Scheduling Constant Bit Rate Flows in Data over Cable Networks

Nir Naaman and Raphael Rom

Department of Electrical Engineering
Technion - Israel Institute of Technology
Haifa 32000, Israel
Email: {mnir@techunix, rom@ee}.technion.ac.il

Abstract

Data Over Cable Systems Interface Specification (DOCSIS) is the leading standard for data over cable networks. We consider the problem of scheduling Constant Bit Rate (CBR) flows in a DOCSIS compliant cable network. CBR flows are required to support the delivery of voice and other real-time applications that generate fixed size data packets on a periodic basis. The primary application of CBR flows is Voice over IP (VoIP) which cable operators intend to use in order to provide cable telephony services. DOCSIS 1.1 is enhanced with Quality of Service (QoS) capabilities; it defines the Unsolicited Grant Service as the mechanism for supporting CBR flows. The scheduling algorithms, however, are not defined by the standard.

We present the scheduling problem and examine several interesting special cases of it. We show that deciding whether a set of CBR flows can be legally scheduled is NP-complete whenever there are two or more different grant intervals. We model the scheduling problem as a variant of bin packing where bin sizes can be modified in a constrained manner. This model enables the development of scheduling algorithms which are based on known algorithms for bin packing. We present an algorithm based on the Next-Fit algorithm and investigate its performance. We show that under certain assumptions which typically hold for VoIP and many other practical applications, a simple polynomial time scheduling algorithm is sufficient.

keywords: scheduling, cable telephony, CATV, CBR, DOCSIS, HFC, QoS, VoIP, bin packing.

1 Introduction

The rapidly growing use of the Internet and multimedia applications created the need for high bandwidth services at subscribers' homes and small offices. The local loop, or last mile, presents the main obstacle to delivering such broadband services. One of the main technologies for delivering broadband services over the local loop is data over CATV (Community Antenna Television). Cable operators are now using their CATV networks to deliver broadband data services, in addition to standard TV channels. The existing infrastructure of Hybrid-Fiber-Coax (HFC), used by most modern CATV networks, enables cable operators to provide bi-directional high bandwidth data channels, at relatively low cost. Initially cable operators offered only best effort type of service

by sharing the available bandwidth equally among all subscribers. In recent years, however, cable operators have been looking for ways to provide additional services such as telephony and video. Since these new services cannot rely on best effort type of service, a mechanism for supporting different Quality of Service (QoS) requirements must be implemented. In this paper we refer to the Data-Over-Cable Service Interface Specification (DOCSIS) standard [8, 5] which is the leading standard for data over CATV networks. We adopt the DOCSIS mechanism for supporting QoS and focus on the problem of scheduling flows with Constant Bit Rate (CBR) type of service; a problem which is left open by DOCSIS. The main application of CBR flows today is the delivery of voice over the CATV media. Cable operators intend to use CBR flows to offer Voice over IP services that would compete with telephony services offered by telephone companies (see [1, 2, 19] for more details).

CATV networks are characterized by a tree-and-branch topology. The Cable Modem Termination System (CMTS), at the root of the tree, controls all traffic in the network. Subscribers of data services use a Cable Modem (CM) to connect to the CMTS. The available bandwidth is divided into channels. Downstream channels (from the CMTS to the CMs) are used by the CMTS. Upstream channels (from the CMs to the CMTS) are shared by all subscribers connected to the same fiber node (typically 500 to 2000 subscribers). To share the upstream channel a TDMA MAC protocol with dynamic bandwidth allocation is implemented. As the downstream is used only by the CMTS no MAC protocol is needed.

DOCSIS based networks transfer Internet Protocol (IP) datagrams between the CMTS and the CMs. The CMTS is responsible for the scheduling of all transmissions in the upstream. Scheduling is done by dividing the upstream, in time, into a sequence of numbered mini-slots. A mini-slot is the unit of granularity for upstream transmission; transmitting a datagram may require one or more mini-slots. The CMTS and a CM may establish a *service flow* between them; a service flow describes the type of connection between the CMTS and the CM and is identified by a service identifier (SID). A CM can support multiple (typically up to four) active service flows simultaneously. When a service flow is created the CMTS starts allocating bandwidth to the CM. For CBR service flows the CMTS allocates the bandwidth without any interaction with the CM. In other services, such as variable bit rate (VBR) and best effort, the CM sends requests to the CMTS indicating the amount of bandwidth it needs, the CMTS then decides if and when to allocate the bandwidth. From time to time, the CMTS publishes a *MAP* in which it allocates mini-slots to the different CMs. The allocation of each mini-slot must appear in one of the MAPs. The scheduling problem is how to allocate the mini-slots in the MAP to the different CMs.

