Minimum Bandwidth Reservations for Periodic Streams in Wireless Real-Time Systems

Jun Yi, Christian Poellabauer, *Senior Member, IEEE,* Xiaobo Sharon Hu, *Senior Member, IEEE,* and Liqiang Zhang, *Member, IEEE*

Abstract—Reservation-based (as opposed to contention-based) channel access in WLANs provides predictable and deterministic transmission and is therefore able to provide timeliness guarantees for wireless and embedded real-time applications. Also, reservation-based channel access is energy efficient since a wireless adaptor is powered on only during its exclusive channel access times. While scheduling for Quality of Service at the central authority (e.g., base station) has received extensive attention, the problem of determining the actual resource requirements of an individual node in a wireless real-time system has been largely ignored.

This work aims at finding the minimum channel bandwidth reservation that meets the real-time constraints of all periodic streams of a given node. Keeping the bandwidth reservation of a node to a minimum leads to reduced energy and resource requirements and leaves more bandwidth for future reservations by other nodes. To obtain a solution to the minimum bandwidth reservation problem, we transform it to a generic uniprocessor task schedulability problem, which is then addressed using a generic algorithm. This algorithm works for a subclass of priority-driven packet scheduling policies, including three common ones: fixed-priority, EDF, and FIFO. Moreover, we then specialize the generic algorithm to these three policies according to their specific characteristics. Their computation complexities and bandwidth reservation efficiencies are evaluated and guidelines for choosing scheduling policies and stream parameters are presented.

Index Terms—Bandwidth reservation, schedulability test, earliest deadline first, fixed-priority, first-in-first-out, medium access control, real time, wireless

1 Introduction

Wireless embedded real-time systems are becoming prevalent with the continuous increase in streaming applications such as video/audio communications, industrial automation, networked and embedded control systems, and wireless sensor and actuator networks. This has called for research efforts to enhance the support of timeliness and Quality of Service (QoS) in wirelessly networked embedded environments. Wireless networks are inherently broadcast and media-shared. Contention-based media accesses such as CSMA are non-deterministic and thus incapable of providing predictable QoS support to periodic communications often found in wireless real-time systems. Moreover, multiple nodes are active simultaneously and continuously sense and contend for the shared media, leading to excessive energy consumption.

Recently, reservation-based channel access protocols that explicitly allow wireless devices to negotiate channel access intervals have been receiving increasing attention. Such access mechanisms allow for contention-free and exclusive accesses, providing deterministic bounds on the delays experienced by the traffic streams and conserving energy (since wireless adaptors having no channel access can be temporarily powered down). Therefore, such access mechanisms are ideally suited for providing *real-time services* in wireless environments. For example, in ad-hoc networks, coordinated sleep mechanisms have been designed [1] to allow wireless devices to coordinate medium access with their neighbors and to reduce their energy requirements. Similarly, in infrastructure-based systems, wireless end devices can coordinate medium access with base stations (BSs) using protocols such as IEEE 802.11e [2].

Reservation-based channel management requires each node to negotiate its desired channel access duration for a given period based on its traffic constraints. However, the computation of such requirements has largely been ignored which has often resulted in poor real-time support, overprovisioning of valuable resources, and poor scalability. The goal of this work is to develop a strategy for the computation of the required channel access reservations for a given packet scheduling policy, such that (i) the real-time constraints of each node's traffic are satisfied and (ii) resource reservations are minimized.

To solve the minimum bandwidth reservation problem at a given node, we treat the complement of the periodic bandwidth reservation as a special periodic stream (the periodic *sleep stream*), i.e., the bandwidth reservation per channel access period is equal to the complement of the execution time (*sleep time*) of the sleep stream with period equal to the channel access period. We add the sleep stream

- *J. Yi, C. Poellabauer, and X.S. Hu are with the Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN, 46556. E-mail: {jyi, cpoellab, shu}@nd.edu*
- *L. Zhang is with the Department of Computer and Information Sciences, Indiana University, South Bend, 46634. E-mail: liqzhang@iusb.edu*

This work is supported in part by NSF under grant numbers CNS-0834180, CNS-0720457, CNS-0834230, and CPS-0931195.

to the original stream set to form an extended stream set. Accordingly, the scheduling policy for the extended stream set is extended from the original scheduling policy for the original stream set such that (a) the sleep stream always has the highest priority and is non-interruptible and (b) the priority relationship among the original stream set is unchanged. As a consequence, we transform the minimum bandwidth reservation problem to the maximum sleep time problem. In other words, there exists a schedule for the original stream set with a given scheduling policy if and only if there exists a schedule for the extended stream set using the extended scheduling policy. Therefore, minimizing the bandwidth reservation is equivalent to maximizing the sleep time of the sleep stream.

Based on supply/demand analysis, we develop a generic algorithm to compute the maximum sleep time for a subclass of priority-driven packet scheduling policies, including three common ones: fixed-priority (e.g., RM and DM), EDF, and FIFO. This subclass exhibits two properties: (1) the scheduling policy is static at the job level but can be either static or dynamic at the task level and (2) the scenario of the worst-case response time of the stream set occurs within the synchronous (in-phase) busy interval of the stream set, i.e., if all stream instances released within this interval meet their deadlines, then all stream instances will meet their deadlines everywhere. The generic algorithm is further modified to provide customized (improved) solutions for each of the three common policies according to their individual characteristics.

The main contributions of this paper are (1) the transformation of the problem of periodically reserving a time interval of minimum length to serve all real-time streams of a node (a problem in the network community) into a dual problem on a dedicated resource (a problem in the real-time scheduling community); and (2) present an alternative solution as in [3] to this problem by applying an augmented supply/demand analysis on the transformed problem in an efficient manner. The resulting solution is general, efficient, and of practical use.

The remainder of this paper is structured as follows. We discuss related work in section 2. In section 3 we present the generic traffic model, the network access model, and the problem this paper is investigating. The model transformation is presented in section 4. Then, the generic algorithm of the minimum bandwidth reservation problem is presented in section 5, and its refinements for fixed-priority policies (section 5.2), EDF (section 5.3), and FIFO (section 5.4) are given subsequently. In section 6, we present simulation results for various stream configurations under various packet scheduling polices. Finally, we conclude this paper in section 7.

2 Related Work

Scheduling and schedulability analysis have been extensively studied in previous work, particularly for processing resources. In networking environments, reservation-based mechanisms are becoming highly prominent in supporting latency-critical and energy-aware traffic. In this section, we discuss existing protocol standards and techniques related to resource and channel access reservations.

A well-known wireless standard that offers channel access reservations is the IEEE 802.11e protocol [2]. The IEEE 802.11e standard proposes a Hybrid Coordination Function (HCF) that provides both contention-based and contention-free channel accesses through two modes: the Enhanced Distributed Channel Access (EDCA) and the HCF Controlled Channel Access (HCCA) [2]. With HCCA, the Hybrid Coordinator (HC), which usually resides at the base station, continuously polls every node. TXOPs (Transmit Opportunities) are assigned by the HC to a node at a regular interval and for a specified duration, which are determined based on the node's traffic specification. The research results reported in this work can be applied to the HCCA mode to help each node reserve the smallest amount of bandwidth necessary to meet all packet deadlines.

RI-EDF [4] is a table-driven, slotted reservation protocol based on earliest-deadline first (EDF). It uses the periodic nature of traffic in a fully connected network to deduce a shared packet transmission schedule. Packets are transmitted during their allocated slots, thus avoiding contention. Traffic from the same node is interspersed into discrete slots. In contrast, our reservation model reserves a continuous time interval for every individual node and nodes only wake up during their allocated intervals.

An earlier work studying the same problem [5] relies on several strong assumptions. It assumes that (1) the channel access period is smaller than every stream period, (2) datagram deadlines must be equal to or less than their respective periods, and (3) datagrams must be transmitted consecutively and without interruption. Also, the work implicity assumes that the underlying scheduling policy is FIFO. Although it has a linear complexity, it over-reserves bandwidth in general cases. This paper removes these restrictions, takes into account the impact of different scheduling policies on the computation of the required bandwidth, and presents algorithms for several scheduling policies to efficiently compute the minimum bandwidth reservation, although at the cost of higher complexity compared to [5]. However, even with increased complexity, the presented algorithms are practical considering that the computational capacity of wireless end devices continuously increases and that a wireless end device usually has only a limited number of concurrent real-time streams.

In this work, we transform the minimum bandwidth reservation problem to the maximum sleep time problem, which is then solved by computing the schedulable execution time of the sleep stream for a subclass of scheduling policies, including fixed-priority, EDF, and FIFO. A similar approach has been taken in [6] to solve the minimum EDF-feasible deadline problem of a given task, given its period

and execution time.

