# Optimal Receiver Scheduling Algorithms for a Multicast Problem

A.A. Bertossi* M.C. Pinotti† R. Rizzi‡

**Abstract**

This paper studies a multicast problem arising in wavelength division multiplexing single-hop lightwave networks as well as in Video-on-Demand systems. In this problem, the same message of duration $\Delta$ has to be transmitted to a set of $n$ receivers which are not all available simultaneously. The receivers can be partitioned into subsets, each served by a different transmission, with the objective of minimizing their overall waiting cost. When there is a single data channel available for transmission, a dynamic programming algorithm is devised which finds an optimal solution in $O(n \log n + \min\{n^2, n\Delta^2\})$ time, improving over a previously known $O(n^3)$ time algorithm. When multiple data channels are available for transmission, an optimal $O(n)$ time algorithm is proposed which finds an optimal solution if the message has constant transmission duration, whereas an NP-completeness proof is given if the message has arbitrary transmission duration.

**Keywords:** multicast algorithm, dynamic programming, batch-processing machine scheduling, wavelength division multiplexing lightwave networks, receiver scheduling, Video-on-Demand systems.

## 1 Introduction

Multicasting transmission is a one-to-many communication primitive where one single transmitter has to send the same message, of fixed duration, to many destinations (or receivers). A message transmission is simultaneously received in a single hop by all the destinations which are available for reception. An obvious solution requires a single transmission which is delayed until all the receivers are available. Such a solution, however, may lead to inefficient use of the resources because all the receivers have to wait the receiver which becomes available the last. To overcome such a drawback, the transmitter can schedule many transmissions each serving those receivers which became available after the previous transmission started. Assuming that the availability times of all the receivers are known in advance, the problem then becomes that of partitioning the receivers into subsets, each served by a different transmission, so that no two transmissions overlap and the overall waiting cost of the receivers is minimized.

Such a multicast problem has been first studied in the context of wavelength division multiplexing single-hop lightwave networks. An example of such networks is the Passive Star Coupler which has one reserved channel employed as a control channel along with several channels used for data broadcasting. Roughly speaking, each node is equipped with a fixed transceiver tuned

*Department of Computer Science, University of Bologna, Mura Anteo Zamboni, 7, 40127 Bologna, ITALY, bertossi@cs.unibo.it, http://www.cs.unibo.it/∼bertossi

†Department of Computer Science and Mathematics, University of Perugia, Via Vanvitelli, 1, 06123 Perugia, ITALY, pinotti@unipg.it, http://www1.isti.cnr.it/∼pinotti

‡Department of Mathematics and Computer Science, University of Udine, Via delle Scienze, 208, 33100 Udine, ITALY, rrizzi@dimi.uniud.it, http://users.dimi.uniud.it/∼romeo.rizzi

on the control channel and a tunable transceiver which can be tuned on any data channel. When a message has to be multicast from a transmitter node to the other network nodes, the transmitter selects a single data channel which is available, partitions the receivers into subsets according to their availability times known through the control channel, informs the receivers of the multicast schedule also using the control channel, and finally performs the scheduled transmissions on the selected data channel [4, 3].

The above mentioned multicast problem can be viewed also as batch machine scheduling problem, where there is a single batch-processing machine which is able to simultaneously process a group of jobs. The problem then consists in grouping the dynamically arriving jobs into non-preemptive batches so as to optimize a properly defined objective function [6]. Unlike the batch-processing problem arising in productive contexts, where the machine has a capacity constraint which limits the size of each batch, the multicasting problem is an incapacitated one [5] since the machine can process a batch of arbitrary size.

Another timely application of the just defined multicast problem arises in Video-on-Demand (VoD) services, which enable customers at home to choose a movie to watch at any time they wish via public communication network. Since dedicating a transmission stream for each viewer is not possible due to the limited available bandwidth, the customer requests are grouped together using batching. Then, each batch is served by a transmission stream broadcast so as to minimize the customer average waiting time. According to the taxonomy of [2], our multicast problem models a VoD system with a hybrid batching policy of service, where each customer does a request as in a *user-centered* mode and he waits until the movie is broadcast as in a *data-centered* mode.

So far, the multicast problem considered in this paper has been studied only in [4, 5]. In particular, Jue and Mukerjee [4] first formalized the problem as a combinatorial optimization one, while Sung and Rim [5] later proposed a dynamic programming off-line algorithm which finds an optimal multicast schedule in $O(n^3)$ time, where $n$ is the number of receivers. Moreover, several on-line and off-line heuristics have also been proposed in [4, 5] which find sub-optimal solutions. Such heuristics either are faster than $O(n^3)$ or attack some variants of the problem, and the goodness of their sub-optimal solutions has been experimentally evaluated through simulations. In this paper, we continue the study of the off-line multicast problem started in [4, 5]. Referring to the VoD application mentioned above, the off-line setting can be justified considering a VoD system with booking that allow customers to indicate in advance their availability time for viewing the desired movie. Once the schedule has been found, the server communicates to the customers the actual start time of the video stream. Specifically, we first show a new faster dynamic programming algorithm for finding an optimal multicast schedule when there is a single data channel for transmission, and then we deal also with the most general case where multiple data channels are available for message transmissions. Such a generalization, which allows the simultaneous overlapping of as many message transmissions as the number of data channels, is considered here for the first time, at the best of our knowledge.

The rest of this paper is organized as follows. Section 2 formally defines the problem, while Section 3 shows useful properties that hold for optimal multicast schedules. Section 4 is the core of the paper and devises a dynamic programming algorithm which finds an optimal solution in $O(n \log n + \min\{n^2, n\Delta^2\})$ time, improving over the previously known $O(n^3)$ time algorithm, where $\Delta$ is the duration of the message transmission. Section 5 considers the case where more than a single data channel is used for multicasting: when $\Delta$ is a constant, an optimal $O(n)$ time algorithm is proposed, whereas when $\Delta$ is arbitrary, an NP-completeness proof is given. Finally, conclusions are offered in Section 6.

## 2 Multicast problem definition

This section will formally define the multicast problem when there is a single data channel available for transmission.

**Problem instance.** A problem instance can be identified by the following parameter set:

- a positive integer $n$ denoting the number of receivers;
- $n$ distinct integers $t_1 < \cdots < t_n$ giving the receiver availability times;
- a positive integer $\Delta$ indicating the duration of a transmission;
- $n$ positive reals $c_1, \ldots, c_n$ providing the waiting costs per time unit of the receivers.

A problem instance will be concisely denoted by $\langle \Delta; (t_1, c_1), \ldots, (t_n, c_n) \rangle$. Note that the assumption of integer availability times and transmission duration is not restrictive. Indeed, in the VoD application mentioned above the movie duration is typically measured in minutes and hence a more accurate precision for the availability times is useless. However, an extension to real availability times and transmission duration will be dealt with at the end of Section 4.

**Feasible solutions.** A feasible solution of the problem is a multicast schedule, that is a subset $S \subset \mathbb{N}$ such that

$$\forall s, u \in S \qquad s \neq u \rightarrow |s - u| \geq \Delta \tag{1}$$
$$\forall s \in S \qquad s \geq t_1 \tag{2}$$
$$\forall s, u \in S \qquad s < u \rightarrow (\exists i \in \{1, \ldots, n\} | \quad s < t_i \leq u) \tag{3}$$
$$\exists s \in S \qquad s \geq t_n \tag{4}$$

Each element of $S$ is the starting time of a single transmission. Constraint (1) requires that the minimum separation between two consecutive transmissions is at least $\Delta$, so that no two transmissions overlap. Constraints (2) and (3) ensure that only useful transmissions are scheduled, that is, each transmission satisfies at least one receiver. Finally, Constraint (4) guarantees that all the receivers are served. By the above constraints, note that the number of transmissions of each feasible schedule ranges between 1 and $n$.

