

# Providing Quality of Service for Wireless Links in Integrated Wireless/Wired Networks

Norival R. Figueira

norival@nortelnetworks.com

Enterprise Solutions Technology Center  
Nortel Networks  
4401 Great America Parkway  
Santa Clara, CA 95054

Joseph Pasquale

pasquale@cs.ucsd.edu

Computer Systems Laboratory  
Dept. of Computer Science and Engineering  
University of California, San Diego  
La Jolla, CA 92093-0114

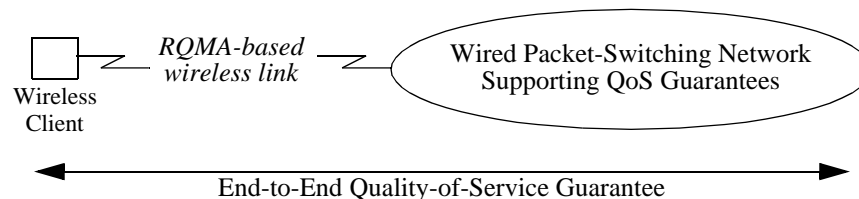
## Abstract

Over the last ten years, a number of scheduling disciplines have been developed to address the problem of providing quality-of-service (QoS) guarantees in packet-switched networks. When one considers the integration of wireless links with these networks, difficulties arise in their implementation because of the relatively large and time-varying bit error rates of wireless links, due to impairments that are difficult to predict. We address this problem at the (wireless) link level, where we have developed a link access scheme called Remote-Queueing Multiple Access (RQMA). RQMA supports the flexible and efficient allocation of bandwidth for real-time sessions in a way that allows the wireless link to be seamlessly integrated with a wired network that supports QoS guarantees. We describe RQMA, and present performance results.

## 1 Introduction

Packet-switching networks are increasingly being required to provide support for traffic types such as voice, video, and other kinds of interactive data. These traffic types, generally called *real-time*, require performance guarantees in terms of throughput, end-to-end delay, and packet loss rate statistics. Most of the research on quality of service (QoS) in wired packet-switching networks has focused on resource reservation and scheduling within switches, without much concern for the bit error rate of the links since it is generally very small and predictable. However, wireless links have a relatively large and time-varying bit error rate due to impairments that are difficult to predict. How to properly integrate wireless links within a wired network that supports QoS guarantees is an important question.

We have developed a new link-access scheme, Remote-Queueing Multiple Access (RQMA), that supports packet transmission over wireless links, allowing the use of scheduling disciplines that have been proposed for wired networks. Importantly, RQMA can be used with *deadline-ordered* [7] scheduling disciplines such as: Delay Earliest-Due-Date (Delay-EDD) [4], Jitter Earliest-Due-Date (Jitter-EDD) [17], VirtualClock [19], Leave-in-Time [6], and Rate-Function Scheduling (RFS) [8]. The end result is that RQMA allows a wireless link to be seamlessly integrated with a wired network that supports QoS guarantees, as illustrated in Figure 1.



**Figure 1:** RQMA allows the QoS guarantees supported by a wired packet-switching network whose routers use deadline-ordered scheduling disciplines to be extended over wireless links.

RQMA assumes a microcell with one base station, called the *base*, and a number of mobile stations, called the *mobiles*. All communications are between a mobile and the base (mobiles do not communicate directly with other mobiles). Mobiles send information to the base about real-time packets that have arrived recently. Enough information is sent, in an efficient and reliable way, so that a scheduling discipline executed by the base can establish a transmission schedule in accordance with a QoS provision (such as delay bounds). The base then sends transmission *permits* to mobiles according to this schedule. Packets that are destined for mobiles within its cell are simply broadcast by the base.

To counteract the wireless link's channel errors, RQMA allows some portion of the link bandwidth to be reserved for the retransmission of real-time packets that are not transmitted correctly. Real-time packets are retransmitted until: (1) they are received correctly, or (2) their deadlines are violated, or (3) there is not enough link capacity to keep trying more retransmissions. This error control is in addition to the use of forward error correction (FEC), which by itself (using even more FEC bits) is significantly less effective than their combination.

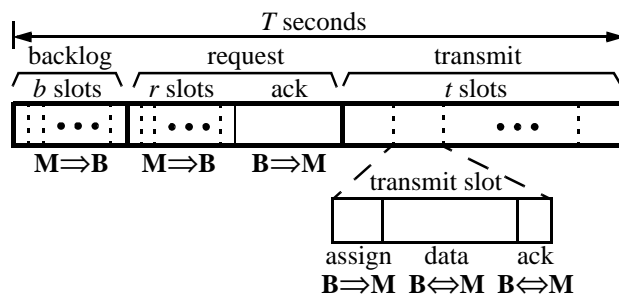
The remainder of this paper is organized as follows. Section 2 describes the basic algorithms of RQMA. In Section 3, the performance of RQMA is demonstrated through simulation experiments. Section 4 presents some possible extensions of RQMA. In Section 5, we show how RQMA can extend the real-time service guarantees of a wired network to include wireless links. In Section 6, we discuss related work, and in Section 7 we present conclusions.

