A Generic Quantitative Approach to the Scheduling of Synchronous Packets in a Shared Medium Wireless Access Network

Reuven Cohen  Liran Katzir
Department of Computer Science
Technion
Israel

Abstract—We present a scheme for allocating unsolicited grants to the end hosts of synchronous applications of a wireless access network, in accordance with the condition of the channel, the importance of each packet and the specific loss recovery mechanism employed in the channel. The proposed scheme is generic in the sense that it maximizes the effectiveness of the channel under various conditions and it can be used along with every FEC-based or retransmission-based error recovery strategy.

I. INTRODUCTION

A synchronous application, like streaming (one way voice/video) or telephony (two way voice), is an application that demands from the network guaranteed maximum delay and loss rate. Such a service usually requires a mechanism for bandwidth reservation. However, even if bandwidth is reserved in advance, meeting the delay and loss requirements of synchronous applications while using the network efficiently is difficult.

In wireless access networks, like IEEE 802.16 or IEEE 802.11 (when “Point Coordination Function” is used in the infrastructure mode), there is a common channel that needs to be shared by many stations using a MAC (Medium Access Control) protocol. There are several possible MAC mechanisms for allocating bandwidth on the upstream channel. The most useful mechanisms for synchronous traffic are UGS (Unsolicited Grant Service) and UGS-AD (UGS with Activity Detection). 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 the traffic generated by the application is unpredictable. For example, this is the case when silence suppression is used for VoIP (Voice over IP). In such a case UGS-AD is employed. With UGS-AD, the base-station views an unused grant to a VoIP application 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 a contention-based mechanism or using a polling-based mechanism. Throughout the paper we use the term UGS for both UGS and UGS-AD.

UGS, with or without activity detection, is inefficient when the channel is erroneous, in which case it might be worthwhile to delay the transmission of a packet until the likelihood of successful transmission increases, or to employ some retransmission strategy. In this paper we propose an enhanced version of UGS, where the base-station issues unsolicited grants to the end hosts in accordance with the condition of the channel, and the specific loss recovery mechanism. The proposed scheme is described in the context of upstream transmission. However, with minor adjustments it is also applicable for the downstream transmission as well.

The most common techniques for increasing the quality of a synchronous call in an erroneous shared-medium wireless channel are as follows:

1) Using a re-transmission strategy. While for two-way synchronous applications, like VoIP, the time-sensitive nature of the stream does not allow for end-to-end retransmission of a packet, re-transmission is possible and useful if it is used over a short erroneous segment. In a wireless network, if the base-station does not receive a correct synchronous packet from a host, it can immediately ask the host to re-transmit this packet over some allocated upstream slots. The whole process may take no longer than a few mili-seconds, which is usually well within the MAC tolerated grant jitter budget.

2) Using a FEC-based scheme. The idea behind FEC is to add some repair data into the transmitted packets in order to reconstruct the missing packets. FEC schemes are divided into two classes: media-independent schemes, and media-dependent schemes. Media-independent FEC schemes are based on Reed-Solomon codes or on a XOR function. Reed-Solomon is more efficient, but leads to a higher processing cost. The notation of a media-independent FEC code is \((n; k)\) where \(n\) is the number of total units in a FEC block and \(k\) is the number of units the receiver should correctly receive in order to decode the whole block successfully. However, this approach requires that the tolerated jitter will be \(n\) times larger than the packetization interval. Media-dependent FEC schemes are usually more efficient, but
The main idea behind the proposed scheme is assigning a merit to the transmission of each synchronous packet at every time slot. For example, when media-dependent FEC is used, the merit of a packet increases if the previous packet of the same call was not successfully delivered. If re-transmission is allowed, the merit of transmitting a packet at slot \( t \) is usually higher than at slot \( t' > t \) because an early transmission is more likely to leave enough time for a possible re-transmission.

The purpose of the proposed scheme is three-fold, as described in the following and summarized in Figure 1. The first purpose (Scenario A in Figure 1) is determining which of the synchronous packets should be dropped and which of them should be transmitted. Such a decision is important only if the channel bandwidth and the tolerated jitter are not enough large compared to the demand of the synchronous applications. This is well shown in Figure 2, that depicts the loss rate of VoIP packets for various loads. When the normalized jitter is higher than 10, losses due to scheduling conflicts are not likely to take place even if the load of the synchronous traffic is very close to 100%. Note that the non-synchronous traffic has no effect on the graph because it is accommodated only when there is no synchronous traffic. In an erroneous channel, the scheduler has an important task even if the channel is under-loaded. If the tolerated jitter is long enough compared to the average length of an error burst (Scenario B in Figure 1), e.g. as in the case of video streaming, the task is determining the best transmission time for each channel. This will maximize the number of synchronous packets that are received on time with no error. When the tolerated jitter is not long enough (Scenario C in Figure 1), e.g. as in the case of packetized telephony, the scheduler does not have enough flexibility to wait until an error burst is likely to end. In such a case, the task of the scheduler is determining whether or not to transmit a packet, such that the number of unsuccessful transmissions is minimized. This can increase the available bandwidth for non-synchronous (best-effort) applications.

To summarize, the proposed quantitative-based approach is generic because it is applicable (a) regardless of what mechanism is used for increasing the reliability of the channel; (b) regardless of the properties of the considered synchronous traffic; and (c) regardless of the load imposed by the synchronous traffic on the channel.

The rest of this paper is organized as follows. In Section II we present the main concepts of the quantitative scheduling scheme. This section also discusses related work. In Section III we present the quantitative scheme under the assumption that there is no information regarding the channel condition. In Section IV we present algorithms for finding a schedule with an optimal profit. In Section V we present some simulation results for the proposed scheme. Finally, Section VI concludes the paper.

