Quality of Service Scheduling in Cable and Broadband Wireless Access Systems

Mohammed Hawa and David W. Petr
Information and Telecommunications Technology Center
University of Kansas, Lawrence, Kansas, 66045
email: {hawa, dwp}@ittc.ku.edu

***Abstract*—In recent years, several last mile high-speed technologies have been explored to provide Internet access and multimedia services to end users [1]. Most notable of those technologies are Hybrid Fiber Coax (HFC) cable networks, Digital Subscriber Line (DSL), Satellite Access, and fixed Broadband Wireless Access (BWA) systems. The *de facto* standard for delivering broadband services over HFC networks is the Data Over Cable Service Interface Specifications (DOCSIS) protocol. For BWA systems, on the other hand, a new protocol, called IEEE 802.16, was developed for the same purpose. This paper presents a new and efficient scheduling architecture to support bandwidth and delay Quality of Service (QoS) guarantees for both DOCSIS and IEEE 802.16. Our design goals are simplicity and optimum network performance. The architecture we develop here supports various types of traffic including constant bit rate, variable bit rate (real-time and non-real-time) and best effort.**

I. INTRODUCTION

HYBRID Fiber Coax (HFC) cable networks have been mainly used in the past to deliver broadcast-quality TV signals to homes. The wide availability of such systems and the extremely wide bandwidth they provide allows extending their functionality to deliver high-speed broadband data signals to end users [2]. To provide such support, Data Over Cable Service Interface Specifications (DOCSIS) protocol [4, 5] was developed by a group of major cable operators called Cable Labs. DOCSIS was later adopted by the ITU and is now supported by many vendors. Versions 1.0 and 1.1 of DOCSIS were completed by 1999, and version 2.0 was introduced in early 2002.

Broadband Wireless Access (BWA) systems, on the other hand, are easier to implement and can be installed rapidly without extensive underground cable infrastructure. They can also be used to provide high-speed data access to subscribers [3]. The IEEE 802.16 standard was developed for BWA systems for that purpose, and was formally approved by the IEEE Standards Association in 2001 [6]. It is worth mentioning that IEEE 802.16 was a consolidation of two proposals, one of which was based on DOCSIS. This is mainly due to the many striking similarities between the HFC cable environment and the wireless BWA environment.

In this paper we develop a new scheduling architecture for both DOCSIS and IEEE 802.16 to allow them to efficiently support bandwidth and delay Quality of Service (QoS) guarantees. The developed architecture supports various types of traffic including constant bit rate, variable bit rate and best effort. The rest of this paper is organized as follows. In the next section, we provide a quick overview of both DOCSIS and IEEE 802.16 standards. We also describe some of their QoS-related features. We then propose our scheduling architecture in Section III. Sections IV and V provide more details about specific issues of the suggested scheduler operation. Finally, we conclude with a summary in Section VI.

This work was supported by Sprint Corporation.

II. BACKGROUND

A. DOCSIS

DOCSIS assumes an architecture in which a headend, called a Cable Modem Termination System (CMTS), controls the operations of many terminating Cable Modems (CMs) at subscriber locations. The medium between the CMTS and the different CMs is a two-way shared medium, in which downstream channels carry signals from the headend to users and upstream channels carry signals from users to the headend. Upstream and downstream channels in DOCSIS are separated using Frequency Division Duplex (FDD). DOCSIS defines both the physical layer and the Medium Access Control (MAC) protocol layer to be used on these channels.

A CM normally tunes to one upstream channel and an associated downstream channel. Each upstream channel is inherently a shared medium, and the CMTS controls access of the CMs to such a medium in an orderly manner by means of the MAC protocol. The main access scheme in DOCSIS 1.0 and 1.1 is time division multiple-access (TDMA). DOCSIS 2.0 also allows frequency division multiple-access (FDMA) and synchronous code division multiple-access (S-CDMA) to complement the original TDMA access scheme. Each upstream channel is further divided into a stream of fixed-size time minislots.

The DOCSIS MAC protocol utilizes a request/grant mechanism to coordinate transmission between multiple CMs. If a CM needs to transmit anything on the upstream channel, it first *requests*, from the CMTS, an opportunity to transmit a certain amount of data. The CMTS is then responsible for allocating such a transmission opportunity (called a *data grant*) in the next upstream frame(s) if capacity is available. Periodically, the CMTS sends a *bandwidth allocation map* (MAP) message over

the downstream channel to indicate to the CMs the specific time minislots allocated to them as their corresponding upstream transmission opportunities. As a result of reserving bandwidth, the CMs are guaranteed a collision-free transmission. Besides indicating the transmission opportunities for the different CMs, the MAP message indicates in which time intervals the different CMs are allowed to send their requests for transmission. This reservation interval is a contention interval in which collisions may actually happen. A contention resolution algorithm is used to resolve such collisions. DOCSIS uses the simple *binary exponential backoff* algorithm for contention resolution. Requests for transmission can also be piggybacked on data packets transmitted by the CMs on the upstream channel.

DOCSIS supports fragmentation and concatenation of data packets. Fragmentation happens when the CMTS provides a data grant to a CM that is smaller than the one the CM actually requested. In such a case, the CM fills the partial grant it receives with the maximum amount of data possible, and sends the rest of the data payload in the next allocated data grant.

