Scheduling of Voice Packets in a Low-Bandwidth Shared Medium Access Network

Reuven Cohen  Liran Katzir
Department of Computer Science
Technion
Israel

Abstract
This paper addresses the schedulability of VoIP packets in a low-bandwidth shared medium broadband access network. We show that EDF (Earliest Deadline First) without preemption is an optimal scheduling algorithm for the case where the size of the packets and the tolerated jitter are equal for all the packets, in the sense that no other algorithm, with or without preemption, may schedule more packets on time. We then employ this algorithm in order to concentrate upon two issues related to the effect of VoIP codec functionality on the schedulability of VoIP packets: packetization delay and silence suppression. We show that in both cases there are two counter-forces affecting the schedulability of the VoIP packets, and we use simulations in order to get in-depth understanding of these counter-forces under different conditions.

1 Introduction
Future wireless networks will employ packet switching not only for data applications but for voice applications as well. This is a shift from current wireless architectures that employ the hybrid model, where the same network uses circuit switching for voice applications and packet switching for data applications. One of the most difficult issues in the implementation of a packetized voice system is meeting the tough delay and loss requirements of voice packets. Packetized voice, also known as Voice-over-IP (VoIP), is very sensitive to dropped packets and delayed packets. Packet loss causes voice clipping and skips, whereas packet delay can cause voice quality degradation due to echo and talker overlap. Echo becomes a significant problem when delay is greater than 50 ms, but it is usually removed by the VoIP codec. Talker overlap becomes significant if one-way delay is greater than 150 ms, and there is nothing a codec can do to reduce its effect. Hence, the end-to-end delay should not exceed this threshold. The various components of VoIP delay are as follows:

1. Accumulation Delay: This delay is caused by the need to collect a block of voice samples to be processed by the voice coder. It varies from a single sample time (0.125 ms) to many milliseconds.

2. Vocoder look-ahead (“algorithmic”) delay: Upon creating a voice block, the compression algorithm must have some knowledge of the voice samples of succeeding block. The value of this delay ranges between 0 ms (for a G.726 vocoder) and 7.5 ms (for a G.723.1 vocoder).
3. Packetization delay: Often, multiple voice-coder blocks are collected in a single packet to reduce overhead. For example, three 10-ms blocks of G.729 may be collected and packed into a single packet. This delay is therefore a function of the voice block size required by the vocoder and the number of blocks placed in a single packet.

4. Codec compression and decompression delay: the time it takes to the Digital Signal Processor (DSP) to translate a sequence of PCM samples into a voice block and vice versa. This delay varies with the codec used and processor’s speed, and it is relatively small compared to the other codec-dependent delay.

5. Network delay: propagation delay on the network links, plus queuing delay at the network routers.

6. The MAC delay and jitter. This delay component exists only in networks that use a MAC protocol in order to coordinate the access to a shared channel. In such networks a node that has a ready VoIP packet should wait according to the MAC rules before transmitting this packet.

The codec at the receiving side employs a jitter buffer in order to smooth the jitter created in the routers and the MAC. Every 1 ms jitter is translated by this buffer to a 1 ms delay. Therefore, for the rest of the paper the MAC delay and jitter is referred to as the “MAC scheduling delay”.

In order to ensure that the end-to-end delay does not exceed the delay budget of 150ms, each of these components should be minimized as possible. This paper addresses two issues related to the codec functionality, that have a strong impact on the MAC scheduling delay:

- The effect of packetization delay.
- The effect of silence suppression.

In terms of packetization delay, as already indicated, the codec can collect one or more voice blocks in a single packet. Increasing the number of voice blocks in a packet decreases the overhead of the packet header. Such a decrease has a positive impact on the MAC scheduling delay because it is easier to schedule the transmission of a packet on time when the load on the channel is smaller. On the other hand, when collecting \( i \) voice blocks, the packetization delay is \( i \)-time larger than the packetization delay when collecting a single voice block. Hence, we have here two counter-forces that have to be examined. In terms of the silence suppression, we also have two counter-forces. On one hand, by employing silence suppression we reduce the load on the channel substantially. On the other hand, the first packet of a new talk-spurt often has a much smaller tolerated delay because of the time it takes to signal to the head-end the start of the new talk-spurt.

The results in this paper are applicable for any access network with a shared medium channel, like cable-modems (DOCSIS [5]) and fixed broadband wireless access (IEEE-802.16 [6]). However, these results are of importance only for “low-bandwidth” (5-30Mb/s) shared channels when the VoIP traffic is likely to consume a significant portion (\( \approx 50\% \)) of the channel bandwidth. This is likely to be the case in mobile access networks.
The rest of this paper is organized as follows. In Section 2 we discuss related work. In Section 3 we present our network model and a formal definition of the problem addressed in this paper. In Section 4 we show that if all the VoIP packets have (a) the same size and (b) the same tolerated jitter, then EDF (Earliest Deadline First) without preemption is an optimal scheduling algorithm in the sense that no other algorithm, with or without preemption, may schedule more packets on time. This new theoretical result, proven in Lemma 2 and Theorem 3 of this paper, is significant in homogeneous cellular access networks where conditions (a) and (b) are likely to be held. In Section 5 we discuss the effect of increasing the packetization time on the schedulability of VoIP packets. We show that under certain conditions, increasing the packetization time, and therefore reducing the packetization overhead, has a negative effect on the VoIP loss rate. In Section 6 we discuss the effect of silence suppression on the schedulability of VoIP packets. Once again, we show that under certain conditions silence suppression might have a negative effect on the packet loss rate, despite of the fact that it significantly reduces the load on the channel. Finally, Section 7 concludes this paper.

2 Related Work

A great deal of work has been done in the theoretical area of process (job) scheduling. The results that are relevant to this paper are presented and discussed in Section 4, where we describe an optimal scheduling algorithm for the model considered in this paper. We therefore limit the discussion in this section to relevant work in other areas related to this paper.

Since we seek to minimize the loss rate of VoIP packets, it is important to understand the correlation between the loss rate and the voice quality as measured using the MOS (Mean Opinion Score) test. This issue, along with discussion on loss repair methods, is addressed in [20] for various voice coding schemes. It is shown there that the effect of loss on the MOS is critical. For instance, with G.729 voice coding, and a 20ms packetization time, a loss rate of 5% decreases the MOS value from 4.1 to 3.4. Ref. [20] also studies the correlation between the packetization time and the MOS degradation. Somewhat surprisingly, it is shown that under burst loss increasing the packetization time improves the MOS, either with or without FEC (Forward Error Correction).