## 2 Remote-Queueing Multiple Access

### 2.1 RQMA Overview

We define a *session* simply as a sequence of related packets. A *real-time* session is one for which the packets have (transmission) deadlines. A *continuous bit rate (CBR)* session is one whose packets are generated at a constant rate, and whose deadlines are known a priori to be spaced at equal intervals. While CBR is actually a special case of real-time, we consider it separately. Finally, a *best-effort* session is one whose packets do not have deadlines.

To analyze the access protocol, we divide the time axis into frames of length  $T$  seconds. Each frame is subdivided into three fields: *request*, *transmit*, and *backlog* (Figure 2). The request field is subdivided into  $r$  request slots and an acknowledgment (ack) subfield. The transmit field is subdivided into  $t$  transmit slots, each composed of assign, data, and ack subfields. Finally, the backlog field is subdivided into  $b$  backlog slots.



**Figure 2:** The structure of a frame.  $M \Rightarrow B$  means transmission from mobile (M) to base station (B), and vice-versa for  $B \Rightarrow M$ . The result of the transmit field may depend on the backlog field of the same frame; hence, the backlog field comes at the beginning to give the base more time to process its contents before the first transmit slot assignment.

Mobiles send *requests* to the base to either establish real-time or CBR sessions, or to send one or more best-effort packets. There are four types of requests (described further in Section 2.1.1): Alloc\_real-time, Alloc\_CBR, Alloc\_best-effort, and Cancel (applies to both real-time and CBR). Requests are sent in request slots using a random access protocol such as slotted Aloha. When the base successfully receives a request, it sends an acknowledgment in the request field's ack subfield.

Once a session has been established, the base decides when a mobile can send or receive the actual data packets of a session. Using a transmit slot's assign subfield, the base assigns the data subfield to a particular session, indicating which mobile is to make use of it. A mobile may only transmit a packet in the data subfield if permission was given. The transmit slot's ack subfield is used to acknowledge successfully received<sup>1</sup> data packets by a mobile or the base, depending on the direction of the transmission indicated in the transmit slot's assign subfield.

A backlog slot is associated with each established real-time session of a mobile. Backlog slots are used to inform

the base about newly arrived packets of real-time sessions at a mobile. This information is used by the base's packet scheduling algorithm to determine when the mobile should transmit these packets.

Table 1 summarizes the contents of the various RQMA fields.

**Table 1: Contents of RQMA Fields.**

Fields	Contents
Backlog slot	sequence number of first, deadline of last, number of packets, new_info flag
Request slot	mobile identifier, type, sequence last   rate   session identifier (“ ” means “or”)
Request: Ack subfield	$r$ mobile identifiers
Transmit: Assign subfield	session identifier, sequence number, direction, first_of_frame
Transmit: Data subfield	data bits, deadline for real-time/CBR   piggy-back for best-effort
Transmit: Ack subfield	acknowledgment

Parameters  $r$ ,  $t$ , and  $b$  are separately configurable for different implementations of RQMA, and have the following implications. The number of backlog slots  $b$  defines the maximum number of real-time sessions that can be established. The number of request slots  $r$  is selected taking into consideration the maximum number of mobiles (usually, increasing  $r$  decreases the average number of collisions of requests). The number of transmit slots  $t$  is the prime contributor to the frame length and the overhead imposed by RQMA. A smaller value of  $t$  implies a smaller frame length but a larger RQMA overhead. Larger values of  $r$  and  $b$  also imply a larger RQMA overhead. Section 3 provides a sample configuration for these parameters.

### 2.1.1 Requests

All requests carry a mobile identifier and a request type. Besides these parameters, the best-effort request also carries either the last sequence number of best-effort packets, the desired rate for a CBR session, or a session identifier for a real-time session.

**Real-time allocation:** An Alloc\_real-time request is used by a mobile to establish a real-time session. After receiving (and acknowledging) an Alloc\_real-time request, the base executes an admission control procedure required by its scheduling algorithm. The admission control procedure verifies that there are enough resources (e.g., link capacity) to accept this new real-time session without violating the QoS guarantees given to all other established real-time and CBR sessions. Once established, the base allocates a backlog slot number to the real-time session. Information exchange during the admission control procedure as well as informing a mobile about its backlog slot number allocation is carried out using a higher layer protocol (further discussion of such protocols is beyond the scope of this paper).

**CBR allocation:** An Alloc\_CBR request is used when a mobile wants to establish a CBR session. Alloc\_CBR requests are processed in the same way as Alloc\_real-time requests. After accepting the session, the base regularly grants some transmit slots for the mobile to transmit CBR packets. Since the base knows the rate of packet transmission, CBR sessions do not use backlog slots.

**Best-effort allocation:** An Alloc\_best-effort request is used by a mobile to transmit best-effort packets. After receiving (and acknowledging) an Alloc\_best-effort request (which may request the transmission of multiple packets), the base station eventually allocates transmit slots for the transmission of these packets.

