

# ON THE COMPLEXITY OF CHANNEL ASSIGNMENT FOR REAL-TIME FLOWS

Jorge A. Cobb

Department of Computer Science  
The University of Texas at Dallas  
Richardson, TX 75083-0688  
email: cobb@utdallas.edu

## ABSTRACT

Consider a computer network in which adjacent nodes exchange messages via multiple communication channels. Multiple channels between adjacent nodes are desirable due to their cost effectiveness and improved fault-tolerance. We consider the problem of providing deterministic quality of service guarantees in this network. In particular, we consider a set of flows that traverse between a pair of neighboring nodes. Each flow must be assigned to one and only one of the channels joining the pair of nodes. We study the complexity of this flow assignment problem. We show the problem to be NP-hard, even under significant restrictions on the quality of service parameters of flows. We also present a pseudopolynomial solution to the problem when the number of channels is fixed.

## KEY WORDS

Quality of Service, MultiChannels, Scheduling, Complexity.

## 1 Introduction

Packet scheduling protocols that provide deterministic quality of service guarantees flourished in the previous decade (for a survey, see [1]). Many of these protocols are based, one way or another, on earlier work on task scheduling. In particular, they are based on the techniques given in the landmark paper of Liu and Layland on periodic task scheduling [2].

In [2], all tasks share a single resource. In the last few years, there has been significant work in the scheduling of periodic tasks over *multiple* resources [3][4][5]. Even though the theory of periodic task scheduling over multiple resources has begun to show promise, there has been little work to develop packet scheduling protocols over multiple channels between network nodes. This is due in part to the belief that multiple channels between nodes is either not practical or uncommon. However, there is significant evidence to the contrary.

In a recent paper [6], it was argued that packet reordering is not a “pathological” problem, but rather a normal occurrence. That is, packets are reordered not only due to route changes (which are rare), but also due to inherent parallelism in the network. One cause for this parallelism

is the aggressive deployment of parallel channels between nodes. As stated in [6], in a survey of 38 major service providers in 1997, only two had no parallel channels between its nodes. The reason for this approach is that it often reduces equipment and trunk costs. That is, it is often more cost effective to put two components in parallel than to use one component that has twice the speed. In addition, it improves fault-tolerance.

Another technology that provides multiple channels between nodes is the establishment of light-paths in wave-division multiplexed (WDM) optical networks. Although the establishment of light-paths is usually semi-permanent, recent work allows the establishment of light-paths on-demand, to reflect the changes in network load over time [7]. If there is a significant load between two nodes in the network, it is possible that a single light-path may not provide enough bandwidth between them, which calls for the establishment of additional light-paths between these nodes. Thus, multiple communication channels may be established between two nodes (for more examples of multi-channel systems, see [8].)

Given the evidence of multiple channels between nodes presented above, it is likely that multiple channels will continue to exist. Therefore, it is reasonable to assume that if a guaranteed quality of service protocol is deployed on a global scale, it will likely traverse at some point network nodes with multiple channels between them. Thus, the problem of scheduling packets for guaranteed quality of service in the presence of multiple channels must be studied.

Two methods exist to forward packets over multiple channels between nodes. One is to distribute the packets of each real-time flow over all the channels. A few recent papers deal with this approach [8][9]. The other approach consists of forwarding the packets of each real-time flow over one and only one of the channels, effectively fixing the set of flows that share each of the channels. We focus on this latter approach.

Determining if a set of real-time flows can satisfy their QoS deadlines over a single channel has been studied significantly in the past [10][11][12]. In our approach, since the set of flows at each channel is fixed, we can easily check each channel to decide if its flows are schedulable. However, the complexity lies in deciding which channel

will be assigned to each flow.

In this paper, we study the complexity of assigning real-time flows to channels. We show the problem to be NP-complete, even under significant restrictions on the quality of service parameters of flows. We also present a pseudopolynomial solution to the problem when the number of channels is fixed. This implies that the problem is tractable in the case where the parameters of the flow (such as rate, delay, etc.) have a moderate range of values.

## 2 Real-Time Flow Model

In this section, we define the network model for single-channel scheduling, and also define the quality of service that the model assigns to each flow of packets. We base our model on the models of [11] and [13]. We present multiple-channel scheduling in Section 4.

A *network* is a set of nodes interconnected by point-to-point communication channels. For every pair of nodes  $a$  and  $b$ , there is at most one channel from  $a$  to  $b$  and at most one channel from  $b$  to  $a$ . Every output channel in a node is equipped with a scheduler. From the input channels, the scheduler receives packets from flows whose path include the output channel of the scheduler. The scheduler then chooses the transmission order and transmission time of these packets over its output channel. This is shown in Figure 1.

We say a packet is *forwarded* to the output channel when its first bit is transmitted over the output channel. We say a packet *exits* a scheduler when the last bit of the packet is transmitted by the output channel of the scheduler, and hence, the output channel becomes idle at this moment. To simplify our discussion, we ignore channel propagation delays, since they simply add a constant delay to each packet.

A *flow* is a sequence of packets that traverse the network without reorder from a source node to a destination node. We adopt the following notation for each flow  $f$  and each scheduler  $s$  along the path of  $f$ .