There exist a lot of efforts on exact schedulability tests for various scheduling policies, e.g., fixed-priority [7], [8], [9], EDF [10], [11], [12], [13], and FIFO [14]. Our generic algorithm for the bandwidth reservation problem is based on the time-demand analysis techniques provided by these earlier research results, but applied to a new problem.

Our work is also closely related to previous work on resource partition/composition models, which usually focus on processor resources in real-time systems. Such models include the static resource partition model [3], the bounded delay resource partition model [3], [15], [16], the periodic resource model [17], [18], and the explicit deadline periodic resource model [19], [20]. These prior efforts differ from each other mainly in the chosen scheduling model. The resource partition/reservation model used in this paper corresponds to the single time slot periodic partition (STSPP) model (a special case of the static resource partition model) introduced in [3].

The schedulability test scheme for the STSPP model in [3] can also be used to solve the problem, by iteratively executing the schedulability test algorithm proposed in [3] (similar to a binary search). Therefore, this approach would be very costly. It is particularly inefficient for the fixed-priority policy since the change of the resource supply and the change of job response times are non-linear and irregular. Our algorithm augments the traditional time-demand analysis to compute the finished/unfinished portion before a job's deadline, which avoids computing fixed-point equations iteratively. As a result, our solution to the reverse problem (the minimum resource requirement problem) has the same complexity as the original problem (the exact schedulability test problem).

Besides resource partition/composition models, hierarchical schedulers (e.g., [21], [22]) can also provide temporal isolation among applications on a uniprocessor. This property prevents a misbehaving task from interfering with other tasks in another application, i.e., only the tasks within the same application as the misbehaving one could be affected. In hierarchical scheduling, each application is composed of a set of correlating entities (e.g., tasks or streams), where applications are scheduled by a global scheduler and each application schedules its tasks using its local scheduler. The approach proposed in this paper can be applied to hierarchical schedulers, i.e., it can be used to determine the minimum resource requirements of an application, given this application's local scheduling policy (fixed-priority, EDF, or FIFO).

3 Bandwidth Reservation Model

This section presents our network access model, traffic model, and the problem statement.

3.1 Network Access Model

We briefly discuss the concept of reservation-based channel access model (which corresponds to the single time slot periodic partition model introduced in [3]) since it forms the basis for the problem we intend to solve. Such a mechanism uses resource reservations to ensure contention-free accesses. This is achieved through a central authority at a base station (BS) that regulates the channel accesses of individual nodes. Here, the BS takes control of the channel and starts polling each of the nodes in a pre-determined order (e.g., round-robin). Upon reception of a polling frame, a node gains access to the channel. The HCCA mode defined in the IEEE 802.11e standard [2] is an example of a protocol which adopts the reservation-based channel access approach to enhance the QoS support for real-time applications in wireless environments.

Borrowing the terminology from the IEEE 802.11e standard, in a reservation-based channel access mechanism, each node is provided a *Service Period* (SP), during which the node has exclusive access to the wireless medium. Polling frames issued by the BS specify the start time and maximum duration of the SP allotted to a node. At the end of an SP for a node, the BS begins polling the next node in its schedule. The period of recurrence of the SPs is referred to as the *Service Interval (SI)*, which is usually specified by the BS in advance and equal to a multiple of the beacon interval of the BS shared by all client nodes. We call the pair $B = (SP, SI)$ the bandwidth profile of a node (shown in Figure 1). The SP parameter at each node must be negotiated with the BS based on the requirements of the node's expected real-time traffic. This reservation-based

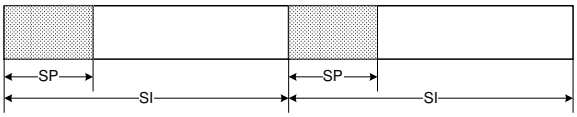

Figure 1. A wireless device's bandwidth profile (SP, SI). Shaded intervals (SPs) are the exclusive access periods for a node, which are repeated every SI time units.

channel access model is very practical and valuable in both wireless local area networks (WLANs) and wireless sensor networks (WSNs): a set of client nodes in a WLAN usually share the same beacon period from the base station [2] and a cluster of nodes in a WSN usually communicate with the cluster head in a duty-cycled manner [23]. There are three advantages of using STSPP in these network areas: (1) it saves energy since nodes only need to wake up to communicate within their respective reserved time intervals; (2) it leads to better latency predictability and possibly higher throughput since wireless contention is avoided a priori; (3) it greatly decreases the run-time complexity of resource partition scheduling due to its simple partition structure.

3.2 Traffic Model

We consider a set of wireless nodes with applications on each node generating one or more periodic real-time streams. Nodes connect wirelessly to a common BS to access an external network. We denote the set of periodic streams generated by a node as $\mathcal{S} = \{\mathcal{S}_1, \cdots, \mathcal{S}_n\}$. Each stream $\mathcal{S}_i$ periodically generates a certain number (worst-case or average-case) of bytes (called a datagram) for a given period p_i for transmission. The datagram generated at the beginning of the j^{th} period of $\mathcal{S}_i$ for transmission is denoted as $J_{i,j}$. Wireless channel conditions are time-varying and error-prone. The worst-case estimation (denoted as e_i) of the transmission time of a datagram of $\mathcal{S}_i$ is needed, which has been the focus of many prior efforts (e.g., in [5] and [24]). Each datagram of $\mathcal{S}_i$ has a relative transmission completion deadline D_i. The release time and deadline of $J_{i,j}$ are denoted as $r_{i,j}$ and $d_{i,j} = r_{i,j} + D_i$, respectively. Our framework requires no specific relationship between stream periods and datagram deadlines, i.e., D_i can be less than, equal to, or greater than p_i. Due to the similarity between the concept of tasks in the literature and the concept of streams in this paper, we will use stream and task interchangeably in this paper. Similarly, the terms datagram and job are also used interchangeably.

Datagrams are often fragmented at the network and/or link layer, depending on the datagram size, network parameters (e.g., the maximum transfer unit or MTU), and the scheduling policy. Therefore, a datagram can be also treated as a logical conglomeration of a series of physical packets. The maximum size of packets cannot be greater than the MTU. When a packet is in flight, no other packet can interrupt it. Therefore, the worst-case non-preemption portion of a datagram is equal to the MTU. Since all streams at the same node share the same MTU value, we set the non-preemption portion of every stream to be the MTU value and denote it as θ. Although packet interruption is not allowed, interleaved transmissions of packets of a datagram with other packets of other datagrams are allowed. For example, when packet $pkt_{1,1,1}$ from datagram $J_{1,1}$ of stream $\mathcal{S}_1$ finishes its transmission and a more urgent datagram $J_{2,1}$ from stream $\mathcal{S}_2$ arrives, the scheduler may need to transmit packets from $J_{2,1}$ before other packets from $J_{1,1}$ (e.g., $pkt_{1,1,2}$). As another example, after packet $pkt_{1,1,1}$ finishes its transmission, the node's allocated time interval for network access is used up and the node is forced to sleep. During the sleep duration, more urgent packets may arrive at the network queue. When the network resource is available again (after the sleep duration), the scheduler may need to transmit these newly arrived urgent packets before packets from $J_{1,1}$. These interleaved transmissions happen frequently since applications treat the network as a dedicated resource and issue packets regardless of the network reservation.

In this paper, we investigate the impact of traffic scheduling policies on bandwidth reservation from the perspective of a single node. The policies under consideration are a subclass of priority-driven polices, and we assume there is a shared queue for all released packets. The priority-driven scheduling policies under consideration include fixed-priority policies, e.g., rate/deadline monotonic scheduling (RM and DM), and dynamic-priority policies, e.g., EDF, and FIFO. All nodes in a WLAN (including the BS) can use their own scheduling policies, i.e., we do not impose restrictions on the choice of scheduling policy.

3.3 Problem Definition and Objectives

Each client node requests its desired bandwidth reservation from the BS and the normalized bandwidth (i.e., the ratio of SP to the given SI) of a node should be minimum as long as all real-time streams meet their deadlines. Minimizing the reserved bandwidth ensures that a node has the maximum amount of sleep time, thereby minimizing the energy consumption of its wireless network card. From the perspective of the BS, the normalized bandwidth of each node should be minimum as well so that the BS's throughput is maximized and the maximum amount of bandwidth is available for potential future reservation requests of other nodes. Therefore, the problem and objective can be stated as:

Problem 3.1: **Minimum Bandwidth Reservation (MBR).** Given a node's set $\mathcal{S} = \{\mathcal{S}_i\}_1^n$ of periodic streams, a fixed service interval SI, and the node's scheduling policy A, determine the minimum SP such that all streams in $\mathcal{S}$ meet their deadlines.