In this paper, we assume that a mobile station has only one uplink best-effort session, which is automatically established, and which is composed of all the best-effort traffic generated at the mobile. Thus, a best-effort request

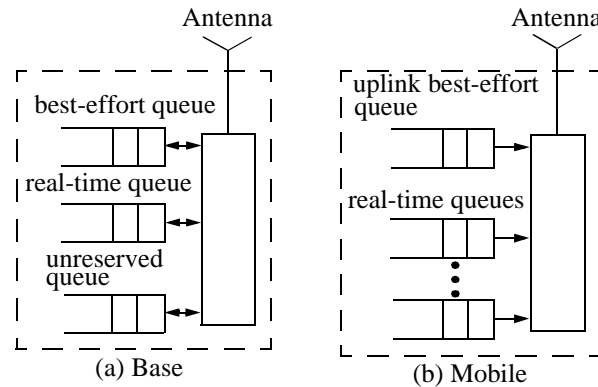
<sup>1</sup> By “successfully received,” we mean that correct data was recovered after FEC is processed at the receiver. In this paper, we assume that FEC processing occurs in parallel with the reception of a packet.

implicitly identifies this “session”. This assumption is made to simplify the description and operation of RQMA; however, RQMA can be easily extended to deal with the general case of more than one best-effort session per mobile.

**Cancel request:** A mobile cancels a real-time or a CBR session by sending a Cancel request.

### 2.1.2 Base Station Operation

The base maintains three queues: a *real-time/CBR queue*, a *best-effort queue*, and an *unreserved queue* (Figure 3a). The real-time/CBR queue keeps track of real-time and CBR packets, the best-effort queue keeps track of best-effort packets, and the unreserved queue is used to deal with transmission errors (explained in Section 2.2.1). These queues keep track of *local* packets, which are stored at the *base* waiting to be transmitted to mobiles, and *remote* packets, which are stored at *mobiles* waiting to be transmitted to the base. Note that the contents of remote packets are not actually stored in these queues, but references to them are.



**Figure 3:** Queues at a (a) base station and (b) mobile station. The mobile’s queues are used for uplink transmissions, whereas the base’s queues are for both uplink and downlink.

The real-time/CBR queue is sorted according to packet deadlines. The deadlines of real-time packets are calculated by the base if the direction of the real-time session is from base to mobile; otherwise, they are calculated by the mobile and sent to the base using backlog slots. Deadlines of real-time packets are calculated based on a real-time scheduling discipline. The base also assigns deadlines to CBR packets, calculated based on the (fixed) rate of the session using the formula:

$$\text{deadline\_of\_next\_packet} = \text{deadline\_of\_previous\_packet} + \text{packet\_length} / \text{CBR\_session\_rate}.$$

Local packets are queued for downlink transmission as soon as they arrive at the base. Remote real-time packets are queued for uplink transmission after the base receives arrival information in backlog slots. For remote CBR packets, queueing for uplink transmission is done periodically and automatically based on the known rates of the established CBR sessions. Finally, remote best-effort packets are queued for uplink transmission after the base receives Alloc\_best-effort requests.

The real-time/CBR queue is served using the *earliest deadline first* policy: transmit packets in increasing order of deadline (ties are ordered arbitrarily). The base only serves the best-effort queue when the real-time/CBR queue is empty.<sup>2</sup> RQMA does not define how the best-effort queue is scheduled. For example, one can use first-come first-served or round robin (for the latter, the base has a queue for each best-effort session).

To “serve” a packet, the base uses a transmit slot’s assign subfield to indicate which mobile should make use of the corresponding data subfield, and the direction of transmission (mobile to base, or vice-versa). A mobile can transmit a packet in a data subfield only if permission was given by the base in the corresponding assign subfield. A packet is considered successfully transmitted only if the receiver sends an ack in the corresponding ack subfield.

### 2.1.3 Mobile Station Operation

A mobile maintains a queue for each of its uplink real-time sessions (a mobile can have more than one real-time session) and one queue for its uplink best-effort session (Figure 3b). The mobile listens to and inspects all the transmit

<sup>2</sup> In this case, real-time sessions must be rate controlled to avoid indefinite starvation of best-effort sessions. Alternatively, one could allocate some small part of the (wireless) link capacity to best-effort sessions and treat them as real-time to guarantee that they will not suffer from indefinite starvation.

slots' assign subfields to determine whether it should use the corresponding data subfield to transmit (or receive) a packet of data. The assign subfield is composed of a session identifier, the sequence number of the packet, direction of the transmission, and a binary flag called first-of-frame (used to compensate for transmission errors, and discussed in Section 2.2.1). The session identifier is unique among all mobiles; thus, there is no need to explicitly identify the mobile.

For each established real-time session, the mobile transmits (in every frame) the following information in the corresponding backlog slot (recall that a real-time session is allocated a backlog slot when the session is established): the sequence number of the first packet of that session that arrived during the previous frame, the number of packets of that session that arrived during the previous frame, the deadline of the last packet of that session that arrived during the previous frame, and the `new_info` bit.

The binary flag `new_info` is used to indicate if a backlog slot contains new information. `New_info` is set to 0 if there were *no* new packet arrivals during the previous frame. In this case, the other fields of the backlog slot contain the same information of the previously transmitted backlog slot. Otherwise, `new_info` is set to 1 indicating there were one or more new packet arrivals during the previous frame, and the rest of the backlog slot contains new information. The `new_info` flag is useful when backlog slots are lost due to transmission errors (see Section 2.2.2).