To support QoS, DOCSIS 1.1 introduces the concept of *service flows*. At least one service flow must be setup between any particular CM and the CMTS to carry best-effort traffic. However, to support other types of traffic, the CM may opt to set up multiple service flows to the CMTS with each flow having its own characteristics and traffic parameters.

An upstream service flow in DOCSIS 1.1 and DOCSIS 2.0 can be classified within one of the following Upstream Service Flow Types: Unsolicited Grant Service (UGS), Real-Time Polling Service (rtPS), Non-Real-Time Polling Service (nrtPS) and Best Effort (BE). The way DOCSIS treats each of those service flow types is explained in Section II-C.

B. IEEE 802.16

The IEEE 802.16 standard for fixed BWA systems supports a metropolitan area network architecture. It assumes a point-to-multipoint topology, with a controlling base station (BS) that connects subscriber stations (SS) to various public networks linked to the BS. The BS and SSs are stationary and one SS typically serves one business or residential building.

The IEEE 802.16 standard defines a connection-oriented MAC protocol that supports multiple physical layer specifications. The physical layer air interface is optimized for bands from 10 to 66 GHz. IEEE 802.16 uses wider channels, compared with DOCSIS, for the downstream and upstream channels, which are separated using either Frequency Division Duplex (FDD), as in DOCSIS, or using Time Division Duplex (TDD). The access mode for the upstream channel is Time-Division Multiple Access (TDMA).

IEEE 802.16 utilizes contention and piggybacking, as in DOCSIS, to send requests to the BS for transmission opportunities on the upstream channel. The BS is the one responsible for assigning such transmission opportunities to different SSs and also for assigning a certain contention interval where such reservations can be made. IEEE 802.16 uses a *binary truncated exponential backoff* as its contention resolution protocol and maintains the concept of a bandwidth allocation MAP as in DOCSIS. Fragmentation and concatenation of data packets are also allowed.

To support QoS, IEEE 802.16 maintains the concept of a service flow. The Upstream Service Flow Types defined in IEEE 802.16 are, again: Unsolicited Grant Service (UGS), Real-Time Polling Service (rtPS), Non-Real-Time Polling Service (nrtPS) and Best Effort (BE).

An extra feature in IEEE 802.16, not available in DOCSIS, is that a SS is allowed to request transmission opportunities either as Grants per Connection (GPC), which is exactly the way DOCSIS works, or as Grants per Subscriber Station (GPSS), in which a SS requests transmission opportunities as a bundle for all the service flows it is maintaining. The SS then holds the responsibility for reassigning the received transmission opportunities between the different service flows. This allows hierarchical and distributed scheduling to be used and is not supported by DOCSIS.

C. QoS Service Flows in DOCSIS and IEEE 802.16

Both DOCSIS and IEEE 802.16 define different types of service flows that should be treated differently by the MAC protocol scheduling process. Those service flow types are identical for both DOCSIS and IEEE 802.16 and are explained below.

Unsolicited Grant Service (UGS) Flows: UGS is designed to support real-time service flows that generate fixed size data packets on a periodic basis, such as Voice over IP. The service offers fixed size *unsolicited* data grants (transmission opportunities) on a periodic basis. This eliminates the overhead and latency of requiring the CM to send requests for transmission opportunities. In UGS, the CM is prohibited from using any contention requests and the CMTS does not provide any *unicast* request opportunities[1] for the CM. Piggyback requests are also prohibited in UGS.

The key service parameters for UGS service flows are: *Unsolicited Grant Size*, *Grants Per Interval*, *Nominal Grant Interval* and *Tolerated Grant Jitter*. The ideal schedule for enforcing such parameters is defined by a *Reference Time* t_0, with the desired transmission times being $t_i = t_0 + i * interval$, where $interval$ is the Nominal Grant Interval. The actual grant times t'_i must be in the range $t_i \leq t'_i \leq t_i + jitter$, where $jitter$ is the Tolerated Grant Jitter. When multiple grants per interval are requested, all grants must be within this jitter interval.

Real-Time Polling Service (rtPS) Flows: rtPS is designed to support real-time service flows that generate variable size data packets on a periodic basis, such as MPEG video. The service offers periodic unicast request opportunities, which meet the flow's real-time needs and allow the CM to specify

[1] A unicast request opportunity is an interval of the upstream channel in which only one particular CM is allowed to send a bandwidth request to the CMTS. This is different from the contention interval in which many CMs contend to transmit their bandwidth requests.

the size of the desired grant. The CM is prohibited from using any contention or piggyback requests. The key service parameters here are: *Nominal Polling Interval*, *Tolerated Poll Jitter* and *Minimum Reserved Traffic Rate*. The ideal schedule for enforcing such parameters is very similar to that for UGS service flows.

Non-Real-Time Polling Service (nrtPS) Flows: The nrtPS is designed to support non-real-time service flows that require variable size data grants on a regular basis, such as high bandwidth FTP. The service offers unicast polls on a periodic basis, but using more spaced intervals than rtPS. This ensures that the flow receives request opportunities even during network congestion. In addition, the CM is allowed to use contention and piggyback request opportunities. The key service parameters here are: *Nominal Polling Interval*, *Minimum Reserved Traffic Rate* and *Traffic Priority*.