**Objective.** Given a feasible schedule $S$, let

$$w_i(S) = \min\{s - t_i | s \in S, s \geq t_i\}, \qquad i = 1, \ldots, n \tag{5}$$

be the waiting time of receiver $i$, which clearly gives the time elapsing from the availability instant of receiver $i$ to the starting time of the first subsequent transmission.

The goal is to find a feasible schedule $\hat{S} \subset \mathbb{N}$ that minimizes the total waiting cost of all the receivers

$$W(\hat{S}) = \sum_{i=1}^{n} c_i w_i(\hat{S}). \tag{6}$$

Note that no cost is associated with transmissions, and hence there might be distinct optimal solutions with the same waiting cost but with a different number of transmissions. Moreover, although the receiver availability times have been assumed to be distinct, the most general case with possibly coincident availability times, that is $t_1 \leq \cdots \leq t_n$, can be reduced to the case of

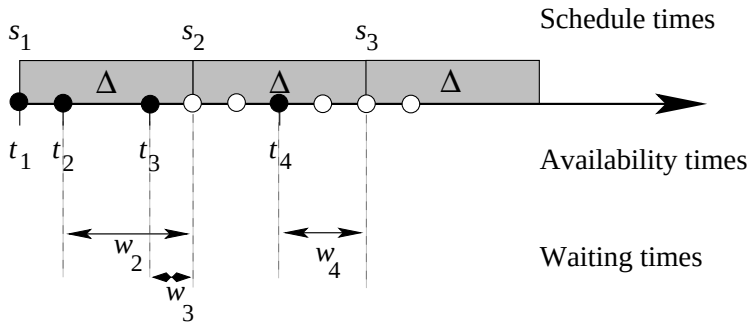


Figure 1: An instance along with a feasible schedule.

distinct times by collapsing all the receivers with the same availability time into a single receiver whose waiting cost is given by the sum of the waiting costs of the coincident receivers.

As an example, an instance with $n = 4$ receivers, whose availability times are $t_1 = 0$, $t_2 = 1$, $t_3 = 3$, and $t_4 = 6$, and with transmission duration $\Delta = 4$ is depicted in Figure 1, along with a feasible schedule.

## 3 Fundamental properties

In order to devise efficient algorithms for the problem, let us prove some features of an optimal schedule.

Let $S$ be a feasible schedule of cardinality $\ell \leq n$ and assume that the elements of $S$ are sorted. By Constraint (1), $S = \{s_1, \ldots, s_\ell\}$, where $s_{j+1} \geq s_j + \Delta$, for $j = 1, \ldots, \ell - 1$. We say that receiver $i$ is served by transmission $j$, or equivalently, when there is no ambiguity, that $t_i$ is served by $s_j$, if transmission $j$ is the first one to start at an instant $s_j$ greater than or equal to $t_i$, that is the availability time of receiver $i$. Formally, $t_i$ is served by $s_1$ if $t_i \leq s_1$, while it is served by $s_j$, with $j > 1$, if $s_{j-1} < t_i \leq s_j$. When $t_i$ is served by $s_j$, it is clear by (5) that $w_i(S) = s_j - t_i$.

**Lemma 3.1.** *Let $\hat{S} = \{s_1, \ldots, s_\ell\}$ be an optimal schedule and let $t_i$ be served by $s_j$. Then, $s_j - t_i < 2\Delta$.*

*Proof.* Suppose by contradiction that $s_j - t_i \geq 2\Delta$. Observe that in $\hat{S}$ a previous transmission must finish before $t_i + \Delta$, and there is no transmission starting in the interval $[t_i + \Delta, s_j)$. Therefore, one can obtain a new schedule $S = \hat{S} \cup \{s'\}$ by introducing a new transmission starting at time $s' = t_i + \Delta$. In this way, $\forall\ h \neq i$, one has that $w_h(S) \leq w_h(\hat{S})$, whereas $w_i(S) = s' - t_i < s_j - t_i = w_i(\hat{S})$, which contradicts the optimality of $\hat{S}$. □

**Lemma 3.2.** *Let $\hat{S} = \{s_1, \ldots, s_\ell\}$ be an optimal schedule and assume there are $t_i$ and $t_{i+1}$ with $t_{i+1} - t_i \geq 2\Delta$. Then, $t_i$ and $t_{i+1}$ are served by two distinct transmissions $s_j$ and $s_{j+1}$ such that $s_j + \Delta \leq t_{i+1}$.*

*Proof.* By Lemma 3.1, $t_i$ and $t_{i+1}$ cannot be served by the same $s_j$ with $s_j \geq t_{i+1}$. Therefore, $t_i$ and $t_{i+1}$ are served, respectively, by $s_j$ and $s_{j+1}$, which are distinct and consecutive by Constraint (3).

Since a previous transmission must finish before $t_i + \Delta$ and there are no further available receivers in the time interval $(t_i, t_{i+1})$, it holds $s_j \leq t_i + \Delta$. Hence, $s_j + \Delta \leq t_i + 2\Delta \leq t_{i+1}$. □

Lemma 3.2 implies that, whenever $t_{i+1} - t_i \geq 2\Delta$, the instance $\langle \Delta; (t_1, c_1), \ldots, (t_n, c_n) \rangle$ can be split into two instances $\langle \Delta; (t_1, c_1), \ldots, (t_i, c_i) \rangle$ and $\langle \Delta; (t_{i+1}, c_{i+1}), \ldots, (t_n, c_n) \rangle$ which can be

solved independently. Therefore, from now on, any two consecutive receiver availability times $t_i$ and $t_{i+1}$ are assumed to satisfy $t_{i+1} - t_i < 2\Delta$.

**Lemma 3.3.** *Let $\hat{S} = \{s_1, \ldots, s_\ell\}$ be an optimal schedule. Then, for each $j = 1, \ldots, \ell$, one of the two following assertions is true:*

1. *either $s_j = t_i$ for some $i = 1, \ldots, n$, or*
2. *$j > 1$ and $s_j = s_{j-1} + \Delta$.*

*Proof.* By contradiction, let $\hat{S}$ be an optimal schedule and let $\bar{\jmath}$ be the smallest index such that neither assertion is true. Consider the time

$$t = \begin{cases} max\{\{t_i | i = 1, \ldots, n; t_i \leq s_{\bar{\jmath}}\} \cup \{s_{\bar{\jmath}-1} + \Delta\}\} & \text{if } \bar{\jmath} > 1 \\ max\{t_i | i = 1, \ldots, n; t_i \leq s_{\bar{\jmath}}\} & \text{if } \bar{\jmath} = 1 \end{cases} \tag{7}$$

Since $\hat{S}$ is an optimal schedule, it is also feasible, and thus Constraint (2) is satisfied and the value of $t$ in Equation (7) exists. By hypothesis $\bar{\jmath}$ does not satisfy either assertion, and thus $t < s_{\bar{\jmath}}$. Let $S = \hat{S} \setminus \{s_{\bar{\jmath}}\} \cup \{t\}$ be the schedule obtained by anticipating the $\bar{\jmath}$-th transmission at time $t$. By Constraint (3), if $\bar{\jmath} > 1$ then $s_{\bar{\jmath}-1} \leq t - \Delta$, and hence schedule $S$ is also feasible. Let $I$ be the set of receivers served by transmission $\bar{\jmath}$ in schedule $\hat{S}$. By Equation (7), all the receivers in $I$ are served at time $t$ in the modified schedule $S$. Therefore, the waiting times of the receivers become:

$$\begin{cases} w_i(S) = t - t_i < s_{\bar{\jmath}} - t_i = w_i(\hat{S}) & \text{if } i \in I \\ w_i(S) = w_i(\hat{S}) & \text{otherwise.} \end{cases}$$