In this paper we concentrate on the issue of minimizing the loss due to scheduling conflicts, while ignoring the losses that occur due to transmission errors. This is not because we believe that these losses are negligible, but because we would like to understand the effect of packet size and the effect of silence suppression on packet schedulability under various assumptions. Several papers published in the last few years take into account also the effect of the channel condition on packet loss. In [19] it is claimed that the Earliest Deadline First (EDF) policy is not always optimal, in the sense of minimizing the number of packets that cannot meet their deadlines, even if the channel state is perfectly known and this policy is implemented only over channels in a “good” condition. In [16] the authors propose a new model for wireless fair scheduling based on the adaptation of fluid fair queuing to handle location-dependent error bursts. They
describe an ideal fair scheduling algorithm that assumes full knowledge of the channel conditions. In [13] a scheduling algorithm for real-time packets on the down-link channel of a DS-CDMA wireless network is presented. This algorithm tries to meet the hard deadlines while minimizing the resource usage by taking into account both the channel condition and the delay requirement of the traffic. Ref. [1] also addresses the problem of scheduling in the presence of transmission errors, but in the context of wireless LAN. It presents several algorithms and compares their performance from the perspective of both throughput and fairness.

A stochastic study of EDF is presented in [18]. The considered model is a single-server queue with customers with deadlines. Both continuous and discrete time queues are studied. Unlike in most of the models, it is assumed there that the service times are not known at the beginning of the service. It is shown that if unforced idle times are not allowed, EDF is optimal for the non-preemptive M/G/1 queue. When the service time is exactly one time unit, EDF is shown to be optimal for a G/D/1 queue.

In order to handle silence suppression efficiently, the MAC protocol should allow the stations to signal the beginning of a new talk-spurt. In this paper we consider a generic scheme that can be based upon contention, polling or any other approach. Several specific protocols for this purpose have been proposed in the past. Ref. [2], that presents a survey of MAC protocols for real-time traffic in wireless network, describes some of them. One of the classical protocols is PRMA (Packet Reservation Multiple Access) [3]. uses slotted ALOHA. In [4] a similar approach is proposed with only one exception: instead of sending the first packet of a new talk-spurt in the contention channel, this channel is used for sending a short signaling message. This reduces the collision penalty, on one hand, but increases the delay of the first packet in a talk-spurt on the other hand.

3 Preliminaries

We consider a wireless access network with a set of nodes connected to a shared channel. The channel is controlled by a centralized node, referred to as a base-station. The upstream channel carries information from the end nodes to the base-station. This information is then forwarded from the base-station to the rest-of-the-world. The downstream channel carries information in the reverse direction: through the base-station to the end nodes. The downstream channel is accessed by the base-station only, and no MAC protocol is therefore needed. In this paper we consider only the access to the upstream channel, which is characterized by many transmitting nodes and one receiving node (the base-station).

There are many proposals for the “next generation mobile networks”. However, TDMA (Time Division Multiple Access) is likely to be an important building block in every system. We therefore assume that in the upstream channel the time is divided into slots, providing for Time Division Multiple Access at regulated time ticks. A slot is the unit of granularity for upstream transmission opportunities. There is no implication that any data packet can actually be transmitted in a single slot. The base-station provides the time reference and controls the allowed usage for each slot. A slot can be granted for transmissions by particular nodes, or for contention by all nodes. For convenience we assume that a sequence of $F$ consecutive slots are grouped
into a logical entity, called frame. However, the results presented in this paper are valid even if framing is not used on the upstream channel. The allocation of upstream slots is controlled by special MAC messages, called MAPs, sent out by the base-stations on the downstream channel. The MAP messages describe how should each upstream slot be used, namely whether it is assigned to some particular node or it is open for contention. More specifically, we assume that just before the time of a new upstream frame is defined, the base-station sends on the downstream channel a MAP message that describes the status of each slot in this upstream frame.

In order to provide QoS on the upstream channel, the upstream data packets are classified into service flows [5, 6]. A service flow is a unidirectional flow of data packets that is provided a particular set of Quality of Service (QoS) parameters. The end node and the base-station provide this QoS by shaping, policing, and prioritizing traffic according to the QoS parameters defined for each service flow. For the sake of this paper we can assume that each end node assigns the VoIP packets of every voice call to a dedicated service flow.

There are several possible MAC mechanisms for allocating bandwidth on the upstream channel to each service flow. The most important are:

- **Unsolicited Grant Service (UGS):** With this mechanism, bandwidth is allocated by providing the service flow fixed size grants at periodic intervals on the upstream channel.

- **Polling Service:** With this mechanism the end host can send real-time periodic request opportunities for bandwidth. This service requires more request overhead than UGS, but it increases the efficiency by supporting grants for variable size packets. It is therefore more applicable for video traffic than for VoIP.

- **Contention Service (BE):** This mechanism is intended mainly for best-effort applications.

With UGS, the host is guaranteed to receive from the base-station fixed-size grants at periodic intervals, without the need to explicitly send requests. The tolerated grant jitter, the grant size and the grant periodicity are all negotiated. UGS reduces latency by eliminating the request-grant cycle for every packet. However, UGS is inefficient when silence suppression is used. In such a case a variant of UGS, referred to as UGS-AD (UGS with Activity Detection) is used. With UGS-AD, the base-station views an unused grant as a signal to voice inactivity. It therefore stops allocating unsolicited grants to the host until the host signals the start of a new talk-spurt – either using the contention service or using the polling service. In this paper we study both the case where UGS and the case where UGS-AD are used.

We assume that each service flow is associated with the following UGS parameters:

- The nominal grant interval is $\delta$.
- The grant size is $\sigma$.
- The tolerated grant jitter is $D_{\text{sched jitter}}$. 
Consider a VoIP service flow. Let $\text{Create}(P_j)$ be the time when the $j$'th packet is ready for transmission. Therefore, for every $j > 0$, $\text{Create}(P_j) = \text{Create}(P_1) + (j - 1) \cdot \delta$. Let $\text{Grant}(P_j)$ be the time when the $j$'th transmission opportunity is granted to SF. The jitter for the $j$'th packet of SF is defined as $\text{Grant}(P_j) - \text{Create}(P_j)$. The target tolerated grant jitter for SF is said to be met if for every $j$, $\text{Grant}(P_j) - \text{Create}(P_j) \leq D_{\text{sched}\text{-jitter}}$. Note that when we discuss the case where the codec employs silence suppression, the above definition holds for each independent talk-spurt.

Since we consider phone calls originated at the same access network, we assume throughout the paper that the VoIP codec at each host employs the same algorithm with the same parameters. This implies that the nominal grant interval, the grant size and the tolerated jitter are identical for all the service flows, with the values $\delta$, $\sigma$ and $D_{\text{sched}\text{-jitter}}$ respectively. Let the codec packetization delay be $D_{\text{packet}}$ and the access network transmission delay be $D_{\text{transmit}}$. Hence we have

$$D_{\text{access}\text{-budget}} = D_{\text{packet}} + D_{\text{sched}\text{-jitter}} + D_{\text{transmit}}.$$  

Let the delay budget allocated to the codec packetization plus MAC scheduling plus access network transmission time be $D_{\text{access}\text{-budget}}$. When a new VoIP call is set up, the base-station is notified, and it needs to issue periodic unsolicited grants to the new service flow while targeting the tolerated grant jitter. Recall that in the considered model, “just before” the time of a new upstream frame is defined, the base-station needs to determine the allocation of slots in this frame, and to notify the hosts of its decision. Since the base-station knows when each service-flow needs to send a packet during the next upstream frame, it can employ any off-line scheduling algorithm in order to determine which slot in this frame should be allocated to which service-flow.