In order to support different QoS demands DOCSIS defines several types of service flows. The main services are Unsolicited Grant Service (UGS) intended for CBR flows, Real-Time Polling Service

(rtPS) intended for VBR flows, Non-Real-Time Polling Service (nrtPS) intended for non real-time VBR flows, and Best Effort service. We are interested in the Unsolicited Grant Service which is designed to support real-time service flows that generate fixed size data packets on a periodic basis. The main application of UGS flows is the delivery of Voice over IP (VoIP). When a CM establishes a UGS service flow the CMTS must allocate a fixed number of mini-slots to the CM at periodic intervals, called the grant interval. An active service flow is assigned a set of parameters that describe it. The relevant parameters for scheduling UGS flows are the grant size (number of bytes to be allocated in each grant), grant interval (the delay between successive grants), and the tolerated grant jitter. Each flow may have a different set of parameters depending on factors such as the type of CODEC (Coder-Decoder) used, the bandwidth and buffering requirements of the application, and the number of active sessions (grants per interval) supported by the flow. The scheduling algorithm at the CMTS must allocate the available time slots to the different service flows in a way that ensures that each service flow is scheduled according to its specifications.

Previous work on QoS scheduling for DOCSIS compliant cable networks have been published. Bandwidth allocation for non real-time applications has been considered in [6, 7, 20]. Scheduling of Variable Bit Rate MPEG video has been considered in [3, 12]. Simulations of simple scheduling algorithms for CBR flows have been presented in [21, 22]. The problem of assigning upstream channels to cable telephony and moving calls from one channel to another, under the assumption that all calls (CBR flows) have the same parameters, is analyzed in [23]. Our work, to the best of our knowledge, is the first to analyze the general problem of scheduling CBR flows. We show that the general problem is NP which rules out the possibility of finding an optimal polynomial time scheduling algorithm (in [7] and [20] polynomial algorithms are nevertheless suggested). We suggest to model the scheduling problem as a new variant of bin packing. This model enables the development of efficient scheduling algorithms which are based on known algorithms for bin packing.

The general scheduling problem, with an arbitrary number of different grant intervals, is complex. We therefore start by analyzing several simpler special cases. In Section 3 we study the case of a single grant interval and show that the scheduling problem in this case is easy. In Section 4 we consider the case of flows with two different grant intervals. We analyze the scheduling problem by converting it to a variant of bin packing in which bin sizes can be modified in a constrained manner. We show that deciding whether a set of flows with two different grant intervals has a legal schedule is NP-complete. We then examine a special case where the grant sizes are smaller than the tolerated jitter and present an optimal polynomial time algorithm. In Section 5 we briefly discuss the extension of the analysis to cases where there are flows with more than two different grant intervals.

2 Problem Definition

We consider the problem of scheduling a set F of CBR flows over a slotted TDMA channel. Each flow $f_i \in F$ is characterized by three parameters

- Grant Size - $S(i)$ - the number of mini-slots that must be allocated to f_i in each grant.
- Grant Interval - $I(i)$ - the nominal time between every two grants to f_i.
- Grant Jitter - $J(i)$ - the tolerated delay of an actual grant from its nominal time.

Scheduling Rules: Each flow f_i is associated with a time reference which we denote by $t_1(i)$. The time reference is chosen by the scheduling algorithm and provides a way for calculating the nominal grant times; the k^{th} nominal grant time of flow f_i is $t_k(i) = t_1(i) + (k-1)\,I(i)$. Flow f_i must get an allocation of $S(i)$ consecutive slots in each grant. The actual grant time must not precede the nominal grant time and must not exceed it by more than the grant jitter. To state it formally, let $\tau_k(i)$ be the actual time of the k^{th} grant of f_i, then it must satisfy $t_k(i) \leq \tau_k(i) \leq t_k(i) + J(i)$.

We define the *actual jitter* of the k^{th} grant of f_i as $j_k(i) \equiv \tau_k(i) - t_k(i)$, where $t_k(i)$ and $\tau_k(i)$ are the nominal and actual times of the k^{th} grant of flow f_i, respectively. To maintain a legal schedule of f_i the actual jitter of each grant must not exceed the grant jitter, i.e., $0 \leq j_k(i) \leq J(i),\ \forall k$.

Our objective is to schedule the flows such that the timing requirements of all the flows are maintained. We call a schedule that achieves this goal a ***legal schedule***. We call a set of flows for which a legal schedule exists a ***feasible set***. Figure 1 presents an example of a legal schedule.