Since $w_i(S)$ is strictly less than $w_i(\hat{S})$ when $i \in I$, and all the waiting costs in the objective function (6) are positive, one derives $W(S) < W(\hat{S})$, contradicting the optimality of $\hat{S}$. □

**Theorem 3.4.** *Let $\hat{S} = \{s_1, \ldots, s_\ell\}$ be an optimal schedule. Then, for each $j = 1, \ldots, \ell$, $s_j$ can be written as*

$$s_j = t_i + k\Delta$$

*where $i$ and $k$ are integers such that $1 \leq i \leq n$ and $0 \leq k \leq \ell - 1$.*

*Proof.* The proof is by induction on $j$. The basis of the induction is true because $s_1 = t_i$ for some $i$ by Lemma 3.3. Let $j > 1$ and suppose by inductive hypothesis that $s_{j-1} = t_{i'} + k\Delta$ for some $i' < n$ and $k < j - 1$. If $s_j = t_i$ for some $i$, then the thesis holds true. Otherwise, by Lemma 3.3 and by inductive hypothesis, $s_j = s_{j-1} + \Delta = t_{i'} + (k+1)\Delta$, where $k + 1 < j$. Since $j \leq \ell \leq n$, the proof follows. □

An important consequence of Theorem 3.4 is that all the transmission times can be chosen among a small set of candidates, each being either a receiver availability time $t_i$ or one of its *echos*, that is an instant that differs from $t_i$ by a multiple of $\Delta$.

It is worth to note that all the results shown in the present section hold true also when all the availability times and the transmission duration assume real values.

# 4 Improved polynomial algorithm

In this section, a dynamic programming approach is proposed which finds an optimal multicast schedule in $O(n \log n + \min\{n^2, n\Delta^2\})$ time, assuming that both the availability times and the transmission duration are integers, thus improving over the $O(n^3)$ algorithm previously proposed in [5].

The algorithm finds an optimal multicast schedule by selecting the transmission starting times among a superset $P$ of candidates, which includes all the receiver availability times along with all their echos. Precisely,

$$P = \{t_i + k_i\Delta | 1 \le i \le n, 0 \le k_i \le \bar{k}_i, t_i + (\bar{k}_i - 1)\Delta \le t_n < t_i + \bar{k}_i\Delta\} \tag{8}$$

As an example, Figure 1 depicts the candidates in $P$ as small circles, where black circles correspond to availability times and white circles represent their echos. Recalling that $t_{i+1} - t_i < 2\Delta$ by assumption and observing that at most $2i$ echos can appear in $[t_i, t_{i+1})$, it follows that the cardinality $m$ of $P$ is bounded from above by $O(\min\{n\Delta, n^2\})$.

For each receiver availability time $t_i$, let $n_i$ be the number of candidates in $P$ which belong to the interval $[t_i, t_i + 2\Delta)$. Moreover, let $s_1^i < s_2^i < \cdots < s_{n_i}^i$ be the ordered set of such candidates. Note that $s_1^i = t_i$, but a generic candidate $s_h^i$, with $2 \le h \le n_i$, can be either a receiver availability time or an echo. Referring again to Figure 1, consider $i = 2$. Then $n_2 = 7$ and $s_1^2 = t_2 = 1$, $s_2^2 = t_3 = 3$, $s_3^2 = t_1 + \Delta = 4$, $s_4^2 = t_2 + \Delta = 5$, $s_5^2 = t_4 = 6$, $s_6^2 = t_3 + \Delta = 7$, and $s_7^2 = t_1 + 2\Delta = 8$. Note also that $n_i \le 2n$ because in the interval $[t_i, t_i + 2\Delta)$ there are at most $2i$ echos of $t_1, \ldots, t_i$ and at most $2(n - i)$ receiver availability times $t_{i+1}, \ldots, t_n$ and their echos. Since there are $m$ candidates and each candidate can appear as an $s_h^i$ for at most $2\Delta$ distinct indices $i$, one has that $\sum_{i=1}^{n} n_i = O(\min\{n^2, m\Delta\}) = O(\min\{n^2, n\Delta^2\})$.

For each $i = 1, 2, \ldots, n$ and each $h = 1, 2, \ldots, n_i$, let

$A[i,h] \triangleq$ the cost of an optimal solution for the subproblem $\langle \Delta; (t_i, c_i), \ldots, (t_n, c_n) \rangle$ assuming that the first transmission starts exactly at time $s_h^i$

$B[i,h] \triangleq$ the cost of an optimal solution for the same subproblem but where the first transmission can start at any time greater than or equal to $s_h^i$

Clearly, by definition, $B[1,1]$ is the cost of an optimal solution for the original instance $\langle \Delta; (t_1, c_1), \ldots, (t_n, c_n) \rangle$. Fixed $i$, $t_i$ can be served only by a transmission starting earlier than $t_i + 2\Delta$ by Lemma 3.1 and the time candidates for such a transmission in the interval $[t_i, t_i + 2\Delta)$ are $s_1^i, \ldots, s_{n_i}^i$ by Theorem 3.4. Thus, $B[i,h] = \min\{A[i,h], A[i,h+1], \ldots, A[i,n_i]\}$.

On the other hand, $A[i,h]$ assumes that $t_i$ is served by a transmission starting exactly at time $s_h^i$, and thus finishing at time $s_h^i + \Delta$. Since such a transmission does not satisfy any receiver with availability time $t_\ell > s_h^i$, the computation of $A[i,h]$ depends on the optimal solution of $\langle \Delta; (t_{i'}, c_{i'}), \ldots, (t_n, c_n) \rangle$, where $i' = \min\{\ell : t_\ell > s_h^i\}$ (if such a set is empty, let $i' = n + 1$). Since the next transmission cannot start earlier than $\max\{s_h^i + \Delta, t_{i'}\}$, the optimal solution for $\langle \Delta; (t_{i'}, c_{i'}), \ldots, (t_n, c_n) \rangle$ is given by $B[i', h']$ where $h' = 1$ if $i' = n + 1$ or $t_{i'} \ge s_h^i + \Delta$, while $h'$ satisfies $s_{h'}^{i'} = s_h^i + \Delta$, if $t_{i'} \in (s_h^i, s_h^i + \Delta)$.

In conclusion, for $i = n, n-1, \ldots, 1$ and $1 \le h \le n_i$, one has:

$$A[i,h] = \sum_{t_i \le t_\ell \le s_h^i} c_\ell (s_h^i - t_\ell) + B[i', h'] \tag{9}$$

while the recurrence of $B[i,h]$, for $i = n, n-1, \ldots, 1$ and $h = n_i, n_i - 1, \ldots, 1$, is:

$$B[i,h] = \begin{cases} 0 & \text{if } i = n+1, h = 1 \\ A[i,h] & \text{if } 1 \le i \le n, h = n_i \\ \min\{A[i,h], B[i,h+1]\} & \text{if } 1 \le i \le n, 1 \le h < n_i \end{cases} \tag{10}$$

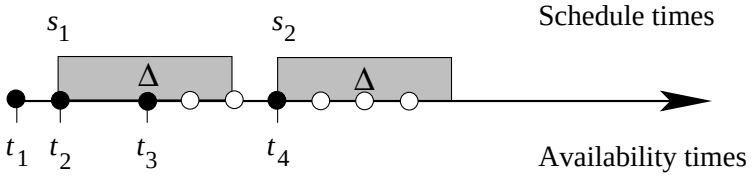


Figure 2: An optimal schedule for the instance of Figure 1.