In computer networks packet loss usually happens due to transmission errors or buffer overflow. However, in a system with hard delay and jitter bounds, as considered in this paper, packet loss also takes place when the scheduler is unable to meet the delay requirement of a packet. We assume that at every node each UGS service flow is assigned a private buffer. This buffer is very small since it needs to contain no more than 1-3 packets. If a grant is not provided to the service-flow on time, the packet in the head of the buffer is discarded. This is the only way a packet can be lost due to scheduling constraints, and it is attributed not to buffer overflow but to delay bound violation.

**Definition 1 (UGS Load)** Consider a network with $N$ UGS service flows. The total normalized bandwidth required by the UGS service flows is $N \sigma / \delta$.

**Definition 2 (UGS Scheduling Performance)** The performance of a UGS scheduling algorithm is defined as the fraction of packets that can be scheduled on time for a given tolerated grant jitter and a given normalized load of the UGS service flows.

Definitions 1 and 2 ignored the non-UGS service flows. Namely, the performance of the UGS scheduling algorithm is defined independently of the load imposed on the system by the non-UGS traffic. The
justification for this is that the UGS service flows have the strictest QoS requirements, and the base-station is supposed to address their needs before addressing the needs of the non-UGS service flows. Hence, the non-UGS service flows are not supposed to affect the UGS scheduling algorithm.

Recall that in the considered network architecture the upstream channel is slotted and there is no implication that any data packet can actually be transmitted in a single slot. When a packet is longer than a single slot, the base-station should allocate the transmitting node multiple slots for the same packet. In some network models fragmentation, sometimes referred to as “preemption”, is allowed. In these models, the slots used for the transmission of a packet can be non-consecutive. In general, packet fragmentation may increase the performance of the UGS and non-UGS scheduling. However, this does not come with no cost, because fragmentation has CPU, memory and bandwidth overhead. The scheduling algorithm presented in this paper for the VoIP packets is shown to be optimal despite of the fact that it does not require these packets to be fragmented.

4 An Optimal Scheduling Algorithm for the Considered Model

The scheduling problem we address here is as follows:

Problem 1 Find a schedule that minimizes the number of packets that cannot be scheduled while meeting their deadlines.

Just before the time of a new upstream frame is defined, frame(i) say, the scheduler is invoked in order to determine which UGS packets will be scheduled in this frame. At this time there exists a set of packets that were released in the past, but have not been scheduled yet. These packets are classified into two sub-sets: a sub-set of packets whose deadline has expired, and a sub-set of packets whose deadline has not yet expired. The former sub-set is ignored by the scheduler. The latest subset is considered as backlog(i). The scheduler tries to accommodate in frame(i) all the packets in backlog(i) as well as all the packets created during the time of frame(i), referred to as create(i). We denote active(i) = backlog(i) \cup create(i). The set of packets scheduled during frame(i) is denoted schedule(i). Note that a packet from backlog(i) can be scheduled at any time during frame(i), whereas a packet from create(i) can be scheduled only after its creation time. Denote by lost(i) the packets in active(i) that are not scheduled during frame(i) whose deadline is passed during this frame. Hence, backlog(i + 1) = active(i) − schedule(i) − lost(i). We now consider a sub-problem of Problem 1, which deals with the scheduling during a single upstream frame:

Problem 2 For a given set of packets schedule(i), find a schedule for frame(i) that minimizes the number of packets in lost(i).

Problem 2 is easier because it considers a single frame. We show latter how to solve Problem 1 using a solution for Problem 2.
Problem 2 was well studied in the context of real-time systems and theoretical computer science as the “uni-processor scheduling” problem\(^1\). With arbitrary release times, due dates and packet sizes, this problem is NP-hard [14]. However, many special cases and relaxations of the problem are polynomially solvable. Some of them are as follows:

- If all the packets are available at the beginning of the time frame, namely \(create(i) = \phi\), the problem is polynomially solvable in \(O(n \log(n))\), where \(n = |active(i)|\). See Ref. [17].

- If the size of every packet is one slot, the problem is polynomially solvable in \(O(n \log n)\) using the EDF (Early Deadline First) algorithm [8].

- If the release times and due dates of the packets are similarly ordered (see Definition 3 below), and preemption is allowed, the problem is polynomially solvable [9, 11]. Recall that in our system “preemption” is translated into “packet fragmentation”.

- When all the packets are of equal size \(K\), but \(K > 1\), the problem is polynomially solvable using dynamic programming if preemption is allowed [10]. A polynomial time dynamic algorithm also exists when preemption is not allowed [10]. However, in both cases the running time is very high \(O(n^7)\) and \(O(n^{10})\) respectively – so both algorithms are impractical.

The following definition is due to [9]:

**Definition 3 (Similarly Ordered)** For every packet \(P\) in \(active(i)\), let \(Release(i; P) = \max\{Create(P), t^i_s\}\), where \(t^i_s\) is the time when \(frame(i)\) starts, and let \(Due(P) = Create(P) + D_{s,hed,itter}\). The release times and due dates of two packets \(P_1, P_2 \in active(i)\) are said to be similarly ordered if \(Release(i, P_1) \leq Release(i, P_2) \Rightarrow Due(P_1) \leq Due(P_2)\).

In what follows we present the EDF-with-preemption and EDF-w/o-preemption algorithms. Throughout the paper we assume that EDF does not schedule packets that are already past their due dates. In some papers [18], such a policy is referred to as Shortest Time to Extinction (STE).

**Algorithm 1 (EDF-with-preemption)** Assign the next slot to the packet with the earliest possible due date from all the packets that are currently available for processing, i.e. packets whose release dates have been passed, whose due dates have not been passed and whose processing has not been completed. More formally, Let the number of slots required in order to complete the transmission of packet \(P\) be \(Complete(P)\). For every slot \(t\) of \(frame(i)\), assign this slot to the packet \(P\) that fulfills the following requirements: (a) \(Release(i, P) \leq t\); (b) \(Complete(P) > 0\); (c) there is no other packet \(P'\) for which (a)-(c) hold and \(Due(P') < Due(P)\).