$p_{f,i}$	$i^{\text{th}}$ packet of $f$ , $i \geq 1$ .
$L_{f,i}$	length of packet $p_{f,i}$ .
$L_f^{\text{max}}$	maximum packet length of $f$ .
$L_f^{\text{min}}$	minimum packet length of $f$ .
$L_{\text{max}}^s$	maximum packet length at $s$ .
$A_{f,i}^s$	arrival time of $p_{f,i}$ at scheduler $s$ .
$E_{f,i}^s$	exit time of $p_{f,i}$ from $s$ .
$C^s$	output bandwidth of scheduler $s$ .

A *real-time flow*  $f$  is a flow with real-time constraints. These constraints are expressed by the following two parameters.

$\delta_f^s$	per-hop delay of $f$ at $s$ .
$R_f$	forwarding rate reserved for $f$ .

In principle, the end-to-end delay of packets from  $f$  should be the sum of  $\delta_f^s$  for all schedulers  $s$  along the path of  $f$ . This would be true if the deadline of each packet

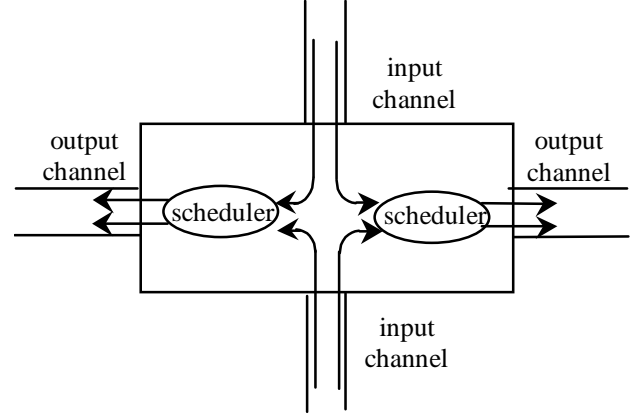


Figure 1. Output channels and their schedulers.

were measured with respect to its arrival time to a scheduler. However, due to possible burstiness in the flow, the deadline of a packet is instead defined with respect to the virtual arrival time (also known as start-time) of the packet [11][13], which is defined as follows.

Consider a scheduler  $s$  and a flow  $f$ . We define the *start-time*  $S_{f,i}^s$  and *finish-time*  $F_{f,i}^s$  of packet  $p_{f,i}$  at scheduler  $s$  as follows. Assume  $s$  were to forward the packets of  $f$  at exactly  $R_f$  bits/sec.. Then,  $S_{f,i}^s$  is the time at which the first bit of  $p_{f,i}$  is forwarded by  $s$ , and  $F_{f,i}^s$  is the time at which the last bit of  $p_{f,i}$  is forwarded by  $s$ . More formally, let  $f$  be an input flow of scheduler  $s$ . Then,

$$\begin{aligned}
 F_{f,i}^s &= S_{f,i}^s + \frac{L_{f,i}}{R_f}, \text{ for every } i, i \geq 1 \\
 S_{f,1}^s &= A_{f,1}^s \\
 S_{f,i}^s &= \max(A_{f,i}^s, F_{f,(i-1)}^s), \text{ for every } i, i > 1
 \end{aligned}$$

The deadline of each packet is derived from its start time and its per-hop delay, as follows [11, 14, 13, 15].

### Definition 1 (Packet Deadline)

The deadline,  $D_{f,i}^s$  of packet  $p_{f,i}$  at scheduler  $s$  is defined as follows, for every  $i, i \geq 1$ ,

$$D_{f,i}^s = S_{f,i}^s + \delta_f^s$$

■

This definition of delay is broad enough to encompass the delay provided by many scheduling protocols. For example, by choosing  $\delta_f^s = L_f^{\text{max}}/R_f$ ,  $\delta_f^s$  becomes the delay of virtual-clock and weighted-fair-queuing protocols [16][17]. Another example is the real-time channel model, [12][10] where each flow has constant packet size and a minimum packet inter-arrival time.

In this paper, we focus on a single-hop along the path of flow  $f$ . For completeness, however, we state below the

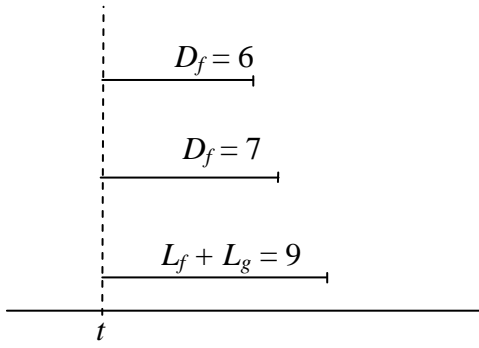


Figure 2. Scheduling conflict example

well-known end-to-end delay bound of a packet across a sequence of schedulers. The end-to-end bound below which was proven in [14][13], and it also follows from the results in [10].