Note that, fixed $i$, the $B[i,h]$'s need to be calculated backwards. Moreover, the computations of arrays $B$ and $A$ have to be interleaved. Indeed, each single $B[i,h]$ depends on $A[i,h], A[i,h+1], \ldots, A[i,n_i]$, while each $A[i,h]$ depends on $B[i',h']$ with $i' > i$. So, the entries of $A$ must be filled by decreasing values of $i$. Following such an order, each $B[i,h]$ can be computed in $O(1)$ time. To compute also each $A[i,h]$ is $O(1)$ time, some data structures have to be introduced as it will be shown in the next subsection.

To compute the optimal multicast schedule, one has to keep track of the actual transmission times. This can be done by defining

$$T[i,h] = \begin{cases} \langle s_h^i, i', h' \rangle & \text{if } B[i,h] = A[i,h] \\ T[i,h+1] & \text{if } B[i,h] = B[i,h+1] \end{cases} \tag{11}$$

In this way, the schedule can be easily built starting from $T[1,1]$. Indeed, letting $T[1,1] = \langle \overline{s}, \overline{i}, \overline{h} \rangle$, the first transmission is taken at time instant $\overline{s}$ and hence the schedule is concatenated with that of $T[\overline{i}, \overline{h}]$, repeating until $\overline{i} = n+1$.

As an example, referring again to the problem instance of Figure 1, the resulting arrays $A$, $B$, and $T$ are given below, where the waiting costs per time unit are assumed to be $c_i = 1$, for $1 \leq i \leq 4$. The corresponding optimal schedule, obtained by tracking the bold entries of $T$, is depicted in Figure 2.

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| 1 | 6 | 4 | 6 | 10 | 14 | 14 | 18 |
| 2 | 3 | 3 | 6 | 9 | 8 | 11 | 14 |
| 3 | 1 | 3 | 5 | 3 | 5 | 7 | 9 |
| 4 | 0 | 1 | 2 | 3 | | | |

$A$

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| 1 | 4 | 4 | 6 | 10 | 14 | 14 | 18 |
| 2 | 3 | 3 | 6 | 8 | 8 | 11 | 14 |
| 3 | 1 | 3 | 3 | 3 | 5 | 7 | 9 |
| 4 | 0 | 1 | 2 | 3 | | | |
| 5 | 0 | | | | | | |

$B$

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| 1 | **1,3,3** | 1,3,3 | 3,4,2 | 4,4,3 | 6,5,1 | 6,5,1 | 7,5,1 |
| 2 | 1,3,3 | 3,4,2 | 4,4,3 | 6,5,1 | 6,5,1 | 7,5,1 | 8,5,1 |
| 3 | 6,4,2 | 6,5,1 | **6,5,1** | 6,5,1 | 7,5,1 | 8,5,1 | 9,5,1 |
| 4 | 6,5,1 | 7,5,1 | 8,5,1 | 9,5,1 | | | |

$T$

### 4.1 Efficient implementation

First of all, one needs to generate the set $P$ in sorted order in $O(m)$ time, without using an explicit sorting algorithm that would require $O(m \log m)$ time. For this aim, $P$ is implemented

```
Procedure GENERATE (⟨Δ; (t_1, c_1), ..., (t_n, c_n)⟩);
 1    p[1] := t_1; r[1] := 1; Q := ∅;
 2    INQUEUE (Q, ⟨t_1 + Δ; 1⟩);
 3    k := 1; i := 2;
 4    while Q ≠ ∅ do
 5         k := k + 1;
 6         ⟨time; pred⟩ := OUTQUEUE (Q);
 7         while i ≤ n and t_i < time do
 8              p[k] := t_i; r[k] := i;
 9              INQUEUE (Q, ⟨t_i + Δ; k⟩);
10              i := i + 1; k := k + 1;
11         p[k] := time; echo[pred] := k;
12         if i ≤ n and time = t_i
13              then r[k] := i
14              else r[k] := 0;
15         if p[k] ≤ t_n
16              then INQUEUE (Q, ⟨p[k] + Δ; k⟩)
17              else echo[k] := 0;
18    m := k; succ := 0;
19    for k := m downto 1 do
20         next[k] := succ;
21         if r[k] ≠ 0 then succ := k;
```

Figure 3: *Procedure for initializing the arrays p, r, echo, and next.*

as a sorted array $p[1], \ldots, p[m]$. For each $k$, $1 \leq k \leq m$, one has to maintain, besides the starting transmission time $p[k]$ and whether $p[k]$ is a receiver availability time or one echo, also the index $echo[k]$ of the next echo of $p[k]$, and the index $next[k]$ of the first receiver availability time following $p[k]$. Formally:

$$r[k] = \begin{cases} i & \text{if } \exists\ \ i \mid t_i = p[k],\ 1 \leq i \leq n \\ 0 & \text{otherwise} \end{cases}$$

$$echo[k] = \begin{cases} h & \text{if } \exists\ \ h \mid p[h] = p[k] + \Delta \\ 0 & \text{otherwise} \end{cases}$$

$$next[k] = \begin{cases} \ell & \text{if } \exists\ \ell = \min\{h | h > k, r[h] \neq 0\} \\ 0 & \text{otherwise} \end{cases}$$

Figure 3 illustrates the generation of the sorted array $p$, along with the initialization of the arrays $r$, *echo*, and *next*. Procedure GENERATE uses a `FIFO` queue $Q$ to maintain the echos and outputs in $p$ the merge of the availability times $t_1 < t_2 < \cdots < t_n$ and their echos.

In the figure, the first candidate $p[1]$ is set to the first availability time $t_1$, its first echo $t_1 + \Delta$ is inserted into $Q$, and the index of such an availability time is kept in $r[1]$, (lines 1–2). As long as $Q$ is not empty (line 4), the oldest echo (i.e., *time*) is extracted from $Q$ (line 6) so as to become a candidate in $p[k]$ (line 11). If the echo coincides with an availability time $t_i$ (line 12), the index $i$ of such an availability time is kept in $r[k]$ (line 13), and its next echo $p[k] + \Delta$ is inserted into $Q$, provided that it is no larger than the last availability time $t_n$ (lines 15–17). However, before doing these actions, each availability time $t_i$ smaller than the current echo becomes a candidate in $p$, its index is kept in $r$, and a new echo $t_i + \Delta$ is inserted into $Q$ (lines 8–9). Once $p$ is built, the array *next* is derived by scanning $r$ backwards (lines 19-21).

Since the outmost `while` (lines 4–17) generates in $O(1)$ time an entry of $p$ for each echo,

whereas the inmost `while` (lines 7–10) generates in $O(1)$ time an entry of $p$ for each receiver availability time, the overall time complexity of procedure GENERATE is $O(m)$.