---

\(^1\)Notations of problems from these family have the form \(\alpha[|\beta|]|\gamma\), where \(\alpha = 1\) indicates a single machine (a single channel in our case), \(\beta\) indicates the characteristics of the jobs (packets) to be scheduled, e.g. whether they are preemtable or non-preemtable, and \(\gamma\) indicates the optimality criterion (maximum lateness, total tardiness etc.). The specific problem which is equivalent to Problem 2 is denoted \(1|r_j|\Sigma_jU_j\), where \(r_j\) indicates that the release times are not equal, and \(\Sigma_jU_j\) indicates that the optimization criterion is minimizing the number of tardy packets.
**Algorithm 2 (EDF-w/o-preemption)** *The same as Algorithm 1, with only one exception: the scheduler selects a new packet only after the previously selected packet is fully transmitted.*

It is interesting to mention that in some cases a scheduling algorithm may find a schedule that accommodates all the packets if such a schedule does exist, but the same algorithm does not necessarily minimize the number of lost packets when a schedule that accommodates all the packets does not exist (obviously, the opposite direction always holds). As an example, consider the case where all the packets have an equal size – of $K$ slots, that preemption is permitted, and that nothing can be assumed regarding the release times and due dates of the packets. It is well known that in such a case Algorithm 1 will find a schedule that accommodates all the packets if such a schedule exists. To show that this algorithm is not optimal when such a schedule does not exist, consider a frame of 20 slots and suppose that the size of each packet is 10 slots. Suppose also that we need to schedule three packets: $P_1$, $P_2$ and $P_3$, whose release times are 1, 9 and 11 respectively, and whose due dates are 19, 18 and 20 respectively. It is easy to see that the best we can do is to schedule $P_1$ during slots 1-10 and then $P_3$ during slots 11-20. However, Algorithm 1 (EDF-with-preemption) will schedule packet $P_1$ during slots 1-8 and then packet $P_2$ during slots 9-18. Consequently, it will not be able to finish the transmission of $P_1$ on time, and it will therefore lose two packets. Algorithm 2 will also fail to find the optimal schedule if we increase the frame length to 21 slots and change the release time and due date of $P_3$ to 12 and 21 respectively. Note that in [10] an algorithm that solves this problem in $O(n^5)$ is presented. This algorithm has two phases. In the first phase the algorithm uses dynamic programming in order to determine which of the packets should be scheduled and which of the packets should be discarded due to scheduling conflicts. In the second phase it uses EDF-with-preemption (Algorithm 1) in order to determine the scheduling order of the selected packets.

Despite of this counter-example, in our specific case where the tolerated jitter is equal for all the packets, EDF is an optimal solution for Problem 2 even without requiring packet preemption. This is proven in the rest of this section.

**Definition 4 (A Feasible Schedule)** A schedule on a subset $S$ of packets is said to be *feasible* if according to this schedule all the packets in $S$ are transmitted on time.

**Lemma 1** Suppose that preemption is allowed, and that some scheduler finds a feasible schedule for a set of packets $I$. Then, when $I$ is given as input to EDF-with-preemption, this algorithm will also find a feasible schedule.

This Lemma is proven in [15]. It provides an easy tool for schedulability test when preemption is possible: a set of packets (jobs) can be scheduled on time if and only if it can be scheduled on time by EDF-with-preemption.

**Lemma 2** If the release times and due dates are similarly ordered (Definition 3), the packets are of equal size, and *preemption is not allowed*, EDF-w/o-preemption (Algorithm 2) finds the optimal solution for Problem 2.
Proof: To prove this lemma we consider the algorithm presented in [9] in the context of “one-machine scheduling without preemption”. In [9] it is proven that this algorithm is optimal when the release times and due dates are similarly ordered, preemption is not allowed, and the packets are of arbitrary size. We now show that in the private case where the packets are of equal size this algorithm is equivalent to EDF-w/o-preemption.

Let the packets in active(i) be \( \Pi = \{P_1, P_2, \ldots, P_N\} \). Without loss of generality, assume that for every 1 \( \leq i < N \) \( \text{Create}(P_i) \leq \text{Create}(P_{i+1}) \). For every 1 \( \leq j \leq N \), define \( \Pi_j = \{P_1, P_2, \ldots, P_j\} \). Following [12], the concept of “\( j \)-optimal” set is defined as follows. The set \( E(\Pi_j) \) is said to be \( j \)-optimal if it is the subset of packets in \( \Pi_j \) that can be scheduled on time by an optimal scheduler.

Define \( E(\Pi_0) = \phi \). Let \( T(I) \) be the earliest finishing time of all the packets in \( I \) that are not late, and let \( t^*_j \) be the time of the last slot in frame(i). The algorithm presented in [9] for constructing \( E(\Pi_N) \) from \( E(\Pi_0) \) in the order of \( j = 1, 2, \ldots, N \) is as follows:

\[
E(\Pi_j) = \begin{cases} 
E(\Pi_{j-1}) \cup \{P_j\} & \text{if } T(E(\Pi_{j-1}) \cup \{P_j\}) \leq t^*_j \\
E(\Pi_{j-1}) \cup \{P_j\} - \{P_i\} & \text{else}.
\end{cases}
\]  

(2)

where \( P_i \) is a packet satisfying \( T(E(\Pi_{j-1}) \cup \{P_j\} - \{P_i\}) \leq T(E(\Pi_{j-1}) \cup \{P_j\} - \{P_i\}) \) for every \( P_i \in E(\Pi_{j-1}) \cup \{P_j\} \). Less formally, in order to construct \( E(\Pi_j) \) from \( E(\Pi_{j-1}) \) the algorithm checks if \( P_j \) can be added to the subset of packets from \( P_1 \cdots P_{j-1} \) that can be scheduled on time. If yes, we have \( |E(\Pi_j)| = |E(\Pi_{j-1})| + 1 \). If not, we have \( |E(\Pi_j)| = |E(\Pi_{j-1})| \). However, the algorithm replaces \( P_j \) with a packet \( P_i \) in \( E(\Pi_{j-1}) \) such that the time it takes to schedule the packets in \( E(\Pi_{j-1}) \) with \( P_i \) and without \( P_j \) is minimum (and, in particular, it is smaller than the time it takes to schedule the packets in \( E(\Pi_{j-1}) \) without this substitution). The proof that \( E(\Pi_N) \) as produced by this algorithm is optimal, namely that \( |E(\Pi_N)| \) is larger than or equal to the size of any other subset of packets that can be scheduled on time is presented in [9].

However, if the packets are not only similarly ordered, but are also of equal size, then there is no \( l \leq i \) such that \( T(E(\Pi_{j-1}) \cup \{P_j\} - \{P_i\}) \leq T(E(\Pi_{j-1}) \cup \{P_j\}) \). This follows from the fact that the finishing time of \( P_j \) can never be earlier than the finishing time of \( P_i, 1 \leq l < j \). Therefore, the scheduled produced by Eq. 2 is equivalent to the scheduled produced by EDF-w/o-preemption.

\[\blacksquare\]

**Theorem 3** If the release times and due dates are similarly ordered and the packets are of equal size, EDF-w/o-preemption (Algorithm 2) is an optimal algorithm for Problem 2. That is, no other scheduling algorithm – with or without preemption – is able to schedule on time more packets than EDF-w/o-preemption.