Note that in the MBR problem minimizing the SP value is equivalent to minimizing the bandwidth (SP/SI) since the SI value is given and fixed. Borrowing the scheduling model concept from [17], where a scheduling model $M = (S, B, A)$ is defined as a tuple of a resource/bandwidth partition model B, a scheduling algorithm/policy A, and a workload model S, the scheduling model M of the MBR problem can be denoted as $(\mathcal{S} = \{\mathcal{S}_i\}_1^n, B = (SP, SI), A)$. In particular, the resource partition model corresponds to the single time slot periodic partition (STSPP) model introduced in [3]. The MBR problem aims to obtain the minimum SP value while all the other parameters of the scheduling model M are given.

Normally, the SI value given to a node is a multiple of the beacon interval of the BS [2]. The beacon interval is determined by the application scenarios for which the WLAN is deployed. Once bandwidth reservations are allocated to nodes by the BS, a change of the SI value will necessitate changes of all allocated bandwidth reservations and trigger re-negotiations between the BS and all of its nodes. A solution/algorithm to the problem can run at each client node if the SI value is given from the BS to the client node, or run at the BS if the client node communicates all of its stream parameters to the BS. The actual implementation choice depends on communication and computation capacities of nodes and the BS.

4 Model Transformation

To approach the MBR problem, we first transform the scheduling model $M = (\mathcal{S} = \{\mathcal{S}_i\}_1^n, B = (SP, SI), A)$ of the MBR problem to another scheduling model $M' = (S', B', A')$, where the stream set $\mathcal{S}'$ extends $\mathcal{S}$ by adding a sleep stream; the scheduling policy A' extends A by assigning the sleep stream the highest priority; and B' represents the dedicated resource allocation for $\mathcal{S}'$. We show that the two scheduling models are schedulably equivalent, i.e., M is schedulable if and only if M' is schedulable. As a corollary, we show that the MBR problem in M is a dual of the maximum execution time problem of the sleep stream in M'.

We first define the sleep stream $\mathcal{S}_0$ as follows.

Definition 4.1: **Sleep Stream**. Given a bandwidth profile (SP, SI), the corresponding sleep stream $\mathcal{S}_0$ is defined with $p_0 = SI$ and $D_0 = e_0 = \theta_0 = SI - SP$, i.e., its period is equal to the service interval SI; its execution time and deadline are invariantly equal and both are equal to the complement portion of the allocated access interval SP, i.e., $SI - SP$. Furthermore, the sleep stream cannot be interrupted (i.e., its non-preemption portion is equal to the execution time).

Figure 2 shows the transformation of the bandwidth profile and the sleep stream. Since e_0 is invariantly equal to

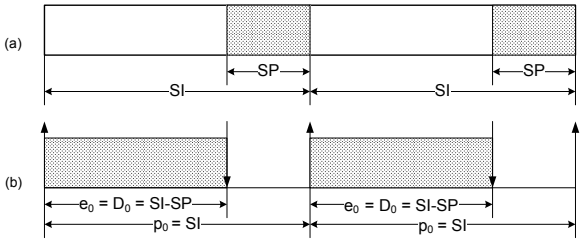

Figure 2. Transformation of the bandwidth profile and the sleep stream: (a) A given bandwidth profile (SP, SI). (b) The corresponding sleep stream $\mathcal{S}_0$ with D_0 = e_0 = SI-SP and p_0 = SI. It always has the highest priority and it is non-interruptible.

D_0, $\mathcal{S}_0$ must have the higher priority than all data streams and no data stream can block it. We define an extended stream set $\mathcal{S}'$ for $\mathcal{S}$ as follows.

Definition 4.2: **Extended Stream Set**. Given a set $\mathcal{S} = \{\mathcal{S}_i\}_1^n$ of periodic streams and a bandwidth profile (SP, SI), the extended stream set $\mathcal{S}'$ is equal to the union of the stream set $\mathcal{S}$ and $\mathcal{S}_0$, i.e., $\mathcal{S}' = \mathcal{S} \cup \mathcal{S}_0 = \{\mathcal{S}_i\}_0^n$.

Moreover, the accompanying scheduling policy A' for $\mathcal{S}'$ is defined as follows.

Definition 4.3: **Extended Scheduling Policy**. Given an extended stream set $\mathcal{S}'$ and the scheduling policy A for $\mathcal{S}$, the extended policy A' for $\mathcal{S}'$ is an extension of A from $\mathcal{S}$ to $\mathcal{S}'$, in which $\mathcal{S}_0$ always has the highest priority and the priority relationships among the streams in $\mathcal{S}$ are unchanged. In other words, the restriction of A' to $\mathcal{S}$ is equal to A, i.e., $A'|\mathcal{S} = A$.

Now, we state the equivalent problem of MBR in terms of A' and $\mathcal{S}'$:

Problem 4.4: **Maximum Execution Time (MET)**. Given a node's set $\mathcal{S} = \{\mathcal{S}_i\}_1^n$ of periodic streams, a service interval SI, and the node's scheduling policy A, determine the A'-schedulable maximum execution time (which is invariantly equal to the relative deadline) of $\mathcal{S}_0$ among the extended stream set $\mathcal{S}'$.

The scheduling model M' of the MET problem can be denoted as $\mathcal{M}' = (\mathcal{S}' = \mathcal{S} \cup \mathcal{S}_0, B' = (SI, SI), A')$. The MET problem aims to obtain the maximum execution/sleep time of the sleep stream $\mathcal{S}_0$ while all the other parameters of the scheduling model M' are derived as described above. Specifically, we observe that $\mathcal{S}'$ is schedulable in M' if and

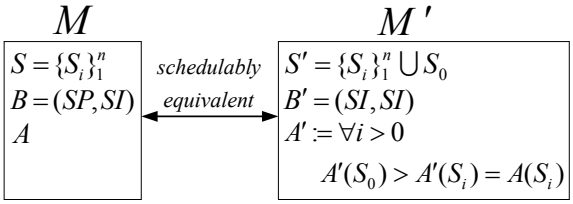

Figure 3. Schedulably equivalence between the original scheduling model $M = (\mathcal{S}, B, A)$ and the transformed scheduling model $M' = (\mathcal{S}', B', A')$, where $\mathcal{S}_0$ is the sleep stream, $B' = (SI, SI)$ represents dedicated resource allocation, and A' is an extension from A with $\mathcal{S}_0$ assigned the highest priority.

only if $\mathcal{S}$ is schedulable in M (as schematically shown in Figure 3), leading us to the following theorem.

Theorem 4.5: M and M' are schedulably equivalent.

Proof: Since $\mathcal{S}_0$ has always the highest priority, regardless of which priority-driven scheduling policy is in use (i.e., once a job of $\mathcal{S}_0$ is released, it will be executed immediately until its completion without interruption), the schedule $\mathcal{T}_A$ of $\mathcal{S}$ using policy A will be exactly the same as the sub-schedule of $\mathcal{S}$ in the schedule $\mathcal{T}_{A'}$ of $\mathcal{S}'$ using A', no matter what the values of SI and SP are. As a result, $\mathcal{S}$ is schedulable using A if and only if $\mathcal{S}'$ is schedulable using A'. □

As a corollary of Theorem 4.6, the MBR and the MET are dual problems. Since the execution (sleep) time e_0 of $\mathcal{S}_0$ is equal to SI - SP and the SI value is fixed, minimizing SP is equivalent to maximizing e_0. We state this conclusion formally as

Corollary 4.6: The problem MBR on M and the problem MET on M' are dual.

In the following sections, we will, based on the existing exact uniprocessor task schedulability test and exploiting the characteristics of the new scheduling model M', develop our algorithms for a class of priority-driven scheduling policies, including three common ones: fixed-priority policies (e.g., RM and DM), dynamic-priority priorities (e.g., EDF), and FIFO.

5 Minimum Bandwidth Reservation

A priority-driven scheduling policy (a set of priority rules) can be considered as a time-varying function $A(\mathcal{J}_{i,k}, \mathcal{J}_{j,l}, t)$ for any two jobs $\mathcal{J}_{i,k}$ and $\mathcal{J}_{j,l}$, taking on values -1, 0, and 1. Function $A(\mathcal{J}_{i,k}, \mathcal{J}_{j,l}, t)$ = 1 (-1 and 0, respectively) if and only if $\mathcal{J}_{i,k}$ has a higher (lower and equal, respectively) priority than $\mathcal{J}_{j,l}$ at time t. $A(\mathcal{J}_{i,k}, \mathcal{J}_{j,l}, t)$ is of form

$$A(\mathcal{J}_{i,k}, \mathcal{J}_{j,l}, t) = \begin{cases} 1 & t \in [r_{i,k}, d_{i,k}] \setminus [r_{j,l}, d_{j,l}] \\ \text{policy dependent} & t \in [r_{i,k}, d_{i,k}] \cap [r_{j,l}, d_{j,l}] \\ -1 & t \in [r_{j,l}, d_{j,l}] \setminus [r_{i,k}, d_{i,k}] \end{cases} \quad (1)$$

The symbol $\setminus$ in Equation 1 is the set difference operator. We describe processor idle intervals using a special task $\mathcal{S}_{-1}$. A policy A is priority-driven if and only if (1) a backlogged processor is not idle (i.e., $A(\mathcal{J}_{i,k}, \mathcal{J}_{-1,l})$ = 1 for every job $\mathcal{J}_{i,l}$, where $i \neq -1$) and (2) the processor invariantly executes the highest-priority backlogged job $\mathcal{J}_{i,k}$ at any time t (i.e., $A(\mathcal{J}_{i,k}, \mathcal{J}_{j,l}, t)$ = 1 for every $\mathcal{J}_{j,l} \neq \mathcal{J}_{i,k}$).

