On the Optimal Scheduling of Streaming Applications in Unstructured Meshes

Luca Abeni, Csaba Kiraly, and Renato Lo Cigno*

DISI – University of Trento, Italy
{abeni,kiraly,locigno}@disi.unitn.it

Abstract. Unstructured, chunk-based P2P streaming (TV and Video) systems are becoming popular and are subject of intense research. Chunk and peer selection strategies (or scheduling) are among the main driver of performance. This work presents the formal proof that there exist a *distributed* scheduling strategy which is able to distribute every chunk to all N peers in exactly $\lceil \log_2(N) \rceil + 1$ steps. Since this is the minimum number of steps needed to distribute a chunk, the proposed strategy is optimal. Such a strategy is implementable and an entire class of deadline-based schedulers realize it. We show that at least one of the deadline-based schedulers is resilient to the reduction of the neighborhood size down to values as small as $\log_2(N)$. Selected simulation results highlighting the properties of the algorithms in realistic scenarios complete the paper.

Keywords: P2P, Streaming, Optimality.

1 Introduction

P2P streaming and in particular P2P support for IP-TV are becoming not only hot research topics, but also available systems and services like [1,2,3,4,5].

Fundamental to support live streaming is the guarantee of a low distribution delay of the information to all peers. This is strictly related to the overlay characteristics and the scheduling that distribute chunks to peers.

The community has been divided on whether structured systems, i.e., an overlay with known and controlled topological properties like a tree or a hypercube, or unstructured systems based on general meshes are better for this scope. The advantage of structured systems lies in the possibility of finding deterministic scheduling that achieve optimal performance, but they are normally fragile in face of churn (coming and leaving of nodes), require signaling for the overlay maintenance, and can be complex to manage. Unstructured systems, instead, are robust and easy to manage. Overlay maintenance only requires connectivity: each node autonomously search and contact its own neighbors. Their disadvantage has been so far the impossibility of finding a *distributed* scheduling algorithm that is optimal and robust under normal operating conditions.

* This work is supported by the European Commission through the NAPA-WINE Project (Network-Aware P2P-TV Application over Wise Network – www.napa-wine.eu), ICT Call 1 FP7-ICT-2007-1, 1.5 Networked Media, grant No. 214412.

L. Fratta et al. (Eds.): NETWORKING 2009, LNCS 5550, pp. 117–130, 2009.

This paper tackles this problem, demonstrating the existence of an entire class of optimal schedulers under the assumption that the overlay is fully connected, and showing that at least one of these schedulers is robust against the reduction of the neighborhood down to $\log_2(N)$, where N is the number of peers.

2 Problem Statement

We study the scheduling (chunk and peer selection) for dissemination at each peer in non structured overlay networks. It is well known that the lower bound on the dissemination delay of any piece of information, given that nodes have exactly the bandwidth necessary for the streaming itself, is $\delta_{lb} = (\lceil \log_2(N) \rceil + 1)T$ where T is the transmission time[1]. It is also known [1] that centralized schedulers can distribute every chunk of a stream in exactly δ_{lb}. Also, in [6] it was proved that a bound holds for several *distributed* schedulers if $N \to \infty$ and $M_c \to \infty$ (M_c is the number of chunks). However, when real-time distribution systems are considered such an asymptotic bound is not equivalent to δ_{lb}.

This paper focuses on formally proving the existence of a distributed optimal algorithm, and in finding robust, feasible schedulers that with restricted neighborhoods perform within a reasonable bound of the optimal one. This is the starting point (a reference optimum) for further research on heterogeneous systems, on the interaction of the overlay with the underlying IP network, and on all those 'impairments' that forbid finding closed-form formal solutions to problems in real networking scenarios.

2.1 System Description

We consider an overlay of peers connected with a general mesh topology. The total number of peers is N. Each peer is connected to $\mathcal{NN}$ other peers[2] which constitute its *neighborhood*. A special case is $\mathcal{NN} = N - 1$, which define a fully connected mesh. We consider the presence of one more "special peer" that is the source of the video. The source never receives chunks, so its links are logically unidirectional and it is not part of any neighborhood, i.e., its unidirectional links are additional to the others. Fig. 1 reports two sample topologies.

The source distributes a (possibly live) video or TV program. The video is divided in M_c *chunks* of equal duration emitted periodically. All peers have unit bandwidth (i.e., they can transmit a chunk in exactly the inter-chunk generation time) on the uplink and no limitations on the downlink. We do not consider churn and we focus, as main performance parameter, on the diffusion delay of chunks, which is the delay with which chunks are received by all peers. Formally, if r_i is the emission time of chunk C_i, then its diffusion delay is $f_i = t - r_i$ such that

[1] The bound comes from the fact that each node can transmit the information only after receiving it, and the number of nodes owning the chunk at most doubles every T.

[2] For the sake of simplicity we restrict discussion to n-regular topologies: random graphs with symmetric connectivity and n links per node.

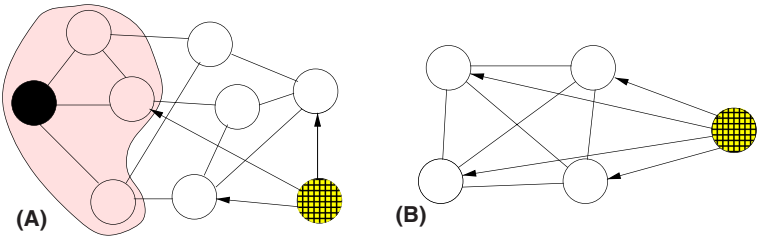

Fig. 1. (A) – General mesh topology with $N = 8$ and $NN = 3$; the shaded (pink) area is the neighborhood of the black node; the source is the checkered (yellow) node; (B) – Full mesh with $N = 4$

all N peers have received C_i. Each peer has a perfect knowledge of the status of its neighbors.