To fill each entry $A[i,h]$ in Equation (9), one has to map each pair of indices $i$ and $h$ to the index $k$ of the corresponding candidate $p[k]$ such that $p[k] = s_h^i$. For this purpose, consider the array $b$ defined as follows:

$$b[i,h] = k + h - 1 \quad \text{if} \quad p[k] = t_i \quad 1 \le i \le n, 1 \le h \le n_i$$

Note that $b[i,1]$ can be set to $k$ in procedure GENERATE as soon as $i$ is assigned to $r[k]$. Then, for each $i$, all the $b[i,h]$'s, with $2 \le h \le n_i$, can be computed in $O(n_i)$ time by a scan of $p$ starting from $p[b[i,1]]$. Specifically, for each $h$, one assigns $b[i,1] + h - 1$ to $b[i,h]$ when $p[b[i,1] + h - 1]$ is less than $t_i + 2\Delta$. Clearly, the last value of $h$ gives the value of $n_i$.

To accomplish the entire dynamic programming scheme, one has to compute, in Equation (9), both indices $i'$ and $h'$ in $O(1)$ time. This can be done as follows:

$$i' = \begin{cases} r[next[b[i,h]]] & \text{if } next[b[i,h]] \neq 0 \\ n+1 & \text{otherwise} \end{cases}$$

$$h' = \begin{cases} echo[b[i,h]] - next[b[i,h]] + 1 & \text{if } i' \neq n+1 \text{ and } echo[b[i,h]] \geq next[b[i,h]] \\ 1 & \text{otherwise} \end{cases}$$

Finally, the sum of the waiting costs in Equation (9) has to be precomputed so that it can be retrieved in $O(1)$ time. This can be achieved by a prefix sum computation using two arrays *cost* and *prefix*, whose entries are defined below:

$$cost[i,h] = \begin{cases} 0 & \text{if } h = 1 \\ cost[i,h-1] & \text{if } r[b[i,h-1]] = 0 \text{ and } 2 \le h \le n_i \\ cost[i,h-1] + c_{r[b[i,h-1]]} & \text{if } r[b[i,h-1]] \neq 0 \text{ and } 2 \le h \le n_i \end{cases}$$

$$prefix[i,h] = \begin{cases} 0 & \text{if } h = 1 \\ prefix[i,h-1] + cost[i,h]\,(p[b[i,h]] - p[b[i,h-1]]) & \text{if } 2 \le h \le n_i \end{cases}$$

Note that $cost[i,h]$ accumulates the waiting costs of all the receivers which are available in the interval $[s_1^i, s_{h-1}^i]$, whereas $prefix[i,h]$ accumulates the product of such waiting costs and the width of the time intervals $[s_1^i, s_2^i], \ldots, [s_{h-1}^i, s_h^i]$. Overall, $O(\min\{n^2, n\Delta^2\})$ time is required to fill the $\sum_{i=1}^{n} n_i$ entries of the arrays *cost* and $prefix$.

Equation (9) can thus be rewritten as $A[i,h] = prefix[i,h] + B[i',h']$, and hence each $A[i,h]$ and $B[i,h]$ can be filled in $O(1)$ time. Therefore, since the number of filled entries of the arrays $A$ and $B$ is $1 + \sum_{i=1}^{n} n_i$, and the receiver availability times $t_1, \ldots, t_n$ need to be sorted, the overall time complexity of the algorithm is $O(n \log n + \min\{n^2, n\Delta^2\})$. It is worth noting that such a time is $O(n^2)$ only if $\Delta = \Omega(\sqrt{n})$, while it is $o(n^2)$ if $\Delta = o(\sqrt{n})$. In particular, it becomes $\Theta(n \log n)$ when $\Delta = o(\sqrt{\log n})$.

In the general case when all the $t_i$'s and $\Delta$ are real numbers, the algorithm remains correct, but its time complexity becomes $O(n^2)$ since the cardinality $m$ of $P$ is $O(n^2)$ and also $\sum_{i=1}^{n} n_i$ is $O(n^2)$, as one can easily check.

# 5 Multiple channels

This section will study the multicast problem when the receiver availability times are integers and there are multiple data channels available for transmissions, showing its NP-completeness in the general case of arbitrary transmission duration $\Delta$ and providing a polynomial time algorithm when the transmission duration $\Delta$ is a constant.

The problem instance is generalized adding a positive integer $K$ denoting the number of available multiple identical channels, and it is concisely denoted by $\langle \Delta, K; (t_1, c_1), \ldots, (t_n, c_n) \rangle$.

Observing that Constraints (3) and (4) imply that $s \leq t_n + \Delta - 1 \ \ \forall s \in S$, a feasible solution must satisfy the previous Constraints (2)-(4), whereas Constraint (1) is replaced by

$$\forall t \in [t_1, t_n + \Delta - 1] \ \nexists \ \ s_1, s_2, \ldots, s_{K+1} \in S \mid s_i \leq t < s_i + \Delta, \quad 1 \leq i \leq K+1 \tag{12}$$

Note that Constraint (12) allows at most $K$ transmissions to overlap at any time instant. Clearly, $K$ is assumed to be strictly less than $\Delta$, since otherwise the above constraint can be trivially satisfied by starting a transmission for every time instant $t$.

## 5.1 NP-completeness for arbitrary transmission duration

When the transmission duration $\Delta$ can be arbitrary, the multicast scheduling problem, in its decisional form, can be shown to be $NP$-complete by a reduction from the VERTEX COVER problem [1]: "Given an undirected graph $G = (V, E)$ and an integer $\phi$, is there a subset $X$ of at most $\phi$ vertices such that each edge has at least one endpoint in $X$?"

Let $G = (V, E)$ be an undirected graph where $V = \{v_1, \ldots, v_n\}$ and $E = \{e_1, \ldots, e_m\}$. We will show how to construct an instance of the multicast problem with $K = 4\,n + 1$ channels admitting a schedule of cost at most $m(9\,n - 1) + \phi$ if and only if $G$ has a vertex cover of size at most $\phi$. In this subsection, $M = 9\,mn$ plays the role of a conveniently large integer. We take $\Delta = n^2$ as the transmission duration. We introduce a set $R_i$ of $7\,(3\,m) + 1 = 21\,m + 1$ receivers for each vertex $v_i \in V$, and a set $R^j$ of 2 receivers for each edge $e_j \in E$. We also introduce a set of $m$ receivers $\overline{R}$, so that the overall set of receivers is thus $R = \bigcup_{i=1}^{n} R_i \cup \bigcup_{j=1}^{m} R^j \cup \overline{R}$. Each receiver in $R$ is represented by a pair $(t, c)$ where $t$ gives the receiver availability time and $c$ gives its cost per time unit. The value $c$ will however be a positive integer whose magnitude is polynomial in $m$ and $n$. Hence the NP-completeness result will also hold for the unweighted problem just replacing the pair $(t, c)$ with $c$ coincident receivers each having unit cost.

Let us now specify the above mentioned receivers. First, for $i = 1, \ldots, n$, we have

$$\begin{aligned} R_i = &\{(7i + j\Delta, M) \mid j = 0, 1, \ldots, 3\,m - 1\} \cup \\ &\{(7i - t + j\Delta, 1) \mid j = 0, 1, \ldots, 3\,m - 1,\ t = 1, \ldots, 6\} \cup \\ &\{(7i - 1, 1)\} \end{aligned}$$

Second, for $j = 1, \ldots, m$, with $e_j = v_a v_b$, we have

$$R^j = \{(7\,a - 2 + (3j - 2)\Delta, M), (7\,b - 2 + (3j - 2)\Delta, M)\}\,.$$

Finally,

$$\overline{R} = \{(3j\Delta, M) \mid j = 0, 1, \ldots, m - 1\}$$

Notice that all receivers we have introduced have either cost 1 or $M$. The $3\,(mn+1)$ receivers with cost $M$ are called *priority receivers*. One useful consequence of the next lemma is that, though the magnitude of the integer $M = 9\,mn$ is only polynomial in $n$ and $m$, still $M$ is large enough to force all priority receivers to be served instantly in any optimal schedule.