In systems that use dynamic job-level scheduling, a datagram (a job) may consist of multiple packets, whose priorities may be different from each other or they may change over time. This, while adding flexibility, would significantly add to the complexity of the network scheduler. Instead, this work assumes that the network scheduler is job-level static, i.e., all packets from the same datagram have the same priority, which is assigned at the release time of the datagram. We would like to point out that many priority-driven scheduling policies (especially widely used ones such as RM, DM, EDF, FIFO, LILO, and round-robin) satisfy these constraints. Policies such as LST (least slack time) do not fall into this category. In the following sections, we restrict our focus to the class of priority-driven policies that satisfy these constraints. Moreover, we require that for a scheduling policy the synchronous (in-phase) busy interval of any task set is the worst-case scenario, i.e., if all task instances released within this interval meet their deadlines, then all task instances will meet their deadlines everywhere. A synchronous busy interval of the stream set starts with all streams generating their datagrams at the same time and ends with the transmission of the last one of these datagrams. We state these two constraints discussed above explicitly as follows.

- **Job-level static**. The scheduling policy is static at the job level, but it can be either static or dynamic at the task level.
- **Critical instant**. The worst-case response time of any given stream set by a given scheduling policy occurs within the synchronous (in-phase) busy interval of the stream set.

5.1 Generic Framework

In this section, we develop a generic algorithmic framework (Algorithm 1) to solve the MET problem, based on an augmented time-demand analysis. In the following description, we distinguish between the *finished portion* and the *unfinished portion* of execution before a given deadline. This concept allows us to conservatively reduce the sleep time of the sleep stream to approach its minimum value, assuming that the reduced sleep time (equal to the unfinished portion) is solely utilized by the job missing its deadline. To compute the finished/unfinished portions of a job, we define a generic time-demand function at every job release event point. The available time at a job release event point for lower-priority jobs is equal to the dedicated time supply minus the generic time demand. As a result, the finished portion of a job before its deadline is equal to, provided it has not finished, the maximum available time over all job release event points between its release time and its deadline. Our approach avoids iterative computations of fixed-point equations (which are usually found in existing response time computations).

Algorithm 1 Minimum bandwidth reservation of stream set $\mathcal{S}'$ using policy A'

```
1:  function MinBW():
2:  /* e_0 is initialized to the maximum value since the goal is to minimize
    bandwidth over-reservation for S' */
3:  e_0=D_0=(1 - sum_{1<=i<=n} e_i/p_i) * SI
4:  /* scan every job J_{i,j} ∈ Π (except the sleep job) in increasing order of
    release times or deadlines, where 0 < i <= n and Π is the synchronous busy
    interval starting at time 0 */
5:  for each J_{i,j} within Π do
6:      loop
7:          compute the finished portion e^+_{i,j} and unfinished portion e^-_{i,j}
            using Equation 5 and Equation 6, and the completion time
            f_{i,j} using Equation 7
8:          if e^-_{i,j} > 0 then
9:              decrement the execution time e_0 of S_0 using Equation 2
10:         else
11:             Break
12:         end if
13:         /* abort when it is impossible to meet the job's deadline even if the
            node were allocated the entire bandwidth (i.e., SP=SI) */
14:         if e_0 < 0 then
15:             Abort
16:         end if
17:     end loop
18:     /* determine if the synchronous busy interval Π is terminated using
        Equation 8 or whether it is impossible to meet the job's deadline even
        if the node were allocated the entire bandwidth (i.e., SP=SI) */
19:     if v(f_{i,j}) <= f_{i,j} then
20:         Break
21:     end if
22:     move on to check next job (other than the sleep job)
23: end for
24: return max{SI - e_0 - θ, 0}
```

This framework scans every job (except jobs of $\mathcal{S}_0$) within the synchronous busy interval Π to check if it meets its deadline (lines 5-23). The timeliness check of the jobs of $\mathcal{S}_0$, referred to as *sleep jobs* in the remainder of the paper, are temporarily skipped until the synchronous busy interval ends. However, their impact on the timeliness of *regular* (non-sleep) jobs is still taken into account. Specifically, every regular job must take into account all sleep jobs released

before its deadline since sleep jobs have the highest priority.

We consider the execution of each job $\mathcal{J}_{i,j}$ to be divided into two distinct portions: the *finished portion* $e^+_{i,j}$ and the *unfinished portion* $e^-_{i,j}$ before the deadline $d_{i,j}$. According to this definition, we have $e_{i,j} = e^+_{i,j} + e^-_{i,j}$. Job $\mathcal{J}_{i,j}$ meets its deadline if and only if $e^-_{i,j} = 0$ or equivalently $e^+_{i,j} = e_{i,j}$. If job $\mathcal{J}_{i,j}$ misses its deadline, the execution time e_0 will be reduced such that the total amount of reduced execution times of all sleep jobs before $d_{i,j}$ is equal to $e^-_{i,j}$, i.e.,

$$e^-_{i,j} = \left\lfloor \frac{d_{i,j}}{SI} \right\rfloor e_0 + \min\{d_{i,j} \| SI, e_0\} - \left\lfloor \frac{d_{i,j}}{SI} \right\rfloor (e_0 - \delta) - \min\{d_{i,j} \| SI, e_0 - \delta\} \quad (2)$$

where δ is the amount by which e_0 needs to be reduced. In Equation 2, $\|$ is the modulus operator. Equation 2 computes the total sleep time before deadline $d_{i,j}$ that $\mathcal{S}_0$ can obtain prior to and after the sleep time deduction, i.e., $\left\lfloor \frac{d_{i,j}}{SI} \right\rfloor e_0 + \min\{d_{i,j} \| SI, e_0\}$ and $\left\lfloor \frac{d_{i,j}}{SI} \right\rfloor (e_0 - \delta) + \min\{d_{i,j} \| SI, e_0 - \delta\}$, respectively. In this manner, the algorithm conservatively expects the total reduced amount to be solely utilized for the execution of the unfinished portion $e^-_{i,j}$. The reduction procedure ends when the job's deadline is met or the algorithm aborts when even supplying the entire bandwidth (i.e., SP=SI) cannot meet the job's deadline. If job $\mathcal{J}_{i,j}$ meets its deadline, the algorithm moves on to the next job until the busy interval Π is terminated.

Figure 4 illustrates the computation of the finished portion, unfinished portion, and completion time of job $\mathcal{J}_{i,j}$.

In order to compute the two distinct portions for a job, we resort to the time-demand analysis of the job. The time-demand $w_{i,j}(t)$ of job $\mathcal{J}_{i,j}$ at time t is defined as the sum of aggregated execution times of all jobs released before t with higher or equal priority than $\mathcal{J}_{i,j}$ and the maximum non-preemption portion θ, as in [11]:

$$w_{i,j}(t) = \sum_{A'(\mathcal{J}_{k,l}, \mathcal{J}_{i,j}, t) \geq 0} \{e_k\} + \theta \quad (3)$$