The assumptions above means that: i) no global ordering of peers is required; ii) the system is not structured; iii) schedulers' decisions are independent one another; vi) peers know exactly the subset of chunks already received or being received by all neighbors; and v) signaling delay is negligible.

The first scheduling decision is whether a peer *pushes* information to other peers or if it *pulls* it from other peers . . . or a mix of the two policies. Sometimes in the literature it is stated that pushing information is a behavior typical of structured systems, and pull methods are more adapt for non-structured overlays. Recent papers like [6,7] instead use push schedulers on non-structured meshes. Indeed, the choice of whether it is better to push or pull information is not related to the structure (or the lack of it) of the system, but to the bandwidth bottleneck, which can create conflicts in scheduling decisions.

Push-based systems are suitable for systems where the bottleneck is the uplink, because this guarantees a priori that only one chunk will be scheduled for transmission on the uplink, and that scheduling conflicts arising from the distributed nature of the scheduling will insist on the downlink of other peers.

If the situation were reversed (uncommon in networks dominated by ADSL access, but technically possible), then pull-based schedulers would solve a priori the conflict on the downlink, and more bandwidth-endowed uplinks would accommodate scheduling conflicts. Interestingly, a scenario with symmetric up- and downlink capacities does not offer an easy logical choice on whether pushing or pulling information is the best choice.

We consider push-based schedulers, but we claim that reversing the bottleneck hypothesis, pull-based schedulers which are dual to those we prove optimal in the sequel can be easily derived.

2.2 Formal Notation and Definitions

A system is composed by a set $\mathcal{S} = \{P_1, \ldots P_N\}$ of N *peers* P_i, plus a special node called *source*. Each peer P_i receives chunks C_j from other peers, and send them out to other peers at a rate $s(P_i)$. The source sends chunks with rate

Table 1. Definitions and symbols used in the paper

Symbol	Definition
$\mathcal{S}$	The set of all the peers
N	The number of peers in the system
M_c	The total number of chunks
P_i	The i^{th} peer
C_h	The h^{th} chunk
r_h	The time when the source generates C_h
$\mathcal{NN}$	The neighbourhood size
f_h	The diffusion delay of chunk C_h (the time needed by C_h to reach all the peers)
$\mathcal{C}(P_i, t)$	The set of chunks owned by peer P_i at time t
$\mathcal{C}'(P_i, t)$	The set of chunks owned by P_i at time t which are needed by some of P_i's neighbours
$\mathcal{N}_i$	The neighborhood of peer P_i
$s(P_i)$	The upload bandwidth of peer P_i

$s(source)$. The set of chunks already received by P_i at time t is indicated as $\mathcal{C}(P_i, t)$.

The source, not included in $\mathcal{S}$, generates chunks in order, at a fixed rate λ (C_j is generated by the source at time $r_j = \frac{1}{\lambda} j$). We normalize the system w.r.t. λ, so that $r_j = j$. Also, we set $\forall i, s(P_i) = s(source) = \lambda = 1$, which is the limit case to sustain streaming.

If $\mathcal{D}_j(t - r_j)$ is the set of nodes owning chunk C_j at time t, the worst case diffusion delay f_j of chunk C_j is defined as the time needed by C_j to be distributed to every peer: $f_j = \min\{\delta : \mathcal{D}_j(\delta) = \mathcal{S}\}$. According to this definition, a generic peer P_i will receive chunk C_j at time t with $r_j + 1 \leq t \leq r_j + f_j$. Considering an unstructured overlay t will be randomly distributed inside such interval. Hence, in an unstructured system P_i is *guaranteed* to receive C_j at most at time $r_j + f_j$. To correctly reproduce the whole media stream, a peer must buffer chunks for a time of at least $F = \max_{1 \leq j \leq M_c}(f_j)$ before starting to play. For this reason, the worst case diffusion delay F is a fundamental performance metric for P2P streaming systems, and this paper will focus on it.

When $\forall i, s(P_i) = \lambda = 1$, at time t the source sends a chunk C_j (with $r_j = t$) to a peer and every peer P_i sends a chunk $C_h \in \mathcal{C}(P_i, t)$ to a peer P_k. All these chunks will be received at time $t + 1$.

As discussed earlier, the minimum possible diffusion delay f_j for chunk C_j is $\lceil log_2(N) \rceil + 1$. Chunk diffusion is said to be optimal if $\forall j, f_j = \lceil log_2(N) \rceil + 1 = F$.

The most important symbols used in this paper are recalled in Table 1.

3 Scheduling Peers and Chunks

In a push-based P2P system, when a peer P_i sends a chunk, it is responsible for selecting the chunk to be sent and the destination peer. The chunk C_j to be sent

is selected by a *chunk scheduler*, and the destination peer P_k is selected by a *peer scheduler*. This paper focuses on algorithms which first select the chunk C_j, and then select a target peer P_k which needs C_j, but the definition of optimality presented in this paper is valid for any chunk-based P2P streaming system.