Every time the base receives a backlog slot with `new_info` equal to 1, it includes references for all the (remote) packets that arrived during the previous frame in the real-time queue. The deadlines of all of these packets are set to the deadline of the one that arrived last (as indicated in the backlog slot). Since these packets are only available for transmission after the backlog slot is received by the base, a mobile needs to add one frame length  $T$  in the deadline calculations of all real-time packets (since the base will not know for at least one frame length what the new arrivals are). Otherwise, the scheduling algorithm executed at the base may not be able to guarantee that all real-time packets will be transmitted before their deadlines (assuming, of course, no transmission errors). Because of this, the delay bounds of all uplink real-time sessions are increased by one frame length. Note that the deadlines of downlink real-time packets do not need to be increased by one frame length since the base already knows about all the newly arrived (downlink) packets.

For uplink transmission of a best-effort packet, the mobile must ask the base for a transmit slot by sending an `Alloc_best-effort` request either in a request slot, or by piggy-backing the request in the uplink transmission of one of its best-effort packets before the next selected request slot arrives. A piggy-back request is considered successful if the data packet carrying the piggy-back request is acknowledged by the base. When the base receives a best-effort request, it assigns sequence numbers to all the new (remote) packets since the last best-effort request, and includes these packets in its best-effort queue.

## 2.2 Transmission Errors

RQMA assumes that some form of FEC is used to reduce the average packet error rate (see Section 3 for an example). However, FEC cannot be used exclusively without incurring excessive overhead. This section describes how RQMA deals with unrecovered transmission errors.

### 2.2.1 Errors in Transmit Slots

RQMA uses sequence numbers to deal with data transmission errors. Each session is assigned a sequence number integer variable that increments by one unit for every transmitted packet of that session.

A real-time or CBR session can be considered a numbered sequence of packets. The (transmission) sequence number of a real-time or CBR packet is equal to the number of the packet within the session. Real-time and CBR packets may be dropped either due to buffer overflows (although this should be avoided by the proper reservation of resources) or due to deadline violations. Thus, sequence numbers of dropped packets will be skipped. ARQ (automatic repeat request) protocols are *not* used for dropped real-time or CBR packets.

All real-time packets also carry their deadlines (deadlines of CBR packets are implicit). The deadline of a successfully received packet is used by the base or mobile to determine when previously missed packets are lost. When the deadline of a successfully received packet is about to expire, all packets with smaller sequence numbers are considered lost.

Real-time scheduling disciplines proposed for wired networks typically assume that all transmissions are successful: after the transmission of a packet, no retransmissions will occur. If retransmissions were possible, the scheduling discipline might not be able to insure QoS guarantees. To support retransmissions, while still maintaining our goal of using existing real-time scheduling disciplines, a separate session is established called the (*real-time*) *retransmission session*. This session reserves some part of the available bandwidth of the link to support the retransmission

of real-time and CBR packets.<sup>3</sup> Thus, if the transmission of a real-time or CBR packet is not successful, the packet is assumed to have just arrived as part of the retransmission session. A new deadline for this packet is calculated. If the new deadline is *later* than the original deadline, the packet is dropped. Otherwise, the packet is kept in the real-time/CBR queue *with its original deadline*, and can be retransmitted *without violating the QoS guarantee of the original session*. This works because loosening up the deadlines of the packets of a session (in this case, the retransmission session) does not affect the schedulability of a service discipline [7].

Besides the above retransmission scheme for real-time packets, a backup scheme can be used for real-time or CBR packets whose deadlines are not yet violated but cannot be considered for the retransmission session (i.e., because there is not enough reserved retransmission bandwidth). In this scheme, the base includes those packets in the unreserved queue (recall that this is one of the three queues maintained by the base). This queue is sorted by deadlines and works just like the real-time/CBR queue. However, the unreserved queue is served as if it were a best-effort queue, after the real-time/CBR queue is emptied.

If a mobile does not receive an ack after transmitting a real-time or CBR packet, the mobile will not know if the base successfully received it. In this case, the mobile will have to wait until the deadline of the packet expires before it can delete the packet from its queue.

For best-effort packets, if a mobile does not receive an ack after transmitting, the following algorithm is used to delete old (transmitted) packets from its best-effort queue. The *first-of-frame* binary flag in a transmit slot's assign subfield is used to indicate whether the slot is the first being used for the corresponding session in the present frame. If this flag is turned on, all of the session's packets that were transmitted with sequence numbers smaller than that of this slot's packet were received by the base and can be deleted from the mobile's queue.

Unlike real-time or CBR sessions, a sequence number can be used repeatedly until some best-effort packet is successfully delivered using this sequence number. Thus, a mobile (or base) is allowed to drop best-effort packets (e.g., due to buffer overflow) without changing the sequence number for the session. RQMA does not define which, if any, ARQ protocol should be used for dropped best-effort packets.

Finally, if the assign subfield of a transmit slot is not correctly received, a mobile cannot make use of the corresponding data subfield. This kind of error will generate missed data packets, and is addressed by the schemes above.