Note that the generic time-demand function defined in Equation 3 is a generalization of the traditional time-demand function for either FP or EDF. The time-demand function at time t for a task for an FP scheduling algorithm is defined as the sum of execution times of jobs (released before t) of higher-priority tasks. The time-demand function at time t for the EDF scheduling algorithm is defined as the sum of execution times of jobs whose deadline is no greater than t. Since a job can only be delayed by at most θ time units (i.e., the non-preemption portion) by a priority-driven policy, we add this value to the time-demand function in Equation 3. Notice that function $w_{i,j}(t)$ is a staircase function, which jumps at release times $\{t_0, t_1, \cdots t_m\}$, where $t_k < t_{k+1}$, $t_0 = r_{i,j}$, and $t_m \leq d_{i,j}$ for all t_k, of jobs with priority no less than $\mathcal{J}_{i,j}$. The value of $w_{i,j}(t_k)$ corresponds to the value right before the jump at time t_k. The computation of the staircase function is different from that of the conventional staircase time-demand function [25], [26], [9], where it may jump at any time point depending on the pattern of the stream set. The conventional staircase time-demand function has been designed solely for computing a job's response/completion time, whereas our staircase function can, besides computing a job's response/completion time if the job's deadline is met, compute a job's finished and unfinished portion before its deadline and determine if a busy interval is terminated (with the help of the additional line $g(t)$ in Figure 4). Specifically, we denote the union of $\{t_0, t_1, \cdots t_m\}$ and $d_{i,j}$ as event points $\mathcal{E}_{i,j} = \{t_0, t_1, \cdots t_m\} \cup d_{i,j}$ of $\mathcal{J}_{i,j}$. Job $\mathcal{J}_{i,j}$ has a chance to be executed before $t \leq d_{i,j}$ if and only if there exists some $s \leq t$ such that $w_{i,j}(s)$ is less than $s + e_{i,j}$. As shown in Figure 4, function $w_{i,j}(t)$ and the vertical straight lines t, where $t \leq d_{i,j}$, intersect the 45° straight line translated upwards by $e_{i,j}$ (i.e., the straight line $g(t)$ in Figure 4) and form isolateral right triangles. The amount of time which $\mathcal{J}_{i,j}$ can utilize before $t \leq d_{i,j}$ (denoted as $e_{i,j}(t)$), can be computed as

$$e_{i,j}(t) = \min\{e_{i,j}, \max_{r_{i,j} \leq s \leq t \leq d_{i,j}} \{0, e_{i,j} + s - w_{i,j}(s)\}\} \quad (4)$$

which corresponds to the maximum length of the vertical legs of those triangles contained within the two straight lines $g(t)$ and $f(t)$ in Figure 4. Since function $e_{i,j}(t)$ is bounded, non-decreasing, and obtains local maximums only at event points $\mathcal{E}_{i,j}$, the finished portion $e^+_{i,j}$ (i.e., $e_{i,j}(d_{i,j})$), and the unfinished portion $e^-_{i,j}$ are computed as follows.

$$e^+_{i,j} = \min\{e_{i,j}, \max_{t_k \in \mathcal{E}_{i,j}} \{0, e_{i,j} + t_k - w_{i,j}(t_k)\}\} \quad (5)$$

$$e^-_{i,j} = e_{i,j} - e^+_{i,j} \quad (6)$$

If job $\mathcal{J}_{i,j}$ meets its deadline, as indicated by the staircase function $w_{i,j}(t)$ intersecting the 45° straight line $f(t)$ in Figure 4(a), its completion time $f_{i,j}$ is equal to the fixed point t such that $w_{i,j}(t) = t$. Alternatively, it is also equal to $w_{i,j}(t_k)$ where t_k is the first event point satisfying $t_k - w_{i,j}(t_k) \geq 0$, i.e.,

$$f_{i,j} = w_{i,j}(t_k) \quad (7)$$
$$\text{where } t_k = argmin_{t_l \in \mathcal{E}_{i,j}}\{t_l - w_{i,j}(t_l) \geq 0\}$$

As a byproduct, we can save substantial computation for the completion time by iteratively solving the fixed-point equation $w_{i,j}(t) = t$, which is the usual practice in the literature. Therefore, the algorithm can compute every job's response time along its finished and unfinished portion, if the job's deadline is met. Otherwise, the completion time is undefined.

Relying on the computation of completion times, we can decide if the synchronous busy interval Π is terminated, via constructing another time-demand function $v(t)$ called

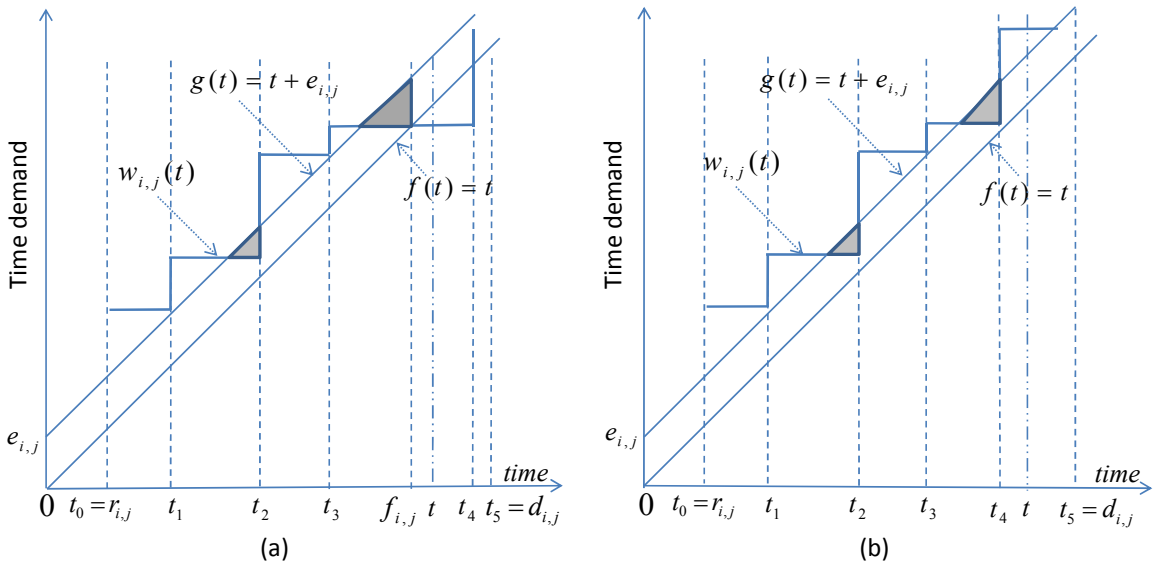

Figure 4. An illustration of the augmented time-demand analysis to compute the finished portion $e^+_{i,j}$, unfinished portion $e^-_{i,j}$, and completion time $f_{i,j}$ of job $\mathcal{J}_{i,j}$, given a priority-driven scheduling policy A'. Job $\mathcal{J}_{i,j}$ is released at $t_0 = r_{i,j}$ and $\{t_k\}_{k=1}^4$ are release times of jobs with higher priority than $\mathcal{J}_{i,j}$ within $[r_{i,j}, d_{i,j}]$, The time-demand function $w_{i,j}(t)$ jumps at every release time. The amount of time that $\mathcal{J}_{i,j}$ can utilize before $r_{i,j} \le t \le d_{i,j}$ is $e_{i,j}(t)$. Its value corresponds to the maximum length of the legs of these isolateral right triangles (shown shaded in the graph) contained within the two straight lines $g(t)$ and $f(t)$ within time range $[r_{i,j}, t]$. It is bounded, non-decreasing, and obtains local maximums only at event points $\mathcal{E}_{i,j} = \{t_k\}_{k=1}^4 \cup d_{i,j}$. The maximum value of $e_{i,j}(t)$ (i.e., $e_{i,j}(d_{i,j})$) is equal to $e^+_{i,j}$. In part (a), $e^+_{i,j} = e_{i,j}$, job $\mathcal{J}_{i,j}$ completes at time $f_{i,j}$ before its deadline. In part (b), $e^+_{i,j} < e_{i,j}$ and therefore job $\mathcal{J}_{i,j}$ has not completed before its deadline.

overall time-demand function. Function $v(t)$ represents the total demand time for all jobs, regardless of their priorities, released before time t, i.e.,

$$v(t) = \sum_{r_{k,l} < t} \{e_k\} + \theta = \sum_{0 \le k \le n} \left\lceil \frac{t}{p_k} \right\rceil e_k + \theta \quad (8)$$

If $v(f_{i,j}) \le f_{i,j}$ (line 19 in Algorithm 1), then evidently Π is terminated before or at $f_{i,j}$.

Finally, we consider the non-preemptive portions of tasks. Sleep jobs should not be blocked by any regular jobs, but it can happen if a regular job is released within less than θ time units before a sleep job. In the worst case, a sleep job can be blocked by regular jobs by θ time units. Therefore, after the synchronous busy interval terminates, the algorithm additionally deducts θ time units from the sleep time of $\mathcal{S}_0$ (line 24 in Algorithm 1), which is equivalent to adding bandwidth reservation of θ to the result regardless of the non-preemptivity of $\mathcal{S}_0$.

We state the correctness and the exactness of Algorithm 1 as a theorem:

Theorem 5.1: For any periodic stream set and any job-level static scheduling policy under which the scenario of the worst-case response time is the synchronous busy interval, the bandwidth minBW computed by Algorithm 1, if successfully terminated, is the exact minimum bandwidth requirement of the stream set.