Some well known chunk scheduling algorithms are *Latest Blind Chunk*, *Latest Useful Chunk*, and *Random Chunk* (again, blind or useful). The Latest Blind Chunk algorithm schedules at time t the latest chunk: $C_j \in \mathcal{C}(P_i,t) : \forall C_h \in \mathcal{C}(P_i,t), r_j \geq r_h$ (C_j is scheduled even if all the other peers already have it). The Latest Useful Chunk (LUc) algorithm selects a chunk that is needed by at least one peer: $C_j \in \mathcal{C}'(P_i,t) : \forall C_h \in \mathcal{C}'(P_i,t), r_j \geq r_h$ where $\mathcal{C}'(P_i,t)$ is a subset of $\mathcal{C}(P_i,t)$ containing only chunks that have not already been received (or are not currently being received) by some other peers. The Random Chunk algorithms select a random chunk in $\mathcal{C}(P_i,t)$ (Random Blind Chunk) or in $\mathcal{C}'(P_i,t)$ (Random Useful Chunk – RUc).

Once the chunk C_j to be sent has been selected, the peer scheduling algorithms selects a peer P_k which needs C_j. The most commonly used peer scheduling algorithm is Random Useful Peer, which randomly selects a peer which needs C_j. In theory, the chunk scheduling algorithm can select $P_k \in \mathcal{S}$, but in practice peer P_i will only know a subset of all the other peers, and will select P_k from a subset of $\mathcal{S}$ called *neighborhood*. The neighborhood of P_i is indicated as $\mathcal{N}_i$. The case in which $\forall i, \mathcal{N}_i = \mathcal{S} - P_i$ is special, and corresponds to a fully connected graph.

3.1 Optimal Peer Scheduling

Random peer selection prevents achieving optimality, because the selected peer might be unable to further distribute the chunk. The rationale behind optimal peer selection should be the following: the selected destination peer should be able to immediately take on the role of redistributing the chunk.

We define the "Earliest-Latest" peer scheduler (ELp) as follows: ELp selects as target a peer P_l that needs C_h and owns the latest chunk C_k with the earliest generation time r_k:

$$C_h \notin \mathcal{C}(P_l,t) \wedge \forall P_j \in \mathcal{N}_l, L(P_l,t) \leq L(P_j,t) \tag{1}$$

where $L(P_i,t) = \max_k\{r_k : C_k \in \mathcal{C}(P_i,t)\}$ is the latest chunk owned by or in arrival to P_i at time t. If at time t P_i has not received any chunk yet, $L(P_i,t) = 0$. If more peers exist that satisfy (1) one is chosen at random.

3.2 Optimal Chunk Scheduling

We show in Theorems 1 and 2 that a LUc/ELp scheduler is optimal in the full mesh case; however, LUc/ELp provides large worst-case diffusion delays when the neighbourhood size is reduced (as will be shown in Section 5). Such a bad behaviour is common to all the LUc schedulers, and is caused by the fact that such schedulers always select the latest useful chunk. Hence, if for some

reason (such as a restricted neighbourhood size or a limited knowledge of the neighbourhood) a chunk C_k with $r_k > r_h$ arrives to a peer before C_h is completely diffused, then the peer is not able to diffuse C_h anymore and its diffusion delay is increased by a large amount. In other words, every time that limited knowledge of the neighborhood makes a later chunk arrive to a peer before an earlier one, the diffusion of this latter might be stopped.

For this reason, a new scheduling algorithm has been developed to be equivalent to LUc/ELp in the full mesh case, and to perform reasonably well when the graph is not fully connected. The new algorithm is based on a deadline-based chunk scheduling algorithm, named Dl. The Dl scheduling algorithm works based on *scheduling deadlines* d_k associated to every chunk instance. The scheduling deadline is initialized to $d_k = r_k + 2$ when the source sends C_k at time r_k. The chunk scheduler then works by selecting the chunk C_k with the minimum scheduling deadline:

$$C_k : \forall C_h \in \mathcal{C}'(P_i, t), d_k \leq d_h; \tag{2}$$

Before sending C_k its scheduling deadline is postponed by 2 time units: $d_k = d_k + 2$ (both P_i and the destination peer will see C_k with its new scheduling deadline, while chunk instances present in other peers are obviously not affected).

The scheduling strategy based on selecting the chunk with a minimum deadline is known in literature as Earliest Deadline First (EDF), and is mentioned as "Deadline Driven Scheduling" in a seminal paper by Liu and Layland [8], but to the best of our knowledge, it has never been applied with dynamic deadlines in distributed systems.

Observation 1. *The scheduling deadline d_k of a chunk instance C_k at peer P_i is equal to $r_k + 2d$, where d is the number of times that C_k has been selected by the Dl schedulers along the path taken by the chunk till P_i.*

4 Analysis with Full Meshes

In this section, some important properties of the LUc/ELp and Dl/ELp scheduling algorithms are proved for the case of a fully connected overlay. In Theorems 1 and 2, it is proved that LUc/ELp achieves optimality, while in Theorem 3 the optimality of Dl/ELp is shown.

Lemma 1. *When using ELp, $\forall i, t \leq \lfloor \log_2(N+1) \rfloor \Rightarrow ||\mathcal{C}(P_i, t)|| \leq 1$.*

Proof. During an initial transient, at time t the system contains $2^t - 1$ chunk instances (because at every time instant the source emits a new chunk and all the peers having at least one chunk send a chunk); hence, there are $N - (2^t - 1)$ peers having no chunks. By definition, the ELp scheduler selects such peers as targets, hence a peer P_i can have more than 1 chunk only if $2^t - 1 > N \Rightarrow 2^t > N + 1 \Rightarrow t > log_2(N + 1)$.