II. A SCHEDULING ALGORITHM FOR MAXIMIZING THE SCHEDULING PROFIT

A. Design considerations and related work

The task of the scheduler logic at the base-station is to minimize the number of losses that take place either due to scheduling conflicts (congestion) or due to transmission

<table>
<thead>
<tr>
<th>Scenario</th>
<th>load of synchronous traffic</th>
<th>channel condition</th>
<th>tolerated jitter</th>
<th>scheduler task</th>
<th>the benefit</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>high</td>
<td>don’t care</td>
<td>short</td>
<td>select the most important packets for transmission</td>
<td>most important synch. packets will be transmitted on time</td>
</tr>
<tr>
<td>B</td>
<td>don’t care</td>
<td>high error rate</td>
<td>long</td>
<td>select the best time for transmitting each packet</td>
<td>more synchronous packets will be transmitted on time</td>
</tr>
<tr>
<td>C</td>
<td>don’t care</td>
<td>high error rate</td>
<td>short</td>
<td>minimizing the number of bad synchronous transmissions</td>
<td>more bandwidth for non-synchronous applications</td>
</tr>
</tbody>
</table>

Fig. 1. The three tasks of the proposed scheduling algorithm

Fig. 2. The loss rate vs. jitter and load for VoIP packets
errors, while taking into account the following scheduling considerations (SCs):

(SC1) In order to avoid losses due to a burst of errors, the scheduler should defer the transmission of a packet in an erroneous channel as much as possible.

(SC2) If packet re-transmission strategy is possible, in terms of the length of the tolerated jitter, the scheduler should leave enough time for re-transmitting a lost packet in the case of transmission error. Note, however, that such a policy contradicts the previous one (SC1).

(SC3) When FEC is used, the scheduler needs to give a special priority to a packet (or packets) of a call that has experienced a recent loss, in order to minimize the number of correlated losses.

We are not aware of previous works that address the scheduling problem while taking into account considerations (SC1), (SC2) and (SC3). Interesting scheduling algorithms that do take into consideration the loss due to transmission errors are proposed in [1]. However, these algorithms seek to distribute the packet loss evenly among the various calls rather than to avoid consecutive losses. Ref. [1] also provides a general guideline for addressing SC1 and SC2 using the notion of “backoff time”. The idea is that if a call has experienced a recent loss due to transmission errors, then a new packet generated by this call at time \( t_0 \), whose deadline is \( t_1 \), will be scheduled during the time interval \( [\frac{t_0 + t_1}{2}, t_1] \) rather than during the time interval \( [t_0, t_1] \), in order to leave some time for a possible re-transmission. However, this scheme is only mentioned as a possible strategy. Its performance is not discussed, and in particular it is not shown to perform well in the case where only (SC1) is effective (i.e. re-transmission is not used), or in the case where only (SC2) is effective, or in the case where both (SC1) and (SC2) are effective. Ref. [5] proposes a model for wireless fair scheduling based on the adaptation of fluid fair queuing. This model is different from the model proposed in this paper. While we assume that the base-station knows the exact time when each host has a new awaiting packet, such an assumption is not made in [5]. On the other hand, issues related to possible re-transmission-based and FEC-based strategies are not taken into account by [5]. In [8] it is shown that when EDF (Earliest Deadline First) is implemented over channels that are in a good condition, then the number of packet lost due to deadline expiration is minimized. This result is important when the channel is congested, in which case it is important to determine which of the packets should be transmitted and which of the packets should be dropped. This is in contrast to the problem considered in this paper where we determine what is the optimal time for transmitting each packet not necessarily under congestion conditions.

The scheme proposed in this paper is based upon quantitative rather than qualitative considerations. For each “scheduling interval”, namely an interval of time for which scheduling decisions are made by the base-station while taking into account all the packets released by this time interval, the scheduler determines the expected profit for scheduling every packet in every time slot. Then, the schedule with maximum expected profit is chosen. The most difficult part of this scheme is finding a good profit function that takes into account the various considerations. If the channel is known to be clean from transmission errors, and our only concern is minimizing the total number of losses, without associating an extra penalty with correlated losses, then we could assign a profit of 1 unit to each packet \( P \) in every slot \( s \in [\text{Release}(P) \cdot \cdot \cdot \text{Deadline}(P)] \), where \( \text{Release}(P) \) is the time slot when packet \( P \) is ready for transmission and \( \text{Deadline}(P) \) is the last slot when it can be transmitted in order to meet the tolerated grant jitter requirement. A a profit of 0 unit is then assigned to every other slot. If the packets are of different length, we can use the number of payload bits as the profit function instead of using an equal profit of ‘1’.

However, when we want to take into consideration the extra penalty associated with correlated losses, the status of each channel, and a possible re-transmission mechanism, determining the profit associated with scheduling a packet \( P \) at slot \( s \) is much more difficult. In fact, even if we assume that there are no transmission errors, and therefore considerations (SC1) and (SC2) are irrelevant, unless \( P=\text{NP} \) no profit function can be used in order to minimize the number of correlated losses as required by (SC3). The proof for this claim is omitted from this version of the paper due to lack of space.

In what follow we discuss several models related to the considered problem, and a generic “profit-based” framework for solving the problem in every model. We start with the basic model, Model 1, where the channel is assumed to be clean of errors. This model is used in order to present the concepts of the proposed framework. We then consider Model 2, where errors are presented but there is no correlation between different packets of the same synchronous call. In Model 3 we assume that errors are present and there is correlation between every two consecutive packets of the same call, but it is impossible to re-transmit a lost packet. Finally, in Model 4 we extend Model 3 by allowing packet re-transmission.

B. The quantitative-based framework

Model 1: Consider a clean channel. In such a channel packets can be lost due to scheduling conflicts only, if the synchronous packets require an excessive percentage of the channel bandwidth or if the tolerated grant jitter is small (see Figure 2). Assume that there is no correlation among consecutive packets of the same call. If all the packets are of equal size, the optimization criterion is maximizing the total number of non-tardy packets. In the more general case where the packets are of variable length, different nodes use different transmission speeds, the optimization criteria is maximizing the number of payload bits in the non-tardy packets transmitted at a single upstream slot.

