle of S, the initial potential value $\Phi(S^{(V)}, 0)$ depends on $s_{V,i}^0$, which is no longer arbitrary but in turn depends on the most recent transmission of page i in the previous blocks for any given $S^{(W)}$ due to the shifted origin.

Let $OPT^{(W)} = \inf_{S^{(W)}} Cost(S^{(W)})$ be the optimal cost of an NLMS schedule with transmission rate W. Then, we have the following theorem.

THEOREM 4.1. *Given a schedule $S^{(W)}$ generated by a good algorithm A for NLMS, its period T', a constant β ($0 < \beta < 1$), and $T_1' = \Omega(T'^2)$, Algorithm 1 generates a schedule block $S^{(V)}$ ($V = \beta W$) with length T_1' that has the following property:*

$$Cost\left(S^{(V)}, T_1'\right) \leq (2-\beta)\alpha \cdot OPT^{(V)} + O(\alpha) + \frac{1}{T_1'} \cdot \Phi\left(S^{(V)}, 0\right)\ ,$$

where $\Phi(S^{(V)}, 0)$ is the initial potential value.

Proof. [Sketch] Since the proof is rather technical, only the general idea is given here. We first define the average cost $Avg(t)$ of $S^{(V)}$ at time t ($t > 0$) as $Avg(t) = \sum_{i=1}^{N} p_i \left(g_{i,t-1} + \sum_{x=1}^{\infty} (1-\beta)^x \tau_i^{(N(i,t-1)+1+x)}\right)$. Then we prove that the greedy choice of the page set J in Algorithm 1 ensures that, for any $T_1^{'} > 0$, the cost of the first $T_1^{'}$ time-slot $Cost\left(S^{(V)}, T_1^{'}\right) \leq \frac{1}{T_1^{'}} \left(\sum_{t=1}^{T_1^{'}+1} Avg(t) + \Phi\left(S^{(V)}, 0\right)\right)$. Finally, we show that, for a constant $0 < \beta < 1$, if $T_1^{'} = \Omega(T'^2)$, then

$\frac{1}{T_1'}\sum_{t=1}^{T_1'+1} Avg(t) \le (2-\beta)\alpha \cdot OPT^{(V)} + O(\alpha)$ and conclude the proof.

5 Layered Multicast Schedule Construction.

In this section, we present our main schedule construction algorithms. The periodic schedule constructions are presented for 2 layers ($L = 2$) and more than 3 layers ($L > 3$) in Section 5.1 and 5.3 respectively, since the 2-layer construction has a different target selection strategy from the $L > 3$ case and can help us to understand the more complicated $L > 3$ case. The 3-layer construction is in principle similar to the 2-layer case and therefore is briefly described in Section 5.2.

5.1 A Schedule Construction When $L = 2$. As outlined in Section 3, there is dependency between adjacent blocks. More specifically, the cost of S in a block B depends not only on how S is generated in B but also on the initial state variables s_i^0 in B. When a schedule is constructed by concatenating schedule blocks, for the cost analysis of S in a block B, the origin of the time axis shifts to the beginning of B and consequently the initial state variables s_i^0 depend on the most recent transmission of page i in the blocks before B. Therefore, the state variables $s_{V,i}^t$ should be bounded in the output subschedule block $S^{(V)}$ of Algorithm 1 so that $S^{(V)}$ only incurs a small cost to the next schedule block. This is achieved by adopting the PTAS in [21] as the good algorithm A, where $\alpha = 1+\epsilon$ and $T' = \frac{N^2+N}{\epsilon}$, and is proved in Proposition 5.1. Moreover, the impact of the dependency is only an additive constant by properly setting the block lengths in Algorithm 2.