Lemma 2. *If $\forall i, s(P_i) = \lambda = 1 \wedge \mathcal{N}_i = \mathcal{S} - P_i$, if a LUc/ELp scheduling algorithm is used, then*

$$\forall \delta,\ 0 < \delta \leq \lfloor log_2(N) \rfloor \Rightarrow ||\mathcal{L}_j(\delta)|| = 2^{\delta - 1}$$

where $\mathcal{L}_j(\delta) = \{P_i : \max_k\{r_k : C_k \in \mathcal{C}(P_i, r_j + \delta)\} = r_j\}$ *is the set of peers having* C_j *as their latest chunk at time* $r_j + \delta$*.*

Proof. The lemma is proved by induction on $\delta = t - r_j$, and by considering the latest chunk owned by the peers at time $t = r_j + \delta$, so that $\mathcal{S}$ is partitioned into three subsets:

- $\mathcal{X}(\delta) = \bigcup\{\mathcal{L}_j(i) : i > \delta\}$ is the set of peers with latest chunk later than C_j;
- $\mathcal{Y}(\delta) = \mathcal{L}_j(\delta)$ is the set of peers having C_j as their latest chunk;
- $\mathcal{Z}(\delta) = \bigcup\{\mathcal{L}_j(i) : i < \delta\}$ the set of peers with latest chunk earlier than C_j.

The above is a partitioning into disjoint subsets, therefore $||\mathcal{X}(\delta)|| + ||\mathcal{Y}(\delta)|| + ||\mathcal{Z}(\delta)|| = ||\mathcal{S}|| = N$. The lemma can be now proved by induction on δ.

Induction base: After chunk C_j is generated by the source at time r_j, it is sent out to a peer P_i, which will receive it at time $t = r_j + 1 \Rightarrow \delta = 1$. Hence,

$$\mathcal{D}_j(1) = \{P_i\} \Rightarrow ||\mathcal{D}_j(1)|| = 1$$

As C_j is the newest chunk in the system, $\mathcal{X}(\delta)$ is empty and C_j becomes the latest chunk on P_i:

$$\forall C_k \in \mathcal{C}(P_i, r_j + 1), r_j > r_k$$

Thus, $\delta = 1 \Rightarrow ||\mathcal{L}_j(\delta)|| = ||\mathcal{D}_j(\delta)|| = 1 = 2^{\delta-1}, ||\mathcal{X}(\delta)|| = 0 = 2^{\delta-1} - 1$. Also note that $||\mathcal{Z}(\delta)|| = N - 1 > ||\mathcal{X}(\delta)|| + ||\mathcal{Y}(\delta)||$.

Inductive step: First of all, it is easy to notice that $||\mathcal{X}(\delta - 1)|| \leq 2^{\delta-2} - 1$: in fact, at every time unit a new chunk $C_k : r_k > r_j$ is generated, and all the peers $P_i \in \mathcal{X}(k-1)$ can send their latest chunk to another peer. As a result, $||\mathcal{X}(\delta-1)||$ will be at most equal to $2||\mathcal{X}(\delta-2)|| + 1$. But $||\mathcal{X}(\delta-2)|| \leq 2^{\delta-3} - 1$ (by induction), so

$$||\mathcal{X}(\delta - 1)|| \leq 2(2^{\delta-3} - 1) + 1 = 2^{\delta-2} - 1$$

Now, if $\delta \leq \lfloor log_2(N) \rfloor$, then

$$\delta \leq \lfloor log_2(N) \rfloor \Rightarrow 2^\delta \leq N \Rightarrow 2(2^{\delta-2} + 2^{\delta-2}) \leq N$$

and since $||\mathcal{L}_j(\delta - 1)|| = 2^{\delta-2}$, $||\mathcal{X}(\delta - 1)|| \leq 2^{\delta-2} - 1$ and $||\mathcal{Z}(\delta - 1)|| = N - ||\mathcal{X}(\delta - 1)|| - ||\mathcal{Y}(\delta - 1)||$, the above equation can be rewritten as

$$2(||\mathcal{X}(\delta-1)|| + 1 + ||\mathcal{Y}(\delta-1)||) \leq N \Rightarrow ||\mathcal{X}(\delta-1)|| + ||\mathcal{Y}(\delta-1)|| \leq ||\mathcal{Z}(\delta-1)|| - 2$$

As a result, at $\delta - 1$, $||\mathcal{Z}(\delta - 1)||$ is more than half of N, therefore there are enough peers with latest chunk older than C_j to receive chunks from both $\mathcal{X}(\delta - 1)$ and $\mathcal{Y}(\delta - 1)$, so $||\mathcal{L}_j(\delta)|| = ||\mathcal{D}_j(\delta)|| = 2^{\delta-1}$, hence the claim.

Theorem 1. *If* $N = 2^i$*, algorithm LUc/ELp is optimal.*

Proof. By definition, an algorithm is optimal iff $\forall j, f_j = \lceil log_2(N) \rceil + 1$. In this case, this means $\forall j, f_j = i + 1$. By Lemma 2,

$$\forall j, \delta \leq \lfloor log_2(N) \rfloor \Rightarrow ||\mathcal{L}_j(\delta)|| = 2^{\delta - 1}$$

hence, $\forall j, ||\mathcal{L}_j(i)|| = 2^{i-1}$. As a result, $||\mathcal{D}_j(i+1)|| = 2||\mathcal{L}_j(i)|| = 2^i = N$, and $f_j = i + 1$.