## 2.2.2 Errors in Backlog Slots

In a frame where a mobile has nothing new to inform about a real-time session, the mobile repeats the same information from the previous backlog slot with `new_info` set to 0. This allows the base to miss a backlog slot that has `new_info` set to 1, and still be able to serve the packets. The mobile does not keep track of which backlog slots the base was able to receive. However, the base must make the necessary adjustments to the deadlines of packets in case it misses a backlog slot, as we explain next.

The base keeps track of the number of consecutive missed backlog slots of a session. Once the base successfully receives a backlog slot, it computes the number of packets it may have missed (using the received sequence number), and assigns as the deadline of *all* the missed packets the received deadline. If the received backlog slot has `new_info` set to 1, the new packets informed in this backlog slot are processed as usual.

If only one backlog slot was missed and the last received backlog slot has `new_info` set to 0, the deadline assigned to the missed packets is the correct one. Otherwise (if either the number of missed backlog slots is greater than one or the last received backlog slot has `new_info` set to 1), the assigned deadline may be larger than the original ones. Since this may increase the delay bound of these packets, the base assigns an expiration time to all of these packets after which they will be dropped. The expiration time guarantees that these packets will not be delivered with a delay larger than the delay bound of the session, and is calculated assuming that the first missed backlog slot had `new_info` set to 1. In this case, if the last received backlog slot has `new_info` set to 1, the expiration time is equal to the assigned deadline plus the "latency" of the scheduling discipline (discussed below) minus one frame length for every missed backlog slot. If the last received backlog slot has `new_info` set to 0, the expiration time is equal to the previous amount plus one frame length.

The latency of a scheduling discipline, which depends on the discipline, is the maximum difference between the delay guarantee given to a packet and the deadline of the packet. For example, in Delay-EDD this amount is equal to zero, and in VirtualClock this amount is equal to one packet length divided by the capacity of the link.

We also need to guarantee schedulability [7]. Thus, (remote) packets originated from missed backlog slots are assumed to be from the retransmission session (as defined in Section 2.2.1). Thus, their original deadlines are compared to their deadlines in the retransmission session. If a packet's original deadline is before its deadline in the

<sup>3</sup>. RQMA does not define whether the reserved amount is fixed or not.

retransmission real-time session, the packet is dropped.

The base needs a mechanism to deallocate backlog slots. RQMA assumes that the base periodically broadcasts data packets containing backlog slot confirmations and deallocations. A mobile can only use a backlog slot for a predefined interval of time after the last confirmation. If a backlog slot was deallocated, the mobile sends an acknowledgment in the backlog slot, and stops further use of it. This ack allows the base to reassign the backlog slot before the predefined timeout period. This algorithm guarantees that a mobile will eventually release a backlog slot even if it does not receive a deallocation from the base. Note that a CBR session can be deallocated by the base by simply denying the allocation of data slots.

### 3 Simulation Experiments

In this section, we evaluate the performance of RQMA through simulation experiments. Since only uplink real-time sessions require the use of the backlog field, we only simulate uplink sessions.

#### 3.1 Traffic Source Models

We use two kinds of traffic sources: ON-OFF and Poisson. ON-OFF sources have been used extensively in recent studies [6, 18] since they can be used to model standard voice sources. In our simulations, all real-time sessions are modeled as ON-OFF traffic sources and all best-effort sessions are modeled as Poisson sources.

**ON-OFF traffic sources:** An ON-OFF traffic source is modeled here as a two-state Markov modulated process. In the ON state, packets are generated at fixed intervals of time  $\tau$ . In the OFF state, no packets are generated. The duration of the ON and OFF states are exponentially distributed with mean  $a_{ON}$  and  $a_{OFF}$ , respectively. The number of packets generated in the ON state is approximated by a geometric distribution with mean  $a_{ON}/\tau$ .

We simulated ON-OFF traffic sources with  $a_{ON} = 352$  ms and  $a_{OFF} = 18.5$  ms. Although a more typical value for  $a_{OFF}$  for a standard voice source is 650 ms [18], we use  $a_{OFF} = 18.5$  ms (also one of the values used in [18]) to increase the link utilization and, consequently, to limit the amount of spare bandwidth available. In our simulations, we set  $\tau = 13.25$  ms, which implies that the generation rate is 32 kbits/s in the ON state, since we use ATM-like packets of 424 bits.

**Poisson traffic sources:** The interarrival time of packets for these traffic sources is exponentially distributed. We use a value of 100 ms for the mean interarrival time to avoid large backlogs of best-effort packets, since our simulations try to allocate most of the capacity of the link to real-time sessions.

#### 3.2 Scheduling Discipline

We use VirtualClock [19] as the packet scheduling discipline at the base because it is simple and easy to implement, while still providing end-to-end delay bounds to real-time sessions given the reservation of a lower bound of bandwidth [5]. All real-time sessions reserved a rate of 32 kbits/s in the wireless link.

#### 3.3 Transmission Errors

We use the two-state Gilbert-Elliot Markov model [3, 9], which is widely used to describe fading channels. This model describes a stochastic sequential machine (SSM) that generates a binary output sequence. In this sequence, a 1 (one) indicates that one bit was transmitted with error, while a 0 (zero) indicates the correct transmission of a bit.