Let  $t_1, t_2, \dots, t_k$  be a sequence of schedulers traversed by flow  $f$ . For all  $i$ ,

$$S_{f,i}^{t_k} \leq S_{f,i}^{t_1} + \sum_{x=1}^{k-1} \delta_f^{t_x} \quad (1)$$

Hence,

$$E_{f,i}^{t_k} \leq D_{f,i}^{t_k} \leq S_{f,i}^{t_1} + \sum_{x=1}^k \delta_f^{t_x} \quad (2)$$

Note that the end-to-end delay, with respect to the start time of the packet at the first scheduler, is just the sum of its per-hop delay  $\delta_f^s$  at each hop. This is regardless of how “bursty” the flow is. That is, a large burst will affect the value of  $S_f^{t_1}$ , but the end-to-end delay with respect to  $S_f^{t_1}$  remains bounded by a per-hop delay of  $\delta_f^s$ .

### 3 Single-Channel Scheduling

In the next section, we consider the case of networks with multiple channels between nodes. However, before doing so, we first overview in this section the main results of scheduling over a single channel.

A set of flows are said to be *schedulable* if all packets of all flows are able to exit the scheduler by their deadline. To determine this, a scheduling test must be performed on the set of flows. That is, simply checking that their reserved rates is less than the channel capacity is not sufficient; the per-hop delays of the flows must also be taken into consideration.

For example, consider two flows  $f$  and  $g$ , with the following parameters.

$$\begin{array}{ll} L_f & = 4 & L_g & = 5 \\ \delta_f & = 6 & \delta_g & = 7 \\ R_f & = 4/10 & R_g & = 5/10 \\ C & = 1 & & \end{array}$$

Consider Figure 2, where one packet from each flow arrives at time  $t$ . Note that  $R_f + R_g \leq C$ , and hence, there is enough bandwidth for the flows. However, regardless of how these two packets are scheduled, at least one of them must miss its deadline.

In [18], a necessary and sufficient condition was developed to ensure flows are schedulable. The condition is based on the notion of the “appetite” of a flow, that is, the deadline density, which is defined as follows

#### Definition 2

**(Appetite)** The appetite of a flow  $f$  during interval  $[t, t']$ , denoted  $\lambda(f, t, t')$ , is the total number of bytes of packets from  $f$  that arrive during the interval and whose deadline is within the interval. That is,

$$\lambda(f, t, t') = \left\langle \sum i : t \leq A_{f,i} \leq t' \wedge t \leq D_{f,i} \leq t' : L_{f,i} \right\rangle$$

Let  $\Lambda(f, \varepsilon)$  be the upper bound on  $\lambda(f, t, t')$  over all intervals  $[t, t']$  of size  $\varepsilon$ , i.e.,

$$\Lambda(f, \varepsilon) = \langle \max t, t' : (t' - t) = \varepsilon : \lambda(f, t, t') \rangle$$

**(Bounded Appetite)** Consider a work-conserving scheduler of capacity  $C$ . The scheduler is said to have bounded appetite iff,

$$\langle \forall \varepsilon :: (\sum f :: \Lambda(f, \varepsilon)) \leq \varepsilon \cdot C \rangle \quad (3)$$

■

The above definition of appetite measures how “dense” the deadline requirements are for each flow. Thus, if too much appetite exists from all the flows, then the scheduler is unable to satisfy all of them. A bounded appetite is thus required to be able to schedule flows [18].

#### Statement 1 (Scheduling Condition)

For a scheduler, if packets are scheduled in a work-conserving manner in order of their deadlines, then each packet exits the scheduler no later than its deadline iff the scheduler has bounded appetite.

From Statement 1, the scheduling test consists only in checking that the the scheduler has bounded appetite. In this paper, we consider three cases for the packet-sizes of the input flows of a scheduler, each of which yields different expressions for the flow appetite. The appetite bounds below have been shown in [11] and [10].

#### Statement 2

**(Intra-flow Max-Packet-Size Appetite)** For a flow  $f$  with a maximum packet size  $L_f^{max}$  and no minimum packet size, the appetite of  $f$  is as follows.

$$\Lambda(f, \varepsilon) = L_f^{max} + (\varepsilon - \delta_f) \cdot R_f \quad (4)$$

if  $\delta_f \leq \varepsilon$ , zero otherwise.

**(Intra-flow Constant-Packet-Size Appetite)** For a real-time flow  $f$  with a fixed packet size  $L_f$ , the appetite of  $f$  is as follows.

$$\Lambda(f, \varepsilon) = L_f + \left\lfloor \frac{\varepsilon - \delta_f}{L_f/R_f} \right\rfloor \cdot L_f$$

if  $\delta_f \leq \varepsilon$ , zero otherwise.

**(Intra-flow Min-Packet-Size Appetite)** For a flow  $f$  with a maximum packet size  $L_f^{max}$  and a minimum packet size  $L_f^{min}$ , where  $2 \cdot L_f^{min} \leq L_f^{max}$ , the appetite of  $f$  is as follows.

$$\Lambda(f, \varepsilon) = \begin{cases} 0 & \text{if } \varepsilon < \delta_f \\ L_f^{max} & \text{if } \delta_f \leq \varepsilon < \delta_f + L_f^{min}/R_f \\ L_f^{max} + (\varepsilon - \delta_f) \cdot R_f & \text{if } \delta_f + L_f^{min}/R_f \leq \varepsilon \end{cases}$$

■

From Statements 1 and 2, to check if a set of flows are schedulable we simply check, for every interval size  $\varepsilon$ , if the sum of the appetites of all flows is at most  $\varepsilon \cdot C$ .