Theorem 2. *Algorithm LUc/ELp is optimal also if $N \neq 2^i$.*

Proof. If $N = 2^i + n$, with $n < 2^i$, by Lemma 2 it comes $\forall j, ||\mathcal{L}_j(i)|| = 2^{i-1}$. Hence, for $\delta = i$ chunk C_j is sent 2^{i-1} times and chunks with $r_k > r_j$ are sent $2^{i-1} - 1$ times. As a result, $||\mathcal{D}_j(i+1)|| = 2^i$, $||\mathcal{X}(i+1)|| = 2^i - 1$, $||\mathcal{Z}(i+1)|| = 0$, and $||\mathcal{L}_j(i+1)|| < ||\mathcal{D}_j(i+1)||$. To compute the exact value of $||\mathcal{L}_j(i+1)||$, let x be the number of chunks sent by peers in $\mathcal{X}(i)$ to peers in $\mathcal{Z}(i)$ and let y be the number of chunks sent by peers in $\mathcal{Y}(i)$ to peers in $\mathcal{Z}(i)$. According to the peer scheduling rules, $x + y = ||\mathcal{Z}(i)||$ (because chunks are sent to peers having the earliest latest chunk). Moreover, $||\mathcal{L}_j(i+1)|| = y + ||\mathcal{L}_j(i)|| - (||\mathcal{X}(i)|| - x)$. Hence,

$$||\mathcal{L}_j(i+1)|| = ||\mathcal{Z}(i)|| - x + 2^{i-1} - (2^{i-1} - 1 - x) =$$

$$= (N - 2^{i-1} - (2^{i-1} - 1)) - x + 2^{i-1} - 2^{i-1} + 1 + x = N - 2^i + 1 + 1 = N - 2^i + 2$$

Finally,

$$||\mathcal{D}_j(i+2)|| = \min\{N, ||\mathcal{D}_j(i+1)|| + ||\mathcal{L}_j(i+1)|| = 2^i + N - 2^i + 2\} = N$$

Hence, $f_j = i + 2 = \lceil log_2(N) \rceil + 1$.

Observation 2. *If an optimal chunk scheduling is used, all the copies of every chunk C_k are forwarded from time r_k to time $r_k + f_k - 2$.*

Based on the optimality of LUc/ELp, it is now possible to prove that Dl/ELp is an optimal algorithm too. This is done by showing that on a full mesh it generates the same schedule as LUc/ELp.

Theorem 3. *If $\forall i, s(P_i) = \lambda = 1$, $\forall i, \mathcal{N}_i = \mathcal{S} - P_i$, then the chunk distribution produced by Dl/ELp is identical to the chunk distribution produced by LUc/ELp.*

Proof. By contradiction: assume that at any time t_0 the chunk distribution produced by Dl/ELp starts to differ from the one produced by LUc/ELp, i.e., assume that Dl at peer P_i at time t_0 selects chunk C_j while LUc would select chunk C_k (so, $r_k > r_j$). However, it will be shown that choosing C_j with Dl implies $r_j \geq r_k$ contradicting the hypothesis $r_k > r_j$.

If $t_0 < \lfloor \log_2(N+1) \rfloor$, then Lemma 1 guarantees that all chunk schedulers are identical under ELp peer scheduling.

If $t_0 \geq \lfloor \log_2(N+1) \rfloor$, we have from the hypotheses that $\forall t < t_0$ the schedules produced by Dl/ELp and LUc/ELp are identical. By definition at time t_0 in P_i LUc/ELp choses

$$C_k \in \mathcal{C}'(P_i, t_0) : \forall C_h \in \mathcal{C}'(P_i, t_0) r_k \geq r_h.$$

Since the source only produces a single chunk at every time unit, r_k and r_h cannot have the same value, hence $r_k > r_h$. To obtain a different schedule Dl/ELp must choose $C_j \neq C_k$.

Since for $t < t_0$ Dl/ELp produced the same schedule as LUc/ELp, C_j and C_k have been transmitted $t_0 - r_j$ and $t_0 - r_k$ times respectively (see Observation 2); hence, $d_i = r_i + 2(t_0 - r_i)$ for both C_j and C_k.

Since Dl/ELp chooses $C_j \in \mathcal{C}'(P_i, t_0) : \forall C_h \in \mathcal{C}'(P_i, t_0) d_j \leq d_h$ we have $d_j \leq d_k \Rightarrow r_j + 2(t_0 - r_j) \leq r_k + 2(t_0 - r_k) \Rightarrow -r_j \leq -r_k \Rightarrow r_j \geq r_k$ which contradicts the hypothesis $r_k > r_j$.

Observation 3. *Note that the Dl scheduler postpones the scheduling deadline by two time units per transmission as $d_k = d_k + 2$. If a generic constant q was used instead of 2 and the scheduling deadline was postponed as $d_k = d_k + q$, then the last equation in the proof of Theorem 3 would have become*

$$r_j + q(t_0 - r_j) \leq r_k + q(t_0 - r_k) \Rightarrow (q-1)r_j \geq (q-1)r_k$$

which contradicts $r_k > r_j$ if $q > 1$. Hence, if a generic constant $q > 1$ is used to postpone the scheduling deadline, then Dl/ELp is still equivalent to LUc/ELp. In this sense, Dl can be seen as a whole class of deadline-based algorithms.

5 Neighborhood Restriction and Selected Results

Although both LUc/ELp and Dl/ELp have been proved to provide optimal performance in the case of a fully connected graph their performance in more realistic situations is still unclear. Besides these two algorithms we consider various combinations with LUc, RUc and RUp algorithms for comparison.

5.1 Simulating P2P Streaming and Measuring Performance

The behavior of the scheduling algorithms introduced in Section 3 is analyzed using the SSSim simulator [9], by setting up an overlay of N peers with unit upload and infinite download bandwidth. The source distributes M_c chunks.