Proof: We first show that Algorithm 1 (lines 5-23) resembles the exact schedulability test [9], [27], [11], [12]. Initially, the bandwidth reservation (thus sleep time of $\mathcal{S}_0$), is set to the theoretical lower bound (line 3). If there is no deadline miss within the synchronous busy interval, then this initial bandwidth reservation is the exact minimum bandwidth requirement. Once a job misses its deadline (lines 8-12), the minimum total amount (equal to the unfinished portion of the job) of the sleep time is deducted on the hypothesis that the deducted amount of time before the job's deadline is solely utilized by the job (Equation 2). Thus, Algorithm 1 never over-deducts the sleeping time of $\mathcal{S}_0$. As a result, the job will finish at a time before its deadline, but as late as possible. This completely resembles the exact schedulability test.

Sleep jobs of $\mathcal{S}_0$ can be blocked, in the worst case, by regular jobs by θ time units. This can be considered by increasing the bandwidth reservation by θ time units (line 23). If the final resulting bandwidth reservation is less than zero, then $\mathcal{S}_0$ does not exist. Thus, we prove this theorem.

□

Although the generic algorithm can solve the MET problem (thus the MBR problem), it incurs unnecessary overheads for some specific scheduling policies. Therefore, the following sections investigate some simplifications for three common scheduling policies by exploiting their individual properties.

5.2 Fixed-Priority

The priority rule of a fixed-priority scheduling policy A is static at stream level, i.e., the second case of Equation 1 is not time-varying. Suppose that streams are indexed in the decreasing order of their priorities, i.e., stream $\mathcal{S}_i$ has a higher priority than $\mathcal{S}_k$ if $i < k$. We denote the subset of streams with equal or higher priority than $\mathcal{S}_i$ as Π_i. The finished portions and completion times of jobs of stream $\mathcal{S}_i$ are only affected by jobs of stream $\mathcal{S}_k$, where $k < i$. Therefore, the time-demand function $w_{i,j}(t)$ (Equation 3) can be customized to

$$w_{i,j}(t) = je_i + \theta + \sum_{k=0}^{i-1} \left\lceil \frac{t}{p_k} \right\rceil e_k$$
$$for \quad (j-1)p_i < t \leq w_{i,j}(t) \tag{9}$$

and the computation of the finished/unfinished portion (Equation 5) can be customized to

$$e^+_{i,j} = \min\{e_{i,j}, \max_{r_{i,j} \leq r_{k,l} \leq d_{i,j} \cap k<i} \{0, e_{i,j} + r_{k,l} - w_{i,j}(r_{k,l})\}\} \tag{10}$$

based on the generic schedulability test of a fixed-priority policy in [9]. As discussed before, $w_{i,j}(t)$ in Figure 4 jumps at the release times of streams with higher priorities than $\mathcal{S}_i$. Since the timeliness of a stream cannot be affected by streams with priorities lower than the stream, not every released job of the stream within the synchronous busy interval needs to be checked. Only jobs within a synchronous busy interval Π_i must be checked [9]. A synchronous busy interval Π_i starts at an instant when (i) all jobs in Π_i released before this instant have completed and (ii) every stream in Π_i releases a job at this instant. The interval ends with the completion of all released jobs in Π_i.

5.3 EDF

In EDF, jobs with priorities higher than the priority of a job $\mathcal{J}_{i,j}$ are those whose deadlines are no later than $d_{i,j}$, and therefore they will be executed before $d_{i,j}$. Thus, the time-demand function (Equation 3), finished/unfinished portion function (Equation 5), and completion time function (Equation 7) can be customized as follows.

$$w(t) = \sum_{0 \leq k \leq n} \left\lfloor \frac{t + p_k - D_k}{p_k} \right\rfloor e_k + \theta \tag{11}$$

$$e^+_{i,j} = \min\{e_{i,j}, \max\{0, w(d_{i,j}) - d_{i,j}\}\} \tag{12}$$

$$f_{i,j} = w(d_{i,j}) \tag{13}$$

which are the same as in [11] and [12]. The jobs released within $(r_{i,j}, d_{i,j}]$ with higher priorities than $\mathcal{J}_{i,j}$ are those whose deadlines are also within $(r_{i,j}, d_{i,j}]$. Since these jobs will complete within $(r_{i,j}, d_{i,j}]$ using the EDF policy, their demand times are equal to their actual execution times. Therefore, they have no impact on the finished or unfinished portions of $\mathcal{J}_{i,j}$. As a result, the unfinished portion $e^-_{i,j}$ of $\mathcal{J}_{i,j}$ is equal to the difference between the demand time $w(d_{i,j})$ at $d_{i,j}$ and the deadline $d_{i,j}$ itself.

We can also check jobs in the order of their deadlines, rather than their release times. A benefit of this is that the completion time of a job $\mathcal{J}_{i,j}$ is equal to the time-demand $w(d_{i,j})$ at $d_{i,j}$. If $e^-_{i,j} > 0$, the sleep time of $\mathcal{S}_0$ must be deducted to compensate for it. Since $\mathcal{J}_{i,j}$ will use up all the deduction, iterative checking of deadline misses is not necessary. Furthermore, if all jobs released before $w(d_{i,j})$ are completed, then the synchronous busy interval Π is terminated. In other words, if the release time of the next job to be analyzed is no less than $w(d_{i,j})$, Π terminates. Other refinements, e.g., conditionally skipping over some jobs within the synchronous busy interval [28], can be used to further improve the algorithm.

5.4 FIFO

In FIFO, only jobs released no later than job $\mathcal{J}_{i,j}$ have higher priority than $\mathcal{J}_{i,j}$. The time-demand function (Equation 3), finished/unfinished portion function (Equation 5), and completion time function (Equation 7) can be customized as:

$$w_{i,j} = \sum_{0 \leq k \leq n} \left\lceil \frac{r_{i,j}}{p_k} \right\rceil e_k + \theta \tag{14}$$

$$e^+_{i,j} = \min\{e_{i,j}, \max\{0, w_{i,j} - d_{i,j}\}\} \tag{15}$$

$$f_{i,j} = w_{i,j} \tag{16}$$

The time-demand function is a constant function within $[r_{i,j}, d_{i,j}]$. Reflecting on the shape of $w_{i,j}(t)$ in Figure 4, we see that the function is a horizontal line. Furthermore, if $w_{i,j}$ is greater than $d_{i,j}$, the unfinished portion $e^-_{i,j}$ can be compensated by the reduction of the sleep time of $\mathcal{S}_0$. In particular, the total reduced amount of sleep times of all jobs of $\mathcal{S}_0$ released before $d_{i,j}$ is equal to $e^-_{i,j}$.

6 Results and Performance

This section describes a set of simulation experiments that have been conducted to evaluate the over-reservation ratio of bandwidth and the computational overhead of our algorithm under typical application scenarios using four different scheduling policies (EDF, FIFO, RM, and DM). These results can provide the system designer and administrator with guidance on which policy to choose in what workload scenarios, also indicating how much bandwidth would be wasted. The periodic streams are generated with random parameters within given ranges and distributions.

Moreover, we experimentally verify the correctness of our algorithm using various test cases.

We use the approach recommended by [6] to generate the task periods. To generate a feasible stream set of N periodic stream with given total utilization U, we first generated a random utilization U_i for each stream $\mathcal{J}_i$ uniformly distributed in (0, 1) and then normalized these utilizations to have U=$\sum_{1\leq i\leq n} U_i$=$\sum_{1\leq i\leq n} \frac{e_i}{p_i}$. We also generated a stream transmission time e_i of stream $\mathcal{J}_i$ as a random variable uniformly distributed in $[e_{min}, e_{max}]$ and calculated the period of each stream as $p_i = e_i/U_i$. The relative deadline D_i of each stream $\mathcal{J}_i$ has been generated according to a random variable V_i (validity) uniformly distributed in $[V_{min}, V_{max}]$ such that $D_i = p_i * V_i$. We set the parameters to match

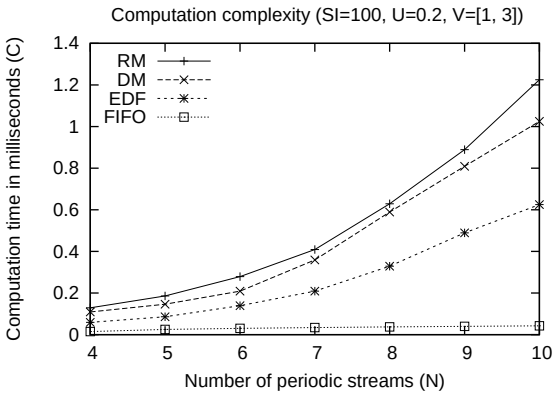

Figure 5. Computation time on an Intel Core2 1.86GHz processor while varying the number N of streams, given the service interval SI=100, the uniform distribution of ratio of deadline to period V=[1, 3], and the total utilization U=0.2.

typical application scenarios in wireless local area networks. The SI is set to 100 milliseconds, which is the default beacon interval in IEEE 802.11 [2]. The range $[e_{min}, e_{max}]$ of datagram sizes is set to $[1, 10]$ milliseconds, which allows a stream $\mathcal{S}_i$ to generate 1 to 5 packets within a single stream period p_i. Default values for N, U, and $[V_{min}, V_{max}]$ are set to 6, 0.2, and $[1, 3]$, respectively. Moreover, we require that the deadline of every generated task is no less than the SI value since 1) the bandwidth for a task with deadline less than the SI value is largely over-reserved, and 2) the SI value in practice should be adjusted to be less than the deadline value. Unless otherwise specified, we use the above default parameters. The non-preemption portion is set to 2 milliseconds. The graphs of all experiments depict the average results over 1000 generated stream sets.