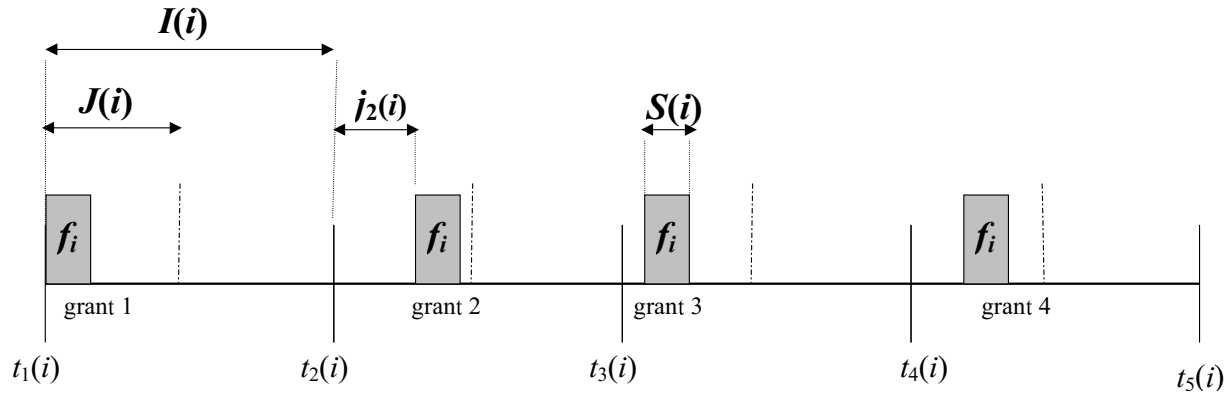

Figure 1: Example of a legal schedule of flow f_i.

The scheduling problem appears in several variations

- Offline setting - The entire set of flows is given before scheduling begins.

- Online setting - Flows are established over time; scheduling decisions are made without any knowledge of future flows.
- Permanent flows - A flow that has been scheduled stays forever.
- Temporary flows - Each flow has a duration for which it must be scheduled. Once a flow completes its duration it leaves.

In this paper we consider the offline version of the problem with permanent flows. We intend to cover the online version in future work. We analyze two different problems.

1. **Feasible-Set** - Given a set of flows F, find whether the set is feasible, i.e., if there is a legal schedule for the set.
2. **Optimal-Schedule** - Given a set of flows F find an optimal legal schedule for a subset of flows $\hat{F} \subseteq F$. We consider two optimization functions: (1) maximize the number of flows in $\hat{F}$, and (2) maximize the channel utilization which is given by $\sum_{f(i) \in \hat{F}} S(i)/I(i)$.

The general scheduling problem is complex. We therefore focus on several interesting special cases. We start with the simplest case where all the flows have the same grant interval and progress to more complex settings.

3 Scheduling Flows with a Single Grant Interval

We assume that all the flows in F have the same grant interval which we denote by I; the flows may have different grant sizes. In this case the answer to most of the scheduling problems is easy.

1. **Feasible-Set** - A legal schedule exists if and only if the sum of grant sizes of all the flows is no more than the grant interval, i.e., if $\sum_{f_i \in F} S(i) \leq I$. A legal schedule for a feasible set is obtained by ordering the flows one after the other over the grant interval and repeating this sequence in all the following intervals. The initial order of the flows is not important.
2. **Optimal-Schedule** - The Optimal-Schedule problem has a simple solution when the goal is to maximize the number of flows. We sort the flows in F according to increasing order of their grant sizes and choose the maximal subset for which the total grant sizes does not exceed I.

   In the case where the goal is to maximize the channel utilization, the problem of Optimal-Schedule is identical to the subset-sum problem. The subset-sum problem is NP-hard but can be solved in pseudo polynomial time [9]. There are also many approximation algorithms for subset-sum (see e.g., [13, 14, 15]).

4 Scheduling Flows with Two Different Grant Intervals

We consider the case where there are two grant intervals, I_1 and I_2 satisfying $I_2 = m \cdot I_1$, where m is an integer. We denote by F_k $(k = 1, 2)$ the set of flows with grant interval I_k and by $\Sigma_k \equiv \sum_{f_i \in F_k} S(i)$ the total size of all grants in F_k.

In order to produce a legal schedule it is sufficient to find a legal allocation of the flows over an interval equal to I_2 which we call the *basic interval*. Once we find such an allocation, we can repeat it over and over to get a legal schedule (see Figure 2). In the basic interval each flow in F_1 must be scheduled m times, while each flow in F_2 must be scheduled once. To find a legal schedule we first group all flow in F_1 and allocate them at the beginning of m periodic intervals of length I_1. Once the flows in F_1 have been scheduled the free slots in the basic interval consists of m gaps which are left between successive allocations of flows in F_1; the size of each gap is $I_1 - \Sigma_1$. Our task now is to allocate the flows in F_2 in those gaps, in order to form a legal schedule. An example of an allocation of flows in F_1 and F_2 over two basic intervals is shown in Figure 2.