Best Effort (BE) Service Flows: In BE service the CM is allowed to use contention and piggyback request opportunities, but neither periodic polls nor periodic data grants will be sent by the CMTS unless they are needed to satisfy the minimum reserved bandwidth for that service. The key service parameters for BE service flows are: *Minimum Reserved Traffic Rate* and *Traffic Priority*.

For nrtPS and BE service flows, the standard specifies that the CMTS should use the Traffic Priority parameter when determining precedence in request service and grant generation, and the CMTS must preferentially provide contention request opportunities based on priority.

Unsolicited Grant Service with Activity Detection (UGS/AD) Service Flows: UGS/AD is a service flow type that is supported by DOCSIS only. It is designed to support UGS flows that may become inactive for substantial portions of time (i.e., tens of milliseconds or more), such as Voice over IP with silence suppression. The service provides unsolicited grants when the flow is active and unicast polls when the flow is inactive. This combines the low overhead and low latency of UGS with the efficiency of rtPS. Though USG/AD combines UGS and rtPS, only one scheduling service should be active at a time.

III. THE SCHEDULING ARCHITECTURE

A few proposals have already been devised to support QoS in HFC networks [7 – 10]. However, most of those proposals do not specifically address the QoS requirements of DOCSIS, which is now the *de facto* standard for HFC systems. For example, in [7] the authors propose a multi-tiered priority-based HFC scheduler, which supports contention-based traffic. The proposed scheduler, however, has no provision for delay-sensitive traffic such as UGS and rtPS service flows. The scheduling architecture proposed in [9], on the other hand, deals with delay-sensitive traffic but cannot be used for DOCSIS because it is based on a hierarchical architecture, which is not supported by the standard. To the best of our knowledge, only the work in [10] addresses the DOCSIS 1.1 standard *per se*, but it also falls short of supporting all the service flow types defined in DOCSIS (it deals only with UGS and BE services). Its treatment of UGS service is also problematic since it does not provide any guarantees in terms of Tolerated Jitter for such UGS service flows.

Hence, the scheduling architecture we present here is the first that truly addresses the QoS needs of DOCSIS and is also the first one to be proposed for the new IEEE 802.16 standard in the context of BWA systems. Although our scheduler supports both DOCSIS and IEEE 802.16, our discussion will be directed more toward the DOCSIS standard to avoid duplicity in technical terms and because of the wide availability of the DOCSIS standard at the time of writing.

Our suggested upstream scheduler architecture is shown in Figure 1. In such an architecture, requests for transmission from the different CMs are received by the CMTS through contention, unicast request opportunities and piggybacking. Those requests are first translated into suitable upstream transmission opportunities (data grants). Such data grants are then scheduled on a frame-by-frame basis by building a corresponding allocation MAP message that describes the usage of each frame interval. The hardware block responsible for creating the MAP message is represented in Figure 1 by a *server* that continuously schedules different data grants (and unicast request opportunities) on the upstream channel. In such a representation, each data grant (or unicast request opportunity) is treated as a *packet* that needs to find its way through the server (scheduler). When such a packet finishes service (i.e., when a data grant gets scheduled), a corresponding entry is logged in the MAP message for the next frame period. The actual transmission of the corresponding data packet, on the other hand, does not take place until the next frame starts.

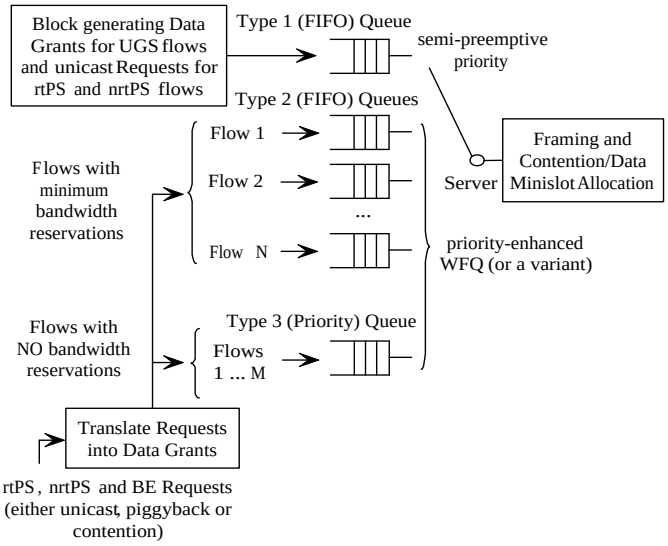

Fig. 1. Architecture of the proposed upstream scheduler for both DOCSIS and IEEE 802.16.

In summary, the server in Figure 1 can be used to represent, in a way, the upstream channel, keeping in mind the difference

in time between the instants of data grant scheduling and the instants in which the corresponding data packets are transmitted. Due to the fact that the actual transmission of data packets happens much later after the scheduling of the corresponding data grants, it is important to point out that for the scheduler in Figure 1 to operate properly, it needs to maintain a time axis that is a slightly different version of the actual system time axis. This would allow us to use the simple model in Figure 1 to represent the CMTS scheduler with the server being equivalent to the upstream channel. We will refer to the new time axis used in the scheduler as the *scheduler time*.

To allow for multiple QoS requirements, the scheduler keeps the data grants to be scheduled in three types of queues, which we will refer to as Type 1, Type 2 and Type 3 queues. More discussion on the properties of these queues and how they are related to the different service flow types is provided below.

