On Slot Reuse for Isochronous Services in DQDB Networks

Ching-Chih Han†, Chao-Ju Hou‡, and Kang G. Shin†

†*Real-Time Computing Laboratory*
Dept. of Elect. Eng. and Comput. Sci.
The University of Michigan
Ann Arbor, MI 48109
{cchan,kgshin}@eecs.umich.edu

‡*Computer Engineering Division*
Dept. of Elect. and Comput. Eng.
The University of Wisconsin
Madison, WI 53706
jhou@ece.wisc.edu

Abstract

DQDB is a MAC protocol jointly adopted by IEEE and ANSI as the candidate protocol for MANs, and has been studied by many researchers. In [1], we laid a formal basis for guaranteeing the timely delivery of isochronous (real-time) messages with hard deadlines, and devised a slot allocation scheme for allocating pre-arbitrated (PA) slots to isochronous message streams in DQDB networks. In this paper, we extend our work in [1] and address on how to improve the performance (in terms of bandwidth utilization) of the slot allocation scheme using the concept of slot reuse. We devise several slot reuse schemes to assign spatially non-intersecting message streams to the same virtual connections (i.e., the sets of PA slots identified by the same VCI numbers). The proposed slot reuse schemes are simple, can be easily incorporated into the slot allocation scheme in [1], and require only a minor change in the current DQDB standards.

1 Introduction

The problem of guaranteeing the timely delivery of messages has drawn considerable attention, especially in the areas of voice/video data transmission over a data network, and message communication in embedded real-time systems. The timing guarantees in both applications are not possible without a network protocol/architecture which supports the timely and predictable delivery of messages. The intent of this paper is thus to use the concept of slot reuse to provide more efficient isochronous services for real-time messages with hard deadlines in a metropolitan area network (MAN) using the Distributed Queue Dual Bus (DQDB) medium access control (MAC) protocol.

DQDB has been jointly adopted by IEEE and ANSI as a standard (IEEE802.6) for MANs [2]. As such, DQDB has become the focus of many studies [1, 3–8]. The DQDB network consists of two high-speed (155 Mb/s) unidirectional *slotted* buses (Bus A and Bus B) running in opposite directions to which every station is connected (Fig. 1). There exists a transmission path from every station to every other station. For each bus, fixed-length slots (each of size 53 bytes) are generated by the slot generator at the head of the bus and transported "downstream." Since the two buses are symmetric, without loss of generality, we consider only data transmission on Bus A throughout the paper.

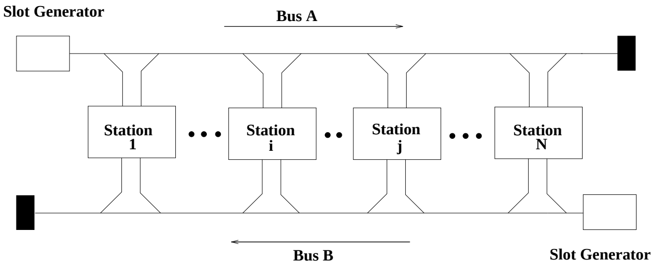


Figure 1: DQDB (IEEE802.6) network configuration.

DQDB provides three distinct classes of services: a *connection-oriented* data service, a *MAC to logical link control* (LLC) data service, and an *isochronous* data service. The first two services are for regular traffic and use the distributed *queue arbitration* (QA) function for slot allocation. The third service is for real-time traffic and uses the *pre-arbitrated* (PA) function. Currently, the DQDB MAC to LLC data service is the only one with defined functions in the existing standards [2]; the other DQDB services remain undefined. In particular, the only specification for the PA function in the existing standards [2] states that: A centralized bandwidth (slot) allocation approach is used in which the *bandwidth manager and VCI server* (BMVS) resides in the slot generator and is responsible for managing/allocating bandwidth. A source station with an isochronous message stream sends a call setup request to BMVS via QA slots. If BMVS grants the call setup request, it assigns a unique *virtual circuit identifier* (VCI) to the message stream, and conveys the information to the source and destination stations via QA slots. BMVS will henceforth reserve empty PA slots by setting their VCI fields to the VCI number of

The work reported in this paper was supported in part by the Wisconsin Alumni Research Foundation (WARF) under Grants 135-2080, by the ONR under Grants N00014-92-J-1080 and N00014-94-1-0229, and by the NSF under Grant MIP-9203895.

the appropriate isochronous message stream. After being notified by BMVS, the station with an isochronous message stream then watches for the PA slots with the appropriate VCI number and transmits its isochronous messages using those slots.

From the above specification, we know that BMVS must ensure that PA slots are properly assigned so as to guarantee the timely delivery of messages in each isochronous message stream. What is lacking in the DQDB standards is *how* the task is accomplished.

To accomplish the task, we laid a formal basis and devised an effective slot allocation scheme in [1] for allocating PA slots to a set of isochronous message streams in a DQDB network. In the context of real-time communications, an isochronous (real-time) message stream M_i is characterized by the following three timing parameters (along with the addresses of the source station N_i^s and the destination station N_i^d): (1) the minimum message inter-arrival time P_i, (2) the maximum message transmission time C_i, and (3) the end-to-end delay bound (deadline) $D_i(\leq P_i)$. The slot allocation scheme assigns the PA slots in such a way that at least C_i slots are allocated to message stream M_i in *any* time window of size D_i slots, for all i. We will summarize the scheme in Section 2.

The performance of the slot allocation scheme (in terms of bandwidth utilization or the number of message streams that can be established) can be improved using the concept of slot reuse. That is, the PA slots that have passed through their destination stations can be reused by the downstream stations. Several slot reuse methods have been suggested for QA services [9–13]. In these studies, slots may be released either by destination stations [9, 10, 12, 13] or by a number of special *erasure* nodes which check every traversing slot and release the ones that have already passed through their destination stations [10, 11, 14]. The destination station of a message stream or the immediate downstream erasure node is equipped with necessary erasure hardware and will mark all the slots assigned to this stream as empty ones so that the downstream stations can reuse them. As indicated in [9, 11], the *destination release* method offers the maximum possible released capacity but suffers from the problems of increased complexity of receiver hardware and increased latency. On the other hand, under the assumption of uniform source-destination traffic, the erasure node method has been shown [10, 14] to overcome the hardware and latency disadvantages at the cost of slightly degraded throughput gain.

As will be clearer in Sections 2–3, the tradeoff between the throughput gain and the associated hardware complexity/latency increase for QA services does not exist for PA services. This is due to the fact that each PA slot used by an isochronous message stream is identified by the PA bit and the VCI field of the slot, both of which are set by BMVS and will remain *unchanged* as the slot traverses the bus. An isochronous message stream authorized by BMVS to use a set of PA slots identified by a unique VCI number will do so when these PA slots traverse the bus, regardless whether or not these PA slots already carry messages (destined for some other stations). It is the responsibility of BMVS to make sure that no PA slot is inappropriately reused before the message carried by the slot is delivered to its destination. Consequently, there is no need for a destination node or an erasure node to change any bit in the *access control field* (ACF) of a PA slot in order to release the slot.