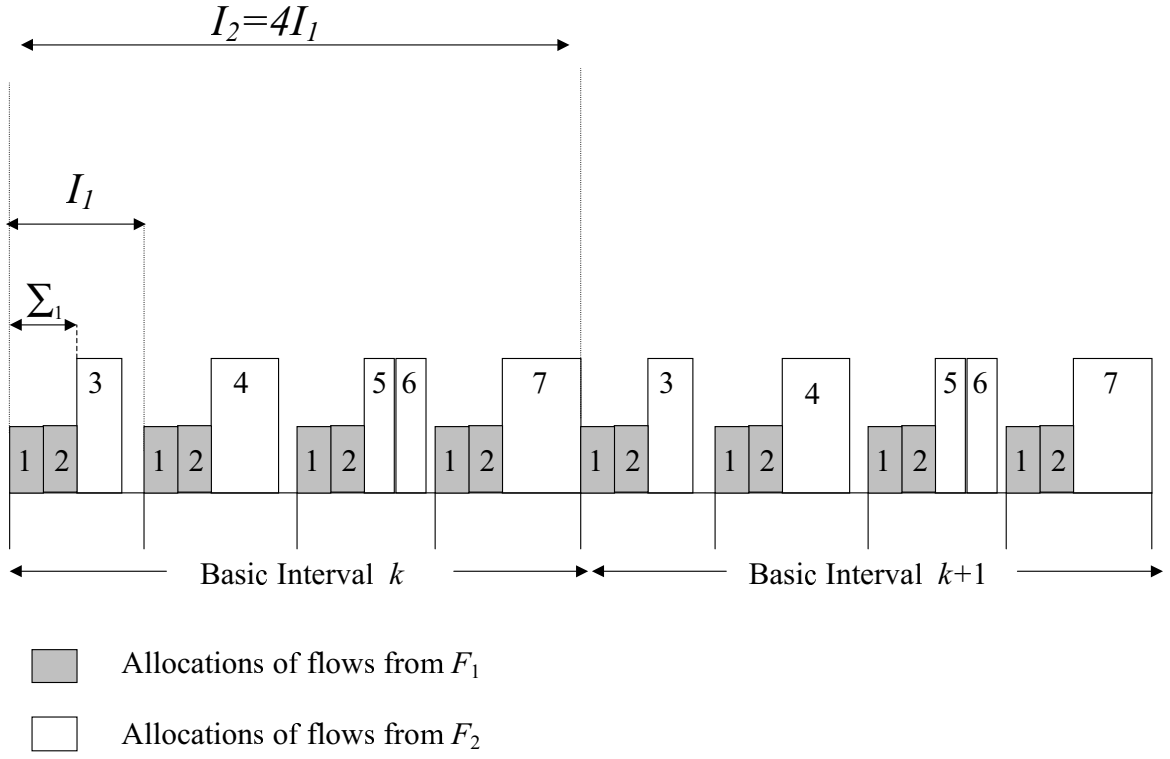

Figure 2: Example of a legal schedule of flows in F_1 and F_2 over two basic intervals.

The problem of scheduling the flows in F_2 immediately brings to mind a bin packing problem. Bin packing problems have been widely researched and investigated (see e.g., [10, 11] and [4] for a comprehensive survey). In the classical one-dimensional bin packing problem, we are given a list of

items $L = (a_1, a_2, ..., a_n)$, each with a size $s(a_i) \in (0, 1]$ and are asked to pack them into a minimum number of unit capacity bins. In the scheduling problem each item corresponds to a flow in F_2 and its size is the flow's grant size. There are m bins of size $B = I_1 - \Sigma_1$ and the goal is to pack the items into the bins in a way that results in a legal schedule. The variant of bin packing we face in the scheduling problem is different from classical bin packing in the following ways:

- The size of the bins may be modified in a constrained manner. This is due to the grant jitter of flows in F_1 that permits us to move them forward in time by as many slots as their grant jitter allows.
- Some items may be bigger than the bin size. Such items may still be packed by increasing the sizes of some of the bins.
- There is a fixed number of bins.

Note that, due to the scheduling rules, in this version of bin packing items may not be fragmented. A similar problem where items may be fragmented has been considered in [16, 17].

4.1 Changing Bin Sizes

The main difference between the scheduling problem and classical bin packing is the ability to change bin sizes. Initially all m bins have the same size; we call this size the *nominal bin size* and denote it by B. If the jitter of all flows in F_1 is J it means that bins of sizes in the range $B - J$ to $B + J$ can be created. Suppose we want to change the size of bin number k. If we want to increase the bin's size, we move the group of flows in F_1 in bin number $k + 1$ forward in time. If we want to decrease the bin's size, we move the flows in F_1 in bin number k forward in time. Clearly the total free space does not change so the total sizes of all bins must remain a constant equal to mB. It is important to note that the sizes of the bins are strongly dependent since increasing the size of one bin results in decreasing the size of the next bin. Because of this dependency there are some sequences of bin sizes which are not feasible, i.e., there is no way to produce them. Take for example five bins of nominal size $B = 4$ and assume $J = 3$; it is easy to see that the set of bin sizes $\{6, 6, 6, 1, 1\}$ is not feasible. We conclude that while it is possible to create any bin size in the range $B - J$ to $B + J$, it may be impossible to create certain sequences of bin sizes even if the bins are all in that range.