The scheduler architecture shown in Figure 1 is easy to implement in hardware thus gaining a performance advantage over software-based alternatives. The architecture also lends itself to easier and straightforward performance analysis via classical queuing theory techniques.

A. Unsolicited Grant Service Flows

UGS packets cannot tolerate excessive delays in their transmission. Hence, the processing of UGS flows should be decoupled as much as possible from all other flows in the scheduler. To achieve this goal a separate hardware block in our scheduler keeps track of all admitted UGS service flows by maintaining a table similar to the one shown in Figure 2. This table is updated whenever the Connection Admission Control (CAC) algorithm admits or releases a new UGS flow. The separate hardware block then uses the information in this table to periodically generate data grants that feed the Type 1 queue in the scheduler.

Fig. 2. The table maintained by UGS dedicated hardware block.

Active UGS Service Flows						
Flow SID	Reference Time, t_0	Nominal Grant Interval	Tolerated Grant Jitter	Grant Size	Grants Per Interval	Other Data
1	X	X	X	X	X	X
2	X	X	X	X	X	X
3	X	X	X	X	X	X

One data grant (or more) is generated per nominal interval for each active UGS service flow. The generation time of each data grant is given by $t_i = t_0 + i * interval$, where $interval$ is the Nominal Grant Interval for that service flow. Each generated data grant is also marked with a delivery deadline equal to $t_i + jitter$, where $jitter$ is the Tolerated Grant Jitter for such a flow. The scheduling algorithm makes sure that data grants fed to the Type 1 queue are served (scheduled) before their corresponding deadlines by providing priority to such data grants. It is important to mention that all the above times are based on the scheduler time discussed earlier and not the actual system time.

The server provides a strict *semi-preemptive* priority to data grants in the Type 1 queue, whereby a grant undergoing service is *sometimes* allowed to complete service without interruption even if a grant of higher priority (a Type 1 grant) arrives in the meantime. This happens *only when* the newly arriving Type 1 grant can still be delivered within its deadline without the need to preempt the grant undergoing service. However, service of a grant must be interrupted (preempted) when a Type 1 grant arrives with a dangerously early deadline. In such a case, the newly arriving Type 1 grant is served first and the remainder of the preempted grant is served afterwards. In DOCSIS, this results in the lower priority data grant being fragmented. Of course, under this priority scheme, when the server becomes free while the Type 1 queue is nonempty, Type 1 grants are always the ones that enter service first. A question here is whether preemption (when needed) should be done just before the Type 1 grant deadline or at an earlier time given that the server knows it needs to perform preemption. Since preemption means fragmentation of a particular data grant, we suggest to preempt at a point convenient for fragmentation (e.g., at a fragment size equal to a power of 2).

For such a scheduling algorithm to work, fragmentation should be enabled for all non-UGS service flows in the network. Otherwise, if fragmentation must be avoided, a simpler architecture can be used in which data grants of all non-UGS service flows are limited by management functions[2] to a certain size that is always smaller than the minimum Tolerated Jitter of all UGS service flows. In this case, no UGS data grant will ever miss its deadline due to a grant being served under a simple strict *non-preemptive* queuing discipline. In fact, in such a design scenario, we can stop attaching deadlines to Type 1 data grants. The CM will be the one responsible for limiting the packet sizes corresponding to non-UGS flows to fit the new data grants with limited sizes.

B. Real-Time Polling Service Flows

There are two portions of rtPS traffic that need to be handled by the scheduler. First, there are the periodic upstream unicast request opportunities to be provided for each rtPS service flow, and second, there are the actual data grants to be allocated to such a flow.

Our scheduler treats rtPS upstream unicast request opportunities in exactly the same way as UGS data grants: A dedicated hardware block generates periodic unicast requests based on information stored in an internal table about the rtPS flows, and feeds those requests to the Type 1 queue in the scheduler. The table structure is the same as that used for UGS traffic (see Figure 2), but with replacing entries corresponding to data grants by entries corresponding to unicast requests (e.g.,

[2]DOCSIS imposes a global limit of 255 minislots on all types of data grants. This is due to implementation considerations. DOCSIS also allows the CMTS to impose a more stringent limit on data grant sizes of rtPS, nrtPS and BE service flows. This is done using the Maximum Traffic Burst management parameter (which has a default value of 1522 bytes).

replacing Nominal Grant Interval by Nominal Polling Interval and Tolerated Grant Jitter by Tolerated Polling Jitter).

A fundamental difference between UGS traffic and rtPS traffic is that UGS reserves a fixed portion of the upstream bandwidth that can only be used by that UGS service flow. In rtPS, however, if the service flow is inactive for a short period of time, the excess reserved capacity can be reused by other rtPS (or nrtPS and BE) flows. Hence, when the scheduler is generating data grants, it should treat rtPS traffic in a different way than UGS traffic. Also, each rtPS service flow may or may not make a minimum bandwidth reservation request at connection setup. The scheduler should also treat various rtPS flows differently based on the amount of bandwidth reservation they make.

Based on the above observations, after a rtPS request for transmission is received on the upstream channel, a corresponding data grant is generated and is fed to either a Type 2 or a Type 3 queue based on whether the corresponding service flow has made a minimum bandwidth reservation or not. The data grant is fed to a Type 3 queue if its corresponding service flow has no bandwidth reservation or is fed to a Type 2 queue if its flow has made such a bandwidth reservation (see Figure 1). The Type 3 queue in the scheduler is shared by all service flows with no explicit bandwidth reservations, while the scheduler provides a dedicated Type 2 queue for each service flow that has already requested some bandwidth guarantees from the CMTS.