PROPOSITION 5.1. *If the PTAS in [21] is adopted as the good Algorithm A, Algorithm 1 generates a subschedule block $S^{(V)}$ of any length T_1' such that every page is transmitted at least once in $T' = \frac{N^2+N}{\epsilon}$ time-slots.*

Proof. [Sketch] The PTAS generates a schedule with the property that the set of all N pages appear contiguously every $T' = \frac{N^2+N}{\epsilon}$ time-slots. Moreover, the greedy choice in Algorithm 1 always selects these contiguous N pages into $S^{(V)}$. Hence, the proposition follows.

A consequence of Proposition 5.1 is that the state variables $s_{V,i}^t$ are bounded by T' except in the first T' time-slots, where $s_{V,i}^t$ depends on the initial state variables $s_{V,i}^0$. Moreover, when we generate a schedule by concatenating several schedule blocks, the upper bound on $s_{V,i}^t$ helps to bound the inter-block dependence between a schedule block and its adjacent blocks.

The periodic schedule construction for two layers is given in Algorithm 2, where the PTAS is used as the good algorithm A.

Algorithm 2 Schedule construction for 2 layers

Input: The number of pages N, the access probabilities p_i $(1 \le i \le N)$;
Output: A schedule S for 2 layers with a period of $3T'^2$;

1: in the first T'^2 time-slots block B_1:
2: set the target set to be $\{0\}$;
3: generate $S^{(1)}$ with a period of T' for the subschedule S_0 using the PTAS, and repeat T' periods to fit in B_1; /*target optimization*/
4: set the subschedule S_1 to be empty; /*non-target optimization*/
5: in the following $2T'^2$ time-slots block B_2:
6: set the target set to be $\{1\}$;
7: generate $S^{(2)}$ with period T' for the combined subschedule $S_{0,\ldots,1}$ using the PTAS, and repeat $2T'$ periods to fit in B_2; /*target optimization*/
8: apply Algorithm 1 to extract subschedule $S^{(1)}$ from $S_{0,\ldots,1}$ by letting $\beta = \frac{1}{2}$ and $T_1' = 2T'^2$, and set S_0 to be $S^{(1)}$; schedule the pages that are not extracted into $S^{(1)}$ to transmit on layer 1; /*non-target optimization*/
9: repeat the resulting $3T'^2$ time-slots period to obtain an infinite schedule S;

REMARK 5.1. *When a schedule is constructed by concatenating several schedule blocks, for a schedule block obtained by extraction as in Algorithm 1, the initial potential value $\Phi(S^{(V)}, 0)$ in Theorem 4.1 is not arbitrary any longer but depends on the previous block in the construction.*

The optimal cost $OPT^{(\cdot)}$ discussed so far is in the NLMS settings, and we next bridge the optimal cost in NLMS and LMS in the following lemma.

LEMMA 5.1. *For any positive integer l, $OPT^{(2^l)} \le OPT_l$.*

Proof. The set of feasible schedules for LMS is a proper subset of the set of feasible schedules for the NLMS problem due to the reliability requirement for LMS schedules. Then, by the definitions of $OPT^{(2^l)}$ and OPT_l, we obtain this lemma.

COROLLARY 5.1. *If $W = 2^{l+1}$, $V = 2^l$,* i.e., *$\beta = 1/2$, then Algorithm 1 generates $S^{(V)}$ of length T_1' with the following property:*

$$Cost\left(S^{(V)}, T_1'\right) \le \frac{3}{2}\alpha \cdot OPT_l + O(\alpha) + \frac{1}{T_1'}\Phi(S^{(V)}, 0)\,.$$

Proof. It follows from Theorem 4.1 and Lemma 5.1.