However, note that only a limited values of  $\varepsilon$  need to be checked. For example, in the case of max-packet-size appetite, only  $\varepsilon$  values which are equal to  $\delta_f$  for each flow  $f$  need to be checked. Details on which values of  $\varepsilon$  suffice to be checked may be found in [11] and [10].

## 4 Multi-Channel scheduling

In this section, we generalize our network model to allow multiple channels between neighboring computers. This is illustrated in Figure 3(a). Here, a computer  $S$  has three neighbors, and it receives input flows from two of these neighbors,  $A$  and  $B$ , and it outputs these flows to the third neighbor,  $T$ . The computer has two input channels with each input neighbor, and it also has two output channels with its output neighbor. For simplicity, we will assume all channels to/from the same neighbor have equal capacity. Channels with differing capacities will be discussed in the concluding remarks section.

Assume  $A$  is forwarding two flows,  $x$  and  $y$ , to  $S$ , and  $S$  forwards these flows to  $T$ . Similarly,  $B$  forwards flows  $f$  and  $g$  to  $S$ , who in turn also forwards these to  $T$ . Note that is irrelevant which channel is used to forward the packets of these flows from  $S$  to  $T$  since both output channels lead to  $T$ .

In principle, there are two paradigms to forward the input flows into the output channels. The first one is shown in Figure 3(b), where a single scheduler is in charge of all the output channels, and the packets of each flow are distributed over all the channels. We have explored this case in our previous work [9] (and work of others [8] and [4]). In [9], we show how any scheduler for a single channel can be adapted to the multi-channel model without a significant increase in packet delay.

The second paradigm is shown in Figure 3(b), where each channel is handled by an independent scheduler, and the packets of each flow are sent over a single channel to the next computer. Since the two channels are independent and do not share any flows, this is simply two independent cases of single-channel scheduling. Thus, the scheduling tests of Section 3 can be applied independently to each channel.

However, note that the problem of assigning flows to channels still remains. That is, the assignment of each flow to an output channel must be done carefully to ensure that, at each channel, its set of output flows are schedulable, i.e., that no packet misses its deadline.

### Definition 3 (Flow Assignment)

Given  $N$  schedulers,  $s_1, \dots, s_n$ , and a set  $\Phi$  of real-time flows, the flow assignment problem consists of dividing  $\Phi$  into  $N$  disjoint subsets,  $\Phi_1, \dots, \Phi_N$ , such that each  $\Phi_i$  is schedulable at scheduler  $s_i$ .

Below, we focus on the complexity of the flow assignment problem. In particular, we show that is NP-complete. We also evaluate the “hardness” of the problem within the NP-complete class, i.e., whether a pseudopolynomial solution exists for the problem.

## 5 Flow Assignment Complexity

As mentioned earlier, a real-time flow is identified by three parameters: its packet size, its reserved rate, and its per-hop delay. It is easy to show that if all three parameters can be different from one flow to another, then the flow assignment problem is NP-complete.

Instead, we show that if two of these parameters are equal for all flows and the remaining parameter may differ from one flow to another, then the problem still remains NP-complete. In particular, we show that NP-completeness is preserved under the following conditions.

- (i) **Inter-flow packet size:** If flows can have different packet sizes, but the rate and delay of all flows are the same.
- (ii) **Inter-flow reserved rate:** If flows can have different rates, but have the same packet size and delay.

It remains an open question whether NP-completeness is preserved in the case where the delay varies from one flow to another but the packet size and rate of all flows are fixed.

Below, we first focus on the intra-flow max-packet-size case of Section 3. I.e., each flow  $f$  has a maximum packet size  $L_f^{max}$  and no minimum packet size. The remaining two intra-flow cases of Section 3 are similar and will be addressed in Section 5.3.

Note that the value of  $L_f^{max}$  may vary from one flow to another in case (i) above of inter-flow packet size, but must be equal for all flows in case (ii) above.

For both (i) and (ii) above, we will use the well-known 3-partition problem in our proofs of NP-completeness, which is defined as follows.

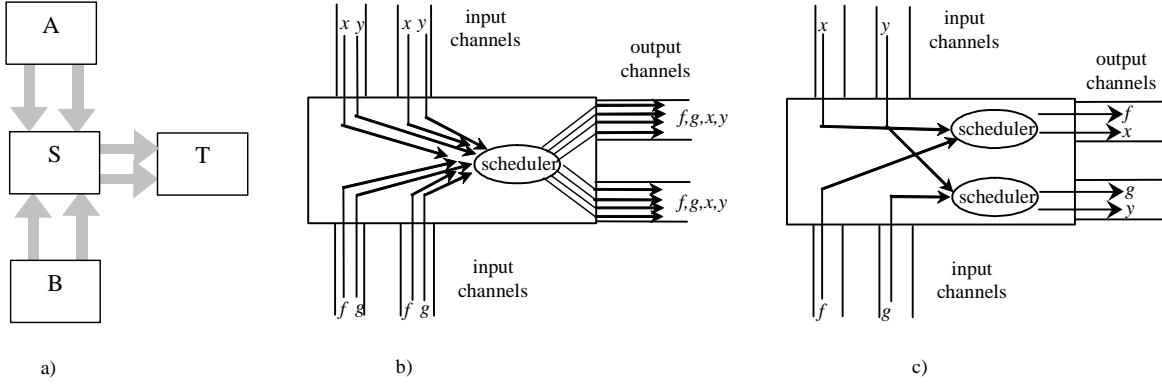


Figure 3. Multi-Channel Scheduling

### Statement 3 (3-Partition)

The following problem is an NP-complete problem.