Proof: Suppose that the packets are similarly ordered and of equal size. Consider an optimal scheduling algorithm, referred to as Alg\( \alpha_{\text{opt}} \), for Problem 2. Suppose that when this algorithm is given an instance \( I \), it determines that the set of packets that can be scheduled on time is \( S(I) \). From Lemma 1 follows that when EDF-with-preemption is given the set \( S(I) \) as input, it will schedule all the packets in \( S(I) \) on time. However, since the packets are similarly ordered, the schedule of EDF-with-preemption does not need to use
preemption at all (at every time, one of the packets with the earliest due date is the last packet to have been chosen for transmission whose transmission has not yet been completed). This implies that for every optimal schedule found by Alg opt there exists a feasible schedule for the same set of packets without preemption. Let Alg opt-w/o-preemption be the algorithm that finds the optimal schedule without preemption. Note that this is not necessarily EDF-w/o-preemption, because by Lemma 1 the latter is guaranteed to schedule all the packets in $S(I)$ on time only if it is given as input the set $S(I)$, and not the whole set $I$. Let $S'(I)$ be the set of packets scheduled on time by EDF-w/o-preemption for the instance $I$. By Lemma 2 $|S'(I)| \geq |S(I)|$. However, due to the optimality of $S(I)$ we have $|S'(I)| = |S(I)|$, and the theorem holds.

**Theorem 4** EDF-w/o-preemption is an optimal solution for Problem 2.

**Proof:** Following Theorem 3, we only need to show that for every $i$ the packets in $frame(i)$ are similarly ordered. Namely, that $Release(i, P_1) \leq Release(i, P_2) \Rightarrow Due(P_1) \leq Due(P_2)$ holds for every two packets $P_1, P_2 \in active(i)$. Recall that in our system $D_{sched\_jitter}$ is equal for all the packet. Consider now the following three cases:

(a) $P_1, P_2 \in backlog(i)$: in this case $Release(i, P_1) = Release(i, P_2) = t^i_s$, where $t^i_s$ is the time of the starting slot in $frame(i)$, and the condition holds regardless of the due dates of these packets.

(b) $P_1, P_2 \in create(i)$: in this case $Due(P_1) = Create(P_1) + D_{sched\_jitter}$ and $Due(P_2) = Create(P_2) + D_{sched\_jitter}$, so the condition holds too.

(c) $P_1 \in backlog(i)$ and $P_2 \in create(i)$: in this case $Release(i, P_1) = t^i_s \leq Release(i, P_2)$ and $Due(P_1) < Due(P_2)$ so the condition holds again.

**Theorem 5** If we allow the scheduler to schedule the transmission of a packet during the end of one frame and the beginning of the succeeding frame, then EDF-w/o-preemption is an optimal solution not only for Problem 2 but also for Problem 1.

**Proof:** Ignore the framing of the upstream channel, and run the EDF-w/o-preemption scheduler. By Theorem 3, this algorithm finds the optimal schedule. However, it is easy to see that the schedule found by this algorithm when applied only once is identical to the schedule found by this algorithm when applied in the beginning of each frame if the latter is allowed to schedule a frame even if the transmission of the frame is completed during the next frame only. Note that in order to use the theorems proven earlier, we need to change the definition of $t^i_s$ as follows. If there is no frame whose transmission starts during $frame(i-1)$ and ends during $frame(i)$, $t^i_s$ is the time when $frame(i)$ starts (as before). If there is packet whose transmission starts during $frame(i-1)$ and ends during $frame(i)$, $t^i_s$ is the first slot after the transmission of this packet ends.
Allowing the transmission of a packet on the upstream channel to start in one frame and continue in the successive frame, without preemption, is straightforward in most of the systems that use framing on the upstream channel. This is because there is no need to transmit a special delimiter on this channel between two consecutive frames, as always required on the downstream channel. If the end of an upstream frame should be signaled to all the stations, this is done by the head-end on the downstream channel, and has no effect on the upstream channel.

After proving the optimality of Algorithm 2 (EDF-w/o-preemption), for the rest of the paper we assume that this algorithm is used for scheduling of the VoIP packets.

5 The Effect of Increasing the Packetization Time on Packet Schedulability

It is well known that there is a tradeoff between bandwidth efficiency and packetization delay. When the packet size decreases, the bandwidth efficiency is reduced because of the overhead of the packet header. On the other hand, if the packet size increases, the packetization delay may become unacceptably high. In this section we study the possibility of compensating for the increased packetization delay by reducing the MAC maximum delay. The rationale is that when the packet overhead decreases, the MAC scheduler can issue grants with a smaller tolerated jitter without increasing the packet loss rate. We start with the following theorem, due to [21], that helps to determine in advance whether the packets generated by a set of service flows can be scheduled on time.

**Theorem 6** Consider a set of $N$ VoIP service flows. Suppose that each service flow generates a VoIP packet every $\delta$ slots. Let the size of each packet be $\sigma$ slots, and the tolerated grant jitter be $D_{sched,\text{jitter}}$ slots. Then, the packets generated by these service flows can be scheduled on time using Algorithm 2 (EDF-w/o-preemption) if and only if

$$\forall \text{ time slot } t \geq 0, \quad N \cdot \left[\frac{t - D_{sched,\text{jitter}}}{\delta}\right]^+ \cdot \sigma \leq t$$

where

$$\left[x\right]^+ = \begin{cases} i & \text{ if } i - 1 \leq x < i, \quad i = 1, 2, \cdots \\ 0 & \text{ else (i.e. } i < 0) \end{cases}$$

**Proof:** This theorem is proven in [21] for the case where the scheduling algorithm is EDF-with-preemption. However, when the release times and due dates are similarly ordered and the packets are of equal size, EDF-with-preemption does not use its preemption capability. Hence, the schedule it produces is identical to the schedule produced by EDF-w/o-preemption.