The key point for BMVS to ensure that no PA slot is inappropriately reused is to ensure that no two *spatially intersecting* message streams use the same set of PA slots. A message stream M_i is said to *spatially intersect* another stream M_j if $N_i^s \leq N_j^s < N_i^d$ or $N_j^s \leq N_i^s < N_j^d$. Note that "spatial intersection" is a symmetric relation, i.e., if M_i intersects M_j, then M_j intersects M_i. It is clear that if M_i and M_j are spatially non-intersecting, either $N_i^d \leq N_j^s$ or $N_j^d \leq N_i^s$, and hence, N_j^s (N_i^s) can reuse the same set of PA slots allocated to M_i (M_j) in the case of $N_i^d \leq N_j^s$ ($N_j^d \leq N_i^s$) without corrupting the messages destined for N_i^d (N_j^d). We henceforth call a set of PA slots identified by a unique VCI number a *virtual connection* (VC), and the key step for slot reuse in isochronous services is to devise a scheme to group message streams so that all the message streams in a group do not spatially intersect one another and can thus use the same virtual connections.

The rest of the paper is organized as follows. For the paper to be self-contained, we summarize in Section 2 the message model, the slot allocation problem for isochronous services, and the slot allocation scheme proposed in [1]. In Section 3, we formally define the slot reuse problem, and discuss the issues that should be considered in devising slot reuse schemes. In Section 4, we present three slot reuse schemes and give illustrative examples. We also argue that the proposed slot reuse schemes are simple, can be easily incorporated into the slot allocation scheme proposed in [1], and require only a minor change in the current DQDB standards. The paper concludes with Section 5.

2 Slot allocation in DQDB networks

In this section, we summarize the message model and the slot allocation scheme proposed in [1] to make the paper self-contained.

2.1 Message model

In the context of real-time communications, each isochronous message stream M_i is characterized by the source station id N_i^s, the destination station id N_i^d, and the following timing parameters:

- P_i: the minimum message inter-arrival time of M_i, i.e., if the j-th message in M_i arrives at time t, then the $(j+1)$-th message in the stream will not arrive before time $t+P_i$, for all $j \geq 1$ (if messages in M_i arrive periodically, then P_i denotes the period);
- C_i: the maximum message transmission time (message size) of M_i, i.e., C_i is the time (measured in slots) needed to transmit a maximum-size message in M_i;
- D_i: the *relative* deadline of M_i, i.e., if a message in M_i arrives at time t, then it must be transmitted by time $t+D_i$ (we require only that $D_i \leq P_i$).

For ease of exposition, we assume that both time and timing parameters (in particular, C_i and D_i) are expressed in slots (cells), and message arrivals are aligned with the beginnings of slots. Under these assumptions, one unit of message (i.e., one cell) needs one unit of time (i.e., one slot) to transmit. We will call the time interval $[t-1, t]$ the t-th (time) slot (or simply, slot t).

We also define the *message density* of a stream M_i as $\rho(M_i) = C_i/D_i$, and the *total message density* of a set of streams, $\mathbf{M} = \{M_1, M_2, \ldots, M_n\}$, as

$$\rho(\mathbf{M}) = \sum_{i=1}^{n} \rho(M_i) = \sum_{i=1}^{n} \frac{C_i}{D_i}.$$

2.2 Slot allocation problem

As discussed in Section 1, BMVS must generate a (virtually) infinite sequence of slots in such a way that the PA slots assigned to a message stream M_i are properly "spaced" so that each message in M_i is transmitted within a time period $\leq D_i$ after its arrival as long as the message inter-arrival time is $\geq P_i \geq D_i$ and the size (transmission time) of the message is $\leq C_i$. We formally define the slot allocation problem as follows.

Problem 1: (**Slot Allocation Problem**) Given a set of real-time message streams $\mathbf{M} = \{M_i = (C_i, D_i, N_i^s, N_i^d) \mid 1 \leq i \leq n\}$, allocate the PA slots in such a way that each stream M_i is guaranteed to transmit each of its messages before the message deadline D_i. That is, if a message of M_i arrives at time t, enough slots must be allocated for M_i during time interval $[t, t+D_i]$ for the transmission of the message. □

Note that in the message model, the *exact* time when a message in a stream M_i arrives and the size of the message are not specified and not known *a priori* (except that the message size is bounded by C_i). Hence, one way for BMVS to ensure the above timeliness criterion is satisfied is to assign at least C_i slots to M_i for *any* time window of size D_i slots, for all i. Consider, for example,

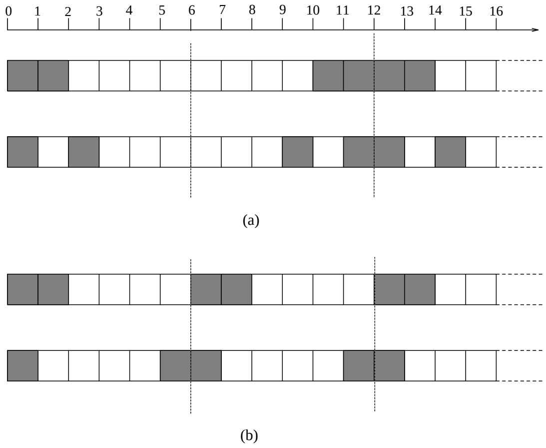


Figure 2: Four possible PA slot allocation patterns for a message stream with $C_i = 2$, $D_i = 6$, and $P_i \geq D_i$.

Fig. 2, where four possible PA slot allocation patterns for a real-time message stream M_i with $C_i = 2$, $D_i = 6$, and $P_i \geq D_i$ are shown. The PA slot allocation patterns in Fig. 2 (a) do not satisfy the criterion that in any time window of size D_i slots, at least C_i slots are allocated to M_i, and a message of M_i which arrives at time t, for $1 \leq t \leq 5$, cannot meet its delivery deadline $t + D_i$. In Fig. 2 (b), the PA slots are so allocated that the above criterion is satisfied, and all messages of M_i can meet their delivery deadlines regardless of their arrival times and sizes.

2.3 Slot allocation scheme

To solve the slot allocation problem, we devised in [1] an on-line slot allocator, called **SlotAllocator**, which can generate, for a given set of message streams $\mathbf{M} = \{M_i = (C_i, D_i, N_i^s, N_i^d) \mid 1 \leq i \leq n\}$, a slot allocation schedule that satisfies the criterion that for any consecutive D_i slots, there are C_i slots allocated to M_i, for all i as long as D_i divides D_j for all $i < j$ and $\rho(\mathbf{M}) \leq 1$.

Succinctly, **SlotAllocator** uses the well-known *rate-monotonic* scheduling algorithm [15] and treats C_i as the computation time and D_i as the period of a task. It assigns priorities to message streams so that the streams with tighter deadlines get higher priorities, i.e., if $D_i < D_j$ then M_i has a higher priority than M_j (ties are broken arbitrarily). After the system is initialized, **SlotAllocator** will assign C_i slots to each message stream M_i during each time period $[(j-1) \cdot D_i, j \cdot D_i]$, for all $1 \leq i \leq n$ and all $j \geq 1$. This is done by assigning the current slot, say $[t-1, t]$, to the message stream with the highest priority among all the *active* message streams, where an active stream M_i is one whose slot requirement with respect to its current time period is *unfulfilled*, i.e., from time $(j-1) \cdot D_i$ to time $t-1$, there are less than C_i slots assigned to M_i, where $(j-1) \cdot D_i \leq t-1 < j \cdot D_i$ for some integer j.