INSTANCE: A sequence of  $3N$  integers,  $A = \langle a_1, a_2, \dots, a_N \rangle$  whose sum equals  $Nb$ .

QUESTION: Can  $A$  be partitioned into  $N$  disjoint subsets of 3 elements each such that the sum of each subset is exactly  $b$ ? ■

Below, we focus on showing the reduction from 3-partition to our flow assignment problem to prove the problem is NP-hard. Showing that the problem is NP-easy is straightforward.

### 5.1 Inter-flow Packet Size

We next proceed to show that the flow assignment problem under case (i) above and the intra-flowMax-Packet-Size scheduling test of Section 3 is NP-complete.

#### Theorem 1 (Inter-flow Packet Size)

The following flow assignment problem is NP-complete.

INSTANCE:  $N$  channels of equal capacity, a set  $\Phi$  of real-time flows, where each flow  $f$  has a maximum packet size  $L_f^{max}$ , and all flows have no minimum packet size, the same reserved rate  $R$ , and the same per-hop delay  $\delta$ .

QUESTION: Can  $\Phi$  be divided into  $N$  disjoint subsets,  $\Phi_1, \dots, \Phi_N$ , such that each  $\Phi_i$  is schedulable in one of the channels.

**Proof:** We must exhibit a polynomial-time reduction from an instance  $I_P$  of the 3-partition problem to an instance  $I_\Phi$  of the flow assignment problem, so that,  $I_P$  can be partitioned iff  $I_\Phi$  can be scheduled.

Our reduction is as follows. If instance  $I_P$  consists of  $3N$  numbers, then  $I_\Phi$  consists of  $N$  channels. Each number in  $I_P$  is mapped to a flow in  $I_\Phi$ . The parameters of the flow are as follows:

- The capacity of the output channel is 1.
- The rate  $R$  common to all flows is  $1/3$  of the capacity of an output channel.

- The delay  $\delta$  common to all flows is equal to parameter  $b$  in  $I_P$ .
- The maximum packet size  $L_f^{max}$  of the flow is equal to the magnitude of its corresponding number in  $I_P$ .

We first show that if  $I_P$  is partitionable, then  $I_\Phi$  is schedulable. That is, we need to show that flows can be assigned to channels such that each channel has bounded appetite, where the appetite of a flow  $f$  is given by Relation (4).

Let each group  $\Phi_i$  be the three flows in  $I_\Phi$  that correspond the three numbers in  $I_P$  that add up to  $b$ .

Given that all three flows in  $\Phi_i$  have the same value of  $\delta$  and  $R$ , from Relations (4) and (3), to be schedulable,  $\Phi_i$  must satisfy the following.

$$\left\langle \sum f : f \in \Phi_i : L_f^{max} + (\varepsilon - \delta) \cdot R \right\rangle \leq \varepsilon \cdot C \quad (5)$$

where  $C$  is the channel capacity, and for any  $\varepsilon$  such that  $\delta \leq \varepsilon$ .

Since  $|\Phi_i| = 3$ , simplifying,

$$\left\langle \sum f : f \in \Phi_i : L_f^{max} \right\rangle + 3 \cdot (\varepsilon - \delta) \cdot R \leq \varepsilon \cdot C$$

Since  $R = 1/3$  and  $C = 1$ , simplifying once more

$$\left\langle \sum f : f \in \Phi_i : L_f^{max} \right\rangle \leq \delta$$

The above relation is true because  $\delta$  was chosen to be equal to  $b$  from  $I_P$ , and the three numbers corresponding to the flows in  $\Phi_i$  from the partition of  $I_P$  add to exactly  $b$ .

We next show that if  $I_P$  is not partitionable, then  $I_\Phi$  is not schedulable.

Since  $R = 1/3$  and  $C = 1$ , at most three flows can be scheduled at each channel. Furthermore, since the number of flows is  $3 \cdot N$ , and there are  $N$  channels, exactly three flows must be scheduled at each channel.

However, if  $I_P$  cannot be partitioned, any partitioning of the flows into groups of three will have at least one group of three where the sum of the numbers (i.e. the packet sizes

in  $I_\Phi$ ) add to more than  $b$  (i.e. more than  $\delta$ ). Hence, Relation (5) above will not be satisfied for  $b = \delta = \varepsilon$ . Intuitively, think of the case where one packet from each of the three flows arrive together. More than  $\delta = b$  time units are necessary to forward them to the output channel. ■

## 5.2 Inter-flow Reserved Rate

We next proceed to show that the flow assignment problem under case (ii) above and the Max-Packet-Size scheduling test of Section 3 is NP-complete.