We suggest using a Weighted Fair Queuing (WFQ) [11] discipline or a simpler variant of it such as Self-Clocked Fair Queuing (SCFQ) [12] or Start-Time Fair Queuing (SFQ) [13] to handle rtPS flows fed to Type 2 and Type 3 queues. A WFQ rate (or weight) is assigned to each Type 2 queue based on the minimum bandwidth reserved for the corresponding service flow. The WFQ rate for Type 3 queue is calculated by subtracting all the reserved rates for Type 2 queues from the aggregate output link capacity (of course, after subtracting the bandwidth reserved for UGS traffic and contention minislots).

It is fair to assume that the number of flows set up with minimum bandwidth reservations will be much smaller than those with no reservations. This is why aggregating all flows with no bandwidth reservations in one Type 3 queue will reduce the complexity of the underlying WFQ algorithm considerably. The choice of per service flow scheduling for Type 2 traffic, on the other hand, is adopted to provide hard bandwidth guarantees for the corresponding service flows that wish to have such guarantees.

We envision that pricing will depend mainly on the amount of minimum bandwidth reserved for a certain service flow. We also envision that for nrtPS and BE service flows, the Traffic Priority parameter will be a second-level pricing criterion. In other words, priority levels can provide finer-grained pricing to be combined with the coarser-grained pricing for the amount of minimum bandwidth reservation a user makes.

As explained earlier, nrtPS and BE service flows can be assigned different priority levels in the range of $0 - 7$, with higher values indicating higher priority. Although the DOCSIS standard does not assign any priority levels to rtPS service flows, we believe that since users are expected to pay more for rtPS than nrtPS and BE services, rtPS should have an implicit priority level of 8.

C. Non-Real-Time Polling Service Flows

There are two differences between nrtPS and rtPS services. First, nrtPS does not depend solely on unicast requests allocated to it by the CMTS but also utilizes contention and piggybacking to send requests to the scheduler. Second, nrtPS flows can be assigned different priority levels while rtPS has only one implicit priority level. In all other aspects, nrtPS and rtPS service flows are identical.

Hence, we handle the periodic upstream unicast requests for nrtPS in exactly the same manner as we handled rtPS (i.e., using a dedicated hardware block feeding requests to the Type 1 queue). In addition, after the nrtPS requests are received at the scheduler, the generated data grants are also fed to a Type 2 or Type 3 queue based on whether they have a minimum bandwidth reservation or not.

Since the standard requires that higher priority service flows be given lower delay and higher buffering preference, given that they are identical in all other QoS parameters besides priority, we propose the following modification of WFQ to produce a *priority-enhanced* WFQ. If two data grants (from two different queues, whether Type 2 or Type 3 queues) have identical[3] WFQ *virtual finish times* [11], then the first grant to be served is not selected randomly but is chosen based on its priority level, with higher priority grants being served first. This makes sure that higher-priority grants always incur less delay.

In addition, since all service flows fed to the Type 3 queue in the scheduler have zero bandwidth reservation, priority can be further invoked by adopting a strict *non-preemptive* priority discipline in serving data grants from Type 3 queue before being handed to the WFQ global server (see Figure 1). Thus, Type 3 grants pass through a non-preemptive priority server first, then pass through a WFQ server in which priority may again be invoked against grants from Type 2 queues.

D. Best Effort Service Flows

BE traffic is treated exactly in the same way as nrtPS traffic except for the fact that no periodic unicast requests are scheduled for any BE service flows.

In the next section we will discuss how nrtPS and BE flows can use contention minislots to send their requests to the CMTS. Our only challenge at this point is that nrtPS and BE flows with minimum reserved bandwidth may not be able to send enough requests to the CMTS to occupy such allocated

[3]The probability of two packets having identical virtual finish times in our scheduler is higher than that in a general variable-length data packet infrastructure. This is because the size of any data grant in DOCSIS, although variable, is always a multiple of the DOCSIS minislot size.

bandwidth because of possible collisions in the contention region (especially at high loads). To avoid this problem, we allocate extra upstream unicast request opportunities at the start of each frame period to all nrtPS and BE flows *with* minimum bandwidth reservations to allow them to at least request such a minimum bandwidth.

The reason for placing such extra unicast requests at the start of each frame period, even before the contention minislots, is an attempt to relieve the contention area by forcing some CMs to use the unicast requests and thus avoid the need for further contention. This should bring collisions to a minimum.

E. Unsolicited Grant Service with Activity Detection

To handle a UGS/AD service flow, a certain portion of the upstream channel bandwidth should be reserved for that flow. This reservation is made fixed when the service flow is active by creating a temporary entry in the UGS table, and treating the flow as if it was a UGS flow. When the flow becomes inactive, the entry in the table is temporarily blanked and instead the service flow is considered a rtPS one with its Minimum Reserved Traffic Rate parameter set to the original UGS traffic rate. This allows any excess bandwidth not used by the service flow to be utilized by other users, but still guarantees the minimum required bandwidth by the service.