We proved in [1] the correctness of **SlotAllocator**:

Theorem 1: For a set of message streams $\mathbf{M} = \{M_i = (C_i, D_i, N_i^s, N_i^d) \mid 1 \leq i \leq n\}$, if D_i divides D_j for all $i < j$ and $\rho(\mathbf{M}) \leq 1$, **SlotAllocator** will allocate C_i slots to M_i in any time window of size D_i, for all i. $\square$

For an arbitrary set of real-time message streams, **M'** $= \{M_i' = (C_i, D_i', N_i^s, N_i^d) \mid 1 \leq i \leq n\}$, in which the deadline constraint set **D'** $= \{D_1', D_2', \ldots, D_n'\}$ (without loss of generality, we assume $D_i' \leq D_j'$ for all $i < j$) does not necessarily consist solely of multiples (i.e., D_i' divides D_j' may not be true for all $i < j$), we first use the *specialization operation* [16,17] to transform the arbitrary stream set **M'** to another stream set $\mathbf{M} = \{M_i = (C_i, D_i, N_i^s, N_i^d) \mid 1 \leq i \leq n\}$, in which the specialized deadline constraint set $\mathbf{D} = \{D_1, D_2, \ldots, D_n\}$ consists solely of multiples and $D_i \leq D_i'$ for all i. Specifically, we find a D_i for each D_i' such that D_i satisfies $D_i = x \cdot 2^j \leq D_i' < x \cdot 2^{j+1} = 2D_i$, for some integer $j \geq 0$, where x is an integer $\in (D_1'/2, D_1']$ that results in the minimum total density increase.[1] This operation is called *specializing* **D'** (**M'**) *with respect to* $\{x\}$ [16,17]. Note that we also use the phrase "specializing **D'** with respect to {a}" (where a is a constant) to denote that the specialization operation is performed with the specialization factor $x = a$. For example, given a deadline constraint set **D'** $= \{4, 7, 8, 13, 24, 28\}$, if the specialization factor $x = 4$, the set after specialization is $\{4, 4, 8, 8, 16, 16\}$; if the specialization factor $x = 3$, the set after specialization is $\{3, 6, 6, 12, 24, 24\}$. Since **D** is more strict than **D'**, if we find a feasible slot allocation schedule for **M**, then the schedule is also feasible for the original constraint set **D'**.

Example 1: Consider a set of real-time message streams, **M'** = {(1,4), (1,7), (2,13), (1,23), (3,28)}.[2] We first specialize the deadline constraint set **D'** = {4, 7, 13, 23, 28} with respect to {3} to **D** = {3, 6, 12, 12, 24}. Since D_i divides D_j, for all $i < j$, and $\rho(\mathbf{M}) = \sum_{i=1}^{5} C_i/D_i = 1/3 + 1/6 + 2/12 + 1/12 + 3/24 = 21/24 < 1$, by Theorem 1, we know that **SlotAllocator** can find a feasible schedule for **M**. Using **SlotAllocator**, we obtain the slot allocation schedule as shown in Fig. 3 in which the schedule repeats every $D_5 = 24$ slots. As one can readily see, there are at least C_i slots assigned to M_i in any time window of size D_i ($\leq D_i'$) slots, and hence, in any time window of size D_i' slots. $\square$

The slot allocation scheme proposed in [1] is simple and effective. Moreover, as long as the total message density $\rho(\mathbf{M})$ after specialization is less than or equal to 1, the deadline constraints for all streams can be guaranteed. However, if $\rho(\mathbf{M}) > 1$, the scheme will not be able to find a feasible schedule. In the following sections, we introduce the concept of slot reuse for isochronous services and propose several schemes to improve the performance (in terms of bandwidth utilization) of the scheme proposed in [1].

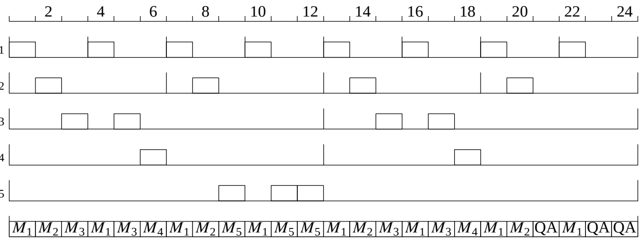


Figure 3: The slot allocation sequence for the set of message streams **M'** = {(1,4), (1,7), (2,13), (1,23), (3,28)}.

3 Slot reuse problem

As discussed in Section 1, the performance of a slot allocation scheme with respect to bandwidth utilization can be improved using the concept of slot reuse. That is, the slots that have passed on to their destination stations can be reused by downstream stations. The slot allocation scheme proposed in [1] needs to be modified in order to exploit the newly released capacity. Specifically, BMVS must not only guarantee the timing constraints of all the message streams but also improve the performance by arranging to have *spatially non-intersecting* message streams reuse the same set of PA slots. BMVS accomplishes the latter by

(1) grouping existing message streams into subsets (groups) in which all the streams in a group do not spatially intersect one another, and

(2) assigning virtual connections to each group so that all the message streams in the group can use the PA slots assigned to these virtual connections, where a virtual connection (VC) is characterized (in addition to its unique VCI number) by its bandwidth requirement, (c, d), where (c, d) denotes at least c slots must be allocated to the virtual connection in any time window of size d slots (for notational convenience, we will use (c, d) or c/d interchangeably to denote the bandwidth of a virtual connection).

A station with a message stream M_i and authorized by BMVS to use some virtual connections will use all the PA slots assigned to those virtual connections when those slots traverse the station, regardless whether the PA slots are currently empty or not. These PA slots may be non-empty due to the fact that they have been used

[1] See [16,17] for details on how to determine the value of x.

[2] We omit the source and destination stations, N_i^s and N_i^d, for each M_i since they are irrelevant in this example.

by upstream stations for transmitting their isochronous messages. However, the slot reuse scheme must ensure that by the time the non-empty PA slots traverse the station of interest, the data contained in the slots have been retrieved by their intended upstream destination stations.

The slot reuse problem for isochronous services in a DQDB network (that uses the slot allocation scheme proposed in [1]) can be formally stated as follows.

Problem 2: (**Slot Reuse Problem**) Given a set of message streams $\mathbf{M} = \{M_i = (C_i, D_i, N_i^s, N_i^d) \mid 1 \leq i \leq n\}$, find a grouping $\mathbf{G} = \{G_1, G_2, \ldots, G_k\}$ of the message streams in $\mathbf{M}$ so that the following conditions are satisfied:

C1. $\cup_{j=1}^{k} G_j = \mathbf{M}$ (note that a grouping $\mathbf{G}$ of $\mathbf{M}$ may or may not be a *partition*[3] of $\mathbf{M}$, i.e., a message stream M_i may belong to more than one group).

C2. All the streams in a group do not spatially intersect one another, and thus can use the same virtual connections.