### Theorem 2 (Inter-flow Reserved Rate)

*The following flow assignment problem is NP-complete.*

*INSTANCE:  $N$  channels of equal capacity, a set  $\Phi$  of real-time flows, where each flow  $f$  has a reserved rate  $R_f$ , and all flows have the same maximum packet size  $L^{max}$ , no minimum packet size, and the same per-hop delay  $\delta$ .*

*QUESTION: Can  $\Phi$  be divided into  $N$  disjoint subsets,  $\Phi_1, \dots, \Phi_N$ , such that each  $\Phi_i$  is schedulable in one of the channels.*

**Proof:** We again exhibit a polynomial-time reduction from an instance  $I_P$  of the 3-partition problem to an instance  $I_\Phi$  of the flow assignment problem, so that,  $I_P$  can be partitioned iff  $I_\Phi$  can be scheduled.

Our reduction is as follows. As before, if instance  $I_P$  consists of  $3N$  numbers, then  $I_\Phi$  consists of  $N$  channels. Each number in  $I_P$  is mapped to a flow in  $I_\Phi$ . However, the parameters of the flow are different as follows:

- The capacity of the output channel equals  $b$ .
- The maximum packet size  $L^{max}$  of all flows is 1.
- The delay  $\delta$  common to all flows is  $3/b$ .
- The reserved rate  $R_f$  of flow  $f$  is equal to the magnitude of  $f$ 's corresponding number in  $I_P$ .

We first show that if  $I_P$  is partitionable, then  $I_\Phi$  is schedulable.

As in the proof of Theorem 1, let each group  $\Phi_i$  be the three flows in  $I_\Phi$  that correspond the three numbers in  $I_P$  that add up to  $b$ .

Given that all three flows in  $\Phi_i$  have the same value of  $L^{max}$  and  $\delta$ , from Relations (4) and (3), to be schedulable,  $\Phi_i$  must satisfy the following.

$$\left\langle \sum f : f \in \Phi_i : L^{max} + (\varepsilon - \delta) \cdot R_f \right\rangle \leq \varepsilon \cdot C \quad (6)$$

where  $C$  is the channel capacity, and for any  $\varepsilon$  such that  $\delta \leq \varepsilon$ .

Simplifying from our choices for the parameters of  $f$ ,

$$\left\langle \sum f : f \in \Phi_i : 1 + (\varepsilon - 3/b) \cdot R_f \right\rangle \leq \varepsilon \cdot b$$

where  $\varepsilon \geq 3$ . Simplifying again,

$$3 + (\varepsilon - 3/b) \cdot \left\langle \sum f : f \in \Phi_i : R_f \right\rangle \leq \varepsilon \cdot b$$

From the partition of  $I_P$ , the sum of  $R_f$  over all three flows in  $\Phi_i$  equals  $b$ . The relation then trivially holds.

We next show that if  $I_P$  is not partitionable, then  $I_\Phi$  is not schedulable.

From the choice for  $\delta$ , at most three packets can be transmitted without violating the deadline, hence, no more than three flows can be assigned to each channel. Furthermore, from the number of flows in  $I_\Phi$ , it must be that three flows are assigned to each channel.

Since  $I_P$  is not partitionable, all partitions of  $I_P$  yields at least one group of three numbers that adds to more than  $b$ . For this group, consider the three flows in the flow group  $\Phi_i$  corresponding to these three numbers. Since  $R_f$  is given the value of the number in  $I_P$  corresponding to  $f$ , the sum of the rates of the three flows in  $\Phi_i$  is greater than  $b$ , which is the rate of the channel. Hence,  $\Phi_i$  is not schedulable. Thus, all partitions of the flow are not schedulable. ■

## 5.3 Intra-flow Scheduling

Above, we have only considered the intra-flow max-packet-size case. Below, we argue that the combination of either of the inter-flow cases with any of the intra-flow cases yields an NP-complete problem.

**Corollary 1** *The combination of any of the three intra-flow cases from Section 3 with either of the two inter-flow cases of Theorems 1 and 2 yield an NP-complete problem*

**Proof:** In the proofs of Theorems 1 and 2, we had to show that  $I_P$  is partitionable iff  $I_\Phi$  is schedulable.

First, we showed that if  $I_P$  is partitionable, then  $I_\Phi$  is schedulable. We showed this by arguing that the sum of the appetites of the flows sharing a channel does not exceed the capacity of the channel, i.e., that the channel has bounded appetite. We used Relation (4) since we considered the intra-flow max-packet-size case.

However, it can be easily shown that the appetite for the intra-flow max-packet-size case is always at least as big as the appetite for the other two intra-flow cases. This is because, since packets can have an arbitrarily small size, “tiny” packets can be used to pack as much appetite as possible within an interval. Hence, if  $I_\Phi$  is schedulable under the intra-flow max-packet-size case, then it is also schedulable under the other two cases.

Second, we showed that if  $I_P$  is not partitionable, then  $I_\Phi$  is not schedulable. We showed this in Theorem 1 by noting that if a maximum-sized packet of each flow arrive concurrently, then at least one packet will miss its deadline. This is independent of the intra-flow packet-size case being chosen. Similarly, in Theorem 2, we argued that the sum of the rates of the flows exceeds the channel capacity. This is also independent of the intra-flow packet-size case being chosen.