IV. CONTENTION MINISLOT ALLOCATION

In DOCSIS and IEEE 802.16, nrtPS and BE service flows use contention to send their requests to the CMTS. We need to allocate an appropriate number of contention request minislots in each frame period to reduce the number of possible collisions and to shorten the contention resolution process. If done properly, this will improve the performance of the MAC protocol under varying traffic load conditions.

A. Frame Structure

In our scheduler we use a variable length upstream frame structure in order to achieve maximum scheduling flexibility and minimum transmission latency. We opt for the frame structure shown in Figure 3, where contention minislots are all clustered adjacently at the beginning of each upstream frame interval. This configuration allows easier implementation at both the CMTS and the CM because both devices have to switch to the contention mode only once at the start of each frame period. Also in such a configuration, the feedback MAP message from the CMTS corresponding to a cluster of minislots can be received prior to the beginning of the next frame period. This reduces latency in receiving request acknowledgments and in contention resolution, which is of great concern in the contention process.

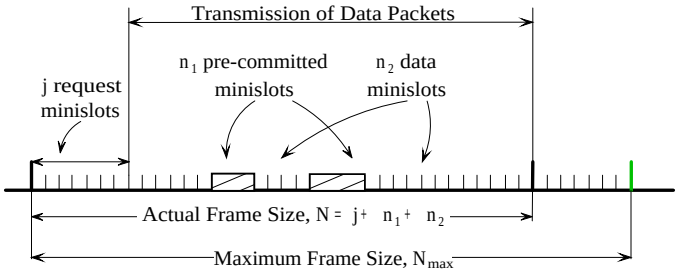

Fig. 3. Upstream frame structure.

B. Contention Minislot Allocation

Different contention minislot allocation schemes have been suggested in the literature [7, 14, 15]. The scheme we propose here is an extension of the mechanisms suggested in [14] and [7] with some modifications to adapt to the minislot structure of DOCSIS, to account for piggybacking, and to allow for variable frame lengths.

Figure 3 above illustrates a typical DOCSIS upstream frame. The frame length (in units of minislots) is variable with a certain maximum limit, $N_{\max}$. The actual frame length depends on the number of data grants pending transmission on the upstream channel. Figure 3 shows the case where there are only n_2 worth of nrtPS and BE data grants pending transmission. UGS grants, rtPS polls and other periodic traffic that do not require contention occupy n_1 minislots of the frame interval. Such n_1 minislots are considered as pre-committed minislots in our algorithm and do not count as part of the frame. A variable number, j, of contention minislots are used in each upstream frame as request minislots. We will develop an algorithm to dynamically calculate the value of j for each frame based on different MAC loading conditions.

The algorithm can be explained as follows. Just enough contention minislots need to be created so that the average throughput (per frame) of the contention request minislots closely matches the number of new data packets that can be transmitted in a maximum frame period. We remember that using a simple contention resolution algorithm such as the random binary exponential backoff, mandated by DOCSIS, gives a throughput efficiency of 33% for contention minislots [14]. Hence, the number of contention minislots in a frame should be adjusted to satisfy the following requirement: the number of requests that can be transmitted within j contention minislots should be equal to *three* times the number of nrtPS and BE data packets that can be transmitted in a frame with maximum length.

To transform the above statement to units of minislots, we note that a data packet may need, on average, l_d minislots to be transmitted. Also a contention request may need, on average, l_c contention minislots to be transmitted[4]. Hence we can say that

[4]A typical value of l_c is 1 minislot. It is also reasonable to assume that $l_d >> l_c$.

the ratio $j/l_c : (N_{\max} - j - n_1)/l_d$ should be at most 3:1. The parameters l_d and l_c can either be fixed *a priori* or measured on periodic basis during the scheduler operation.

Our algorithm should also be able to dynamically reduce the number of required contention minislots per frame as the load on the system increases. This is possible because as the traffic from certain flows become heavier; those flows can utilize piggybacking more often, and hence reduce the load on contention. Taking this into account allows us to reduce the ratio of 3:1 we originally needed for contention minislots to a smaller ratio. To quantify this effect we note that if a multi-packet batch arrives at a certain CM buffer, only the first packet in that buffer generates a contention request. However, future requests for the rest of the multi-packet batch can be sent using piggybacking. Now, assume that the average length (in units of packets) of multi-packet batches arriving at different CMs is k, then we can say that k data packets are utilized for each request that makes it through the minislot contention process. Again, the value of k can be measured during normal operation. In summary, we need to have,

$$j = \frac{3 * (N_{\max} - j - n_1) * l_c}{k * l_d}$$

C. Algorithm

We compute the number of required contention minislots in each frame period based on an estimate of the maximum number of data packets that can be transmitted in such a frame. This estimate is mainly derived from the traffic observed in the previous frame. The following algorithm dynamically calculates the number of contention minislots j_i for each frame i:

Frame 0: Set $j_0 = j_{\min}$ (Initialization)
Frame i: Let, $j_i = \max\left\{\frac{3*(N_{\max}-j_{i-1}-n_{1,i-1})}{(k*l_d/l_c)},\ j_{\min}\right\}$
If $Q \geq \alpha * (N_{\max} - j_{i-1} - n_{1,i-1})$, Set $j_i = j_{\min}$

Q in the above expression is the total number of data minislots requested but not yet allocated by the CMTS (i.e., the aggregate length of bending data grants at the scheduler), and α is a design parameter set to 2.5 or 3.5. The condition including Q means that when there are so many outstanding minislot requests that cannot be handled within the next two frames or so, the CMTS should prevent CMs from sending further contention requests. This will only happen in overload (congestion) situations, and will prevent the buffers in the CMTS from overflowing unnecessarily. It also makes sense to deny piggybacked requests in those overload situations.