C3. Each group is assigned one or more virtual connections with appropriate bandwidth to ensure that the timing constraints for all the message streams in the group are satisfied. However, the total bandwidth assigned to the groups should be reduced as much as possible. □

There are three issues we must consider in devising an effective slot reuse scheme. First, if a slot allocation schedule satisfies that "there are at least c slots allocated to a stream in any time window of size d slots," then it also satisfies that "there are at least $q \cdot c$ slots allocated to the stream in any time window of size $q \cdot d$ slots, for every integer $q \geq 1$." However, the converse is not necessarily true. Therefore, the bandwidth requirement for a stream with $(C_i, D_i) = (q \cdot c, q \cdot d)$ can be fulfilled by a schedule generated for a stream with $(C_i, D_i) = (c, d)$, but not the converse.

Second, although fewer groups in general imply lower bandwidth requirement, this is not always true in our slot reuse problem, i.e., minimizing the number of groups does not always lead to an optimal solution. For example, consider a set of message streams as shown in Fig. 4 (a). Fig. 4 (b) is a grouping which gives the least number of groups. Since the bandwidth assigned to each group must be able to guarantee the timing constraints of all the streams in the group, the total bandwidth required for all the virtual connections, VC1, VC2, and VC3, exceeds one $(1/2 + 1/2 + 1/8)$. This implies that there does not exist a feasible slot allocation schedule for such

[3] A partition of a set $\mathbf{S}$ is a set $\mathbf{P} = \{P_1, P_2, \ldots, P_k\}$, where P_i is a subset of $\mathbf{S}$ for all i, $\cup_{j=1}^{k} P_j = \mathbf{S}$, and $P_i \cap P_j = \emptyset$ for all $i \neq j$.

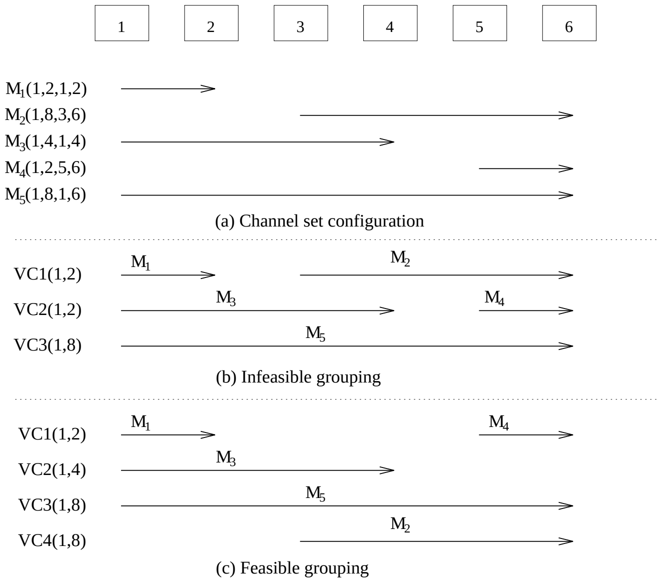


Figure 4: Example showing that minimizing the number of groups is not the only criterion in grouping.

a grouping. However, the grouping shown in Fig. 4 (c) does yield a feasible allocation schedule. Fig. 4 (c) gathers spatially non-intersecting message streams *of similar message densities* into a group, and hence the bandwidth is not as much over-allocated to a group as in Fig. 4 (b). The resulting total bandwidth needed in Fig. 4 (c) is 1 $(= 1/2 + 1/4 + 1/8 + 1/8)$, implying the existence of a feasible slot allocation schedule.

Third, as indicated in **C1** of Problem 2, the grouping $\mathbf{G} = \{G_1, G_2, \ldots, G_k\}$ is not necessarily a partition of $\mathbf{M}$. That is, one message stream may belong to multiple groups. For example, consider the stream set $\{M_1 = (1,4,1,3), M_2 = (1,8,1,3), M_3 = (3,8,5,6)\}$. Since M_1 and M_2 spatially intersect each other, they must be assigned to two distinct groups and use two distinct virtual connections with bandwidth 1/4 and 1/8, respectively. If M_3 joins either group, the bandwidth required for that group has to be increased (by the amount of 1/8 if M_3 joins $G_1 = \{M_1\}$, and 1/4 if M_3 joins $G_2 = \{M_2\}$). A better way is to have M_3 use the bandwidths allocated to both groups. That is, assign $G_1 = \{M_1, M_3\}$ and $G_2 = \{M_2, M_3\}$, and let M_3 use the PA slots assigned to both virtual connections. The timing requirement for M_3 is fulfilled, and yet the total bandwidth requirement is not increased.

4 Proposed slot reuse schemes

In this section, we present three heuristic slot reuse schemes. Each of the three schemes has three phases, namely, specialization, grouping, and bandwidth/VC assignment. They mainly differ in the way and the order in which the three phases are performed. In the following discussion, a group G_j is said to be able to *accommodate* a message stream M_i if M_i does not intersect any stream in G_j, and the *current bandwidth* B_j of G_j is defined to be the largest message density among the message densities of all the message streams in G_j, i.e., $B_j = \max_{M_i \in G_j} C_i/D_i$.

4.1 Scheme A

For a set of arbitrary real-time message streams, **M'** = $\{M'_i = (C_i, D'_i, N_i^s, N_i^d) \mid 1 \leq i \leq n\}$, Scheme A performs the following steps:

(1) Specialization: Scheme A first specializes **M'** (with respect to $\{x\}$) to a more strict stream set $\mathbf{M} = \{M_i = (C_i, D_i, N_i^s, N_i^d) \mid 1 \leq i \leq n\}$, where the deadline constraint set $\mathbf{D} = \{D_1, D_2, \ldots, D_n\}$ consists solely of multiples, and $D_i \leq D'_i$ for all i.

(2) Grouping: Scheme A then finds a grouping $\mathbf{G} = \{G_1, G_2, \ldots, G_k\}$ of the streams in **M** (in this scheme, the grouping is also a partition). All the streams in a group do not intersect one another. Two grouping methods **GM1** and **GM2** are proposed.

Grouping method GM1: Given a specialized set of message streams, **M**, **GM1** performs the following steps:

- **S1.** Sort the message streams in **M** into the order of nondecreasing source station id, and for message streams with the same source station id, into the order of nondecreasing message density (ties are broken arbitrarily). For ease of exposition, let the sorted message streams be still denoted by $M_1, M_2, \ldots, M_n$.
- **S2.** Assign the first message stream M_1 of the sorted streams to G_1. Initialize **G** to $\{G_1\}$.
- **S3.** Consider the other message streams one at a time. Suppose the message stream currently under consideration is M_i, and the current grouping is $\mathbf{G} = \{G_1, G_2, \ldots, G_\ell\}$. If for all $j \in [1, \ell]$, M_i spatially intersects the most recently added stream of G_j,[4] then create a new group $G_{\ell+1}$, add M_i to $G_{\ell+1}$, and add $G_{\ell+1}$ to **G**. Otherwise, let **G'** $\subseteq$ **G** be the set of groups that can accommodate M_i and whose current bandwidths are larger than or equal to $\rho(M_i)$, and **G"** $\subseteq$ **G** be the set of groups that can accommodate M_i but whose current bandwidths are less than $\rho(M_i)$. There are two cases to consider: (i) if **G'** $\neq \emptyset$, then add M_i to a group G_j with the minimum current bandwidth B_j in **G'**. (ii) if **G'** $= \emptyset$, then add M_i to a group G_j with the maximum current bandwidth B_j in **G"**.