**Lemma 5.1.** *If $G$ admits a vertex cover of size at most $\phi$, then there exists a multicast schedule of cost at most $m(9\,n-1)+\phi$.*

*Proof.* Let $X \subseteq V$ be a vertex cover of $G$ with $|X| = \phi$. For $i = 1, 2, \ldots, n$, use the $i$-th channel to transmit at every instant in $T_i = \{7i + j\Delta \mid j = 0, 1, \ldots, 3\,m - 1\}$. Also, for $t = 1, 2, 3$, if $v_i \in X$ then use the $(tn+i)$-th channel to transmit at every instant in $T_{i,t} = \{7i - 2t + j\Delta \mid j = 0, 1, \ldots, 3\,m-1\}$ and otherwise, if $v_i \notin X$, then use the $(tn+i)$-th channel to transmit at every instant in $T_{i,t} = \{7i - 2t + 1 + j\Delta \mid j = 0, 1, \ldots, 3\,m - 1\}$.

In this way we have spared the last $(4\,n + 1)$-th channel where we first schedule all the transmissions for the $m$ priority receivers in $\overline{R}$, that is, we transmit at the instants $\{3j\Delta \mid j = 0, 1, \ldots, m-1\}$. Notice that these $m$ transmissions leave $m$ free intervals onto the last $(4\,n+1)$-th channel, with each of these intervals having length precisely $2\Delta$.

Before indicating how to schedule further transmissions on this last channel, we find it convenient to observe that, if these further transmissions were not introduced, then all receivers in $\overline{R}$ would incur in no cost while the receivers in $\bigcup_{i=1}^{n} R_i$ would collectively incur in a cost of $\phi+(3\,n)(3\,m)$. Moreover, since $X$ is a vertex cover, then for each $e_j = v_a v_b \in E$ we have that at most one of the two priority receivers in $R^j = \{(7\,a-2+(3j-2)\Delta, M), (7\,b-2+(3j-2)\Delta, M)\}$ would be served with some delay and hence incur in some positive cost.

The idea is therefore to exploit the $m$ still free and available intervals onto the last channel to instantly serve these unattended priority receivers, also lowering the above mentioned cost of $\phi + (3\,n)(3\,m)$ associated to the receivers in $\bigcup_{i=1}^{n} R_i$ to only $\phi + 9\,mn - m$. Precisely, for each $e_j = v_a v_b \in E$, at most 3 cases can occur:

- if $a \in X$ and $b \notin X$, then we schedule a transmission at $7\,b - 2 + (3j - 2)\Delta$;
- if $a \notin X$ and $b \in X$, then we schedule a transmission at $7\,a - 2 + (3j - 2)\Delta$;
- if $a \in X$ and $b \in X$, then we schedule a transmission at $7\,a - 1 + (3j - 2)\Delta$.

Notice that after scheduling these last $m$ transmissions on the last channel, all priority requests are then served instantly. Notice also that, whatever of the above 3 cases occurs, whenever we schedule the transmission for the edge $e_j = v_a v_b \in E$, the global cost associated to the receivers in $\bigcup_{i=1}^{n} R_i$ is lowered by 1. It follows that once these last $m$ transmissions have been added onto the last channel the total cost of the schedule produced is $m(9\,n-1)+\phi$. □

We also have the following converse:

**Lemma 5.2.** *If there exists a multicast schedule of cost at most $m(9\,n-1)+\phi$, then $G$ admits a vertex cover of size at most $\phi$.*

*Proof.* First, since $\phi < n < m$ can be trivially assumed, then the cost of the proposed schedule is less than $M$, which implies that the proposed schedule instantly serves all the priority receivers.

By possibly dropping useless transmissions, we can assume that no two transmissions are issued at the same time and also that every transmission issued serves at least one receiver. Two neat properties of the proposed schedule then follow.

($p1$) the starting time $t$ of each scheduled transmission obeys $k\Delta \leq t \leq k\Delta + 7\,n$ for some $k = 0, 1, \ldots, 3m - 1$;

($p2$) for every $j = 0, 1, \ldots, m - 1$, at most $3(4\,n + 1)$ transmissions are scheduled with starting time $j\Delta \leq t \leq (j + 3)\Delta$.

Here, Property ($p1$) follows since $k\Delta \leq t \leq k\Delta + 7\,n$ holds for every receiver $(t, c)$ of the instance obtained with the reduction and also from the fact that a transmission is forced at every $k\Delta + 7\,n$ by the presence of the priority receivers in $R_n$. Actually, for $k \not\equiv 0 \bmod 3$, we have a more narrow admissible interval: $k\Delta < t \leq k\Delta + 7\,n$. As for Property ($p2$), if $3(4\,n+1)+1$ transmissions had been scheduled there, then we would necessarily have a transmission at each of the following 4 instants: $j\Delta$, $(j+1)\Delta$, $(j+2)\Delta$, $(j+3)\Delta$, contradicting what just observed for $k \not\equiv 0 \bmod 3$. A further couple of things should be observed concerning the scheduling instance constructed by our reduction. For each $j = 0, 1, \ldots, m-1$, and $p = 0, 1, 2$, let $R_{j,p}$ denote the set of those receivers $(t, c)$ with $(3j+p)\Delta \leq t \leq (3j+p)\Delta + 7\,n$, and let $T_{j,p} = \{\, t \,:\, \exists (t, c) \in R_{j,p} \,\}$. Notice that $|T_{j,p}| \geq 7\,n$ and $|T_{j,0}| = 7\,n + 1$ for each $j$. Since we only have $4\,n + 1$ channels, this already implies that for every $j = 0, 1, \ldots, m-1$ the cost collectively incurred by the receivers in $R_{j,p}$ is at least $3\,n - 1$ and by those in $R_{j,0}$ is at least $3\,n$. Furthermore, for every pair $(j, p)$ where this lower bound is attained by the proposed schedule, the schedule must have a very precise structure. In particular, when $p = 0$, in order for the lower bound to be attained we have a transmission at instant $(3j + p)\Delta$ to immediately serve a prioritary receiver in $\overline{R}$, and also, for every $i = 1, 2, \ldots, n$, either we have a transmission at the 4 instants $3j\Delta + 7i, 3j\Delta + 7i - 2, 3j\Delta + 7i - 4, 3j\Delta + 7i - 6$ or we have a transmission at the 4 instants $3j\Delta + 7i, 3j\Delta + 7i - 1, 3j\Delta + 7i - 3, 3j\Delta + 7i - 5$ to serve receivers in $R_{j,0}$; whenever this structure is not rigidly followed, then the penalty is at least 2 for every $i$ where a discrepancy is observed still involving 3 transmissions for that $i$, and it is at least 3 when we omit performing all 3 transmissions for that $i$ (whereas when for an $i$ we have an extra transmission we recover only 1 of these 3, with a total loss of at least 2). Consider now a fixed $j$: if the above penalties have not been payed for both $R_{j,0}$ and $R_{j+1,0}$, then it follows by Property ($p2$) that the cost collectively incurred by the receivers in $R_{j,1} \cup R_{j,2}$ is at least $6\,n - 2$. Furthermore, for this lower bound to be attained, the schedule must have the following structure: there is a transmission at instants $3j\Delta$ and $3(j+1)\Delta$ to immediately serve two receivers in $\overline{R}$, and then, for every $i = 1, 2, \ldots, n$, either there is a transmission at each of the 14 instants $3j\Delta + 7i, 3(j+1)\Delta + 7i, (3j+p)\Delta + 7i - 2, (3j+p)\Delta + 7i - 4, (3j+p)\Delta + 7i - 6$ (with $p = 0, 1, 2, 3$) or a transmission at each of the 14 instants $3j\Delta + 7i, 3(j+1)\Delta + 7i, (3j+p)\Delta + 7i - 1, (3j+p)\Delta + 7i - 3, (3j+p)\Delta + 7i - 5$ (with $p = 0, 1, 2, 3$) to serve receivers in $R_{j,0} \cup R_{j,1} \cup R_{j,2} \cup R_{j+1,0}$, plus we must have one further transmission which instantly serves one or two further receivers in $R_{j,1} \cup R_{j,2}$. Again, when this structure is not rigidly followed, then the penalty is at least 2 for every $i$ where a violation is occurring. In this way, we can enforce this structure over the proposed schedule without incrementing the associated costs. Once this has been done, then for every $i = 1, 2, \ldots, n$ it must be the case that either for every $j = 0, 1, \ldots, m-1$ we have a transmission at the 3 instants $3j\Delta + 7i - 2, 3j\Delta + 7i - 4, 3j\Delta + 7i - 6$ or we have a transmission at the 3 instants $3j\Delta + 7i - 1, 3j\Delta + 7i - 3, 3j\Delta + 7i - 5$; when the first case occurs, then we place vertex $v_i$ into $X$. The fact that $X$ is a vertex cover of $G$ follows now from the fact that the proposed schedule instantly serves all requests in $R^j$, $j = 1, \ldots, m$. The fact that $|X| = \phi$ follows from the fact that for every $i = 1, \ldots, n$ there are two requests of the form $(7i - 1, 1)$ into $R_i$. □