LEMMA 5.2. *A schedule S generated by Algorithm 2 has the following property:*

$$Cost(S,0) \leq \frac{4}{3}(1+\epsilon)OPT_0 + O(1) .$$

Proof. We first prove that the inter-block dependency only incurs an additive constant to the cost of a block. Block B_1 is the target phase for layer 0, and the PTAS is used to generate S_0 with period T' where every page is transmitted. Therefore, for layer 0, only the cost of the first T' time-slots in B_1 depend on the previous non-target phase block. Further, since all N pages appear at least once every T' time-slots in the previous non-target phase block due to Proposition 5.1, the initial state variables of B_1 are bounded by T'. Hence, the sum of the cost in the first T' time-slots in B_1 is at most $(T'+1)+(T'+2)+\cdots+(T'+T') = T'^2 + T'(T'+1)/2 = O(T'^2)$. On the other hand, the sum of the cost in the remaining $(T'^2 - T')$ time-slots in B_1 is $(T'^2 - T')(1+\epsilon)OPT_0$ for some constant ϵ due to the PTAS. Hence, in B_1, layer 0 has an average cost no more than $\frac{O(T'^2)+(T'^2-T')(1+\epsilon)OPT_0}{T'^2} = (1+\epsilon)OPT_0 + O(1)$.

Block B_2 is the non-target phase of layer 0, and S_0 is generated by subschedule extraction in Algorithm 1. By Remark 5.1, the initial potential value $\Phi(S^{(V)}, 0)$ in Theorem 4.1 depends on the previous target phase in the construction. We now prove that the value of $\Phi(S^{(V)}, 0)$ is bounded by $O(T'^2)$. First, by the definitions of $g_{i,t}$ and $\tau_i^{(j)}$, we know $g_{i,0} \leq T'$ and $\tau_i^{(j)} \leq T'$ $(j > 1)$, because $g_{i,t}$ and $\tau_i^{(j)}$ depend on $S_{0,\ldots,1}$ and $S_{0,\ldots,1}$ in B_2 has a period of T' due to the PTAS. Second, since S_0 in B_1 is generated by the PTAS and has a period of T', we have $s^0_{V,i} \leq T'$ for B_2. Hence, we obtain $\Phi(S^{(V)}, 0) = O(T'^2)$ by the definition of Φ in Equation (4.10). Then, by Corollary 5.1, the cost of layer 0 in B_2 is no more than $\frac{3}{2}(1+\epsilon)OPT_0 + O(1)$.

Therefore, this lemma follows by averaging the cost of layer 0 in a period.

LEMMA 5.3. *A schedule S generated by Algorithm 2 has the following property:*

$$Cost(S,1) \leq \frac{4}{3}(1+\epsilon)OPT_1 + O(1) .$$

Proof. [Sketch] The idea is similar to that of Lemma 5.2. The cost of layer 1 is at most $2(1+\epsilon)OPT_1 + O(1)$ in B_1, and $(1+\epsilon)OPT_1 + O(1)$ in B_2.

PROPOSITION 5.2. *A schedule S generated by Algorithm 2 has the following property for some constants ϵ:*

$$\lambda_2 \leq \frac{4}{3}(1+\epsilon) \approx 1.333(1+\epsilon) .$$

Proof. It immediately follows from Lemma 5.2 and 5.3.

5.2 A Schedule Construction When $L = 3$. Algorithm 3 shows how to construct a schedule for 3 layers. The 3-layer construction differs from the 2-layer case in the target selection and block lengths. Moreover, in block B_1 in Algorithm 3, only a fraction $(\frac{3}{4})$ of the cumulative rate R_2 is dedicated to layer 2, *i.e.*, the combined subschedule $S_{1,\ldots,2}$.

Algorithm 3 Schedule construction for 3 layers

Input: The number of pages N, the access probabilities p_i $(1 \leq i \leq N)$;

Output: A schedule S for 3 layers with a period of $5T'^2$;

1: in the first $3T'^2$ time-slots block B_1:
2: set the target set to be $\{0, 2\}$;
3: generate $S^{(1)}$ for the subschedule S_0 and $S^{(3)}$ for the combined subschedule $S_{1\ldots2}$ using the PTAS;
4: arbitrarily extract a subschedule S_1 from $S_{1\ldots2}$;
5: in the next $2T'^2$ time-slots block B_2:
6: set the target set to be $\{1\}$;
7: generate $S^{(2)}$ for the combined subschedule $S_{0\ldots1}$ using the PTAS;
8: set S_2 to be an empty subschedule;
9: apply Algorithm 1 to extract $S^{(1)}$ from $S_{0\ldots1}$, and set S_0 to be $S^{(1)}$;
10: repeat the resulting $5T'^2$ time-slots block to obtain a infinite schedule S;