[4] Note that because the streams in **M** are sorted into the order of nondecreasing source station id and all streams added to a group G_j do not spatially intersect one another, G_j can accommodate a new stream M_i if and only if M_i does not spatially intersect the most recently added stream of G_j.

The reason for adding M_i to a group with the minimum current bandwidth in **G'** in case (i) of **S3** is to use the "best-fit" group in **G'** and leave the other groups with larger current bandwidths for subsequent streams. The reason for adding M_i to a group with the maximum current bandwidth in **G"** in case (ii) of **S3** is to minimize the increase in the current total bandwidth.

The problem of finding a grouping (partition) of the message streams in **M** such that all the streams in a group do not intersect one another can be viewed as an *interval-graph coloring problem* [18], where the vertices of the interval graph correspond to the given message streams and an edge exists between two vertices if and only if the two corresponding streams spatially intersect each other. The grouping method **GM1** is the same in essence as the well-known greedy algorithm [18] in interval-graph coloring. The latter gives the smallest number of colors needed to color the vertices so that no two adjacent vertices are assigned the same color, while the former gives the least number of groups.

As demonstrated in Fig. 4 in Section 3, finding the least number of groups does not necessarily render the best solution for our slot reuse problem. This is due to the fact that if message streams which significantly differ in their message densities are assigned to the same group (in order to reduce the number of groups), bandwidth may be unduly assigned. To remedy this drawback, we modify **GM1**, and propose the second grouping method **GM2**. Note, however, that **GM2** may not necessarily render the best solution either, since it may not give the least number of groups.

Grouping method GM2: Given a specialized set of message streams, **M**, **GM2** performs the following steps:

- **S1.** Sort the message streams in **M** into the order of nonincreasing message density, and for message streams with the same message density, into the order of nondecreasing source station id (ties are broken arbitrarily). For ease of exposition, let the sorted streams be still denoted by $M_1, M_2, \ldots, M_n$.
- **S2.** Assign the first message stream M_1 of the sorted streams to G_1. Initialize **G** to $\{G_1\}$.
- **S3.** Consider the other message streams one at a time. Suppose the message stream currently under consideration is M_i, and the current grouping is $\mathbf{G} = \{G_1, G_2, \ldots, G_\ell\}$. If for all $j \in [1, \ell]$, M_i spatially intersects some message stream(s) of G_j, then create a new group $G_{\ell+1}$, add M_i to $G_{\ell+1}$, and add $G_{\ell+1}$ to **G**. Otherwise, let **G'** $\subseteq$ **G** be the set of groups that can accommodate M_i and whose current bandwidths are larger than or equal to $\rho(M_i)$, and **G"** $\subseteq$ **G** be the set of groups that can accommodate M_i and whose current bandwidths are less than $\rho(M_i)$. There are two cases to consider: (i) if **G'** $\neq \emptyset$, then

add M_i to a group G_j with the minimum current bandwidth B_j in **G'**. (ii) if **G'** $= \emptyset$, then add M_i to a group G_j with the maximum current bandwidth B_j in **G"**.

Note that the only differences between **GM1** and **GM2** are: (1) in **S1**, **GM1** sorts message streams into the order of nondecreasing source station id first and then into the order of nonincreasing message density, while **GM2** sorts message streams into the order of nonincreasing message density and then into the order of nondecreasing source station id; (2) to check if a group G_j can accommodate a message stream M_i in **S3**, **GM1** needs only to check whether or not M_i spatially intersects the *most recently added* stream of G_j, while **GM2** needs to check whether or not M_i and *every* stream in G_j are spatially non-intersecting.

(3) Bandwidth/VC assignment: Finally, Scheme A determines the virtual connections and their bandwidths to be assigned to each group G_j as follows. First, the bandwidth B_j of a group G_j is set to

$$B_j = \frac{c_j}{d_j} = \max_{M_i \in G_j} \frac{C_i}{D_i}, \tag{4.1}$$

for all j. Then, B_j is further decomposed into

$$B_j = \frac{c_j}{d_j} = \frac{c_{j0}}{d_{j0}} + \frac{c_{j1}}{d_{j1}} + \cdots + \frac{c_{jm_j}}{d_{jm_j}}, \tag{4.2}$$

where $m_j = \log_2 \frac{d_j}{x}$, $d_{jl} = x \cdot 2^l$ for $0 \le l \le m_j$, and $0 \le c_{j0} < x$ and $c_{jl} = 0$ or 1, for $1 \le l \le m_j$. Note that the decomposition is unique. For example, if $B_j = 35/48$ and the specialization factor $x = 3$, then B_j is decomposed into $2/3 + 1/24 + 1/48$.

Corresponding to each nonzero c_{jl}, we assign a virtual connection with bandwidth c_{jl}/d_{jl} to G_j. Then, we can use **SlotAllocator** to generate a slot allocation schedule for all the virtual connections such that there are c_{jl} slots assigned to the corresponding virtual connection in any time window of size d_{jl} slots. Note that there is usually more than one virtual connection assigned to a group G_j. A message stream in G_j will use the PA slots assigned to all the virtual connections of G_j for transmitting its isochronous messages.

The rationale behind decomposing c_j/d_j into $\sum_{l=0}^{m_j} c_{jl}/d_{jl}$ is best illustrated by the following example. Consider a group G_j which consists of two message streams, M_1 and M_2, with message density 1/8 and 5/32, respectively (note that $\max(1/8, 5/32) = 5/32$). If we simply assign a virtual connection with bandwidth 5/32 to G_j, the timing requirement for M_1 (with message density 1/8) may not be guaranteed. For example, if the PA slots for the virtual connection are so allocated that the 5 slots assigned to this connection in any time window of size 32 slots are not "evenly spaced" in such a way that at least 1 slot is assigned to the connection in any time window of size 8 slots, the timing requirement for M_1 will not be fulfilled. However, if we decompose 5/32 into $1/8 + 1/32$, and assign two distinct virtual connections with bandwidths 1/8 and 1/32, respectively, to G_j, the timing requirements for both streams will be fulfilled.

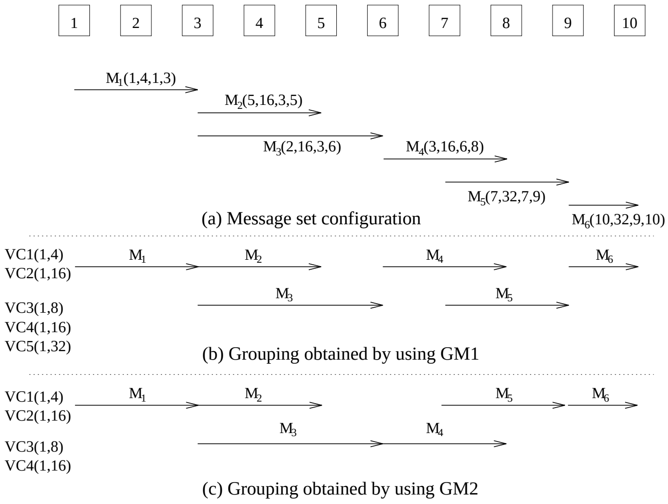


Figure 5: Example which shows how Scheme A works.