D. Dealing with Priorities

Now that we know the number of contention minislots to allocate per frame, we need to divide this capacity of contention minislots between the different service flow priorities. The DOCSIS standard requires a preferential treatment of higher priority traffic allowing it to have a better chance of sending requests through contention. This can be achieved by introducing a set of multiplication factors, a_d, $0 < a_d < 1$ for d = 0, ..., 7 that allow dividing the available amount of contention minislots between the different priorities based on a preference criterion decided by the service provider and, of course, related to pricing. The a_d factors should satisfy the requirement $\sum_{d=0}^{7} a_d = 1$. Hence, after calculating the total number of contention minislots per frame, j, we calculate the number of minislots, j^d, to allocate to each priority d as follows: $j^d = \lfloor j * a_d \rfloor$.

One scenario that might happen in the above minislot allocation scheme is that higher priority flows may get more minislots allocated than their actual need. This will happen if the aggregate load of high priority flows is smaller than anticipated. To overcome this situation and to improve the operation of our allocation scheme, we incorporate another factor related to the observed traffic load of each priority in the system. More specifically, we start by computing a moving average of the observed number of contention minislots used per frame for each priority level. We denote these values by r_d for d = 0, ..., 7. These values will represent estimates of how many minislots each priority should be expected to use in the next frame period. Notice that these estimates are updated periodically.

The idea is that we want to utilize any excess amount of contention minislots for use by other priorities (preferably higher priorities). To do that, after calculating the number of minislots allocated to one priority, j^d, we compare this value to the number of minislots we should expect in the next frame for that priority, r_d. If the value of j^d is much larger than r_d, we borrow a few of the j^d minislots, say δ, and redistribute them among the contention minislots allocated to higher priorities. Such an algorithm is recursive and is illustrated below, where β and δ are design parameters.

Start: $d = 0$
While $d < 7$ do{ (we stop at d = 7 – 1)
If $j^d > \beta\, r_d$, Set $j^d = j^d - \delta$
And set $j^e = j^e + \delta * a_e \Big/ \sum_{f=d+1}^{7} a_f$ for all $e = d+1, ..., 7$
}End While

Another option for distributing the contention minislots between the different priorities is to use a nested priority scheme, in which higher priority flows are allowed to use the whole contention area, while low priority flows are only allowed to use part of such a contention region. The reason we avoid this scheme is that it does not allow complete separation of the different priorities and hence cannot prevent misbehaving high priority traffic from causing undue collisions in the whole contention area, including the contention interval for low priority traffic. In our scheme, on the other hand, the region given to low priority traffic is pre-determined by the service provider using

the a_d parameter (which is mainly determined by pricing) and high priority traffic cannot receive more than its allocated share of the contention interval unless the low priority traffic load is smaller than anticipated.

V. BUFFER MANAGEMENT

This section deals with the problem of allocating buffer space to the different queue Types (Type 1, 2 and 3) of the scheduler to achieve minimum losses of data grants during scheduling. It is important to note at this point that losing a data grant at the scheduler due to buffer overflow does not necessarily mean the loss of the corresponding data packet itself. This is because after the CMTS receives a request for a data grant from the CM, it sends a signal back to the CM in the bandwidth allocation MAP indicating a pending data grant. When this data grant is lost due to buffer overflow at the scheduler, the CM will eventually timeout and will retransmit another request. This will certainly cause the CM buffer to grow monotonically during the timeout period but will not necessarily result in the loss of information. Hence, mapping data grant losses in the scheduler to actual data packet losses in the CM is not an easy task to achieve and is heavily dependent on the buffer space at the different CMs along with the utilized timeout mechanisms.

Now, returning to the scheduler architecture shown earlier in Figure 1, we see that data grants are treated as generic packets that are placed in different queue Types before being served by the scheduler. An important distinction we need to draw here is between the *virtual* and the *actual* meaning of each queue buffer space in such a scheduler. Virtually, the system works as if it is scheduling packets with variable lengths passing through a continuous-time server. Actually, however, when a data grant is generated, the only information that needs to be stored about such a data grant is its length (in units of minislots) and optionally its deadline (in the case of Type 1 data grants). Such information is the only information needed to construct a bandwidth allocation MAP at the end of each processing period to describe the usage of the upstream channel. The size of this data grant *information* is of fixed length whether it corresponds to a 1 KB data grant or a 4 KB data grant. This means that the buffer space allocated to each queue type needs to be measured in units of fixed-size data grant information units rather than units of bytes.

Now, to distribute the total amount of available buffer space locations between the different queue types, we start with the single Type 1 queue. Calculating the maximum number of data grants that can accumulate in such a queue at any moment of time can be done easily because of the periodic nature of UGS traffic feeding this queue. We always allocate such a maximum number of buffer space locations to the Type 1 queue. Remember that we cannot afford to loose any data grants from such a queue since there is no other mechanism for UGS service flows to request new data grants if the scheduler loses them. For Type 2 queues, we can allocate buffer space based on a pricing criterion. Since the price paid for setting up a service flow increases with the amount of minimum bandwidth reserved for that flow and the priority level assigned to it, a service flow that reserves more bandwidth and has a higher priority level should receive a larger buffer space. The remaining buffer space after deciding on Type 1 and Type 2 queue sizes is then used for the Type 3 queue.