EDF is known to produce an optimal schedule for Model 1 in the case where all the packets have the length of \( 1 \)-slot time. When all the packets are of equal size of \( K \)-slot time, EDF is only a
We seek for a schedule \( F \) using dynamic programming [2]. However, the running time complexity of this algorithm – \( O(n^2) \) – renders it impractical. As shown in [3], when the packets do not necessarily have an equal length, the problem considered in Model 1 is NP-Complete.

We now present our generic “profit-based” framework for solving the problem in Model 1. We assume that the head-end executes the scheduling algorithm every scheduling interval of \( T \) upstream slots. The base-station maintains a profit matrix \( M \). Entry \([c, t]\) in this matrix indicates the expected profit for scheduling the current packet of call \( c \) starting from slot \( t \). If the transmission at slot \( t \) is not completed before the deadline of the packet − the profit is 0, otherwise − the profit is equal to the number of payload bits in the packet. It will be convenient to assume in the meantime that each synchronous call has only one “current” (i.e. pending) packet during each scheduling interval. A pending packet is a packet that (1) was released, (2) has not yet been successfully transmitted, and (3) whose due date has not yet expired. This assumption holds only when the tolerated grant grant jitter for a call is smaller than the packetization interval. For instance, when the application is voice, the former ranges between 2-10 ms, while the latter ranges between 10-30 ms. In contrast, for video applications the tolerated grant jitter is much larger than the packetization interval. Namely, if the packet approaches its deadline in order to encourage a transmission even if the channel of call \( c \) is in a bad condition during this time interval. This implies that there is a high probability for a transmission error in this channel if a packet will be scheduled for transmission. However, assuming that the shared channel is not over-loaded, an algorithm that seeks to maximize the expected profit will choose to transmit the packet of call \( c \) at a slot where the error probability is the smallest, even if this probability is still relatively high. The reason is that when the channel is not over-loaded, it is likely that there exist some empty slots that are not used for any other packet, during which the expected profit of transmitting the packet of call \( c \) is small but positive. However, it would probably be much better not to schedule this packet during the current scheduling interval, but to wait for one of the following scheduling intervals \([T + 1 \cdots 2T]\), or \([2T + 1 \cdots 3T]\) etc., when the probability for an error in the channel of call \( c \) might decrease.

There are several possible approaches for addressing this problem. One approach is to change the optimization criterion from maximizing the aggregated expected profit to maximizing the average expected profit per slot. This criterion penalizes the scheduler for transmitting packets in bad slots. However, the drawback of this approach is that the scheduler will avoid transmitting a packet in a bad channel even if this packet is very close to its deadline. We therefore choose another approach for handling this problem. We determine a minimum threshold \( \Delta \) for the probability of a successful transmission. When this probability is smaller than \( \Delta \), the profit is set to 0, and the scheduler does not select the packet for transmission. As we increase the value of \( \Delta \) towards 1, the aggregated profit decreases, but the bandwidth lost due to unsuccessful transmissions decreases as well, and vice versa. Note that this bandwidth can be assigned to other applications of stations whose channel is in a good condition.

We could decrease the value of \( \Delta \) as the packet approaches its deadline in order to encourage a transmission even if the channel is not good when waiting for a future slot is not possible any more. However, this effect is already achieved by
the mechanism that determines the probability of a successful transmission in every slot as described in Section II-C and III.

Model 3: As in Model 2, we consider an erroneous channel. However, we also suppose that media-dependent FEC is used, and therefore there exists a strong correlation among succeeding packets. To be more specific, we assume that for every synchronous call, the \(i'th\) packet contains a high-rate encoding of the \(i'th\) application block and a lower-rate encoding of the \((i-1)'th\) application block. The size of the high-rate encoding block for call \(c\) is \(HR_c\) bits, whereas the size of the low-rate encoding is \(LR_c\) bits.

We use the profit-based framework as described before, with one modification: the expected profit of packet \(i\) of each synchronous call depends now not only on the quality of the channel and on the size of the packet (\(HR_c + LR_c\) bits), but also on the status of the previous packet. If packet \(i-1\) is received on time and with no error, then only \(HR_c\) bits in packet \(i\) are relevant. If packet \(i-1\) is not received on time, all the \(HR_c + LR_c\) bits are relevant. Hence, the scheduler gives a priority to a packet of a call whose previous packet was lost over a packet of a call whose previous packet was not lost.

Model 4: This is the most generic model, that considers an erroneous channel and a possible usage of FEC and/or some re-transmission strategy. The profit assigned to the transmission of each packet \(P\) in every time slot \(t\) takes into account a possible re-transmission of \(P\) in a later slot \(t' > t\) if the transmission at \(t\) encounters an erroneous channel.

While in Model 2 and Model 3 the decisions made by the scheduler are mainly driven by the condition of the channel, in Model 4 we have a new constraint: the scheduler in this case will benefit from scheduling a packet before the packet due date even if the channel is not perfect, because this will allow a re-transmission if the first transmission is lost. If we want to employ the profit-based framework for this model, these considerations should be somehow reflected in the profit function. In the following discussion we assume that only one re-transmission is possible, and then we extend it to allow multiple re-transmissions. The re-transmitted copy is scheduled only after the first copy gets lost. Hence, the expected profit of this packet is determined as discussed before for Model 2 when FEC is not used, and for Model 3 when FEC is used. For the original copy, the expected profit also reflects the chance of a successful re-transmission if this copy is lost, namely:

\[
E(\text{Profit of transmitting at } t \text{ the } 1st \text{ copy of packet } i) = \\
\text{Prob}(\text{this copy is successfully transmitted}) \cdot \\
\cdot \text{number of relevant bits in the first copy} + \\
\cdot \text{Prob}(\text{2nd copy is successfully transmitted} / \\
1st \text{ copy is lost}) \cdot \\
\cdot \text{number of relevant bits in the 2nd copy}.
\]

\(^1\)It is assumed that both the base-station and the transmitting host are aware of an unsuccessful transmission of a packet, and they can therefore perform the necessary actions described in the following discussion.