We use the following example to illustrate the three steps of Scheme A.

Example 2: Given a set of message streams, **M'** $= \{M_1' = (1,5,1,3), M_2' = (5,17,3,5), M_3' = (2,21,3,6), M_4' = (3,17,6,8), M_5' = (7,32,7,9), M_6' = (10,33,9,10)\}$, Scheme A first specializes **M'** with respect to $\{x = 2\}$ to $\mathbf{M} = \{M_1 = (1,4,1,3), M_2 = (5,16,3,5), M_3 = (2,16,3,6), M_4 = (3,16,6,8), M_5 = (7,32,7,9), M_6 = (10,32,9,10)\}$ (Fig. 5 (a)). (Note that without slot reuse, $\rho(\mathbf{M}) = 43/32 > 1$, implying that no feasible slot allocation schedule exists for **M**.)

If Scheme A applies **GM1** to find a grouping for **M**, it considers the streams in the order of M_i, $i = 1, 2, \ldots, 6$, and obtains the groups $G_1 = \{M_1, M_2, M_4, M_6\}$ and $G_2 = \{M_3, M_5\}$ (Fig. 5 (b)). Scheme A then assigns $B_1 = \max_{M_i \in G_1} C_i/D_i = 5/16$ as the bandwidth for G_1, decomposes 5/16 into $1/4 + 1/16$, and assigns two virtual connections, VC1 and VC2, with bandwidths 1/4 and 1/16, respectively, to G_1. The stations with the message streams in G_1 will use the PA slots assigned to both VC1 and VC2 as these slots traverse the stations. Note that because all the message streams in G_j do not intersect one another, they will use the PA slots at *disjoint* time intervals. Similarly, Scheme A assigns $B_2 = \max(C_3/D_3, C_5/D_5) = 7/32$ as the bandwidth for G_2, decomposes 7/32 into $1/8 + 1/16 + 1/32$, and assigns three virtual connections, VC3, VC4 and VC5, with bandwidths 1/8, 1/16, and 1/32, respectively, to G_2. All the message streams in G_2 will use the PA slots assigned to VC3 through VC5 at disjoint time intervals. The total bandwidth required after applying Scheme A (with grouping method **GM1**) is $B_1 + B_2 = 17/32 < 1$.

If Scheme A applies **GM2** to find a grouping for **M**, it considers the streams in the order of M_2, M_6, M_1, M_5, M_4, and M_3, and obtains the groups $G_1 = \{M_1, M_2, M_5, M_6\}$ and $G_2 = \{M_3, M_4\}$ (Fig. 5 (c)). Scheme A then assigns $B_1 = 5/16$ as the bandwidth for G_1, decomposes $5/16$ into $1/4 + 1/16$, and assigns two virtual connections, VC1 and VC2, with bandwidths $1/4$ and $1/16$, respectively, to G_1. Similarly, Scheme A assigns $B_2 = 3/16$ as the bandwidth for G_2, decomposes $3/16$ into $1/8+1/16$ and assigns two virtual connections, VC3 and VC4, with bandwidths $1/8$ and $1/16$, respectively, to G_2. The total bandwidth required after applying Scheme A (with **GM2**) is $B_1 + B_2 = 1/2 < 1$. □

4.2 Scheme B

For a set of arbitrary message streams, **M'**, Scheme B performs the following steps:

(1) Grouping: Scheme B first uses **GM1** or **GM2** to find for the message streams in **M'** a grouping $\mathbf{G} = \{G_1, G_2, \ldots, G_k\}$ in which all the streams in a group do not intersect one another.

(2) Bandwidth/VC assignment and specialization: Scheme B then assigns bandwidth and virtual connections to each group. One plausible method is to assign bandwidth $B'_j = \max_{M'_i \in G_j} C_i/D'_i$ to G_j for all j, specialize **B'** $= \{B'_1, B'_2, \ldots, B'_k\}$ to $\mathbf{B} = \{B_1, B_2, \ldots, B_k\}$, where $B_j = C_j/D_j$, and then decompose B_j as in Eq. (4.2). However, this method is not valid. Consider an example in which $G_j = \{(1,5), (7,33)\}$. Using the above invalid method, we would assign $B'_j = 7/33$, specialize B'_j to $B_j = 7/32$ (suppose the specialization factor $x = 2$), decompose B_j into $B_j = 1/8 + 1/16 + 1/32$, and assign three virtual connections with bandwidths $1/8$, $1/16$, and $1/32$, respectively, to G_j. However, as discussed in Section 3, these three virtual connections are not sufficient to guarantee the timing requirement of the stream $(1,5)$ since $5 < 8$, i.e., there may not be one slot assigned to the virtual connections for G_j in any time window of size five slots. We propose the following valid method:

Given a grouping $\mathbf{G} = \{G_1, G_2, \ldots, G_k\}$ of **M'**, we perform the following steps:

S1. (Intra-group specialization) Let $x_j = \min_{M'_i \in G_j} D'_i$, for all j. For each j, specialize G_j with respect to $\{x_j\}$ to H_j, i.e., for each $M'_i = (C_i, D'_i, N^s_i, N^d_i) \in G_j$, change M'_i to $M''_i = (C_i, D''_i, N^s_i, N^d_i) \in H_j$, where $D''_i = x_j \cdot 2^l \leq D'_i < 2 \cdot D''_i$, for some integer $l \geq 0$.

S2. Let $B'_j = c_j/d'_j = \max_{M''_i \in H_j} C_i/D''_i$, for all j. Decompose B'_j into

$$B'_j = \frac{c_j}{d'_j} = \frac{c_{j0}}{d'_{j0}} + \frac{c_{j1}}{d'_{j1}} + \cdots + \frac{c_{j\ell_j}}{d'_{j\ell_j}}, \qquad (4.3)$$

where $\ell_j = \log_2 \frac{d'_j}{x_j}$, $d'_{ji} = x_j \cdot 2^i$, for $0 \leq i \leq \ell_j$, $0 \leq c_{j0} < x_j$, and $c_{ji} = 0$ or 1, for $1 \leq i \leq \ell_j$.

S3. (Inter-group specialization) Specialize (with respect to $\{x\}$) the composite set $\{(c_{ji}, d'_{ji}) \mid 1 \leq j \leq k, 0 \leq i \leq \ell_j, \text{ and } c_{ji} \neq 0\}$ to the set $\{(c_{ji}, d_{ji}) \mid 1 \leq j \leq k, 0 \leq i \leq \ell_j, \text{ and } c_{ji} \neq 0\}$.

S4. For each nonzero c_{ji}, $1 \leq j \leq k$ and $0 \leq i \leq \ell_j$, assign a virtual connection VCji with a bandwidth $B_{ji} = c_{ji}/d_{ji}$ to group G_j. All message streams in G_j will use the PA slots assigned to the virtual connections VCji, for $0 \leq i \leq \ell_j$ and $c_{ji} \neq 0$, for transmitting their isochronous messages.

We use the following example to illustrate the steps of Scheme B.