LEMMA 5.4. *Given a constant γ $(0 < \gamma < 1)$ and two integers a and b $(b = \gamma a)$, the following inequalities hold:*

1. $OPT^{(b)} \leq \frac{1}{\gamma}OPT^{(a)} + O(1)$;
2. $OPT^{(a)} \leq \gamma OPT^{(b)} + O(1)$.

Proof. The proof works by simple schedule reorganizations.

By Lemma 5.4 and 5.1, the cost of S_2 in block B_1 is no more than $\frac{4}{3}OPT_2 + O(1)$.

PROPOSITION 5.3. *A schedule S generated by Algorithm 3 has the following property for some constant ϵ:*

$$\lambda_3 \leq \frac{8}{5}(1+\epsilon) = 1.6(1+\epsilon) .$$

Proof. The proof is essentially similar to that of Proposition 5.2.

5.3 A Construction When $L > 3$. A construction for more than 3 layers is given in Algorithm 4. Let $S^{(W)}_{PTAS}$ denote a schedule block obtained from truncating an infinite schedule, which is generated by the PTAS with transmission rate W.

In Algorithm 4, a period of the output schedule S consists of three schedule blocks of length T_1, T_2, and T_3, respectively. The length of each block depends on L and is carefully chosen to minimize the objective λ_L (see Table 1). During each block, we select a target layer every three layers, except that layer 0 is always in the target set in the third block. The target layers are rotated among the blocks. For example, if $L = 10$, then the target layer set for each block in a period are $\{0, 3, 6, 9\}$, $\{1, 4, 7\}$, and $\{0, 2, 5, 8\}$, respectively. Observe that each layer becomes a target in one block and non-target in the other two blocks, except that layer 0 is a target in two blocks since layer 0 is unlike other layers in that it has no lower layers to rely on. As outlined in Section 3, the reason to set the distance between adjacent targets in a target set to be 3 is that a non-target layer can achieve moderate cost if it is immediately below or immediately above some target layer. Therefore, we can generate schedule blocks with good costs for target layers (by the PTAS) and moderate costs for non-target layers simultaneously.

Number of layers (L)	T_1	T_2	T_3
4	$7T'^2$	$18T'^2$	$28T'^2$
5	$10T'^2$	$9T'^2$	$13T'^2$
6	$134T'^2$	$66T'^2$	$91T'^2$
≥ 7	$67T'^2$	$54T'^2$	$44T'^2$

Table 1: Schedule block lengths in Algorithm 4.

In Algorithm 4, after setting the target set in each block (Line 3-5), we generate each schedule block vertically from lower layers to higher ones (Line 6-21) in three steps. First, we schedule the lowest target layer l_1 and the layer below it if any (Line 8-9). For layer l_1, its cumulative rate R_{l_1} is dedicated to it, and the layer below it if any is scheduled by subschedule extraction using Algorithm 1. Second, the subschedules on layers between the lowest and highest target layer are generated (Line 10-14). We obtain a combined subschedule $S_{l_{h-1}+1,\ldots,l_h}$ for each target layer l_h ($l_1 < l_h \leq l_k$). Notice that only a fraction of the cumulative rate R_{l_h} is dedicated to the combined subschedule $S_{l_{h-1}+1,\ldots,l_h}$. Then, for the intermediate layers between adjacent target layers, we generate subschedules for them by incremental extractions. Finally, we deal with the subschedules on layers above the highest target layer l_k (Line

Algorithm 4 Schedule construction for L ($L > 3$) layers

Input: The number of pages N and the access probabilities p_i ($1 \leq i \leq N$);