As explained in Section 2 the performance metric considered in this paper is the worst case diffusion time F, and (as stated in Section 3), a scheduling algorithm is optimal iff $F = \lceil log_2(N) \rceil + 1$.

First of all, the algorithms have been simulated on a fully connected graph, as shown in Figure 2. In accordance with Theorems 2 and 3, LUc/ELp and Dl/ELp achieve optimal performance, outperforming the other algorithms (in particular, RUc/ELp achieves a value of f_i near the double of the optimal, and all the other algorithms achieved even worse performance).

5.2 Restricting the Overlay

In realistic situations a restricted overlay is used instead of a fully connected graph. Such a restricted overlay is modeled assuming bidirectional relations and

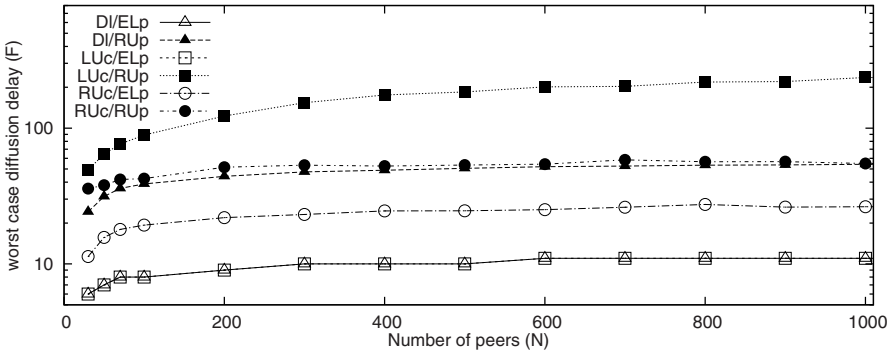

Fig. 2. Full mesh overlay; maximum diffusion delay as a function of N; 500 chunks

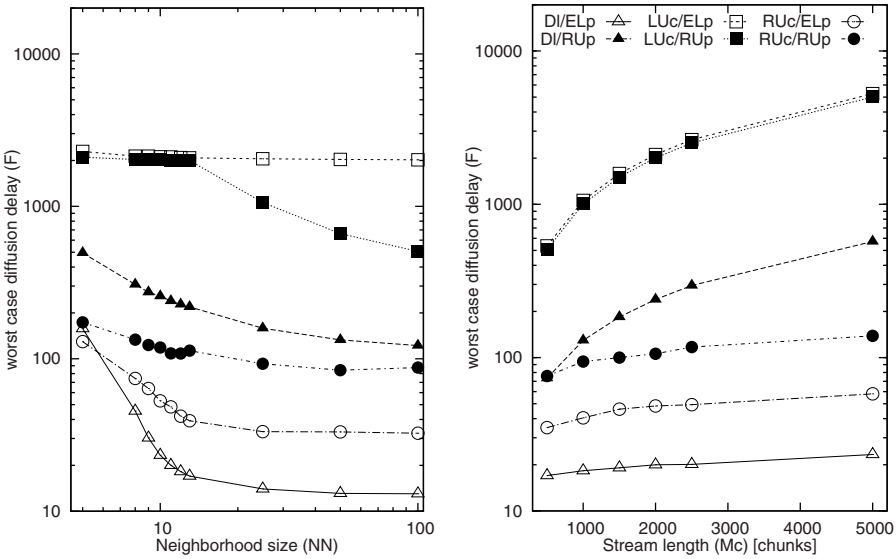

Fig. 3. Worst case chunk diffusion delay of algorithms, with 1000 peers, as a function of: (left) neighborhood size, with 2000 chunks; (right) number of chunks, with $\mathcal{NN} = 11$

a pre-defined number ($\mathcal{NN} = ||\mathcal{N}_i||$) of neighbor nodes. The resulting graph is a random $\mathcal{NN}$-regular graph. In the following simulations, the algorithms are evaluated on 10 instances of the random $\mathcal{NN}$-regular graph. We have verified that confidence intervals were always within 5% of the reported mean values with a confidence level of 90%.

The left hand side of Figure 3 shows performance of different streaming algorithms as a function of $\mathcal{NN}$ and shows how the LUc/ELp algorithm (which is optimal on a full mesh) is highly sensitive to neighborhood restrictions and performs badly when $\mathcal{NN} < N - 1$. Dl/ELp, on the other hand, works better than all the

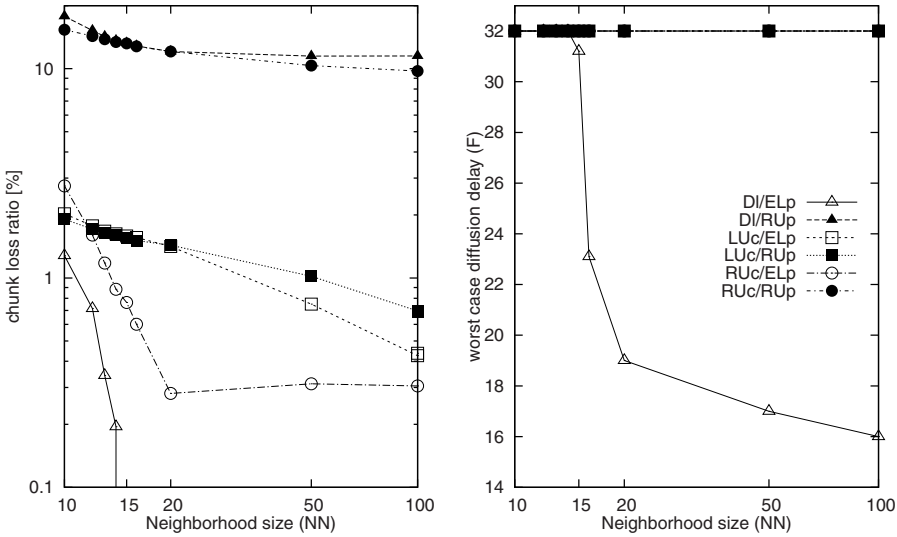

Fig. 4. Chunk loss and F as a function of the neighborhood size ($N = 10000, D = 32$)