There are two basic approaches for changing bin sizes. In the first approach, which we call *state dependent*, the algorithm assumes the bins are ordered in a certain way and this order does not change. The fixed bin order enables the algorithm to calculate the possible sizes each bin may have and thus enables it to maintain a legal schedule throughout the packing process. The second

approach is called *state independent* because the algorithm does not maintain the state of the bins that determines the possible bin sizes. Instead, scheduling is divided into two stages. In the first stage the algorithm packs the items while assuming the size of each bin can change by at most J (or any other constant). In the second stage the algorithm tries to order the bins that were created in the first stage so as to produce a legal schedule.

Due to size constraints this paper describes state dependent algorithms only. A more detailed description of both state dependent and state independent algorithms appears in [18].

4.2 Complexity of the Scheduling Problem

We show that the Feasible-Set problem with two grant intervals is NP-complete and hence the Optimal-Schedule problem is NP-hard. To that end we define a variant of bin packing, which we call *Bin Packing with Jitter*, where the size of each bin may be modified by at most the tolerated jitter J, but the sum of sizes must remain constant. We note that Bin Packing with Jitter is the first stage performed by a state independent algorithm.

We denote by $s(B_i)$ and $c(B_i)$ the size and content (sum of packed items) of bin B_i, respectively.

Bin Packing with Jitter (BPJ):
INSTANCE: A set of m bins of nominal size B, a list $L = \{a_1, ..., a_n\}$ of n items, each of size $s(a_k) \in Z^+$, $1 \leq k \leq n$ and a tolerated jitter $J \in Z^+$.
QUESTION: Is there a legal packing for the list of items, i.e., such that the size of each bin satisfies $B - J \leq s(B_i) \leq B + J$, the total sizes of all bins is $\sum_{i=1}^{m} s(B_i) = m\,B$, and the content of each bin does not exceed its size $c(B_i) \leq s(B_i)$, $1 \leq i \leq m$.

Theorem 1 *The Feasible-Set problem with two different grant intervals is strongly NP-complete.*

Proof: Feasible-Set is at least as hard as bin packing with jitter since any legal solution of Feasible-Set is also legal for BPJ, but a legal solution of BPJ is not necessarily a legal solution for Feasible-Set. It is therefore sufficient to prove the following claim.

Claim 4.1 *Bin packing with jitter is NP-complete in the strong sense.*

We show a transformation form 3-PARTITION (defined below) to a restricted version of Bin Packing with Jitter. 3-PARTITION is known to be NP-complete in the strong sense [9].

3-PARTITION:
INSTANCE: A finite set A of $n = 3k$ integers $a_1, a_2, ..., a_n$ and a bound $C \in Z^+$ such that $C/4 < a_j < C/2$ for $j = 1, ..., n$ and $\sum_{j=1}^{n} a_j = k\,C$.

QUESTION: Can A be partitioned into k disjoint subsets $S_1, ..., S_k$ such that $\sum_{j \in S_i} a_j = C$ for $i = 1, ..., k$?

Consider an arbitrary given instance of 3-PARTITION with a bound C and a set of $n = 3k$ elements $A = \{a_1, a_2, ..., a_n\}$. For a given jitter J we construct the following instance of BPJ: let the nominal bin size be $B = C + J$ and let $m = 2k$. The list of items contains $n + k$ items; the first n items have the sizes of the elements in A and we add k more items of size $B + J$.

In order to pack all items of size $B + J$ there must be k bins of size $B + J$. As a result the sizes of the remaining k bins must be $B - J$. After placing the items of size $B + J$ in bins of size $B + J$ we face the problem of packing items corresponding to the elements in A into k bins of size $B - J = C$. This is exactly the 3-PARTITION problem which means that the answer to BPJ is "yes" if and only if the answer to 3-PARTITION is "yes". ♢

We state, without a proof, one more result regarding the complexity of the scheduling problem. We mentioned that state independent algorithms attempt to construct a legal schedule in two stages. In the first stage an algorithm for BPJ is executed; the algorithm produces a set of variable size bins $Q = \{B_1, ..., B_m\}$. In the second stage a different algorithm is used to order the bins in Q so as to construct a legal schedule; we call this stage *Bin Ordering with Jitter*. In [18] we show that the decision version of Bin Ordering with Jitter, i.e., deciding whether a legal schedule can be constructed from a set of bins produced by BPJ, is also NP-complete in the strong sense.

4.3 State Dependent Approximation Algorithms

Since the scheduling problem is NP-hard we develop approximation algorithms for it. In this section we describe a family of algorithms called state dependent algorithms. A state dependent algorithm maintains a state for each bin. A bin's state is made of two components: the free space in the bin, and the actual jitter of flows in F_1 in the bin. The states are updated whenever an item is packed. To pack an item the algorithm first checks if the bin has enough free space to contain the item. If the bin is too small, the algorithm tries to increase its size; the amount by which a bin's size can be increased may depend on the states of all other bins. This way a legal schedule is maintained at every stage of the packing. The main advantage of using a state dependent algorithm is that it always produces a legal schedule. Maintaining the state is simple and requires at most $O(m)$ operations per item. The main disadvantage is that a state dependent algorithm makes no effort to optimize the packing by ordering the bins and can therefore perform poorly in some cases.