In the first set of experiments, we tested the computational overhead of the algorithm as a function of the number of tasks (Figure 5). We measure the average finish time for the computations on a laptop with an Intel Core2 1.86GHz processor. The algorithms for RM and DM closely approximate the generic solution to bandwidth computation and therefore show larger running times than the other scheduling policies. Moreover, the computation of the minimum bandwidth requirement is much more complicated for FP than either EDF or FIFO, since a job's response time is irregular in reaction to the change of resource supply. FIFO and EDF have lower computational overheads since their computations are extremely simplified compared to the generic algorithm. However, bandwidth computations are infrequent events (e.g., when a new stream joins or leaves), therefore the computational overhead is typically not very significant.

In another set of experiments, we studied the bandwidth over-reservation ratios under various scheduling policies for various stream sets. The over-reservation ratio E is defined as the normalized ratio of the bandwidth reservation $\frac{SP}{SI}$ to the idealized minimum utilization U=$\sum_{1\leq i\leq n} \frac{e_i}{p_i}$. That is,

$$E = \frac{SP}{SI * U} \tag{17}$$

Thus, E=1 means that all reserved bandwidth is used and no bandwidth is wasted (i.e., there is no over-reservation).

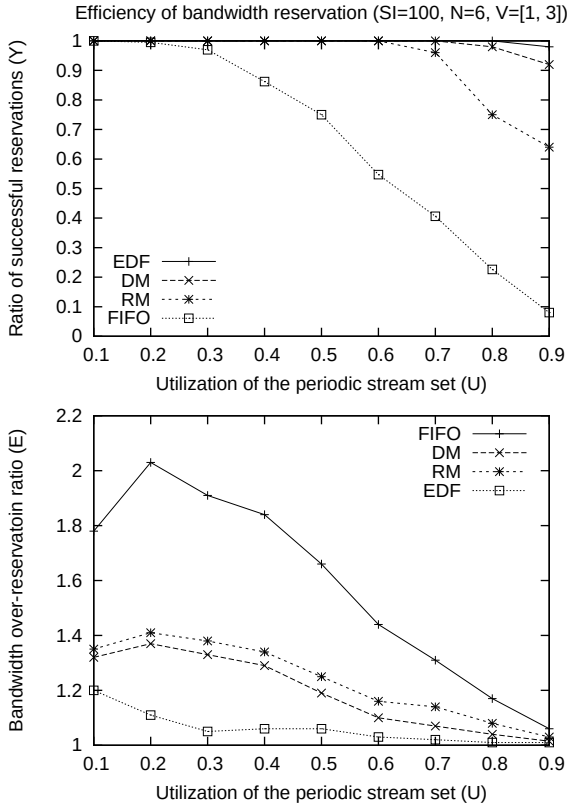

Figure 6. The percentage of successful reservations Y (a) and the over-reservation ratio E (b) while varying the utilization U of the stream set, given the service interval SI=100, a uniform distribution of ratio of deadline to period V=[1, 3], and the number of periodic streams N=6.

In order to demonstrate the correctness and accuracy of our approach (Theorem 5.1), we compute the bandwidth reservation (SP) using Algorithm 1 for every stream set,

and then use the computed bandwidth reservation as a given bandwidth profile (SP, SI) to simulate the packet scheduling and transmission process for the same stream set. The simulation reports no deadline misses. Furthermore, we then set the SP of the bandwidth profile (SP, SI) to a certain amount less than the SP computed by Algorithm 1, and again simulate the packet scheduling and transmission process. The result is that every such simulation reported some deadline misses, indicating that Algorithm 1 is able to determine the minimum necessary bandwidth reservations.

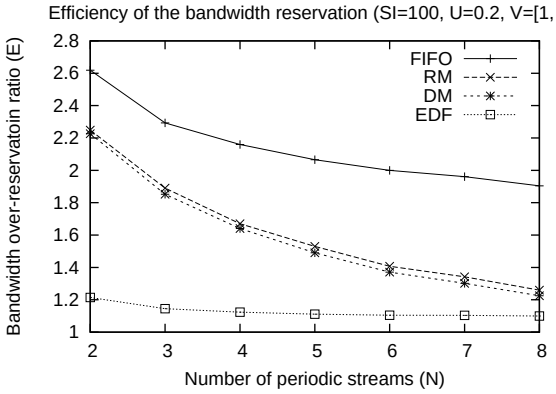

Figure 7. The over-reservation ratio E while varying the number N of streams, given the service interval SI=100, the uniform distribution of ratio of deadline to period V=[1,3], and the total utilization U=0.2.

We measured the over-reservation ratio E as a function of the total utilization U, the number of tasks N, and the packet validity V (Figures 6, 7, 8). For each scheduling policy, the reported ratio is averaged over all stream sets that are schedulable by all four policies. In other words, if a stream set is schedulable by one policy (e.g., EDF) but not schedulable by another (e.g., FIFO), then we exclude this stream set.

Figure 6 displays the over-reservation ratio (E) and the ratio (Y) of successful reservations (i.e., Algorithm 1 does not abort), respectively, over increasing utilization U. The results show that FIFO has the worst performance with respect to the two metrics: while the utilization is low it extremely over-reserves its bandwidth (E is up to 2.05) and as the utilization increases (U=[0.4, 1.0]) the ratio (Y) of successful reservations decreases linearly. On the other hand, EDF has the lowest reservation ratio and the highest ratio of successful reservations. Figures 7, 8, and 9 show the over-reservation ratio (E) as the number of streams N, the validity V, and the service interval SI vary, respectively. In Figure 7, as the number of streams decreases the over-reservation ratio of RM, DM and FIFO decreases since any unused resources have more opportunities to fit in a stream as the number of streams increases. DM does not provide many advantages over RM since relative deadlines of tasks

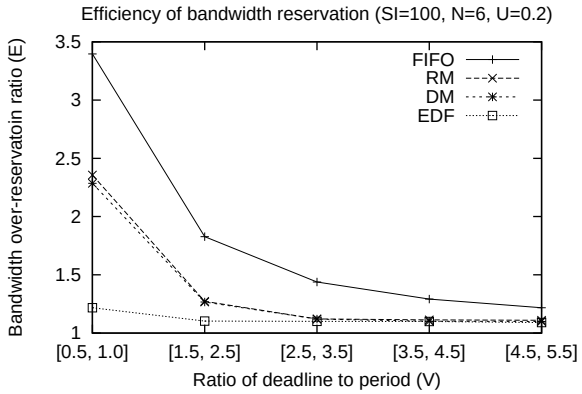

Figure 8. The over-reservation ratio E while varying the ratio V of deadline to period, given the service interval SI=100, the total utilization U=0.2 of the periodic task set, and the number of periodic streams N=6.

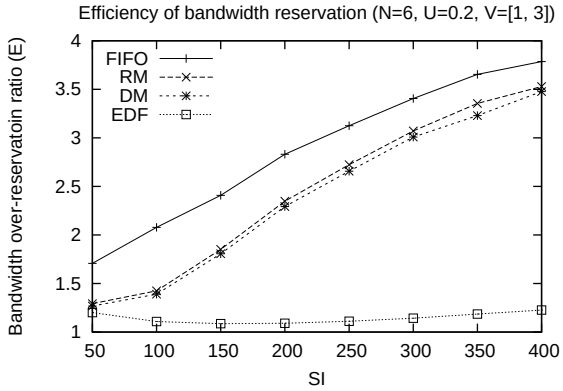

Figure 9. The over-reservation ratio E while varying the SI value provided by the BS, given a uniform distribution of ratio V of deadline to period [1,3], stream set utilization U=0.2, and the number of periodic streams N=6.

are all (much) greater than their respective periods, which is typical in a wireless streaming environment. Changing the number of streams has only a modest impact on EDF. Similarly, Figure 8 shows that increasing validity V only slightly affects the over-reservation ratio of EDF, whereas it has a high impact on RM, DM, and FIFO. The over-reservation ratios of FIFO, RM, and DM decrease rapidly when the validity decreases from 0.5 to 3.5. However when the validity decreases further (>4.5), the over-reservation ratios of all policies stabilize and converge. Figure 9 shows the impact of varying SI on the over-reservation ratio (E). Increasing SI only slightly affects the over-reservation ratio of EDF, but significantly impacts the over-reservation ratios of RM, DM, and FIFO.