other algorithms and is able to achieve values of F near the optimum (which in this case is 11). The right side of Figure 3 shows how the number of chunks affects F for $NN = 11$ (note that $\log_2(N) = 9.9658$). Dl/ELp keeps good performance even for long streams, while for several other algorithms f_i increases with i (the performance of the algorithm depends on the stream length), hence these distribution mechanism results to be unstable in a streaming context.

5.3 Limiting the Chunk Buffer Size

The only solution to the instability problem is to define a playout delay D, and to discard chunks C_j at time $r_j + D$. This causes some chunk loss (for chunks C_j that would have $f_i > D$), but can make the distribution system stable again. Moreover, the playout delay D can be used to dimension the chunk buffers in the peers (in particular, each peer needs to buffer at most D chunks)[3].

Since some chunks can be lost, the performance should be evaluated based on both chunk loss ratio and the maximum delay. Figure 4 plots the chunk loss ratio (left) for the various algorithms as a function of the neighborhood size with $D = 32$. Note that for $NN > 14$ the chunk loss ratio for Dl/ELp is 0, showing that it is possible to dimension the chunk buffer size so that it does not affect the algorithm's performance (to the authors' best knowledge, this is not possible for the other algorithms). The worst case diffusion time F (right) fastly approaches the optimum with Dl/ELp, while it is obviously 32 for all other algorithms.

[3] Implementing the *chunks buffer size* in the simulator can enable optimizations which allow to simulate larger task sets, hence we move to $N = 10000$.

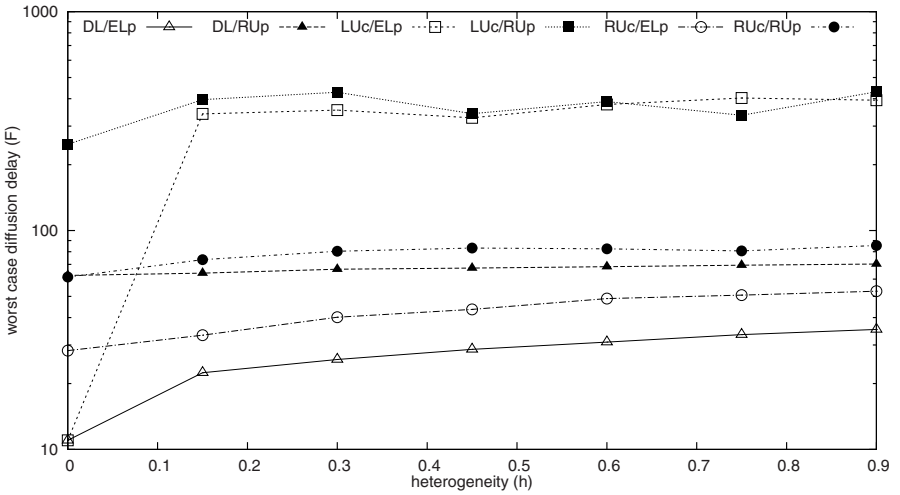

Fig. 5. F as a function of bandwidth heterogeneity ($N = 600$, $M_c = 600$)

5.4 Heterogeneous Upload Bandwidth

Finally, we evaluate the performance of Dl/ELp in heterogeneous networks. We use a scenario similar to that of [6]. The system is composed of N=600 nodes, devided in 3 classes based on their upload bandwidth: bandwidth of 2 for $(h/3)N$ nodes[4], bandwidth of 0.5 for $(2h/3)N$ nodes, and unit bandwidth for $(1-h)N$ nodes, thus keeping the mean bandwidth at 1. We vary h from 0 (homogeneous case) to 1.

Figure 5, plotting the diffusion delays for $M_c = 600$ chunks and an infinite buffer size, shows that Dl/ELp performs better in this specific setting than the other algorithms studied for the whole range of h. These initial studies indicate that Dl/ELp could be a strong contender also in heterogeneous settings. We leave more detailed studies, including studies of the effect of Dl's increment parameter on performance (see observation 3), for future work.

6 Related Work and Contributions

Optimality of schedulers has been extensively studied in the literature. For the case of full mesh overlay and unit upload bandwidth limits, the generic (i.e., valid for any scheduler) lower bound of $(\lceil \log_2(N) \rceil + 1)T$ is well known. [1] proves that this bound is strict in a streaming scenario by showing the existence of a *centralized* scheduler that achieves such bound. A similar proof (although for the case of file dissemination) can be found in [11]. Our work improves on these results by proving

[4] In order to validate our results with heterogeneous bandwidth, we implemented our algorithms also in the P2PTVSim [10] simulator. For this reason, we had to use a smaller number of peers and chunks.

the existence of *distributed* schedulers (LUc/ELp and Dl/ELp) that achieve the same strict bound.