Output: A schedule S for L layers with a period of $(T_1 + T_2 + T_3)$;

1: T_1, T_2, and T_3 are the lengths of three consecutive blocks B_1, B_2, B_3 respectively, and are given in Table 1;
2: let $\mathbb{T}_1$, $\mathbb{T}_2$, $\mathbb{T}_3$ be the target sets in B_1, B_2, B_3 respectively;
3: $\mathbb{T}_1 := \{0, 3, 6, \ldots, 3i, \ldots\}$, where $i \in \mathbb{N}$ and $3i \leq L - 1$;
4: $\mathbb{T}_2 := \{1, 4, 7, \ldots, 3i + 1, \ldots\}$, where $i \in \mathbb{N}$ and $3i + 1 \leq L - 1$;
5: $\mathbb{T}_3 := \{0, 2, 5, 8, \ldots, 3i + 2, \ldots\}$, where $i \in \mathbb{N}$ and $3i + 2 \leq L - 1$;
6: **for** each block B_j ($j = 1, 2, 3$) **do**
7: let $\mathbb{T}_j = \{l_1, l_2, \ldots, l_k\}$, where $l_1, \ldots, l_k \in \mathbb{N}$ and $l_1 < l_2 < \cdots < l_k$;
/**schedule layers* $0, \ldots, l_1$ */
8: $S_{0,\ldots,l_1} := S^{(R_{l_1})}_{PTAS}$ of length T_j;
9: if $l_1 = 1$ then $S_0 :=$ a schedule block $S^{(V)}$ extracted by Algorithm 1 letting $S^{(W)} = S_{0,\ldots,1}$, $T'_1 = T_j$, and $V = 1$;
/**schedule layers* $l_1 + 1, \ldots, l_k$ */
10: **for** each target layer $l_h \in \mathbb{T}_j$ and $h > 1$ **do**
11: $S_{l_{h-1}+1,\ldots,l_h} := S^{(R_{l_h} - R_{l_{h-1}})}_{PTAS}$ of length T_j;
12: $S_{l_{h-1}+1,\ldots,l_h-1} :=$ a schedule block $S^{(V)}$ extracted by Algorithm 1 letting $S^{(W)} = S_{l_{h-1}+1,\ldots,l_h}$, $T'_1 = T_j$, and $V = R_{l_h-1} - R_{l_{h-1}}$;
13: if $l_{h-1} + 1 = l_h - 2$ then $S_{l_{h-1}+1} :=$ a subschedule arbitrarily extracted from $S_{l_{h-1}+1,\ldots,l_h-1}$;
14: **end for**
/**schedule layers* $l_k + 1, \ldots, L - 1$ */
15: **if** ($k \geq 1$) (*i.e.*, there is more than one targets) or ($l_k + 1 = L - 1$) **then**
16: $S_l := S^{(r_l)}_{PTAS}$ of length T_j, for $l = l_k + 1, \ldots, L - 1$;
17: **else**
18: $S_{l_k+1,\ldots,L-1} := S^{(3R_{L-1}/4)}_{PTAS}$ of length T_j;
19: $S_{l_k+1} :=$ a subschedule arbitrarily extracted from $S_{l_k+1,\ldots,L-1}$;
20: **end if**
21: **end for**
22: concatenate blocks B_1, B_2, and B_3 to obtain a period of an infinite schedule S;

15-20). These layers need special handling, for example, the subschedule on the highest layer $L-1$ cannot be obtained via subschedule extraction. The performance guarantee of Algorithm 4 is given in Theorem 5.1.