**Theorem 5.3.** *The decisional multicast scheduling problem with $K$ multiple channels and arbitrary message transmission duration $\Delta$ is $NP$-complete.*

*Proof.* By Lemmas 5.1 and 5.2, VERTEX COVER can be reduced in polynomial time to the multicast scheduling problem. Since VERTEX COVER is $NP$-complete and a polynomial time non deterministic algorithm can be easily designed for the multicast problem, the proof follows. □

### 5.2 Optimal algorithm for constant transmission duration

When $\Delta = O(1)$ a dynamic programming algorithm can be designed which finds an optimal multicast schedule for $K$ channels by selecting the transmission starting times among the set of time candidates consisting of all the integers between $t_1$ and $t_n + \Delta - 1$. This algorithm heavily depends on the assumption that all the $t_i$'s and $\Delta$ are integers, and it does not hold anymore when such data are reals.

Let $R$ be an array storing, for each time candidate, the waiting cost of a receiver, if there is any receiver available. Precisely, assuming without loss of generality $t_1 = 0$, one has:

$$R[t] = \begin{cases} c_i & \text{if } \exists i \mid t = t_i \\ 0 & \text{otherwise} \end{cases} \qquad 0 \le t \le t_n + \Delta - 1$$

Let $V_{\Delta,K}$ be the set of arrays of size $\Delta$ with non-negative components such that the sum of their components is exactly equal to $K$. Formally, $V_{\Delta,K} = \{v \in \mathbb{N}^{\Delta} | \sum_{d=0}^{\Delta-1} v[d] = K\}$. In practice, at any time instant $t$, the component $v[d]$ indicates how many channels become available at time $t + d$.

For each $t = 0, 1, \ldots, t_n + \Delta - 1$ and each $v \in V_{\Delta,K}$, let

$$\begin{array}{rcl} M[t,v] & \triangleq & \text{the cost of the optimal solution for the subproblem } R[t], \ldots, R[t_n + \Delta - 1] \\ & & \text{where the channel availability is given by the array } v \\ T[t,v] & \triangleq & \text{the starting time of the first transmission for the same subproblem} \end{array}$$

Given $v \in V_{\Delta,K}$, let $v_\tau$, with $\tau = \{0, 1\}$, be defined as follows:

$$v_\tau[d] = \begin{cases} v[0] - \tau + v[1] & \text{if } d = 0 \\ v[d+1] & \text{if } 1 \le d \le \Delta - 2 \\ \tau & \text{if } d = \Delta - 1 \end{cases}$$

In words, since the array $v$ gives the channel availability in the time interval $[t, t + \Delta - 1]$ for any time instant $t$, the array $v_\tau$ provides the new channel availability in the time interval $[t + 1, t + \Delta]$ depending on whether a transmission has been done at time $t$ ($\tau = 1$) or not ($\tau = 0$).

To compute $M[t, v]$, observe that, if $v[0] = 0$, then no transmission can start at time $t$. Hence, any receiver available at time $t$ has to wait up to the first transmission of the optimal solution $M[t+1, v_0]$, that is that for the subproblem starting at the next time instant assuming the channel availability given by array $v_0$. Whereas, if $v[0] > 0$, one has to choose whether to transmit immediately, without any additional waiting cost with respect to $M[t + 1, v_1]$, or not. Therefore, the Recurrences for $M[t, v]$ and $T[t, v]$, with $0 \le t \le t_n + \Delta - 1$ and $v \in V_{\Delta,K}$, become:

$$\begin{gathered} M[t,v] = \min\{M[t+1, v_0] + R[t](T[t+1, v_0] - t), \Gamma[t,v]\} \\ T[t,v] = \begin{cases} t & \text{if } M[t,v] = M[t+1, v_1] \\ T[t+1, v_0] & \text{otherwise} \end{cases} \end{gathered} \tag{13}$$

where

$$\Gamma[t,v] = \begin{cases} \infty & \text{if } v[0] = 0 \\ M[t+1, v_1] & \text{if } v[0] > 0 \end{cases}$$

```
Procedure BUILD-SCHEDULE (M, T);
1    s := 0; t := 0; S := ∅;
2    let v be such that M[0, v] = min_u{M[0, u]};
3    while s < t_n do
4        if T[t, v] = t
5            then v := v_1; s := t; S := S ⋃ {s}
6            else v := v_0;
7        t := t + 1;
```

Figure 4: *Procedure for building an optimal schedule $S$.*

When $t = t_n + \Delta$ and $v \in V_{\Delta,K}$, the arrays $M$ and $T$ are initialized as follows:

$$M[t_n + \Delta, v] = \begin{cases} \infty & \text{if } v[0] = 0 \\ 0 & \text{if } v[0] > 0 \end{cases}$$

$$T[t_n + \Delta, v] = \begin{cases} \infty & \text{if } v[0] = 0 \\ t_n + \Delta & \text{if } v[0] > 0 \end{cases}$$

To compute an optimal multicast schedule $S$, one can keep track of the actual transmission times by searching the array $T$ as shown in Procedure BUILD-SCHEDULE, reported in Figure 4. The search can start from any entry $T[0, v]$ for which $M[0, v]$ is minimum on row 0 of $M$ (line 2). Clearly, if such a minimum is not unique, different optimal schedules might be found starting from different $T[0, v]$'s. A new transmission starting at time $s = t$ is added to the schedule whenever $T[t, v] = t$ (line 5). The search proceeds from either $T[t + 1, v_1]$ or $T[t + 1, v_0]$, depending on whether there is a transmission starting at time $t$ or not (lines 3–7).

In order to devise the overall time complexity of the dynamic programming algorithm, observe that the number of rows of $M$ and $T$ is $O(t_n + \Delta)$, the number of columns is $|V_{\Delta,K}| = O(K^\Delta)$, and that $O(1)$ time is needed to fill each entry. Therefore, overall $O((t_n + \Delta)K^\Delta)$ time is required.