The condition given in Theorem 6 can be used during a call admission control (CAC) phase, in order to determine if a new call should be accommodated and in which channel (in the case where there are multiple independent upstream channels). Moreover, [21] shows that in practice it is not needed to check that Eq. 3 holds for an infinite time interval, but only for a finite set of points. However, in this section we are interested
upstream channel rate

<table>
<thead>
<tr>
<th>voice block size</th>
<th>5 Mb/s</th>
<th>10 Mb/s</th>
<th>20 Mb/s</th>
</tr>
</thead>
<tbody>
<tr>
<td>(D_{\text{packet}}=10) ms (140-byte packets)</td>
<td>(D_{\text{transmit}}=0.224) ms</td>
<td>(D_{\text{transmit}}=0.112) ms</td>
<td>(D_{\text{transmit}}=0.056) ms</td>
</tr>
<tr>
<td>(D_{\text{sched jitter}} \approx 20.8) ms</td>
<td>(D_{\text{sched jitter}} \approx 20.9) ms</td>
<td>(D_{\text{sched jitter}} \approx 21) ms</td>
<td></td>
</tr>
<tr>
<td>(D_{\text{packet}}=20) ms (220-byte packets)</td>
<td>(D_{\text{transmit}}=0.352) ms</td>
<td>(D_{\text{transmit}}=0.176) ms</td>
<td>(D_{\text{transmit}}=0.088) ms</td>
</tr>
<tr>
<td>(D_{\text{sched jitter}} \approx 10.7) ms</td>
<td>(D_{\text{sched jitter}} \approx 10.8) ms</td>
<td>(D_{\text{sched jitter}} \approx 11) ms</td>
<td></td>
</tr>
<tr>
<td>(D_{\text{packet}}=30) ms (300-byte packets)</td>
<td>(D_{\text{transmit}}=0.48) ms</td>
<td>(D_{\text{transmit}}=0.24) ms</td>
<td>(D_{\text{transmit}}=0.12) ms</td>
</tr>
<tr>
<td>(D_{\text{sched jitter}} \approx 0.5) ms</td>
<td>(D_{\text{sched jitter}} \approx 0.75) ms</td>
<td>(D_{\text{sched jitter}} \approx 0.9) ms</td>
<td></td>
</tr>
</tbody>
</table>

(a) Scenario A: \(D_{\text{access budget}}=31\) ms

upstream channel rate

<table>
<thead>
<tr>
<th>voice block size</th>
<th>5 Mb/s</th>
<th>10 Mb/s</th>
<th>20 Mb/s</th>
</tr>
</thead>
<tbody>
<tr>
<td>(D_{\text{packet}}=10) ms (140-byte packets)</td>
<td>(D_{\text{transmit}}=0.224) ms</td>
<td>(D_{\text{transmit}}=0.112) ms</td>
<td>(D_{\text{transmit}}=0.056) ms</td>
</tr>
<tr>
<td>(D_{\text{sched jitter}} \approx 25.8) ms</td>
<td>(D_{\text{sched jitter}} \approx 25.9) ms</td>
<td>(D_{\text{sched jitter}} \approx 26) ms</td>
<td></td>
</tr>
<tr>
<td>(D_{\text{packet}}=20) ms (220-byte packets)</td>
<td>(D_{\text{transmit}}=0.352) ms</td>
<td>(D_{\text{transmit}}=0.176) ms</td>
<td>(D_{\text{transmit}}=0.088) ms</td>
</tr>
<tr>
<td>(D_{\text{sched jitter}} \approx 15.7) ms</td>
<td>(D_{\text{sched jitter}} \approx 15.8) ms</td>
<td>(D_{\text{sched jitter}} \approx 16) ms</td>
<td></td>
</tr>
<tr>
<td>(D_{\text{packet}}=30) ms (300-byte packets)</td>
<td>(D_{\text{transmit}}=0.48) ms</td>
<td>(D_{\text{transmit}}=0.24) ms</td>
<td>(D_{\text{transmit}}=0.12) ms</td>
</tr>
<tr>
<td>(D_{\text{sched jitter}} \approx 5.5) ms</td>
<td>(D_{\text{sched jitter}} \approx 5.75) ms</td>
<td>(D_{\text{sched jitter}} \approx 5.9) ms</td>
<td></td>
</tr>
</tbody>
</table>

(b) Scenario B: \(D_{\text{access budget}}=36\) ms

Figure 1: The Scheduling Tolerated Jitter for each Case Study in both Scenarios

in another aspect of Theorem 6: understanding the effect of increasing the number of voice blocks in a single packet on the schedulability of the VoIP packets. When we increase the number of voice block in a packet, we decrease the number of packets, but we do not affect the number of voice blocks transmitted in the channel. In terms of Eq. 3, \(\sigma\) increases in the same factor that \(\delta\) increases. However, when we take into consideration the overhead of the packet headers, i.e. 40 bytes for IP/UDP/RTP, the increase factor of \(\sigma\) is smaller than the increase factor of \(\delta\). This may suggest that the condition for schedulability is more likely to hold when a packet contains more data blocks. However, from Eq. 1 follows that by increasing the number of voice blocks in a packet, and thereby the value of \(D_{\text{packet}}\), we must decrease \(D_{\text{sched jitter}}\). Hence, we see two counter-forces in Eq. 3: the decreased load on the channel versus the decreased tolerated jitter. This implies that although we reduce the bandwidth consumed by the VoIP packets, in some circumstances the schedulability of these packets can be negatively affected.

We used simulations in order to gain better understanding of these counter-forces. We generated VoIP calls using parameters of real codecs under different loads, and executed the EDF-w/o-preemption scheduling algorithm in order to find the correlation between the packet loss rate and the number of voice blocks in a packet for different upstream channel transmission rates. As already said, we concentrate our attention on relatively slow upstream access channels, in the range of 5-30Mb/s, because only in such channels the relative weight of the VoIP packets is expected to be significant. We consider the standard G.711 codec that
generates voice samples at a rate of 64 Kb/s. For each channel we considered three packetization delays as follows:

1. The case where each packet contains 10ms worth of voice samples. Consequently, each packet contains 80 bytes (8KB/s*0.01) of voice samples and 60 bytes of headers (40 bytes for the IP/UDP/RTP plus 20 bytes for the MAC/PHY headers). The total size of a packet in this case is 140 bytes.

2. The case where each packet contains 20ms worth of voice samples. Consequently, there are 160 bytes of voice samples in a packet, and the total size is 220 bytes.

3. The case where each packet contains 30ms worth of voice samples. Consequently, there are 240 bytes of voice samples in a packet, and the total size is 300 bytes.

As already indicated, for toll-quality calls the end-to-end delay should not exceed 150 ms, and $D_{\text{access-budget}}$ is only one component of the end-to-end delay. Throughout the simulation study we considered two scenarios as follows:

**Scenario A:** $D_{\text{access-budget}} = 31$ ms.

**Scenario B:** $D_{\text{access-budget}} = 36$ ms.

We shall show that despite of the small difference between the selected values, we get different behavior for the various packetization delays. By Eq. 1, $D_{\text{sched-jitter}} = D_{\text{access-budget}} - D_{\text{packet}} - D_{\text{transmit}}$. The table in Figure 1 shows the value of $D_{\text{sched-jitter}}$ for each packet size when the upstream channel rate is 5, 10 and 20Mb/s for Scenario A and Scenario B.