However, the number of relevant bits is the same in the first and in the second copy of a packet. When FEC is not used, this number is equal to the number of payload bits. When FEC is used, this number for call \(c\) is \(HR_c + LR_c\) if the previous packet of this call is lost and \(HR_c\) if the previous packet is not lost.

Determining the probability that the 2nd copy of a packet \(P\) is successfully transmitted before its deadline given that the 1st copy was lost is not easy, because this probability depends on the time when the 2nd copy is transmitted. If the first transmission takes place at \(t\) and encounters a transmission error, and there is a correlation between the status of the channel in consecutive slots, the best would be to re-transmit the packet “just” before Deadline(\(P\)), because this is the time during the interval \([t, \text{Deadline}(P)]\) when the error probability is minimal. Therefore, we shall assume that the second transmission indeed takes places just before the deadline of \(P\). Hence,

\[
E(\text{Profit of transmitting at } t \text{ the } 1st \text{ copy}) = \\
(\text{Prob}(\text{this copy is successfully transmitted}) + \\
\cdot \text{number of relevant bits in this packet}.
\]

In the next subsection we show how the probability for successful transmission in every slot can be computed when the loss process is according to Gilbert model. This will allow us to compute Eq.1 and to generalize it to every number of re-transmissions. However, this computation requires a good knowledge of the channel condition which is very difficult to achieve when the stations are mobile. Hence, a more practical approach, that does not require any knowledge on the condition of each channel, is presented in Section III.

C. Computing the Probability for a Successful Transmission

The correlation structure of the loss process of packets in a wireless channel can be modeled with a good approximation by a low order Markovian chain, such as a two state Gilbert model [4], [9]: one state, referred to as state ‘0’, represents an erroneous channel, while the other state, referred to as ‘1’, represents a good channel. Let \(S(n) \in \{0, 1\}\) be the state during slot \(n\). Let \(Prob[S(n+1) = 0 | S(n) = 0] = p\) and \(Prob[S(n+1) = 1 | S(n) = 1] = q\) (see Figure 3). The values of \(p\) and \(q\) can be estimated for each station.

![Fig. 3. Gilbert Model](image-url)
using the statistics the base-station maintains on good and bad transmissions. The following discussion pertains for each specific host \( H \). Let time 0 be the last time when \( H \) transmits any packet, not necessarily of a synchronous call, to the base-station. The base-station knows whether this transmission was good or bad, and needs to compute the probability that the channel is in a bad state at time \( n \) as a function of the channel condition at time 0, that is:

\[
\begin{align*}
\text{(a)} \quad & \text{Prob}\{S(n) = 1 \mid S(0) = 1\} \\
\text{(b)} \quad & \text{Prob}\{S(n) = 1 \mid S(0) = 0\} \\
\end{align*}
\]

Let \( T(n) \) be the probability that the channel is in error state at time \( n \) (i.e. \( T(n) = \text{Prob}\{S(n) = 1\} \)). Hence, we have

\[
T(n + 1) = qT(n) + (1 - p)(1 - T(n))
\]
or equivalently

\[
T(n + 1) = (q + p - 1)T(n) + (1 - p).
\]

The solution for \( T(n + 1) = aT(n) + b \), where \( T(0) = C \) is

\[
T(n) = Ca^n + ba^{n-1} + ba^{n-2} + \ldots + b = Ca^n + b \sum_{i=0}^{n-1} a^i \tag{2}
\]

assuming that \( a \neq 1 \), for \( n > 0 \) we get

\[
T(n) = Ca^n + b\frac{a^n - 1}{a - 1}
\]

and therefore

\[
T(n) = C(p + q - 1)^n + (1 - p)\frac{(p + q - 1)^n - 1}{p + q - 2} \tag{3}
\]

To find \( \text{Prob}\{S(n) = 1 \mid S(0) = 1\} \), we substitute \( C = 1 \) into Eq.(3) and get

\[
\text{Prob}\{S(n) = 1 \mid S(0) = 1\} =
\]

\[
= (p + q - 1)^n + (1 - p)\frac{(p + q - 1)^n - 1}{p + q - 2}. \tag{4}
\]

To find \( \text{Prob}\{S(n) = 1 \mid S(0) = 0\} \), we substitute \( C = 0 \) into Eq.(3) and get

\[
\text{Prob}\{S(n) = 1 \mid S(0) = 0\} =
\]

\[
= (1 - p)\frac{(p + q - 1)^n - 1}{p + q - 2}. \tag{5}
\]

Let \( Err \) be the probability for a bit error in the ‘1’ state. The values of \( p \), \( q \) and \( Err \) can be computed using statistical information from each channel. However, precise measurement is difficult due to the mobility of the hosts. We shall see in Section III that the profit matrix \( M \) can be filled without estimating these values.

III. Assumming No Knowledge on the Channel Condition

The model proposed in Section II-C requires the base station to estimate the value of \( p \) and \( q \) for every host. This computation might be impossible when the channel is unstable, e.g. due to rapid mobility of the hosts. In this section we propose a heuristic that allows the base-station to schedule the packets of each call without any knowledge of \( p \) and \( q \) for the associated host. Rather, the base-station takes into account the status of the channel during the last time a packet was transmitted by the host, regardless whether this packet belongs or does not belong to the considered call.

In the following discussion we consider a given host that needs to transmit a packet \( P \). We start with two observations. First, suppose that the last packet sent by the host on the upstream channel encountered a transmission error at time \( t_0 \). This packet does not necessarily belong to the synchronous call of packet \( P \). It might be, for example, a best-effort packet belonging to another application at the considered host, or a synchronous packet of another call originating at the same host. Suppose that re-transmission is not supported, and we therefore allow the host to transmit each synchronous packet only once. From Eq. 4 follows that

\[
\text{Prob}\{S(n) = 0 \mid S(0) = 1\} = \frac{(p + q - 1)^n - 1}{p + q - 2} \tag{6}
\]

Assuming that \( p + q \geq 1 \) (\( p \) is usually very close to 1, and \( q \) ranges between 0.3 and 0.8), it is clear that this probability increases with the value of \( n \), implying that the maximum will be achieved if packet \( P \) will be scheduled as close as possible to Deadline(\( P \)). On the other, if the previous upstream transmission of the considered host was successful, then from Eq. 5 follows that the packet should be transmitted as close as possible to Release(\( P \)).