We can convert almost any standard bin packing algorithm into a state dependent scheduling algorithm. The algorithm decides in which bin the item is to be packed but the item is not packed

if packing it violates the scheduling rules. If packing an item fails the algorithm may try to pack it in a different bin or discard it. As an example consider a State Dependent version of the First-Fit algorithm [4]. The algorithm goes over the bins according to their order and packs an item into the first bin in which it fits, increasing the bin's size if necessary. If the item does not fit in any of the bins, it is discarded. In the next subsection we present a state dependent version of the Next-Fit algorithm.

4.3.1 The Next Fit with Jitter Algorithm

The Next Fit with Jitter (NFJ) algorithm is a modification of the well-known Next Fit (NF) algorithm. The NF algorithm keeps only one open bin and packs items, according to their order, into the open bin. When an item does not fit in the open bin, the bin is closed, a new bin is opened and the item is packed in the new bin. NFJ is similar to NF but it uses the jitter to change bin sizes. NFJ is a state dependent algorithm; it records only the state of the open bin and uses this state to change the bin's size and to determine the state of the next open bin. This way the algorithm ensures that scheduling the items (flows) according to the order they were packed provides a legal schedule.

Algorithm NFJ

1. Set $i = 1$. // i holds the index of the open bin
2. Set $d = J$. // d holds the increase to the size of the next bin
3. Open bin B_i and modify its size to be $s(B_i) = B + d$.
4. Pack items into B_i until the next item does not fit.
5. Set $d = \min\{s(B_i) - c(B_i),\ J\}$.
6. Close B_i and set its size to $s(B_i) = \max\{c(B_i),\ B - J\}$.
7. Set $i = i + 1$.
8. IF $i == m$ set $d = d - J$. // adjust d for the last bin
9. ELSEIF $i \leq m$, return to step 3.
10. ELSE STOP.

NFJ is a simple algorithm with $O(n)$ running time. To evaluate the performance of the algorithm we consider the case where the goal is to maximize the bin (channel) utilization. For a given list L of n items and algorithm A, let $A(L)$ and $OPT(L)$ be the total size of items from L that A and

an optimal algorithm pack in m bins, respectively. We let $R_A(L) \equiv OPT(L)/A(L)$ and define the asymptotic worst case performance ratio R_A^∞ as

$$R_A^\infty \equiv \inf\{r \geq 1 : \text{ for some } N > 0,\ R_A(L) \leq r \text{ for all } L \text{ with } OPT(L) \geq N\} \tag{1}$$

Theorem 2 *The asymptotic worst case performance ratio of algorithm NFJ is $R_{NFJ}^\infty = 2$.*

Proof: Consider some arbitrary bin B_k, $1 \leq k \leq m-1$. When B_k is opened its size is set to be $s(B_k) = B + j \geq B$. Assume the bin is closed when an item of size x does not fit in it and is therefore packed in B_{k+1}. It follows the that $c(B_k) + c(B_{k+1}) \geq c(B_k) + x > s(B_k) \geq B$ which means that he content of any two consecutive bins is at least $B+1$. We conclude that the average bin content is at least $B/2$ which gives us the upper bound on the asymptotic performance ratio.

We now show an example that proves the lower bound. In our example there are $n+1$ bins of size $B = 2b$, the list of items L has $2n$ items of size b, and $2n$ items of size $J+1$; the items are ordered alternately $L = \{b, J+1, b, J+1, ..., b, J+1\}$. We choose $B > 2n(J+1)$ which means that all items of size $J+1$ fit in one bin, hence $OPT(L) = 2n\,(b + J + 1)$. Algorithm NFJ packs two items in each bin, one of sizes b and one of size $J+1$, hence $NFJ(L) = (n+1)(b+J+1)$. The ratio in this example is $R_{NFJ}(L) = \frac{2n}{n+1}$; the asymptotic worst case performance ratio is therefore $R_{NFJ}^\infty = 2$. ♢

4.4 Legal Schedule when Grant Sizes are Smaller than the Jitter

In this section we consider the scheduling problem under the assumption that grant sizes are smaller than the tolerated jitter. This assumption is expected to hold for many practical applications. For example, in VoIP the grant jitter is normally more than 1 ms, while grant sizes are less than 200 bytes which in a typical CATV network corresponds to less than 0.5 ms [8, 19]. We show that under the above assumption any set of flows with channel utilization less or equal 1 is feasible. Moreover, in this case even a simple algorithm such as NFJ is optimal.

Theorem 3 *Let F be a set of n flows with two grant intervals $I_2 = m\,I_1$. Denote by S_{max} the maximal grant size of flows in F_2 and by J the minimal grant jitter of flows in F_1. Suppose F satisfies the following two properties:*

1. $\sum_{f_i \in F} \frac{S(i)}{I(i)} \leq 1$
2. $S_{max} \leq J + 1$

then F is a feasible set. Furthermore, NFJ always finds a legal schedule for F.