Example 3: Given the same set of message streams, **M'**, as in Example 2, Scheme B first groups the message streams in **M'** into subsets $G_1 = \{M'_1, M'_2, M'_4, M'_6\}$ and $G_2 = \{M'_3, M'_5\}$ (assuming that the grouping method **GM1** is used). Second, Scheme B performs the intra-group specialization, i.e., Scheme B specializes G_1 with respect to $\{x = 5\}$ to $H_1 = \{M''_1 = (1,5,1,3), M''_2 = (5,10,3,5), M''_4 = (3,10,6,8), M''_6 = (10,20,9,10)\}$, and G_2 with respect to $\{x = 21\}$ to $H_2 = \{M''_3 = (2,21,3,6), M''_5 = (7,21,7,9)\}$. Then, Scheme B decomposes $B'_1 = \max\{1/5, 5/10, 3/10, 10/20\} = 5/10$ into $B'_1 = 2/5 + 1/10$, and $B'_2 = \max\{2/21, 7/21\} = 7/21$ into $B'_2 = 7/21$. Then, Scheme B performs inter-group specialization, i.e., specializes the set $\{(2,5), (1,10), (7,21)\}$ (with respect to $\{x = 5\}$) to $\{(2,5), (1,10), (7,20)\}$. Finally, Scheme B assigns two virtual connections with bandwidths $2/5$ and $1/10$, respectively, to G_1 and one virtual connection with bandwidth $7/20$ to G_2. The total bandwidth required after applying Scheme B (with **GM1**) is $2/5 + 1/10 + 7/20 = 17/20 < 1$. □

4.3 Scheme C

In the previous two schemes, the grouping **G** of the streams in **M** (**M'**) obtained by either of the two grouping methods **GM1** and **GM2** is, in fact, a partition of **M** (**M'**), i.e., $\cup_i G_i =$ **M** (**M'**) and $G_i \cap G_j = \emptyset$ for all $i \neq j$. In order to exploit the advantage of assigning a message stream to more than one groups, we propose the following scheme.

For a set of arbitrary message streams, **M'**, Scheme C performs the following steps:

(1) Specialization and stream decomposition: Scheme C first specializes **M'** (with respect to $\{x\}$)

to $\mathbf{M}$, and decomposes each message stream $M_i = (C_i, D_i, N_i^s, N_i^d)$ in $\mathbf{M}$ into a set of sub-streams:

$$S(M_i) = \{M_{ij} = (C_{ij}, D_{ij}, N_i^s, N_i^d) \mid 0 \le j \le m_i \text{ and } C_{ij} \ne 0\},$$

where $m_i = \log_2 \frac{D_i}{x}$, $D_{ij} = x \cdot 2^j$ for $0 \le j \le m_i$, $0 \le C_{i0} < x$, $C_{ij} = 0$ or 1, for $1 \le j \le m_i$, and $C_i/D_i = \sum_{j=0}^{m_i} C_{ij}/D_{ij}$. Note that the decomposition is unique. We say that a sub-stream *belongs to* a message stream M_i if the sub-stream is in $S(M_i)$.

(2) Grouping: Scheme C then finds a grouping $\mathbf{G} = \{G_1, G_2, \ldots, G_k\}$ for the sub-streams. Either **GM1** or **GM2** can be used as the grouping method, except that in this scheme, the grouping is performed on sub-streams, instead of on message streams, and the term "accommodate" is (re)defined as "a group G_j can *accommodate* a sub-stream M_{ix} if M_{ix} does not intersect any sub-stream in G_j that *does not belong to stream* M_i." That is, although two sub-streams M_{ix} and M_{iy} spatially intersect each other, they can be put into the same group since they actually belong to the same stream M_i. Note that $\mathbf{G}$ is a partition of the sub-streams, but may not be a partition of the streams.

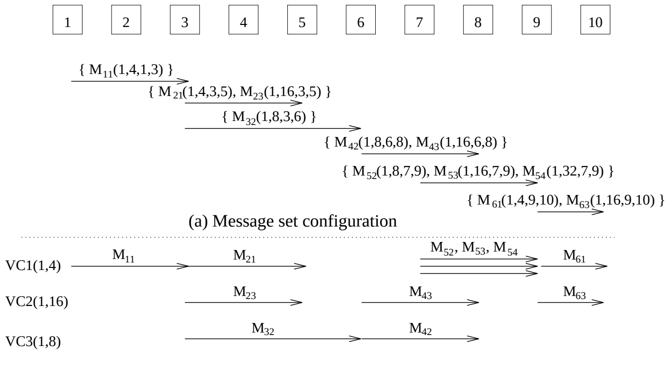
(3) Bandwidth/VC assignment: Let M_{f_j} be a stream in G_j such that

$$\sum_{l \text{ s.t. } M_{f_j l} \in S(M_{f_j}) \cap G_j} C_{f_j l}/D_{f_j l} = \max_{i \text{ s.t. } S(M_i) \cap G_j \ne \emptyset} \sum_{l \text{ s.t. } M_{il} \in S(M_i) \cap G_j} C_{il}/D_{il}.$$

That is, among all streams with sub-streams in G_j, M_{f_j} is the stream whose sub-streams in G_j require the largest bandwidth. For each sub-stream $M_{f_j l}$ with $M_{f_j l} \in S(M_{f_j}) \cap G_j$, Scheme C assigns a virtual connection $\mathrm{VC}f_j l$ with a bandwidth $C_{f_j l}/D_{f_j l}$ to group G_j, for all j. A message stream in G_j will use the PA slots assigned to the virtual connections $\mathrm{VC}f_j l$, for $1 \le j \le k$, and $0 \le l \le m_{f_j}$ and $C_{f_j l} \ne 0$, for transmitting its isochronous messages.

Since a message stream is decomposed into sub-streams each of which is assigned to a (possibly distinct) group, a message stream may be assigned to several distinct groups. Moreover, since the timing requirement for each sub-stream is ensured in the above bandwidth/VCI assignment, the timing requirement for each message stream is also guaranteed.

We use the following example to illustrate the steps of Scheme C.



(a) Message set configuration

(b) Grouping obtained

Figure 6: Example which shows how Scheme C works.