Now, suppose that re-transmission is allowed. The observation made above is applicable to the re-transmitted copy. Hence, the optimal timing for the transmission of this copy is as close as possible to Deadline(\( P \)), because the channel was bad during the first transmission. The only issue we still need to address is when should the first transmission take place (\( a \) if the channel is known to be good when packet \( P \) is released, and \( b \) if the channel is known to be bad when packet \( P \) is released). Let Deadline(\( P \)) \( - t_0 = N + 1 \) slots. Let these slots be numbered 0 \( \cdots \) \( N \). Let \( i \) and \( j \) be the time when the first transmission and the second transmission should take place, respectively. Hence, Release(\( P \)) \( \leq i < j \leq \) Deadline(\( P \)). From Eq. 1 follows that we need to maximize the probability that the first transmission is good plus the conditional probability that the second transmission is good if the first transmission encounters an error. Let this sum be represented by \( F_g(i, j) \) if the channel is known to be good before the first transmission takes place, and by \( F_0(i, j) \) if the channel is known to be bad before the first transmission takes place. Hence, we have

\[
F_g(i, j) = \text{Prob}\{S(i) = 0 \mid S(0) = 0\} +
\]

\[
+ \text{Prob}\{S(i) = 1 \mid S(0) = 0\} \cdot \text{Prob}\{S(j) = 0 \mid S(i) = 1\} \tag{7}
\]

\[
F_0(i, j) = \text{Prob}\{S(i) = 0 \mid S(0) = 1\} +
\]

\[
+ \text{Prob}\{S(i) = 1 \mid S(0) = 1\} \cdot \text{Prob}\{S(j) = 0 \mid S(i) = 1\} \tag{8}
\]
We now want to determine the values of \(i\) and \(j\) that maximize \(\mathcal{F}_g(i, j)\) and the values of \(i\) and \(j\) that maximize \(\mathcal{F}_b(i, j)\). By substituting Eq. 4 and Eq. 5 into Eq. 7 we find that \(\mathcal{F}_g(i, j)\) is maximized when \(i = \text{max}(\text{Release}(P), t_0)\) and \(j = \text{Deadline}(P)\). By Substituting Eq. 4 and Eq. 5 into Eq. 8 we find that \(\mathcal{F}_b(i, j)\) is maximized when \(j = \text{Deadline}(P)\). In such a case we have
\[
\mathcal{F}_b(i) = \text{Prob}[S(i) = 0 \mid S(0) = 1] + 
+ \text{Prob}[S(i) = 1 \mid S(0) = 1].
\]
By differentiating \(\mathcal{F}_b(i)\) with respect to \(i\) and equating to 0 we find that
\[
\text{MAX}_{i=\text{Release}(P)}(\mathcal{F}_b(i)) = \begin{cases} 
N/2 & \text{if } N/2 > \text{Release}(P) \\
\text{Release}(P) & \text{else}
\end{cases} \quad (9)
\]
Recall that \(N\) is the number of slots between the previous faulty transmission of any packet by the considered host.

We now extend this result to an arbitrary number of possible re-transmissions.

**Theorem 1:** Suppose that at time \(t\) a packet \(P\) with a deadline is available for transmission. If this packet has not been transmitted before, then \(t\) is the release time. If this packet has been transmitted unsuccessfully, then \(t\) is the time when the failure of the previous transmission is known to both the head-end and the host. Suppose that the last transmission by the same host of any packet, not necessarily \(P\), takes place at \(t_0 \leq t\). Then,

1. If the last transmission (at \(t_0\)) was unsuccessful, then in order to maximize the probability of a successful transmission of \(P\): (a) if only 1 additional transmission of \(P\) is allowed, this transmission should take place at as close as to \(\text{Deadline}(P)\); (b) if \(X \geq 2\) additional transmissions of \(P\) are allowed, the next transmission should take place at \(\text{Max}\{\text{Release}(P), t_0 + \frac{\text{Deadline}(P) - t_0 }{X-1}\}\).

2. If the last transmission (at \(t_0\)) was successful, then, regardless of the number of allowed re-transmissions, in order to maximize the probability of successful transmission of \(P\), the next copy of \(P\) should be transmitted at \(\text{Max}\{\text{Release}(P), t_0\}\).