Recalling that $t_{i+1} - t_i < 2\Delta$ and that $\Delta = O(1)$ by assumption, one has $t_n + \Delta \leq 2\Delta(n - 1) + \Delta$, and thus the number of rows is $O(n)$. Since $K < \Delta$, the number of columns is $|V_{\Delta,K}| = O(K^\Delta) = O(1)$, and $O(n)$ time is spent to fill all the entries of $M$ and $T$. Since the array $R$ can be constructed in $O(n)$ time, even if the receiver availability times are not sorted, the overall time complexity of the algorithm is $O(n)$, which is clearly optimal because an $\Omega(n)$ time lower bound trivially holds. It is worth noting that, although the complexity is linear, the algorithm is practical only for small values of $\Delta$ and $K$.

Consider again the problem instance in Figure 1, but with $K = 2$ channels. The corresponding receiver availability array is $R = [1, 1, 0, 1, 0, 0, 1, 0, 0, 0]$, while the 10 arrays in the set $V_{4,2}$ are $v^1 = [2, 0, 0, 0], v^2 = [0, 2, 0, 0], v^3 = [0, 0, 2, 0], v^4 = [0, 0, 0, 2], v^5 = [1, 1, 0, 0], v^6 = [1, 0, 1, 0], v^7 = [1, 0, 0, 1], v^8 = [0, 1, 1, 0], v^9 = [0, 1, 0, 1], v^10 = [0, 0, 1, 1]$. For each array $v^j$, the indices of the arrays $v_0^j$ and $v_1^j$ are $v_0^1 = 1, v_1^1 = 7, v_0^2 = 1, v_0^3 = 2, v_0^4 = 3, v_0^5 = 1, v_1^5 = 7, v_0^6 = 5, v_1^6 = 9, v_0^7 = 6, v_1^7 = 10, v_0^8 = 5, v_0^9 = 6, v_0^{10} = 8$. The resulting arrays $M$ and $T$ are given as follows, where column $j$ refers to array $v^j$ defined above, with $1 \leq j \leq 10$.

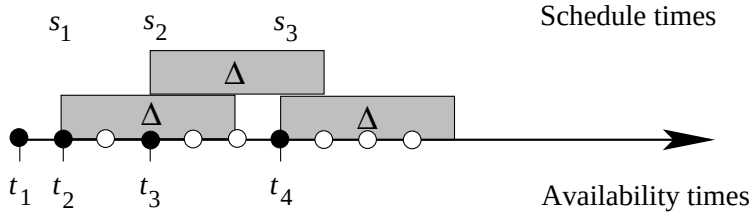


Figure 5: An optimal schedule for the instance of Figure 1, when $K = 2$.

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 3 | 5 | 1 | **1** | 1 | 1 | 1 | 3 |
| 1 | 0 | 1 | 2 | 4 | 0 | 0 | 1 | 1 | 2 | 2 |
| 2 | 0 | 0 | 1 | 2 | 0 | 0 | 0 | 0 | 0 | 1 |
| 3 | 0 | 1 | 2 | 3 | 0 | 0 | 0 | 1 | 1 | 2 |
| 4 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 5 | 0 | 0 | 1 | 2 | 0 | 0 | 0 | 0 | 0 | 1 |
| 6 | 0 | 1 | 2 | 3 | 0 | 0 | 0 | 1 | 1 | 2 |
| 7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 8 | 0 | 0 | 0 | ∞ | 0 | 0 | 0 | 0 | 0 | 0 |
| 9 | 0 | 0 | ∞ | ∞ | 0 | 0 | 0 | 0 | 0 | ∞ |
| 10 | 0 | ∞ | ∞ | ∞ | 0 | 0 | 0 | ∞ | ∞ | ∞ |

$M$

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 2 | 3 | 0 | **1** | 1 | 1 | 1 | 2 |
| 1 | 1 | 2 | 3 | 4 | **1** | 1 | 1 | 2 | 3 | 3 |
| 2 | 2 | 3 | 4 | 5 | 2 | 3 | **3** | 3 | 3 | 4 |
| 3 | 3 | 4 | 5 | 6 | 3 | **3** | 3 | 4 | 4 | 5 |
| 4 | 4 | 5 | 6 | 7 | 4 | 4 | 5 | 5 | **6** | 6 |
| 5 | 5 | 6 | 7 | 8 | 5 | **6** | 6 | 6 | 6 | 7 |
| 6 | 6 | 7 | 8 | 9 | **6** | 6 | 6 | 7 | 7 | 8 |
| 7 | 7 | 8 | 9 | 10 | 7 | 7 | 7 | 8 | 8 | 9 |
| 8 | 8 | 9 | 10 | ∞ | 8 | 8 | 10 | 9 | 10 | 10 |
| 9 | 9 | 10 | ∞ | ∞ | 9 | 10 | 10 | 10 | 10 | ∞ |
| 10 | 10 | ∞ | ∞ | ∞ | 10 | 10 | 10 | ∞ | ∞ | ∞ |

$T$

The bold entries of $M$ and $T$ are used to actually build the optimal schedule shown in Figure 5, where small circles represent time candidates.

# 6 Conclusion

This paper studied a multicast problem where the same message of duration $\Delta$ has to be transmitted to a set of $n$ receivers which are not all available simultaneously. Multiple transmissions of the message can be scheduled with the objective of minimizing the overall waiting cost of the receivers.

In the case that a single channel is available for transmission, a dynamic programming off-line algorithm has been proposed which finds an optimal multicast schedule in $O(n \log n + \min\{n^2, n\Delta^2\})$ time, when the availability times and the transmission duration are integers, and in $O(n^2)$ time, when such data are reals, thus improving over a previously known $O(n^3)$ time algorithm [5]. In the case of multiple channels, an optimal off-line algorithm running in $O(n)$ time has been proposed for constant transmission duration, and hence constant number of channels, whereas an NP-completeness proof has been provided for arbitrary transmission

duration and arbitrary number of channels. Such results have been proved in the case that the availability times and the transmission duration are integers.

As a matter of further research, one could investigate the time complexity of the multicast problem for arbitrary transmission duration and constant number of channels. Moreover, one could design on-line and off-line heuristic algorithms for arbitrary transmission duration and arbitrary number of channels, for example, by generalizing to multiple channels those heuristics already proposed in [4, 5] for a single channel. Finally, another extension that could be of interest in future work, is to study the multicast problem when the number of transmissions in the schedule is limited and/or the receivers can tune on a restricted subset of channels.

## References


[1] M.R. Garey and D.S. Johnson, *Computers and Intractability*, Freeman, San Francisco, 1979.

[2] D. Ghose and H.J. Kim, Scheduling video streams in Video-on-Demand systems: A survey. *Multimedia Tools and Applications* 11 (2000) 167–195.

[3] A.M. Hamad and A.E. Kamal, A survey of multicasting protocols for broadcast-and-select single-hop networks. *IEEE Network* 16 (2002) 36–48.

[4] P. Jue and B. Mukherjee, The advantages of partitioning multicast transmissions in a single-hop optical WDM network. *Photonic Network Communications* 1 (1999) 111–124.

[5] C.S. Sung and H.C. Rim, Receiver set partioning and sequencing for multicasting traffic in a WDM lightwave single-hop network. *Journal of the Operational Research Society* 55 (2004) 630–639.

[6] S. Webster and K.R. Baker, Scheduling groups of jobs on a single machine. *Operations Research* 43 (1995) 692–703.