Hence, any choice of intra-flow packet size will yield an NP-complete problem. ■

## 6 Pseudopolynomial Solution

For some NP-complete algorithms, it is possible to find an algorithm that solves the problem in a time polynomial in the length of the input, provided each of the values used in the input is not “too large”. Therefore, a tractable solution is known if the input values are within a sensible range. These types of solutions are said to be *pseudo-polynomial* solutions.

**Definition 4** *An algorithm that solves a problem  $\Pi$  will be called a pseudo-polynomial time algorithm for  $\Pi$  if its time complexity function is bounded above by a polynomial function of the two variables  $\text{Length}[I]$  and  $\text{Max}[I]$ , where  $I$  is the input to the problem.*

*Integer  $\text{Length}[I]$  corresponds to the number of symbols used to describe any instance  $I$  under some reasonable encoding scheme for  $\Pi$ .*

*Integer  $\text{Max}[I]$  corresponds to the magnitude of the largest number in  $I$ .*

For some NP-complete problems, such as the 3-PARTITION problem, there is no pseudopolynomial solution (unless  $P = NP$ ). For others, however, such as the simple PARTITION problem, a pseudo-polynomial solution is known.

Some scheduling problems over multiprocessors are known to have pseudopolynomial solutions in the case where the number of processors are fixed [19]. Below, we consider the flow assignment problem where  $N$ , the number of channels, is fixed. In particular, we consider the case of intra-flow max-packet-size, where each flow has an upper bound and no lower bound on its packet size. We make no inter-flow assumptions, and hence, flows can have different deadlines, rates, and maximum packet sizes.

### Theorem 3 (Pseudopolynomial Solution)

*A pseudopolynomial solution exists for the following problem.*

*INSTANCE: channel capacity, a set  $\Phi$  of real-time flows, where each flow  $f$  has a maximum packet size  $L_f^{max}$ , a reserved rate  $R_f$ , and a per-hop delay  $\delta_f$ .*

*QUESTION: Can  $\Phi$  be divided into  $N$  disjoint subsets (where  $N$  is constant),  $\Phi_1, \dots, \Phi_N$ , such that each  $\Phi_i$  is schedulable in one of the channels.*

**Proof:** First, we order the flows in  $\Phi$  by increasing deadline. That is, let  $f^1$  be the flow with least  $\delta$  value (ties broken arbitrarily), and  $f^{|\Phi|}$  be the flow with the greatest  $\delta$  value. Our algorithm iterates over the flows in  $\Phi$  according to the order defined above.

The algorithm is based on dynamic programming, similar to the pseudopolynomial method for the PARTITION problem [19].

We build a boolean-bit array. This array has  $2 \cdot N + 1$  indices. As we iterate over the ordered flows in  $\Phi$ , the first index indicates the flow number we currently considering. We denote this index by  $i$ .

There are two remaining indices for each of the channels. Let  $R_n$  and  $A_n$  be the indices corresponding to channel  $n$ .  $R_n$  corresponds to the sum of the rates of the flows assigned to channel  $n$ , and  $A_n$  corresponds to the sum of the appetites of the flows assigned to channel  $n$  during an interval of size  $\delta_{f^i}$ .

For example, let  $N = 2$ , and let  $B$  be our bit array. Then,  $B[i, R_1, A_1, R_2, A_2]$  will be set to true if and only if, the first  $i$  flows of  $\Phi$  can be separated into two sets  $\Phi_1$  and  $\Phi_2$ , such that:

- The sum of the rates of  $\Phi_1$  (respectively  $\Phi_2$ ) is  $R_1$  (respectively  $R_2$ ).
- The maximum appetite of  $\Phi_1$  (respectively  $\Phi_2$ ) over a period of length  $\delta_{f^i}$  is  $A_1$  (respectively  $A_2$ ).
- Each of  $A_1$  and  $A_2$  is at most  $\delta_{f^i} \cdot C$ , where  $C$  is the bandwidth of each channel (i.e.  $\Phi_1$  and  $\Phi_2$  are schedulable).

We continue our example with  $N = 2$ ; the generalization to any constant  $N$  is straightforward. Consider now filling all entries in  $B$  with first index  $i$  assuming all entries for index  $i - 1$  have been filled.

Assume first for simplicity that  $\delta_{f^i} = \delta_{f^{i-1}}$ . Note that flow  $i$  would add a rate of  $R_{f^i}$  to the channel it is assigned to and an appetite of  $L_{f^i}^{max}$ . Then, entry  $B[i, R_1, A_1, R_2, A_2]$  can be set to true iff adding the flow to the first or to the second channel will result in a schedulable set of flows at each channel. That is, iff one of the following two conditions hold,

$$B[i - 1, R_1 - R_{f^i}, A_1 - L_{f^i}^{max}, R_2, A_2] \wedge A_1 \leq \delta_{f^i} \cdot C$$

$$B[i - 1, R_1, A_1, R_2 - R_{f^i}, A_2 - L_{f^i}^{max}] \wedge A_2 \leq \delta_{f^i} \cdot C$$

The case where  $\delta_{f^i} > \delta_{f^{i-1}}$  is similar, except that we have to consider that the appetites from index  $i - 1$  need to be adjusted for the larger interval  $\delta_{f^i}$ . That is, one of the following two conditions hold.