This SSM has two states: a “good state” and a “bad state.” In the good state, the probability that the SSM will generate a 1 is  $p_0$ , while in the bad state this probability is  $p_1$ . The SSM makes a transition from the good state to the bad state with probability  $p_b$ , and vice-versa with probability  $p_g$ . Define the average error rate  $P_{av}$  as the probability that the next output of the SSM is equal to 1 (see [11] for a derivation):

$$P_{av} = \left( \frac{p_g}{p_b + p_g} \right) p_0 + \left( \frac{p_b}{p_b + p_g} \right) p_1 .$$

We simulated two kinds of channels: a slow fading channel and a fast fading channel. The fading parameters we use are similar to those used in [11]. For the slow fading channel:  $p_b = 10^{-6}$  and  $p_g = 3 \times 10^{-6}$ . For the fast fading channel:  $p_b = 0.2$  and  $p_g = 0.6$ . For both kinds of channels,  $p_0 = 10^{-6}$  and  $p_1 = 3.9997 \times 10^{-2}$ . This results in a fairly high average error rate  $\hat{P}_{av}$  of  $10^{-2}$  for both channels.

### 3.4 Forward Error Correction

FEC is generally used to reduce errors in wireless communications; we use FEC in our simulations of RQMA. We use the Bose-Chaudhuri-Hocquenghem (BCH) codes and the tables in [14] to calculate the number of FEC (overhead) bits. To simplify our simulations, we assumed that all errors are either corrected or detected.

We experimentally determined the point at which increasing the number of FEC bits, which reduces the usable bandwidth, produced diminishing returns in reducing the packet loss rate. Determining the number of FEC bits is complicated in that some frame bits are more important than others, and so one must also determine which frame bits they should correct. In the Appendix section, we present an analysis on how the number of FEC bits for data packets was determined.

Table 2 shows the length of all the RQMA parameters and the number of FEC bits used in the simulations.

**Table 2: Parameters Used in Simulation Experiments.**

Field	Parameter Sizes (bits)	FEC Bits (Error-corrected Bits)	Total Bits
Backlog Slot	sequence first (24), number of packets (8), deadline last (16), new_info_slot (1)	54 (8)	103
Request Slot	mobile identifier (8), type (2), sequence last   rate (24)	12 (2)	46
Request: Ack subfield	mobile identifier (8) * $r$ ( $r = 20$ )	15 (2)	175
Transmit: Assign subfield	session identifier (9), sequence number (24), direction (1), first_of_frame (1)	45 (7)	80
Transmit: Data sub- field	data bits (424), deadline   piggy-back best-effort (24)	91 (10)	539
Transmit: Ack subfield	ack (7)	8 (2)	15

### 3.5 Frame and Link Parameters

We assumed a microcell with a radius of 450 m, and a link capacity of 2 Mbits/s. We configured our frames to contain 40 backlog slots, 20 request slots, and 100 transmit slots. To prevent uplink signals from different mobiles from overlapping, a *guard time* must be provided for each separate uplink communication. The guard time must be at least equal to the maximum difference in the round-trip radio propagation delay between any two mobiles in the same cell, which is approximately 3  $\mu$ s given our microcell size. Finally, we use 10 bits for synchronization of packets. Given these parameters, the frame time is approximately 37 ms.

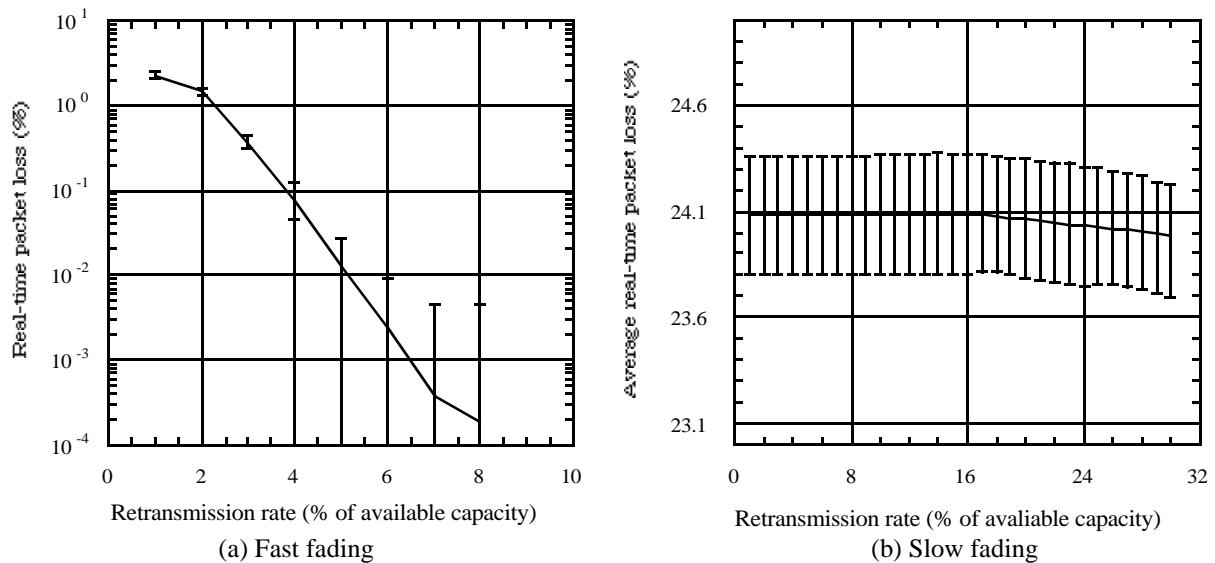
### 3.6 Simulation Results

We present the results of an experiment that uses 33 mobiles, 33 best-effort sessions, and 25 real-time sessions. Figure 4 shows the percentage of real-time packets that are lost, i.e., that are not successfully transmitted before their deadlines, versus the amount of bandwidth that is reserved for the retransmission of real-time packets. Recall that the channel's average error rate is  $10^{-2}$ .