The simulation results are interesting and non-intuitive, since they reveal that the balance point between the two counter-forces discussed before is very hard to predict. Figure 2 shows the packet loss rate versus number of VoIP calls for each packet size and each upstream channel rate in Scenario A. We prefer using the “number of calls” metric rather than the “UGS load” because the latter depends on the number of voice blocks in a packet and on the channel speed whereas the former is independent of these two variables. Consider the 5Mb/s upstream channel first (Figure 2(a)). It is evident that for a relatively light load of VoIP (less than 30 calls, which is equivalent to $\approx 40\%$ of the total bandwidth if the header overhead is ignored) it is better to use the 300-byte packets because the loss rate is negligible ($\approx 0$) for the three cases. However, when the VoIP load increases, we see that with 220-byte packetization we get the best performance. The 300-byte packets give lower loss rate when we increase the load of the VoIP packets further, to 75 calls. However, this load is well above the channel capacity so the results for this range are irrelevant.

When we increase the channel rate to 10Mb/s (Figure 2(b)), and then increase it further to 20Mb/s and 30Mb/s (Figure 2(c) and (d)) we see a very interesting phenomena: while the difference between the 140-byte packets and the 220-byte packets sustains, the curve of the 300-byte packets is shifted right as the channel rate increases. Consequently, when the rate is 30Mb/s, we see that with 300-byte packetization we get the best performance.
Figure 2: Packet loss rate for different packetization times and upstream channel rates in Scenario A

Figure 3 shows the results for Scenario B, for a 5Mb/s channel and a 10Mb/s channel. By comparing Figure 2(a) to Figure 3(a) and Figure 2(b) to Figure 3(b) we see that as the value of $D_{access-budget}$ increases, it is better to use larger packets. For instance, while in Scenario A ($D_{access-budget} = 31 ms$) for an upstream channel of 5Mb/s the 300-byte packets have the higher loss rate, in Scenario B ($D_{access-budget} = 36 ms$) for an upstream channel of 5Mb/s the 300-byte packets have the lower loss rate. The main conclusions we draw from the graphs in Figure 2 and Figure 3 are:

(C1) When the transmission rate of the upstream channel increases, the advantage of large packetization time increases. See Figure 2(a) vs. (b), (c) and (d).

(C2) When the value of $D_{sched-jitter}$ (or the value of $D_{access-budget}$) increases, the advantage of large packetization time increases. See Figure 2(a) vs. Figure 3(a).

(C3) When the load of the upstream channel increases, the advantage of large packetization time increases. See the various curves in Figure 2(a) or in Figure 2(b).
To explain these findings, note that there are two factors for loosing packets: scheduling conflicts and channel overflow. The pseudo-linear increase in the loss rate which we see for all the curves after a certain threshold is related to channel overflow: when the load on the channel is $x > 1$, the loss due to overload is $x - 1$, and the loss rate is $(x - 1)/x = 1 - 1/x \approx 1 - (1 - x) = x$. The loss rate that exceeds this linear increase when the channel is not overloaded is attributed to scheduling conflicts. The explanation to (C1), (C2) and (C3) can be found in the way the two counter-forces in of Eq. 3 affect each of these two factors: the decreased load on the channel mainly affects the losses due to channel overflow, whereas the decreased tolerated jitter affects only the losses due to scheduling conflicts.

Let us start with (C1). When the transmission rate increases, there are more slots in a single time unit. Hence, the number of slots in the tolerated jitter increases. Figure 4 shows the loss rate as a function of number of slots in $D_{sched, jitter}$ for 60%, 80% and 100% load, regardless of the channel rate and packetization time. This independence is achieved by considering 1-slot VoIP packets. As shown in this figure, when the number of slots in the jitter is sufficiently large (the exact number depends on the load, but it ranges
between 5 and 10), we shall see no scheduling conflicts, and the main loss factor is channel overflow. Under these conditions, the long packets, that have smaller overhead, have a pure advantage. The explanation to (C2) is similar, because increasing $D_{sched,jitter}$ is equivalent to increasing the number of slots in the tolerated jitter. To some extent, the explanation of (C3) is using an opposite argument. First, note that the losses due to scheduling conflicts are usually limited to 1-2%. Hence, when the load on the channel increases, the main factor is packet overflow. When this is the case, the effect of decreasing the load on the channel by using larger packets is stronger than the effect of decreasing the number of slots in the tolerated jitter. Another explanation to (C3) is that when the load on the channel is significantly high, i.e. beyond 100%, the loss due to insufficient jitter decreases. This is shown in Figure 5. The graph in this figure describes the case where 300-byte packets are used in a 5Mbps channel in Scenario A. The upper curve represents the total packet loss. The lower curve represents the packet loss due to the channel overflow, computed as explained earlier (i.e. if the channel load is $x > 1$, the loss due to overload is $x - 1$, and the loss rate is $(x - 1)/x$), and the vertical lines between the two curve represent the differences between the two curves, namely the packet loss rate due to a small jitter. It is evident from the graph that when the load on the channel is higher than 100%, the length of the vertical lines decreases as the load increases. The reason for this is that when we have a higher load, we have more packets to choose from in order to fill the available slots while meeting the jitter constraints.

6 The Effect of Silence Suppression on VoIP Packet Schedulability

Silence suppression is known to be a very effective approach for decreasing the bandwidth consumed by a VoIP call. Silence suppression involves detecting parts of a signal where there is no speech and discontinuing the codec output. In normal conversation, silence suppression decreases the number of voice blocks generated by the codec by 40-60%. A shared medium broadband access network can take advantage of the bandwidth saving of silence suppression using the UGS-AD (UGS with Activity Detection) access scheme.
With UGS-AD, the base-station views an unused grant as a signal of voice inactivity. It therefore stops allocating unsolicited grants to the host until the host signals the start of a new talk-spurt – either using a polling mechanism or using some ALOHA-like contention algorithm. If the base-station is informed by a host in some upstream frame of a new talk-spurt, this new talk-spurt can be taken into consideration by the upstream scheduling algorithm only in the succeeding upstream frame\(^{2}\). The implication of this is that the first packet of a new talk-spurt experiences an average delay of at least \(\mathcal{F}/2\) slots before it can be taken into consideration by the scheduler, where \(\mathcal{F}\) is the length of the upstream frame. Such a delay can harmfully affect the schedulability of the first packet in every talk-spurt. Unlike in the case discussed in Section 5, here we have a strong correlation between the size \(\mathcal{F}\) of a frame and the performance of the scheduler: the longer the frame size, the lower the probability that the first packet can be scheduled on time.

We first prove that Algorithm 2 (EDF-w/o-preemption) is also the best scheduling algorithm in this case.

**Theorem 7** Let \(P\) be the first packet of a new talk-spurt. Suppose that the scheduler is informed of this new talk-spurt during frame\((i - 1)\). Define Release\((i, P) = t_i^i\). If we allow the scheduler to schedule the transmission of a packet during the end of one frame and the beginning of the succeeding frame, then EDF-w/o-preemption is an optimal solution for Problem 1 when silence suppression is used.