**Proof:** Part 1(a) of the theorem follows directly from Eq. 6. Let \(\text{Deadline}(P) - t_0 = N\). We prove part 1(b) by induction on \(X\). The induction basis is for \(X = 2\), and it follows from Eq. 9. Assume that the claim holds when \(X - 1\) transmissions are allowed. Let \(\mathcal{F}_b(X, N)\) be the probability for a successful transmission of \(P\) when \(X\) transmissions are allowed and \(\text{Deadline}(P) - t_0 = N\). Assuming that the first transmission takes places at \(t_0 + \alpha\), we have
\[
\mathcal{F}_b(X, N) = \text{Prob}[S(\alpha) = 0 \mid S(0) = 1] + 
+ \text{Prob}[S(\alpha) = 1 \mid S(0) = 1].
\]
For a given value of \(\alpha\), this equation is maximized when \(\mathcal{F}_b(X-1, N-\alpha)\) is maximized. By the induction assumption, the maximum of \(\mathcal{F}_b(X-1, N-\alpha)\) is achieved when the first transmission takes place at \(t_0 + \alpha + (N-\alpha)/(X-1)\), because \(t_0 + \alpha\) is the time when the last transmission took place. Note that \(\text{Release}(P)\) must be earlier than \(t_0\) and hence \(\text{Max}\{\text{Release}(P), t_0 + \alpha + (N-\alpha)/(X-1)\}\) is \(t_0 + \alpha + (N-\alpha)/(X-1)\). Therefore, we have
\[
\mathcal{F}_b(X-1, N-\alpha)_{\text{max}} = 
\text{Prob}[S\left(\frac{N-\alpha}{X-1}\right) = 0 \mid S(0) = 1] + 
+ \text{Prob}[S\left(\frac{N-\alpha}{X-1}\right) = 1 \mid S(0) = 1].
\]
Since both \(\text{Prob}\left[S\left(\frac{N-\alpha}{X-1}\right) = 0 \mid S(0) = 1\right]\) and \(\text{Prob}\left[S\left(\frac{N-\alpha}{X-1}\right) = 1 \mid S(0) = 1\right]\) are constant, it is clear that \(\mathcal{F}_b(X-1, N-\alpha)\) gets its maximum when \(T\left(X-2, \frac{(N-\alpha)(X-2)}{X-1}\right)\) gets its maximum. We can now use the induction assumption once again to find when this happens. By repeating this process \(X - 1\) times we get that \(\mathcal{F}_b(X-1, N-\alpha)_{\text{max}}\) has the same form as found for \(\text{Max}(n)\) in Eq. 2, namely
\[
\mathcal{F}_b(X-1, N-\alpha)_{\text{max}} = Ca^X-1 + b\alpha^X-1 \frac{1 - \alpha}{a-1}. \quad (11)
\]
where \(a = \text{Prob}\left[S\left(\frac{N-\alpha}{X-1}\right) = 1 \mid S(0) = 1\right]\), \(b = \text{Prob}\left[S\left(\frac{N-\alpha}{X-1}\right) = 0 \mid S(0) = 1\right]\), and \(C = \mathcal{F}_b(0, \alpha)\). Substituting this equation into Eq. 10, differentiating it with respect to \(\alpha\), and then equating the result to 0 yield that if \(\text{Release}(P) > t_0 + \frac{N}{X}\) then \(\mathcal{F}_b(R, N)\) gets its maximum at \(\alpha = \text{Release}(P) - t_0\), whereas if \(\text{Release}(P) \leq t_0 + \frac{N}{X}\) then \(\mathcal{F}_b(R, N)\) gets its maximum at \(\alpha = \frac{N}{X}\). This completes the proof of 1(b).

To prove part (2), note that the probability for success if \(X\) transmissions are allowed and the channel is known to be good at \(t_0\) is given by
\[
\mathcal{F}_g(X, N) = \text{Prob}[S(\alpha) = 0 \mid S(0) = 0] + 
+ \text{Prob}[S(\alpha) = 1 \mid S(0) = 0]\cdot \mathcal{F}_b(X-1, N-\alpha). \quad (12)
\]
By substituting Eq. 11 into this equation, we find that the maximum is achieved for \(\alpha = 0\).

As an example for using the results of Theorem 1, consider a packet \(P\) whose release time is \(t_0\). Suppose that the last transmission of the same host before time \(t_0\) was unsuccessful. Assuming that the deadline of the packet is \(t_1\), and that up to \(N\) transmissions are allowed, the best would be to schedule the packet for transmission at \(t_0 + \frac{t_1-t_0}{X}\). Now consider two sub-cases of this scenario:

1. If the same host transmits another packet at \(t \in [t_0, t_0 + \frac{t_1-t_0}{X}]\) successfully, then the optimal time for transmitting \(P\) is shifted to \(t\).
2) If the same host does not transmit another packet successfully during \([t_0, t_0 + \frac{t_1 - t_0}{N}]\), but due to scheduling conflicts the base-station schedules \(P\) for transmission at \(t \neq t_0 + \frac{t_1 - t_0}{N}\), and this transmission is unsuccessful, the next transmission of \(P\) should take place at \(t + \frac{t_1 - t_0}{N}\).

Using these rules, we now show how the profit matrix \(M\) should be updated for every synchronous call \(c\) and every time slot \(t\), without assuming any knowledge regarding the values of \(p, q\) and \(Err\), for the channel of call \(c\). For each packet this algorithm needs only know whether the last transmission on the same channel, not necessarily by the same synchronous call, was successful or not. This algorithm uses a continuous linear increasing or linear decreasing function \(\phi(t)\) whose value is 1 at the optimal slot and 0 at the sub-optimal slot. In addition, this algorithm uses the threshold \(\Delta\), discussed in Section II, in order to avoid transmissions when \(\phi(t)\) is not sufficiently large.

**Algorithm 1 (filling up the profit matrix \(M\)):** Let \(t_0\) be the next time when the scheduling algorithm is invoked. For every active synchronous call \(c\) with a pending packet \(P\) do:

1) Determine the value of the auxiliary variable \(v(P)\), based on the number of bits in packet \(P\), whether or not FEC is used for this packet and call and whether or not the previous packets of the same call were successfully delivered.

2) \(\forall t, t_0 \leq t \leq \text{Deadline}(P), \) set:

\[
M[c, t] \leftarrow v(P) \cdot \phi(t),
\]

where \(\phi(t)\) is computed as follows:

(a) If the last transmission by the same host was successful then according to Theorem 1, the optimal time for transmitting packet \(P\) is as early as possible. Hence, \(\forall t, t_0 \leq t \leq \text{Deadline}(P), \) \(\phi(t) = \frac{\text{Deadline}(P) - t}{t - t_0}\) (see Figure 4(a)).

(b) If the last transmission of the same host, at time \(t_1\) say, was unsuccessful and packet \(P\) can be transmitted at most \(X \geq 1\) additional times, then according to Theorem 1, the optimal time for transmitting packet \(P\) is \(\tau = \max(t_0, t_1 + \frac{1}{X} \cdot \text{Deadline}(P) - t_0\). Hence, \(\forall t, t_0 \leq t \leq \tau, \) \(\phi(t) = \frac{\text{Deadline}(P)}{\tau - t_0}\); \(\forall t, \tau \leq t \leq \text{Deadline}(P), \) \(\phi(t) = \frac{\text{Deadline}(P) - 1}{\text{Deadline}(P) - \tau}\) (see Figure 4(b)).