Proof: To construct a legal schedule we first schedule the flows in F_1 over an interval of length I_2 creating m bins of size B. We then apply the NFJ algorithm to schedule the flows in F_2.

To prove the theorem we show that whenever NFJ closes a bin, the bin does not contain unused space. Recall that a bin is defined as the gap between two consecutive blocks of flows in F_1 and that the actual jitter is the difference between the nominal and actual allocation times. We define the actual jitter of a bin as the actual jitter of the flows in F_1 in the block that starts the bin. When NFJ opens bin number k the actual jitter of the bin may be in the range $0 \leq j_k \leq J$. Initially the bin size is $s(B_k) = B - j_k$ but the algorithm increases the size of the bin (by moving the block of F_1 flows that ends the bin) to be $s(B_k) = B - j_k + J$. Assume B_k is closed when the algorithm tries to pack an item of size x but the item does not fit since $c(B_k) + x > B + J - j_k$. It follows that the content of the bin satisfies $c(B_k) > B + J - x - j_k$. Since $x \leq J + 1$ we have $c(B_k) \geq B - j_k$ which means that the content of the bin is at least the original bin size, hence the bin does not contain free space. Since there is no wasted space in any bin that NFJ closes, the algorithm can schedule any set of flows whose channel utilization is no more than 1. Condition 1 ensures that F is such a set. $\diamondsuit$

To see that condition 2 in Theorem 3 is tight consider an example where $S_{max} = J + 2$. We choose a set F consisting of one flow in F_1 with $S(1) = 2$, $I(1) = 10$ and $J(1) = 3$, and eight flows in F_2 with $S(i) = 5$, $I(i) = 50$ and $J(i) = 3$, for $2 \leq i \leq 9$. After allocating f_1 there are five bins of size $B = 8$ to which the flows in F_2 must be assigned. It is easy to verify that there is no way to legally schedule all flows in F_2, hence F is not a feasible set.

5 Scheduling Flows with more than Two Different Grant Intervals

In this section we briefly discuss the scheduling problem when there are $K \geq 3$ different grant intervals. We only have space to point out the differences between the case of two grant intervals and more than two grant intervals. We also explain why the problem becomes even harder when the grant intervals are not integer multiple of each other.

Let us first assume that the grant intervals are an integer multiple of each other, that is, if the intervals are sorted in increasing order they satisfy $I_{j+1} = m_j \cdot I_j$, for every $1 \leq j < K$, where m_j is an integer. To produce a legal schedule we must find a way to allocate the flows over the basic interval which is I_K. Once we find such an allocation, we can repeat it over and over to create a legal schedule. Although there are other possibilities, we assume that the scheduling algorithm allocates the flows in increasing order of their grant intervals. First flows in F_1 are allocated creating fixed size bins. The flows in F_2 are then allocated into those bins but this time variable size bins are created. After allocating flows in F_3 a new set of variable size bins are created and the process continues until

all groups have been allocated. Similar to the case of two intervals, the bin sizes can be modified. This time, however, each bin is bounded by a different set of flows which means that the constraints on modifying a bin's size are different from bin to bin.

In subsection 4.4 we showed that when grant sizes are smaller than the jitter it is possible to solve the scheduling problem in polynomial time. This property remains true even when there are $K \geq 3$ different grant intervals (see [18] for details). This means that for many practical applications a simple scheduling algorithm is sufficient.

Finally, let us consider the general case where the different grant intervals are not necessarily integer multiples of each other. In this case the problem becomes even harder because of two reasons

1. In order to produce a legal schedule we must now find a legal allocation of the flows over a basic interval whose length equals the least common denominator of all grant intervals. Only then can we repeat the allocations to produce a legal schedule.

2. It is impractical to model the scheduling problem as a bin packing problem since the different grant intervals do not overlap.

6 Concluding Remarks

In this paper we addressed the problem of scheduling CBR flows over a DOCSIS compliant cable network. CBR flows are scheduled using Unsolicited Grant Service where each flow is characterized by its grant size, grant interval, and grant jitter. We considered several special cases of the scheduling problem and suggested a way for developing scheduling algorithms by modeling the problem as a variant of bin packing. We showed that the general scheduling problem is NP-hard even when there are only two different grant intervals; however, when the tolerated jitter is larger than all grant sizes the problem is simpler and a polynomial time scheduling algorithm is sufficient. This result provides good news since in VoIP, and many other practical applications, the tolerated jitter is (considerably) larger than grant sizes.

The scope of this work is rather limited due to size constrains. Several extensions to this work are found in [18] where we present average case analysis of NFJ, define more state dependent algorithms and present the class of state independent algorithms in details. We consider the general scheduling problem (with many grant intervals) in more details and suggest both scheduling and admission control algorithms for it. We also analyze the online version of the scheduling problem where flows are established and torn down over time. There are many other subjects that await further research; the problem of scheduling VBR flows to support video applications is particularly interesting.