In summary, the results derived from the simulation experiments reveal the following. (1) Increasing the validity (ratio of deadline to period) can greatly increase the bandwidth usage efficiency (less bandwidth waste) and increase

the probability of successful bandwidth reservation. Once the minimum bandwidth computation aborts, the most effective remedy is to increase the deadlines for some streams. (2) When the validity is less than 4.5, EDF requires less bandwidth reservation than either RM, DM, or FIFO, whereas when the validity goes beyond 4.5, the advantage of using complex scheduling policies diminishes and their minimum bandwidth reservations converge. In this case, FIFO works perfectly considering its low system overhead and scheduling complexity. Similarly, decreasing the given SI value has a similar effect as increasing the validity.

7 Conclusions and Future Work

The benefits of reservation-based channel accesses are two-fold: (1) they provide contention-free access within allocated/reserved channel access intervals to meet timing constraints predictably and (2) they allow a wireless radio to be powered down when the channel is not needed. Careless resource allocations may lead to poor support for real-time traffic or over-provisioning of scarce network resources. This paper solves the minimum bandwidth reservation problem to allow all streams to meet their timing constraints. To obtain a solution to the minimum bandwidth reservation problem, we transform it to a generic uniprocessor task schedulability problem, which is then addressed using a generic algorithm based on time-demand analysis. The generic minimum bandwidth reservation algorithm works for a subclass of priority-driven packet scheduling policies, including three common ones: fixed-priority (e.g., RM and DM), EDF, and FIFO. Refinements of the generic solution to these three types of policies are presented and discussed as well. The simulation results show that the generic algorithm is correct and practical in terms of computation complexity. The proposed bandwidth reservation scheme leads to minimal amounts of bandwidth waste if appropriate scheduling policies and stream parameters are selected for a given stream set. However, it also leads to potentially large energy savings, while being simple to implement and deploy. In our future work, we will address (1) how the base station chooses the optimal SI value to minimize the bandwidth/energy consumption of the entire wireless local area network, (2) how different client nodes use an SI which is a multiple of the beacon interval of the base station, and (3) how reservations (SPs) of nodes with different SIs can be composed efficiently to form a super frame.

References

[1] L. Wang and Y. Xiao, A survey of energy-efficient scheduling mechanisms in sensor networks, Mobile Networks and Applications, vol. 11, no. 5, pp. 723-740, 2006.

[2] IEEE 802.11 WG, Part II: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications: Medium access control (MAC) enhancements for quality of service (QoS), IEEE 802.11e Standard, Nov 2005.

[3] A. K. Mok, X. A. Feng, and D. Chen, Resource partition for realtime systems, in Proceedings of the Seventh Real-Time Technology and Applications Symposium. Washington, DC, USA, IEEE Computer Society, 2001, p. 75.

[4] T. L. Crenshaw, S. Hoke, A. Tirumala, and M. Caccamo, Robust implicit EDF: A wireless MAC protocol for collaborative realtime systems, Transaction. on Embedded Computing Systems., vol. 6, no. 4, p. 28, 2007.

[5] D. Rajan, C. Poellabauer, X. S. Hu, L. Zhang, and K. Otten, Wireless channel access reservation for embedded real-time systems, in Proceedings of the 7th ACM international conference on Embedded software, Atlanta, GA, USA, 2008, pp. 129-138.

[6] H. Hoang, G. Buttazzo, M. Jonsson, and S. Karlsson, Computing the minimum edf feasible deadline in periodic systems, in Proceedings of the 12th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications, Washington, DC, USA, 2006, pp. 125-134.

[7] O. Redell and M. Torngren, Calculating exact worst case response times for static priority scheduled tasks with offsets and jitter, in Proceedings of the Eighth IEEE Real-Time and Embedded Technology and Applications Symposium, Washington, DC, USA, 2002, p. 164.

[8] R. I. Davis, A. Zabos, and A. Burns, Efficient exact schedulability tests for fixed priority real-time systems, IEEE Trans. Comput., vol. 57, no. 9, pp. 1261-1276, 2008.

[9] J. Lehoczky, Fixed priority scheduling of periodic task sets with arbitrary deadlines, in Proceedings of the 11th Real-Time Systems Symposium, December 1990, pp. 201-209.

[10] 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.

[11] S. Baruah, A. Mok, and L. Rosier, Preemptively scheduling hardreal-time sporadic tasks on one processor, in Proceedings of the 11th IEEE Real-time Systems Symposium, December 1990, pp. 182-190.

[12] S. K. Baruah, L. E. Rosier, and R. R. Howell, Algorithms and complexity concerning the preemptive scheduling of periodic real-time tasks on one processor, Real-Time Systems, vol. 2, no. 4, pp. 301-324, 1990.

[13] F. Zhang and A. Burns, Schedulability analysis for real-time systems with edf scheduling, IEEE Transactions on Computers, vol. 118, no. 1, pp. 100-120, 2009.

[14] L. George and P. Minet, A FIFO worst case analysis for a hard real-time distributed problem with consistency constraints, in Proceedings of the 17th International Conference on Distributed Computing Systems, Washington, DC, USA, 1997, p. 441.

[15] X. A. Feng and A. K. Mok, A model of hierarchical real-time virtual resources, in Proceedings of the 23rd IEEE Real-Time Systems Symposium. Washington, DC, USA, IEEE Computer Society, 2002, p. 26.

[16] A. K. Mok and X. A. Feng, Towards compositionality in realtime resource partitioning based on regularity bounds, in Proceedings of the 22nd IEEE Real-Time Systems Symposium. Washington, DC, USA, IEEE Computer Society, 2001, p. 129.

[17] I. Shin and I. Lee, Periodic resource model for compositional real-time guarantees, in Proceedings of the 24th IEEE International Real-Time Systems Symposium. Washington, DC, USA, IEEE Computer Society, 2003, p. 2.

[18] I. Shin and I. Lee, Compositional real-time scheduling framework with periodic model, ACM Trans. Embed. Comput. Syst., vol. 7, no. 3, pp. 1-39, 2008.

[19] A. Easwaran, M. Anand, and I. Lee, Compositional analysis framework using edp resource models, in Proceedings of the 28th IEEE International Real-Time Systems Symposium. Washington, DC, USA, IEEE Computer Society, 2007, pp. 129-138.

[20] N. Fisher and F. Dewan, Approximate bandwidth allocation for compositional real-time systems, in Proceedings of the 21st Euromicro Conference on Real-Time Systems. Washington, DC, USA, IEEE Computer Society, 2009, pp. 87-96.

[21] J. Regehr and J. A. Stankovic, HLS: A framework for composing soft real-time schedulers, in Proceedings of the 22nd IEEE Real-Time Systems Symposium, London, UK, Dec. 2001, pp. 3-14.

[22] F. Zhang and A. Burns, Analysis of hierarchical edf pre-emptive scheduling, in Proceedings of the 28th IEEE International Real-Time Systems Symposium. Washington, DC, USA, IEEE Computer Society, 2007, pp. 423-434.

[23] W. Ye, J. Heidemann, and D. Estrin, Medium access control with coordinated adaptive sleeping for wireless sensor networks, IEEE/ACM Trans. Netw., vol. 12, no. 3, pp. 493-506, 2004.
[24] P. Puschner and A. Burns, A review of worstcase execution time analysis, Real-Time Systems, vol. 18, no. 2, pp. 115-130, 2000.
[25] N. Audsley, A. Burns, M. Richardson, K. Tindell, and A. J. Wellings, Applying new scheduling theory to static priority preemptive scheduling, Software Engineering Journal, vol. 8, pp. 284-292, 1993.
[26] J. Lehoczky, L. Sha, and Y. Ding, The rate monotonic scheduling algorithm: exact characterization and average case behavior, in Proceedings of the 10th Real Time Systems Symposium, Washington, DC, USA, 1989, pp. 166-171.
[27] J. W. S. Liu, Real-Time Systems. Upper Saddle River, NJ: Prentice Hall, 2001.
[28] K. Albers and F. Slomka, Efficient feasibility analysis for realtime systems with EDF scheduling, in Proceedings of the conference on Design, Automation and Test in Europe, Washington, DC, USA, 2005, pp. 492-497.