IV. ALGORITHMS FOR FINDING THE OPTIMAL SCHEDULE IN \(M\)

So far we have concentrated on the way the matrix \(M\) should be configured such that \(M[c, t]\) would reflect the profit of transmitting the packet of synchronous call \(c\) at time \(t\). However, after \(M\) is configured, the base-station needs to run an algorithm for finding an optimal schedule in \(M\) for the following \(T\) slots, where \(T\) is the length of the scheduling interval. Formally, we seek a vector of transmission \(F\) such that \(\text{Profit}(F) = \sum_{t=1}^{T} M[F(t), t]\) is maximum. The development of such an algorithm is orthogonal to the profit based scheduling mechanism presented in this paper. However, for the sake of completeness we discuss in this section potential algorithms in this section.

In the hypothetical case where each synchronous packet fits a single slot, an optimal \(F\) can be found using the concept of maximum matching in a bi-partite graph [3]. The idea is to build a bi-partite graph whose nodes of one set are the active synchronous calls and the nodes of the second set are the time slots. The set of edges is constructed such that each node that represents an active call \(c\) is connected to each node that represents a time interval \(t\). The weight associated with such an edge is \(M[c, t]\).

However, even if we can assume that all of the synchronous packets are of equal length, it is unlikely to assume that this length is equal to the size of one slot. In order to allow good utilization of the channel bandwidth in the case where data packets are transmitted, it is more efficient to use much smaller slots, and to assign multiple consecutive slots to every packet. In [3] we show that in such a case the problem of finding an optimal transmission vector \(F\) is NP-complete. We also present three efficient algorithms for solving this problem. These algorithms are outlined in the rest of this section.

The simplest algorithm is a greedy algorithm referred to as the “maximum local profit algorithm”. To determine which packet should be transmitted at \(t\), this algorithm inspects all the packets whose transmission starting at slot \(t\) would yield some profit. It selects the packet whose normalized profit, i.e. the profit divided by the number of slots \(L\) required for transmitting this packet, if transmitted starting at \(t\), is maximum. This packet is then scheduled to be transmitted during slots \(t, t + 1, \ldots, t + L - 1\). The time complexity of this algorithm is \(O(TC)\) where \(T\) and \(C\) are the dimensions of the matrix \(M\), namely the length of the scheduling interval and the number of active synchronous calls respectively.

A slightly more complicated algorithm is the “maximum global profit algorithm”. Like the “maximum local profit algorithm”, this algorithm is also greedy. However, it makes a greedy decision based on the entire profit matrix, rather than on the information of a given time slot only. The algorithm scans the whole matrix \(M\) and chooses a transmission instance, i.e. a combination of a packet and a sequence of consecutive time slots, with the maximum normalized profit, provided that the selected packet has not been chosen yet and that this transmission does not collide with the transmissions of previously selected packets. This process is repeated until no more profit can be achieved. The time complexity of a naive implementation of this algorithm is \(O(T^2C)\). However, it can be reduced to \(O(T \cdot C \cdot \log(TC))\) using a heap data structure.

A third possible algorithm, referred to as “2-approx”, is rather complex. Due to space constraints we refer the reader to [3]. The time complexity of this algorithm is \(O(T \cdot C \cdot \log(TC))\). This algorithm guarantees in the worst case a profit which is not smaller than half of the maximum possible profit. Such a worst case performance does not exist for the previous two algorithms. However, in practice 2-approx does
not always perform better than the two greedy algorithms. Figure 5 depicts simulation results for the average profit of each algorithm vs. \(C/T\). This profit is normalized to the profit achieved by the 2-approx algorithm. Hence, the normalized profit of the 2-approx is '1' for every value of \(C/T\). As expected, the maximum global profit algorithm is always better than the maximum local profit algorithm. Moreover, this algorithm is better than 2-approx for \(C/T < 4.3\), whereas for larger values of \(C/T\) the 2-approx algorithm performs better.

Recall that during the discussion so far it was assumed that each call has only one pending packet. As already noted, this assumption does not hold when the packetization interval is smaller than the tolerated grant jitter. Figure 6(a) shows a case where the tolerated jitter is \(2/3\) of the packetization interval and Figure 6(b) shows a case where the tolerated jitter is \(2\) times larger than the packetization interval. For example, for a one-way video application, typical values of the packetization interval and of the tolerated jitter are 20 ms. and 2 sec. respectively. Theoretically there might be around 100 pending packets for each call. However, in practice, under “normal” conditions where the channel associated with a particular call is good and the system is not congested, a call will have only one pending packet because every new packet is likely to get transmitted before the successive packet arrives.

There are several approaches to accommodate multiple pending packets per call. In what follows we distinguish between the case where re-transmissions are not allowed (Model 3) and the case were re-transmissions are allowed (Model 4). In the former case, the matrix \(M\) will contain a row for each pending packet of each call rather than a row for each call. It is therefore possible for the scheduler to choose two or even more packets from the same call during the same scheduling interval. Since re-transmissions are not allowed, there is no problem with transmitting the \(i\)'th packet of a call before the host knows whether the \((i-1)\)'th packet has been correctly received. However, since typical synchronous applications require that if two packets are correctly received – their receiving time will be in the same relative order of their transmission time, we need to make sure that the packets are not transmitted out of order. While it is likely that the scheduler will schedule packets from the same call in their original order during the same scheduling interval, since an older packet has a closer due date and therefore a higher profit, this condition is not guaranteed in the general case. This issue is addressed by adapting the scheduler output to the FIFO rule. For instance, if the scheduler determines for some reason that the \(i\)'th packet of a certain call will be transmitted before the \((i-1)\)'th packet, this decision is changed such that the \((i-1)\)'th packet will be transmitted first.