THEOREM 5.1. *For any constant $\epsilon > 0$, an L layer schedule S can be constructed in polynomial time with the following property:*
(i) $\lambda_L = 1 + \epsilon$ *when* $L = 1$*;*
(ii) $\lambda_L = \frac{4}{3}(1+\epsilon) \approx 1.333(1+\epsilon)$ *when* $L = 2$*;*
(iii) $\lambda_L = \frac{8}{5}(1+\epsilon) = 1.6(1+\epsilon)$ *when* $L = 3$*;*
(iv) $\lambda_L = \frac{88}{53}(1+\epsilon) \approx 1.660(1+\epsilon)$ *when* $L = 4$*;*
(v) $\lambda_L = \frac{197}{112}(1+\epsilon) \approx 1.759(1+\epsilon)$ *when* $L = 5$*;*
(vi) $\lambda_L = \frac{11216}{6111}(1+\epsilon) \approx 1.835(1+\epsilon)$ *when* $L = 6$*;*
(vii) $\lambda_L = \frac{6448}{3465}(1+\epsilon) \approx 1.861(1+\epsilon)$ *when* $L \geq 7$.

Proof. [Sketch] Claim (i) immediately follows, since if $L = 1$, the LMS problem becomes the NLMS problem where the PTAS can be applied. Claim (ii) and (iii) have been verified in Proposition 5.2 and 5.3 respectively, and the remaining claims can be proved using similar ideas as in Claim (ii) and therefore are omitted.

References

[1] M. Altinel, D. Aksoy, T. Baby, M. Franklin, W. Shapiro, S. Zdonik: DBIS Toolkit: Adaptable Middleware for Large Scale Data Delivery. In Proc. of ACM SIGMOD, 1999.

[2] M.H. Ammar, J.W. Wong: The Design of Teletext Broadcast Cycles. Performance Evaluations. **5**:4 (1985) 235–242

[3] M.H. Ammar, J.W. Wong: On the Optimality of Cyclic Transmission in Teletext Systems. IEEE Transactions on Communications.**35**:1 (1987) 68–73

[4] S. Anily, C. A. Glass, R. Hassin: The scheduling of maintenance service. Discrete Applied Mathematics. **80** (1998) 27–42

[5] S. Banerjee, B. Bhattacharjee, C. Kommareddy: Scalable Application Layer Multicast. In Proc. of SIGCOMM 2002. 205–217

[6] A. Bar-Noy, R. Bhatia, J. Naor, B. Schieber: Minimizing Service and Operation Costs of Periodic Scheduling. Mathematics of Operations Research. **27:3** (2002) 518–544

[7] A. Bar-Noy, B. Patt-Shamir, I. Ziper: Broadcast Disks with Polynomial Cost Functions. Proc. of IEEE INFOCOM'00. 575–584

[8] A. Bar-Noy, Y. Shilo: Optimal Broadcasting of Two Files over an Asymmetric Channel. Journal of Parallel and Distributed Computing. **60**:4 (2000) 474–493

[9] J. Beaver, N. Morsillo, K. Pruhs, P. K. Chrysanthis, V. Liberatore: Scalable Dissemination: What's Hot and What's Not. Proc. of WebDB 2004. 31–36

[10] Y. Birk, D. Crupnicoff: A Multicast Transmission Schedule for Scalable Multi-Rate Distribution of Bulk Data using Non-Scalable Erasure-Correcting Codes. Proc. of IEEE INFOCOM'03. 1033–1043

[11] J. Byers, M. Luby, M. Mitzenmacher: Fine-grained Layered Multicast. Proc. of IEEE INFOCOM'01. 1143–1151

[12] Q. Cai, V. Liberatore: Approximation Algorithms for Layered Multicast Scheduling. Proceedings of the 16th International Symposium on Algorithms and Computation (ISAAC). (2005) 974–983

[13] C. Chen, R. Bhatia, R. K. Sinha: Multidimensional Declustering Schemes Using Golden Ratio and Kronecker Sequences. IEEE Transactions on Knowledge and Data Engineering, vol 15, no 3, 2003. 659–670

[14] P. A. Chou, A. E. Mohr, A. Wang, S. Mehrotra: Error Control for Receiver-driven Layered Multicast of Audio and Video. IEEE Transactions on Multimedia, Vol. 3, No. 1, 2001.