Figure 4 expresses the reserved retransmission rate in terms of the available capacity of the link, i.e., the amount of bandwidth that is left for the transmission of payload data after all the overhead is taken into account. The available capacity is about 1.15 Mbits/s, of which about 30.4% (or 350 Kbits/s) is available for retransmissions. The rest of the available bandwidth (about 800 Kbits/s) is reserved for the 25 real-time sessions. To isolate the contribution of the bandwidth reservation for retransmission of real-time packets, the experiments shown in Figure 4 are for a base sta-

tion that is *not* using the unreserved queue (as defined in Section 2.1.2).

For the fast fading channel, bandwidth reservation for retransmission provides a good reduction in the packet loss rate of  $10^{-5}$  (or  $10^{-3}\%$  as shown in Figure 4a). However, for the slow fading channel, bandwidth reservation for retransmission did not provide a reduction in the packet loss rate even at high levels of reservation. This is because retransmissions are ineffective when the channel fades for long periods of time, as does our simulated slow fading channel. In this case, real-time packets are dropped (because their deadlines expire) before they can be successfully transmitted. While RQMA cannot deal with this problem, we expect that any access protocol that is exposed to this kind of slow fading channel and high average channel error rate will experience a similar problem.



**Figure 4:** Average real-time packet loss (as a percent of all packets) versus the reserved retransmission rate (as a percent of available capacity) for (a) a fast fading channel, and (b) a slow fading channel, with a channel average error rate of  $10^{-2}$ . The bars indicate the maximum and the minimum packet loss over all real-time sessions.

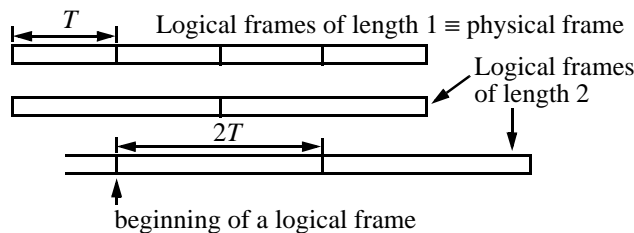
In the Appendix section, we carry out similar experiments for channels with lower average error rates, ranging from  $10^{-3}$  to  $10^{-5}$ . In these cases, the results were as good as (and generally much better than) those in Figure 4a, for *both* fast and slow fading channels. Under these better conditions, RQMA is capable of delivering real-time traffic with a reasonably small average packet loss rate, on the order of  $10^{-5}$ .

## 4 Extensions

### 4.1 Logical Frames

Since the number of real-time backlog slots in a frame is finite, the maximum number of real-time sessions that can share a base is also finite. To allow the number of real-time sessions to exceed the number of backlog slots in a frame, one can associate real-time sessions with *logical frames* instead of physical frames. A logical frame is composed of an integral number of consecutive physical frames.

For example, if logical frames are composed of two physical frames, half of the logical frames beginning at odd physical frame boundaries and the other half beginning at even physical frame boundaries, we can have twice as many real-time sessions sharing the base. However, this scheme has the drawback that logical frames are longer than physical frames. This can be resolved by using a hierarchy of logical frames where the maximum number of logical frames that begins at any given physical frame is at most equal to the number of backlog slots of a physical frame. In this case, sessions can choose from a set of logical frames with various lengths. For example, a backlog field with two backlog slots can be used for three logical frames: one logical frame with one physical frame and two logical frames with two physical frames each (see Figure 5).



**Figure 5:** A hierarchy of logical frames.

## 4.2 Time Slots, Codes, and Frequency Slots

This paper presents RQMA based on *time slot* allocation. However, nothing in RQMA dictates that these slots must represent time. Consequently, RQMA can easily be generalized to allocate frequency slots or codes (as in CDMA).

## 5 Providing Real-Time Services in an Integrated Wireless-Wired Network using RQMA.

Applications such as interactive voice and video communicate real-time traffic that require performance guarantees in terms of throughput, end-to-end delay, and packet loss rate statistics. These performance guarantees may be provided by real-time communication services of the network. To provide such services, network nodes (routers or switches) employ a certain type of scheduling discipline. As mentioned before, several such scheduling disciplines have been recently proposed for wired networks, including Delay-EDD [4], Jitter-EDD [17], VirtualClock [19], Leave-in-Time [6], and RFS [8]. All of these scheduling disciplines are deadline-ordered and can be used with RQMA.