**Proof:** The considerations here are similar to those discussed in Theorem 4 and Theorem 5. The only difference is in the proof that the packets in frame\((i)\) are similarly ordered. As in the proof to Theorem 4 we consider two packets \(P_1, P_2 \in active(i)\) and show that Release\((i, P_1) \leq\) Release\((i, P_2)\) implies \(Due(P_1) \leq Due(P_2)\). If neither \(P_1\) nor \(P_2\) signals a new talk-spurt, the proof is the same as in Theorem 4. If both \(P_1\) and \(P_2\) signal a new talk-spurt (of different service flows) then we have Release\((i, P_1) = Release(i, P_2) = t_i^i\), and the condition holds. Finally, if only one packet, \(P_1\) say, signals a new talk-spurt while the other packet \(P_2\) does not signal a new talk-spurt, we distinguish between the following two cases:

(a) If \(P_2 \in backlog(i)\), Release\((i, P_1) = Release(i, P_2) = t_i^i\), and the condition holds.

(b) If \(P_2 \in create(i)\), Due\((P_1) < Due(P_2)\) and Release\((i, P_1) < Release(i, P_2)\), so the condition holds too.

\[\square\]

In order to simulate the effect of silence suppression on the schedulability of VoIP packets, we consider the standard ON/OFF model for the talk-spurts and silence periods generated by the codec. According to [7], in most cases the talk-spurt distribution is slightly more “heavy-tailed” than exponential, whereas the silence distribution deviates strongly from an exponential model. However, since we found almost no correlation between the exact distribution of the ON/OFF periods and the performance of the scheduler, we consider in our study the standard approach where the talk-spurts and silence periods alternate according a two-state Markov chain with average periods of 1.0 and 1.3 seconds respectively. We assume that when the

\(^{2}\)One may consider an approach where the host does not only signal the existence of a new voice packet, but it also uses the polling opportunity or the contention channel in order to send this packet. However, this approach is known to be inefficient because it increases the contention overhead.
first packet of a new talk-spurt is ready, the base-station is immediately notified. However, this packet can be scheduled for transmission only in the next upstream frame. We did not simulate the exact scheme, like ALOHA or polling, used for the notification step, because the details of such a scheme are likely to vary from one network to another.

The simulation results are presented in Figure 6 for a 5 Mb/s upstream channel, and in Figure 7 for a 10 Mb/s upstream channel. In each figure we consider 3 possible lengths of the upstream channel frame: 5ms, 10ms and 15ms, in order to show the strong correlation between the length of the upstream frame and the schedulability of the first packet in every talk-spurt. It is interesting to compare the graph in Figure 6 to the graph in Figure 3(b) because a 5 Mb/s channel with silence suppression ratio of 1:1.3 is equivalent to a 11.5 Mb/s channel without silence suppression. Indeed, when the frame size is 5ms (Figure 6(a)) the two graphs are almost identical. The implication of this similarity is that when the frame size is relatively small compared to $D_{\text{sched jitter}}$, the scheduler does not encounter difficulties in accommodating the first packet of every talk-spurt.

However, when the frame size increases to 10ms (Figure 6(b)) we see a significant loss rate for the
300-byte packets. Recall that in Scenario B, when the channel rate is 5Mb/s and $D_{\text{packet}}=30\text{ms}$ (300-byte packets) then $D_{\text{sched\_jitter}} = 5.5\text{ms}$. Hence, with a probability of $1 - 5.5/10 = 0.45$ the first packet of a talk-spurt has no chance to be transmitted on time. Since a new talk-spurt starts every 2.3 seconds, and the number of 300-byte packets generated during every 1-second ON period is $1000/30$, 3% of the packets start a new talk-spurt. Consequently, we expect to get a loss rate of $0.45 \times 3\% = 1.35\%$ independently of the load on the channel, as indeed is the case in Figure 6(b). When the frame size increases to 15ms, the probability to loss the first packet of a talk-spurt increases to $1 - 5.5/15 = 0.63$, and the loss rate increases to $0.63 \times 3\% = 1.9\%$, as we can see indeed in Figure 6(c).

Due to the loss rate of the first packet of every talk-spurt, we see that the advantage of the 300-byte packets disappears. With packetization time of 220-byte only, the value of $D_{\text{sched\_jitter}}$ is $15.7\text{ms}$. Hence, even if the frame size is 15ms the scheduler is able to accommodate the first packet in each talk-spurt and the loss rate due to the transition from silence to activity is avoided. When the load on the channel is high, we start seeing loss due to congestion. In this case, the advantage of bandwidth saving with 300-byte packetization wins the disadvantage attributed to the high loss probability of the first packet in a talk-spurt.
Throughout this section we have seen the strong correlation between the upstream frame size and the performance of the scheduling algorithm. However, many systems will employ a continuous upstream channel. The results in this sections are applicable for such systems as well with only one difference: the value of $F$ will not represent the length of the frame, but the maximum time interval between the signaling of a new talk-spurt by an end host and the allocation of a slot to this packet by the head-end.

7 Conclusions

This paper addressed the scheduling of VoIP packets in a shared medium broadband access network. We have proven that in a system where the size of the packets and the tolerated jitter are equal, EDF-w/o-preemption is an optimal scheduling algorithm, in the sense that no other algorithm, with or without pre-emption, may schedule more packets on time.

We have then concentrated upon two issues related to the effect of VoIP codec functionality on the schedulability of VoIP packets: packetization delay and silence suppression. We showed that in both cases there are two counter-forces that have to be carefully examined. Increasing the packetization delay has a positive effect on the schedulability of the VoIP packets due to the decreased load on the channel, and a negative effect due to the decreased tolerated jitter. The implementation of silence suppression has a positive effect on the schedulability of the VoIP packets due to the decreased load on the channel, and a negative effect due to the decreased tolerated jitter of the first packet in every talk-spurt.

We used simulations in order to get in-depth understanding of the counter-forces under different conditions. We found that it is indeed very hard to predict in advance the effect of the packetization delay and silence suppression on the schedulability of the VoIP packets, because by changing the tolerated jitter, the load on the channel, or the rate of the channel, we get different balance points. We can summarize our findings as follows. Increasing the size of the VoIP packets improves the schedulability of these packets except in the case where (a) the rate of the channel is small ($< 10 Mb/s$), the delay budget of the access network is small ($< 30 - 35 ms$) and the bandwidth consumed by the VoIP packets is relatively high ($> 40 - 50\%$ of the channel bandwidth); or (b) the length of the upstream frame is small ($\approx 5 ms$) and silence suppression is used.

We can generalize the results we got as follows. Regardless of the load on the channel, when silence suppression is not used, no loss is encountered due to scheduling conflicts if the length of the scheduling jitter is 10-15 times larger than the size of the packets. When silence suppression is used, it is also required to guarantee that the length of the scheduling jitter will be larger than the length of the upstream frame, or more generally – than the time it takes to signal the transition of the voice from an inactive state to an active state.

Throughout the paper we compared the various schemes according to their loss rates. We did not try to differentiate between the loss penalty in each case. However, an important related research is to examine for every case the real impact a lost packet has on the voice quality, as measured using the Mean Opinion Score
References