For buffer management of Type 2 and Type 3 queues, we suggest using the Random Early Detection (RED) and multi-priority RED schemes, respectively. RED [16] is a buffer management scheme that avoids congestion by randomly (probabilistically) dropping packets when the buffer occupancy reaches a certain limit. Multi-priority RED is just an extension of RED to support multi-priority flows sharing the same buffer, as is the case for the Type 3 queue.

The reason we suggest using RED for buffer management in our scheduler is that RED was designed to work hand-in-hand with the TCP congestion control algorithm, and hence is best suited for Internet traffic. Since DOCSIS and IEEE 802.16 are mainly Internet oriented, RED would be the best candidate for buffer management in our scheduler.

VI. SUMMARY

In this paper we introduced a new scheduling architecture for both DOCSIS and IEEE 802.16. The new architecture supports diverse QoS guarantees for various service flow types suggested by the above standards. More specifically, it supports tight delay guarantees for UGS traffic and minimum bandwidth reservations for rtPS, nrtPS and BE flows. It is worth mentioning that vendor-specific QoS parameters can also be used in DOCSIS and IEEE 802.16. This means that users can also request QoS delay bounds for their rtPS and nrtPS service flows. Because we are using a fair queueing algorithm in our scheduler, providing such guarantees is feasible and can be implemented easily given that the service flows feeding the scheduler are properly policed (either at the CMTS or at the CM level).

We also introduced a dynamic minislot allocation scheme that should improve the performance of our scheduling algorithm under varying load conditions. It speeds the contention phase by providing extra bandwidth for contending packets. The loss of throughput due to this operation should not be a severe one. This is mainly due to the fact that request packets are much smaller than actual data packets, and because our algorithm allocates fewer contention minislots as the load on the system increases.

We are currently in the process of performing analytical and simulation studies to demonstrate the efficiency and performance of our scheduler. The results of such analysis will be provided in future publications.

REFERENCES

[1] M. Gagnaire, “An Overview of Broad-band Access Technologies,” Proceedings of the IEEE, Vol. 85, No. 12, pp. 1958–1972, December 1997.

[2] Y. Lin, W. Yin, C. Huang, "An Investigation into HFC MAC Protocols: Mechanisms, Implementation, and Research Issues," IEEE Communications Surveys, Third Quarter 2000, http://www.comsoc.org/pubs/surveys/.

[3] C. Shirali, M. Shahar, K. Doucet, "High-bandwidth Interface for Multimedia Communications over Fixed Wireless Systems," IEEE Multimedia, Vol. 8, No. 3, pp. 87–95, 2001.

[4] D. Fellows and D. Jones, "DOCSIS^{TM} Cable Modem Technology," IEEE Communications Magazine, Vol. 39, No. 3, pp. 202 –209, March 2001.

[5] Data-Over-Cable Service Interface Specifications, Radio Frequency Interface Specification, SP-RFIv1.1-I07-010829, http://www.cablemodem.com/.

[6] IEEE 802.16 Standard, IEEE Draft Standard for Local and Metropolitan Area Networks - Part 16: Air Interface for Fixed Broadband Wireless Access Systems, IEEE Draft P802.16/D5, http://grouper.ieee.org/groups/802/16/published.html.

[7] N. Golmie, F. Mouveaux, D. Su, "Differentiated Services over Cable Networks," GLOBECOM'99, December 1999.

[8] R. Rabbat and K. Y. Siu, "QoS Support for Intergrated Services over CATV," IEEE Communications, Vol. 37, No. 1, pp. 64–68, January 1999.

[9] Xiaojun Xiao, W.K.G. Seah, Yong Huat Chew, Chi Chung Ko, "Upstream Resource Reservation and Scheduling Strategies for Hybrid Fiber/Coaxial Networks," APCC/OECC '99.

[10] M. Droubi, N. Idirene, C. Chen, "Dynamic Bandwidth Allocation for the HFC DOCSIS MAC Protocol," International Conference on Computer Communications and Networks, pp. 54 –60, 2000.

[11] A. K. Parekh, R. G. Gallager, "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single-Node Case," IEEE INFOCOM'92, Vol. 2, pp. 915-924, May 1992.

[12] S. Golestani, "A Self-Clocked Fair Queueing Scheme for Broadband Applications," INFOCOM'94, pp. 636–646, April 1994.

[13] P. Goyal, H. M. Vin, H. Chen, "Start-Time Fair Queueing: A Scheduling Algorithm for Integrated Services Packet Switching Networks," SIGCOMM'96, pp.157–169, September 1996.

[14] K. Sriram, "Performance of MAC Protocols for Broadband HFC and Wireless Access Networks," Advances in Performance Analysis, Vo1. 1, No. 1, pp. 1–37, 1998.

[15] W. M. Yin and Y. D. Lin, "Statistically Optimized Minislot Allocation for Initial and Collision Resolution in Hybrid Fiber Coaxial Networks," IEEE JSAC, Vol. 18, No. 4, September 2000.

[16] S. Floyd and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance," IEEE/ACM Transactions on Networking, Vol. 1, No. 4, August 1993.