Generic upper bounds as well as upper bounds on the distribution times achieved by different distributed schedulers can also be found in literature. The fundamental work of [12] studies asymptotic properties of distributed gossiping algorithms in a similar setting, showing an upper bound for any pull based algorithm of all the messages in $\mathcal{O}(M_c + log(N))$ time with high probability even for blind algorithms. Generic asymptotic bounds are also shown for blind push based algorithms, although in this case full dissemination cannot be guaranteed. A blind algorithm that distributes chunks with a high probability in $(9{*}M_c{+}9{*}\log_2(N))T$ is also shown. Note that this suggest a distribution delay for the individual chunk that grows with M_c.

Authors of [11] also evaluate blind distributed strategies in the case of file distribution, showing distribution delays dependent on the number of chunks.

[6] studies upper bounds for specific well known algorithms, showing that the combination of random peer selection and LUc achieves asymptotically good delays, however this demonstration is provided in the case of upload bandwidth higher than 1.

The distributed LUc/ELp and Dl/ELp schedulers presented in our paper perform significantly better than the generic upper bounds shown in [12] and [11] in that it achieves full diffusion of all chunks in $(M_c + \lceil log_2(N) \rceil)T$, i.e. a chunk diffusion delay independent of M_c.

It also differs from the streaming algorithms studied in [6], since for LUc/ELp and Dl/ELp this strict delay bound holds for any N(not just asymptotically), and it is valid even in the boundary case of unit upload bandwidth, without relying on redundant source coding.

[6] uses ER graphs to model the restricted neighborhood. With $N = 600$ and $\overline{NN} = 10$, authors find that the studied algorithms suffer significant losses. These chunk losses are confirmed by our results (even if our random graph model is slightly different) for the algorithms considered therein. However, we also show (through simulations) that the new Dl/ELp algorithm performs near the optimum with any M_c and any N, even with significant overlay restrictions. Namely, reducing the neighborhood size to any $NN >= \lceil log_2(N) \rceil$, our algorithm keeps distributing all chunks with a delay only slightly above the lower bound and always (on all simulated NN-regular random graphs) below $2 * (\lceil log_2(N) \rceil + 1)T$. Note that the neighborhood of $\lceil log_2(N) \rceil$ practically means less than 30 in any reasonable setting.

7 Conclusions and Future Work

This paper presented the formal proof that *distributed* algorithms can achieve optimal diffusion for streaming applications in unstructured meshes. The paper introduced a class of deadline based algorithms Dl/ELp which are optimal in full meshes and maintain very good properties also in realistic scenarios with small neighborhoods.

Future work includes on the one hand extending the theoretical results to scenarios with different constraints, including large bandwidth and heterogeneous scenarios, and, on the other hand, exploiting these algorithms to implement real P2P streaming systems.

References

1. Liu, Y.: On the minimum delay peer-to-peer video streaming: how realtime can it be? In: MULTIMEDIA 2007: Proceedings of the 15th international conference on Multimedia, Augsburg, Germany, pp. 127–136. ACM Press, New York (2007)
2. Hefeeda, M., Habib, A., Xu, D., Bhargava, B., Botev, B.: Collectcast: A peer-to-peer service for media streaming. In: ACM Multimedia 2003, vol. 11, pp. 68–81 (2003)
3. Hei, X., Liang, C., Liang, J., Liu, Y., Ross, K.W.: Insights into pplive: A measurement study of a large-scale p2p iptv system. In: Proceedings of the Workshop on Internet Protocol TV (IPTV) services over World Wide Web in conjunction with WWW 2006 (2006)
4. Chu, Y., Ganjam, A., Ng, T.S.E., Rao, S.G., Sripanidkulchai, K., Zhan, J., Zhang, H.: Early experience with an internet broadcast system based on overlay multicast. In: ATEC 2004: Proceedings of the annual conference on USENIX Annual Technical Conference, Boston, MA, June 2004, USENIX Association (2004)
5. Pianese, F., Keller, J., Biersack, E.W.: Pulse, a flexible p2p live streaming system. In: IEEE INFOCOM (2006)
6. Bonald, T., Massoulié, L., Mathieu, F., Perino, D., Twigg, A.: Epidemic live streaming: optimal performance trade-offs. In: Liu, Z., Misra, V., Shenoy, P.J. (eds.) SIGMETRICS, Annapolis, Maryland, USA, pp. 325–336. ACM, New York (2008)
7. Couto da Silva, A., Leonardi, E., Mellia, M., Meo, M.: A bandwidth-aware scheduling strategy for p2p-tv systems. In: Proceedings of the 8th International Conference on Peer-to-Peer Computing 2008 (P2P 2008), Aachen (September 2008)
8. Liu, C.L., Layland, J.: Scheduling alghorithms for multiprogramming in a hard real-time environment. Journal of the ACM 20(1) (1973)
9. Abeni, L., Kiraly, C., Cigno, R.L.: TR-DISI-08-074: SSSim: Simple and Scalable Simulator for P2P streaming systems. Technical report, University of Trento (2008), `http://disi.unitn.it/locigno/preprints/TR-DISI-08-074.pdf`
10. The NAPA-WINE Project: P2PTVSim home page, `http://www.napa-wine.eu/cgi-bin/twiki/view/Public/P2PTVSim`
11. Mundinger, J., Weber, R., Weiss, G.: Optimal scheduling of peer-to-peer file dissemination. J. of Scheduling 11(2), 105–120 (2008)
12. Sanghavi, S., Hajek, B., Massoulié, L.: Gossiping with multiple messages. In: Proceedings of IEEE INFOCOM 2007, Anchorage, Alaska, USA, May 2007, pp. 2135–2143 (2007)