Practically speaking, the scheduler does not have to tell the host which packet should be transmitted. Rather, it sends the host a grant for the considered call. This will be the role of the host to determine which of the call’s pending packets is the next one to be transmitted, e.g. by maintaining a simple FIFO queue and discarding from the head of the queue packets whose deadlines have expired.

The advantage of having an entry in \(M\) for each outstanding packet is that in such a case the scheduler will be able to benefit from periods of time during which the channel is in a good condition after it was in a bad condition. The scheduler will be able to give the host multiple grants during the same scheduling interval, instead of at most one grant per each scheduling interval if each call was represented by a single row in \(M\). However, when re-transmissions are used (Model 4) this is not possible any more. In order to guarantee that packets
of a the same call are received in their original order, the host should not transmit packet $i$ before it knows that packet $i - 1$ was successfully received. Hence, for this model the matrix $M$ will contain a single entry per every call, and this entry will indicate the profit for transmitting the oldest pending packet of this call in every time slot.

Another issue related to Model 4 when a call may have multiple pending packets is determining the re-transmission interval for each packet. If we allow the first pending packet to be re-transmitted during the maximum possible interval, we reduce the period of time during which the next pending packets can be scheduled. However, the following observation explains why such a policy works well.

The difference between the due times of two successive packets is always greater than or equal to the packetization time. By the rules described in Algorithm 1, in the worst case the last transmission of the first pending packet might take place just before the deadline of this packet. If this transmission is bad, the channel is in a bad condition during a long interval and it does not matter which packets were transmitted during this interval. On the other hand, if the transmission is good, the channel is in a good condition and we have a burst of pending packets awaiting transmission. Each of these packets has an “independent tolerated jitter”, i.e. a tolerated jitter that is not affected by the scheduling of previous packets from the same call. This independent tolerated jitter is equal to the packetization time, and is much longer than the packet transmission time. Hence, by the graph in Figure 2, the scheduler is likely to schedule on time the whole burst with no problem.

V. SIMULATION RESULTS

Due to lack of space, we present in this section only the simulation results for Scenario C in Figure 1. This is the scenario where the tolerated jitter is short compared to the length of an error burst, and the load of the synchronous traffic is not necessarily high. Recall that the main motivation for using a smart scheduler in this scenario is minimizing the number of packets that are transmitted while the channel is noisy. As already said, we are not aware of any previous scheme that addresses considerations SC1 (an erroneous channel), SC2 (re-transmission strategy) and SC3 (FEC-based strategy) together. Hence, in this section we compare the results of the proposed quantitative approach without full knowledge of the channel condition (i.e. Algorithm 1), to the performance of a Strawman algorithm which uses EDF policy in order to schedule the first copy of each packet. If the $i$'th copy is lost at time $t$, the Strawman algorithm schedules the $(i+1)$'th copy for transmission at $t + 0.5(\text{Deadline}(P) - t)$ [1].

The synchronous application we consider is a VoIP G.711 codec that generates voice samples at a rate of 64 Kbps, with packetization time of 10ms. Consequently, each packet contains 80 bytes (8KB/s*0.01) of voice samples and 60 bytes of headers. The tolerated jitter of the VoIP packets is 10ms. The scheduling algorithm at the base station is executed every 9ms. Since a new packet is generated every 10ms, each VoIP call has at most one pending packet during every scheduling instance.

In what follows we concentrate upon a the case where only one re-transmission is allowed, FEC is not used and the load of the VoIP calls is only 30%. Figure 7 depicts the fraction of bandwidth consumed by our scheme compared to the Strawman scheme vs. the “locality” of errors, measured as the average time period during which the channel is in an erroneous state. We ran the simulations for several values of average error rates (5%, 10% and 20%). However, the improvement is consistent, depending only upon the locality index. Although not shown in this figure, the number of delivered packets was almost equal (see discussion below).

One of the most important parameters in the quantitative-based algorithm is the threshold $\Delta$ for the probability of a successful transmission. As explained in Section 2, this parameter determines how aggressive is the algorithm when the channel conditions are poor. The results shown in Figure 7 where achieved for $\Delta = 0.01$. In order to measure the effect of $\Delta$ we ran our simulations for different $\Delta$ and locality values. The results are presented in Figure 8. Figure 8(a) shows the percentage number of packets transmitted by the Quantitative algorithm compared to the Strawman algorithm. Figure 8(b) shows the percentage number of packets delivered.
starting from slot $c$, that requires a good knowledge regarding the mechanism are used for each synchronous call. We proposed whether a re-transmission-based and/or a FEC-based recovery maximum expected profit.

VI. CONCLUSIONS

We proposed a generic quantitative-based scheme for scheduling the transmission of synchronous packets over a wireless access channel. This scheme allows the base-station to determine when should every packet be scheduled for transmission. The paper considered the transmission on the upstream channel, but a similar scheme is applicable for the downstream channel as well. The base-station maintains a profit matrix $M$. Entry $[c, t]$ in this matrix indicates the expected profit for scheduling the current packet of call $c$ starting from slot $t$. After the base-station creates this matrix, it executes an algorithm that searches for the schedule with maximum expected profit.

The most important part of the proposed scheme is a profit function that takes into account the status of the channel, and whether a re-transmission-based and/or a FEC-based recovery mechanism are used for each synchronous call. We proposed such a scheme, that requires a good knowledge regarding the statistical behavior of the channel. Then, we extended this scheme to address the case where there is no information regarding the statistical behavior of the channel. In such a case the only information employed by the algorithm is whether the last transmission of a packet over the channel was successful or not.

Depending on the load of the synchronous traffic, the channel condition and the length of the tolerated jitter, the benefit of the proposed scheduling algorithm is three-fold: (a) selecting the most important packets for transmission; (b) increasing the number of synchronous packets that are transmitted on time, and (c) decreasing the number of packets that are transmitted when the channel is noisy.

REFERENCES