[15] M. J. Donahoo, M. H. Ammar, E. W. Zegura: Multiple-Channel Multicast Scheduling for Scalable Bulk-data Transport. Proc. of IEEE INFOCOM'99. 847–855.

[16] I. El Khayat, G. Leduc: Congestion Control for Layered Multicast Transmission. Networking and Information Systems Journal, vol. 3, 2000. 559–573

[17] S. Floyd, V. Jacobson, CG. Liu, S. McCanne, L. Zhang: A reliable multicast framework for light-weight sessions and application level framing. IEEE/ACM Transactions on Networking, 1997. 784–803

[18] A. Itai, Z. Rosberg: A golden ratio control policy for a multiple-access channel. IEEE Transactions on Automatic Control. **29**:8 (1984) 712–718

[19] M. Jung, J. Nonnenmacher, E. W. Biersack: Reliable Multicast via Satellite: Uni-directional vs. Bi-directional Communication. Proc. of KiVS, 1999.

[20] C. Kenyon, N. Schabanel: The Data Broadcast Problem with Non-Uniform Transmission Times. Algorithmica. **35** (2003) 146–175

[21] C. Kenyon, N. Schabanel, N. Young: Polynomial-Time Approximation Scheme for Data Broadcast. Proceedings of the 32nd Annual ACM Symp. on Theory of Computing (STOC). (2000) 659–666

[22] A. Legout, E. W. Biersack: PLM: Fast Convergence for Cumulative Layered Multicast Transmission Schemes. In Proc. of ACM SIGMETRICS'2000. 13–22

[23] X. Li, S. Paul, M. Ammar: Layered Video Multicast with Retransmission (LVMR): Evaluation of Hierarchical Rate Control. Proc. of IEEE INFOCOM'98.

[24] V. Liberatore: Multicast scheduling for list requests. Proc. of IEEE INFOCOM'02. 1129–1137

[25] J. Liu, B. Li, Y. Zhang: A Hybrid Adaptation Protocol for TCP-friendly Layered Multicast and its Optimal Rate Allocation. Proc. of IEEE INFOCOM'02. 1520–1529.

[26] J. Liu, B. Li, Y. Zhang: An End-to-End Adaptation Protocol for Layered Video Multicast Using Optimal Rate Allocation. IEEE Transactions on multimedia, **6**

(2004):1. 87–102
[27] S. McCanne, V. Jacobson, M. Vetterli: Receiver-driven Layered Multicast. ACM Special Interest Group on Data Communication (SIGCOMM 1996). 117–130
[28] A. Ninan, P. Kulkarni, P. Shenoy, K. Ramamritham, R. Tewari: Cooperative leases: Scalable consistency maintenance in content distribution networks. Proc. of International WWW Conference, 2002.
[29] L. Peterson, D. Culler, T. Anderson, T. Roscoe: A Blueprint for Introducing Disruptive Technology into the Internet. In Proc. of the 1st Workshop on Hot Topics in Networks (HotNets-I), Princeton, New Jersey, USA, October 2002.
[30] R. Rummler, A. H. Aghvami: End-to-end IP multicast for software upgrades of reconfigurable user terminals within IMT-2000/UMTS networks. IEEE International Conference on communications, vol. 1, 2002. 502–506
[31] N. Schabanel: The Data Broadcast Problem with Preemption. Proc of the 17th International Symp. on Theoretical Computer Science (STACS 2000). 181–192
[32] P. A. Silva Gonçalves, J. F. Rezende, O. C. M. B. Duarte, G. Pujolle: Improving Feedback Merging for Source-Adaptive Layered Multicast Schemes. Cluster Computing, vol 8, no. 1, 2005. 77–88
[33] L. Vicisano, J. Crowcroft, L. Rizzo: TCP-like Congestion Control for Layered Multicast Data Transfer. Proc. of IEEE INFOCOM'98.
[34] W. Zhang, W. Li, V. Liberatore: Application-Perceived Multicast Push Performance. Proc. of IPDPS 2004.