Example 4: Given the same set of message streams, **M'**, as in Example 2, Scheme C first specializes **M'** with respect to $\{x = 2\}$ to $\mathbf{M} = \{M_1 = (1, 4, 1, 3), M_2 = (5, 16, 3, 5), M_3 = (2, 16, 3, 6), M_4 = (3, 16, 6, 8), M_5 = (7, 32, 7, 9), M_6 = (10, 32, 9, 10)\}$. Second, Scheme C decomposes all the streams in $\mathbf{M}$ into sets of sub-streams: $S(M_1) = \{M_{11} = (1, 4, 1, 3)\}$, $S(M_2) = \{M_{21} = (1, 4, 3, 5), M_{23} = (1, 16, 3, 5)\}$, $S(M_3) = \{M_{32} = (1, 8, 3, 6)\}$, $S(M_4) = \{M_{42} = (1, 8, 6, 8), M_{43} = (1, 16, 6, 8)\}$, $S(M_5) = \{M_{52} = (1, 8, 7, 9), M_{53} = (1, 16, 7, 9), M_{54} = (1, 32, 6, 8)\}$, and $S(M_6) = \{M_{61} = (1, 4, 9, 10), M_{63} = (1, 16, 9, 10)\}$ (Fig. 6 (a)). (Note that as defined earlier, $M_{ij} = (C_{ij}, D_{ij})$, where $D_{ij} = x \cdot 2^j = 2^{j+1}$). Then, Scheme C performs the grouping method **GM1** and considers the sub-streams in the order of M_{11}, M_{21}, M_{23}, M_{32}, M_{42}, M_{43}, M_{52}, M_{53}, M_{54}, M_{61}, and M_{63}, and obtains the groups: $G_1 = \{M_{11}, M_{21}, M_{52}, M_{53}, M_{54}, M_{61}\}$, $G_2 = \{M_{23}, M_{43}, M_{63}\}$, and $G_3 = \{M_{32}, M_{42}\}$ (Fig. 6 (b)). Finally, Scheme C sets $B_1 = \max\{1/4, 1/4, 1/8 + 1/16 + 1/32, 1/4\} = 1/4$, $B_2 = \max\{1/16, 1/16, 1/16\} = 1/16$, and $B_3 = \max\{1/8, 1/8\} = 1/8$ as the bandwidths of G_1, G_2, and G_3, respectively, and assigns virtual connections, VC1, VC2, and VC3, with bandwidths 1/4, 1/16, and 1/8 to G_1, G_2, and G_3, respectively. The total bandwidth required after applying Scheme C is $B_1 + B_2 + B_3 = 7/16 < 1$. $\square$

4.4 Incorporating slot reuse

The above three slot reuse schemes can be incorporated into the slot allocation scheme proposed in [1] as follows. Given a set of arbitrary message streams **M'** $= \{M'_i = (C_i, D'_i, N_i^s, N_i^d) \mid 1 \le i \le n\}$, we first apply a specific slot reuse scheme to get the set of groups, the virtual connections assigned to each group, and the bandwidth assigned to each virtual connection. Note that a virtual connection is characterized by (in addition to its unique VCI number) its bandwidth requirement, c/d, where c/d denotes that at least c slots should be allocated to the virtual connection in any time window of size d slots. The virtual connections assigned to each group guarantee that a sufficient number of well-spaced PA slots are allocated to the message streams in the group so that their timing constraints can be satisfied. The bandwidth assigned to a group is the sum of the bandwidths of all the virtual connections assigned to the group, and the total bandwidth assigned to the stream set **M'** is the sum of the bandwidths of all the virtual connections.

Note that since specialization is one of the steps performed in each proposed slot reuse scheme, the deadline constraint set of the bandwidth requirements of all the virtual connections consists solely of multiples. As discussed in Section 2, **SlotAllocator** takes a set of message streams whose deadline constraint set consists solely of multiples and whose total density is less than or equal to 1 as the input, and generates a feasible slot allocation schedule which satisfies the timing constraints of all the message streams in the set. If we use the set of the virtual connections as the input to **SlotAllocator**, and if the total bandwidth required for all the virtual connections is less than or equal to 1, then by Theorem 1, **SlotAllocator** can generate a slot allocation schedule which satisfies the bandwidth requirements of all the virtual connections, and hence the timing constraints of all the message streams in **M'**.

5 Concluding remarks

We have proposed three slot reuse schemes to improve the performance (in terms of bandwidth utilization or the number of message streams that can be established) of the slot allocation scheme proposed in [1] for DQDB networks. The proposed slot reuse schemes divide the message streams into a set of groups in which all the streams in a group do not spatially intersect one another and thus can use the same virtual connections, where a virtual connection is a set of "well-spaced" PA slots. The proposed schemes allocate one or more virtual connections to each group to provide the timing guarantees for all the streams in the group. The resulting schemes are guaranteed to find a feasible slot allocation schedule for a set of message streams as long as the total bandwidth assigned to the groups is less than or equal to 1.

We have also briefly discussed how to incorporate the proposed schemes into the slot allocation scheme proposed in [1]. The integration of the slot allocation and slot reuse schemes is simple. Moreover, the only change needed in the current DQDB standards is that a real-time (isochronous) message stream may be assigned more than one virtual connection (and hence VCI number). This can be easily realized by modifying the VCI server.

References

[1] C.-C. Han, C.-J. Hou, and K. G. Shin, "On slot allocation for time-constrained messages in DQDB networks," in *Proc. of INFOCOM'95*, April 1995.

[2] "IEEE Standards for Local and Metropolitan Area Networks: Distributed Queue Dual Bus (DQDB) Subnetwork of a Metropolitan Area Network (MAN)." IEEE802.6, July 1991.

[3] C. C. Biskikian, "Waiting time analysis in a single buffer DQDB (802.6) network," *IEEE J. on Selected Areas in Communications*, vol. 8, pp. 1565–1573, October 1990.

[4] E. L. Hahne, A. K. Choudhury, and N. F. Maxemchuk, "Improving the fairness of distributed-queue-dual-bus networks," in *Proc. of INFOCOM'90*, June 1990.

[5] E. L. Hahne and N. F. Maxemchuk, "Fair access of multi-priority traffic to distributed-queue-dual-bus networks," in *Proc. of INFOCOM'91*, April 1991.

[6] W. Jing and M. Paterakis, "Message delay analysis of the DQDB subnetwork based on an approximate node model," *IEEE Trans. on Communications*, vol. 42, pp. 1120–1130, February/March/April 1994.

[7] D. Saha, M. C. Saksena, S. Mukherjee, and S. K. Tripathi, "On guaranteed delivery of time-critical messages in DQDB," in *Proc. of INFOCOM'94*, June 1994.

[8] O. Sharon, "A proof for lack of starvation in DQDB with and without slot reuse," in *Proc. of INFOCOM'95*, 1995.

[9] M. W. Garrett and S.-Q.-Li, "A study of slot reuse in dual bus multiple access networks," in *Proc. of INFOCOM'90*, 1990.

[10] A. E. Kamal, "Efficient multi-segment message transmission with slot reuse on DQDB," in *Proc. of INFOCOM'91*, April 1991.

[11] M. A. Rodrigues, "Erasure nodes: performance improvements for the IEEE 802.6 MAN," in *Proc. of INFOCOM'90*, 1990.

[12] O. Sharon and A. Segall, "A simple scheme for slot reuse without latency for a dual bus configuration," *IEEE/ACM Trans. on Networking*, vol. 1, Feb. 1993.

[13] O. Sharon and A. Segall, "On the efficiency of slot reuse in the dual bus configuration," *IEEE/ACM Trans. on Networking*, vol. 2, pp. 89–100, February 1994.

[14] G. B. Brewster and M. K. Vernon, "The fairness of DQDB networks with slot reuse," in *Proc. of INFOCOM'95*, April 1995.

[15] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment," *Journal of the ACM*, vol. 20, no. 1, pp. 46–61, 1973.

[16] M. Y. Chan and F. Chin, "Schedulers for larger classes of pinwheel instances," *Algorithmica*, vol. 9, 1993.

[17] C.-C. Han and K.-J. Lin, "Scheduling distance-contrained real-time tasks," *Proc. IEEE 13th Real-Time Systems Symposium*, pp. 300–308, December 1992.

[18] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*. The MIT Press, Cambridge, Massachusetts, 1990.