$$B[i - 1, R_1 - R_{f^i}, A'_1, R_2, A_2] \wedge A_1 \leq \delta_{f^i} \cdot C$$

$$B[i - 1, R_1, A_1, R_2 - R_{f^i}, A'_2] \wedge A_2 \leq \delta_{f^i} \cdot C$$

where

$$A'_1 = A_1 - (L_{f^i}^{max} + (\delta_{f^i} - \delta_{f^{i-1}}) \cdot R_1)$$

$$A'_2 = A_2 - (L_{f^i}^{max} + (\delta_{f^i} - \delta_{f^{i-1}}) \cdot R_2)$$

Therefore, at the end of the algorithm, if there is any element in  $B$  with first index  $i = |\Phi|$  equal to true, then the set of flows are schedulable.  $\blacksquare$

## 7 Concluding Remarks

In this paper, we have studied the complexity of the flow assignment problem over multiple channels. Some open problems remain, such as the complexity when the delay may vary from one flow to another, while the rate and packet size is the same across all flows. In addition, pseudopolynomial solutions for intra-flow constant-packet-size and min-packet-size remain to be found.

A future research direction is the development of efficient heuristics for the flow assignment problem. In addition, we have studied the problem where the set of flows to be scheduled are fixed. On-line algorithms that balance new incoming flows over multiple channels with existing flows should also be pursued.

We have explicitly made the assumption of all channels having equal capacity. Channels with different capacity do not affect the result. In addition, we have implicitly made the assumption in our scheduling tests that packets may be preempted. If not, then the scheduling test needs only to be slightly modified by subtracting the small term  $L_{max}/C$ , i.e., one packet transmission time, from the right-hand-side of Relation (3).

## References

- [1] Zhang H., "Service Disciplines for Guaranteed Performance Service in Packet-Switching Networks", *Proceedings of the IEEE*, Vol. 93, No. 10, Oct. 1995.
- [2] Liu C., Layland J., "Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment", *Journal of the ACM*, Vol. 20, Jan 1973.
- [3] Baruah, S., Cohen, N, Plaxton, G, Varvel, D., "Proportionate Progress: A Notion of Fairness in Resource Allocation", *Algorithmica*, 15:600-625, 1996.
- [4] Baruah, S., Gherke, J., Plaxton, G., "Fast Scheduling of Periodic Tasks on Multiple Resources", *Proc. Intl. Parallel Processing Symposium*, 1995.
- [5] Moir, M., Ramamurthy, S., "Fair Scheduling of Fixed and Migrating Periodic Tasks on Multiple Resources", *IEEE Real-Time Systems Symposium*, 1999.
- [6] Bennet, J.C.R., Partridge, C., Shtetman, N., "Packet Reordering is not Pathological Network Behavior", *IEEE/ACM Trans. on Networking*, Vol 7 No 6, Dec. 1999.
- [7] Fumagalli, A., Cai, J., Chlamtac, I., "A Token Based Protocol for Integrated Packet and Circuit Switching in WDM", *Proceedings of the IEEE GLOBECOM Conference*, 1998.
- [8] Blanquer J.M., Ozden B., "Fair Queuing for Aggregated Multiple Links", *Proceedings of the ACM SIGCOMM Conference*, 2001.
- [9] Cobb J., Lin M., "End-to-End Delay Guarantess for Multi-Channel Schedulers", *Proceedings of the IEEE International Workshop on Quality of Service*, May 2002.
- [10] Zheng Q., Shin K.G., "On the Ability of Establishing Real-Time Channels in Point-to-Point Packet-Switched Networks", *IEEE Transactions on Comm.*, Vol 42, No. 2/3/4, 1994.
- [11] Cobb J., "An In-Depth Look at Flow Aggregation", *Proceedings of the IEEE International Conference on Network Protocols*, 1999.
- [12] D. Ferrari, D. Verma, "A Scheme for Real-Time Channel Establishment in Wide-Area Networks", *IEEE Journal on Selected Areas in Communications*, 8(3), April 1990.
- [13] Figueira N., Pasquale J., "Leave-in-Time: A New Service Discipline for Real-Time Communications in a Packet-Switching Data Network", *Proceedings of the ACM SIGCOMM Conference*, 1995.
- [14] Cobb J., Gouda M., "Flow Theory", *IEEE/ACM Transactions on Networking*, October 1997.
- [15] Goyal P, Lam S., Vin H., "Determining End-to-End Delay Bounds in Heterogeneous Networks", *Proceedings of the NOSSDAV workshop*, 1995.
- [16] Parekh A. K. J., Gallager R., "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single Node Case", *IEEE/ACM Transactions on Networking*, 1(3):344-357, June 1993.
- [17] Xie G., Lam S., "Delay Guarantee of Virtual Clock Server", *IEEE/ACM Transactions on Networking*, Dec. 1995.
- [18] Figueira N., Pasquale J., "A Schedulability Condition for Deadline-Ordered Service Disciplines", *IEEE/ACM Transactions on Networking*, Vol. 5 No. 2, April 1997.
- [19] Garey, M., and Johnson, D., "A Guide to the Theory of NP-Completeness", Freeman and Company, N.Y., 1979.