References

[1] J. Albrycht. VoIP: putting the pieces together. CED, vol. 24, No. 13, pp. 44-48, December 1998.

[2] B. Beser. Providing enhanced services over DOCSIS. CED, vol. 27, No. 4, pp. 80-84, April 2001.

[3] D. Bushmitch, S. Mukherjee, S. Narayanan, M. Ratty and Shi-Qun. Supporting MPEG video transport on DOCSIS-compliant cable networks. IEEE Journal on Selected Areas in Communications, vol. 18, No. 9, pp. 1581-96, Sept. 2000.

[4] E. G. Coffman, Jr., M. R. Garey and D. S. Johnson. Approximation algorithms for bin packing: A survey. In D. Hochbaum (ed), PSW publishing, Boston. Approximation Algorithms for NP-Hard Problems, pages 46-93, 1996.

[5] Cable Television Laboratories Inc., CableLabs. http://www.cablelabs.com.

[6] Chen-Jian, Qian-Ying, Chen-Huimin, Li-Yingchun and Chen-Jainqiang. Dynamic bandwidth allocation scheme for MCNS DOCSIS. Journal of Shanghai University. vol. 2, No. 4, pp. 328-33, 1998.

[7] M. Droubi, N. Idirene and C. Chen. Dynamic bandwidth allocation for the HFC DOCSIS MAC protocol. Proceedings Ninth International Conference on Computer Communications and Networks. Piscataway, NJ, USA, 2000, pp. 54-60.

[8] Cable Television Laboratories Inc., "Data-Over-Cable Service Interface Specifications - Radio Frequency Interface Specification (status: interim)", April 2000.

[9] M. R. Garey and D. S. Johnson. Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman and Co., San Francisco, 1979.

[10] D. S. Johnson, A. Demers, J. D. Ullman, M. R. Garey and R. L. Graham. Worst-case performance bounds for simple one-dimensional packing algorithms. SIAM J. of Computing, vol. 3, pp. 299-325, 1974.

[11] D. S. Johnson. Fast algorithms for bin packing. Journal of computer and system Science. vol. 8, pp. 272-314, 1974.

[12] D. Jianghong, X. Zhongyang, C. Hao, and D. HuiScheduling. Algorithm for MPEG-2 TS Multiplexers in CATV Networks. IEEE Transaction on Broadcastinf, vol. 46, No. 4, pp. 249-255, December 2000.

[13] H. Kellerer, R. Mansini and M.G. Speranza. Two linear approximation algorithms for the subset-sum problem. European Journal of Operational Research, vol. 120, No. 2, pp.289-296, January 2000.

[14] H. Kellerer, U. Pferschy and M.G. Speranza. An efficient approximation scheme for the subset-sum problem. Algorithms and Computation, 8th International Symposium ISAAC '97, Proceedings, Springer-Verlag, Berlin, Germany, xv+426, pp. 394-403, 1997.

[15] S. Martello and P. Toth. Knapsack Problems: Algorithms and computer Implementations. Jhon Wily and Sons, Ltd., New York, 1990.

[16] Nir Menakerman (Naaman) and Raphael Rom. Bin Packing with Item Fragmentation. Algoritm and Data Structures, Proceeding of the 7th International Workshop WADS 2001. Springer pp. 312-324.

[17] Nir Menakerman (Naaman) and Raphael Rom. Analysis of Transmissions Scheduling with Packet Fragmentation. Discrete Mathematics and Theoretical Computer Science 4, pp. 139-156, 2001.

[18] Nir Naaman and Raphael Rom. Scheduling Constant Bit Rate Flows over a TDMA Channel. Technical report, to appeaear.

[19] Cable Television Laboratories Inc., "PacketCable™ Dynamic Quality-of-Service Specification (status: interim)", August 2000.

[20] R. Rabbat and K.Y Siu. QoS support for integrated services over CATV. IEEE Communication Magazin, vol. 37, No. 1, pp. 64-68, January 1999.

[21] V. Sdralia, C. Smythe, P. Tzerefos and S. Cvetkovic. Performance characterisation of the MCNS DOCSIS 1.0 CATV protocol with prioritised first come first served scheduling. IEEE Transactions on Broadcasting, vol. 45, No. 2, pp. 196-205, June 1999.

[22] P. Tzerefos, V. Sdralia, C. Smythe and S. Cvetkovic. Delivery of low bit rate isochronous streams over the DOCSIS 1.0 cable television protocol. IEEE Transactions on Broadcasting, vol. 45, No. 2, pp. 206-214, June 1999.

[23] Wai Sum Lai. Upstream bandwidth allocation for packet telephony in hybrid fiber-coax systems. Proceedings of the 2000 Symposium on Performance Evaluation of Computer and Telecommunication Systems, San Diego, CA, USA, xii+570 pp. 96-100.