Consider a network that provides real-time communication services and that has wireless links employing RQMA. Consider an application that has throughput and end-to-end delay requirements for transmitted packets (packet loss requirements will be discussed later). The application can use a signaling protocol such as the one defined for RSVP [20] to request these performance guarantees from the network. The network then employs an *admission control procedure* to verify if it can provide the desired performance guarantees given the path of the new flow.

The admission control procedure of the network needs to take into account the current resource commitments in all the network nodes in the path of the flow. This is done by the execution of *admission control tests* for every network node in the path of the flow. These admission control tests are specific to the deadline-ordered service discipline used by the network node. With respect to delay and throughput bounds, *the admission control tests for a wireless link using RQMA are the same as the ones the network node would use for a wired link*. However, the actual delay bound of the network node is increased by one frame length due to the RQMA algorithm.

Besides requiring throughput and end-to-end delay bounds, some real-time applications may also require bounds on packet loss rate. Unfortunately, wireless links suffer from relatively large and time varying bit error rates due to impairments that are difficult to predict. Consequently, guarantees on packet losses are much harder to achieve for wireless links. However, that this is not an intrinsic problem of RQMA, as all transmission schemes for wireless links suffer from this problem. Therefore, integrated wireless-wired networks can only support applications that tolerate packet losses. This is the case with voice applications which can accept a small percentage of lost packets (usually less than 1%) without severely impacting user-perceived voice quality.

Although the above example uses the integrated services (IntServ) approach defined by the Internet Engineering Task Force (IETF), where network nodes provide hard real-time guarantees, RQMA can also be used in networks that use the Differentiated Services (DiffServ) approach [2].

DiffServ tries to address the drawbacks of the IntServ architecture, which requires flow state to be saved in every router traversed by a flow, by moving complex tasks to the edge of the network, and by operating only on traffic aggregates in the core network devices. DiffServ networks classify packets at the boundaries of the network to a small number of aggregated flows. In a network node, each aggregated flow is forwarded according to the per-hop behavior (PHB) that was assigned to the aggregate.

The expedited forwarding PHB [12], which can be implemented with a simple strict priority queue, can be used to provide a service that resembles a leased line. This service can be appropriate for some real-time applications such as Internet telephony.

RQMA can be used to integrate wireless links into a wired network employing the DiffServ architecture. One way to support the expedited forwarding PHB is to use a single real-time flow per mobile to aggregate all the

mobile's "leased line flows." For all packets of the aggregated leased line flows of all the mobiles, RQMA can use a fixed and predefined delay bound. In this case, the delay bound is used to regulate retransmission attempts, and work simply as an expiration time for the packets. Since all packets have the same deadline, the scheduling discipline of the base's real-time queue reduces to the first-in-first-out (FIFO) scheduling discipline, which greatly simplifies the implementation of the base station.

Note that DiffServ also defines traffic conditioners ("shapers" and "policers") that should be used in a per-flow basis at edge routers, i.e., routers connected to end systems. If the base station is an edge router (which is a reasonable assumption), it will need to provide traffic conditioning. In the above example, traffic conditioning can only be provided on a per-mobile basis, since the base station receives a single aggregated leased line flow from a mobile. This seems to be a reasonable constraint since a mobile will usually serve a single user. Optionally, one can use as many aggregated leased line flows per mobile as required to separate the traffic flows from the same user or from different users. This, of course, consumes more backlog slots in a frame.

## 6 Related Work

RQMA builds on ideas found in Centralized Packet Reservation Multiple Access (C-PRMA) [1], Distributed-Queueing Request Update Multiple Access (DQRUMA) [13], and the work described in [16]. RQMA can be compared to C-PRMA, which supports "periodic" (voice) traffic and "random" (data) traffic sources. In C-PRMA, periodic sources contend for wireless channel time slots using random access, and then reserve future slots (similar to RQMA CBR sessions). C-PRMA enhances Packet Reservation Multiple Access (PRMA) [10] [15] by assigning to the base station a central function in scheduling the transmission of mobiles. This allows packet retransmissions and different delay constraints to be more easily dealt with. However, unlike RQMA, C-PRMA and PRMA do not support more general real-time traffic, where packets may have arbitrary deadlines and must be scheduled despite distributed information.

In DQRUMA, mobiles send transmission requests to the base, which broadcasts transmission permits. RQMA uses a similar request-permission scheme. However, RQMA differs by providing explicit support for real-time and CBR sessions; once these sessions are set up, packet delivery does not involve a random access protocol. Also, RQMA uses backlog slots for real-time sessions. A real-time session is allocated a backlog slot in every frame to allow it to send information about its newly arrived packets, which is used by the base's scheduling discipline to determine the transmission schedule of real-time packets and support performance guarantees.

A "dynamic TDMA" scheme is described in [16], where a TDMA frame is subdivided into request slots and message slots. Each message slot provides for the transmission of an ATM-like packet. Request slots are comparatively short and are used for initial access in slotted ALOHA contention mode. RQMA uses a similar framing structure. However, although RQMA defines a frame structure to allow for periodic request intervals, it is *not* based on TDMA since all data packet transmissions are explicitly controlled and assigned by the base in every message slot, leading to a more flexible bandwidth allocation that increases efficiency.

## 7 Conclusions

We presented a new scheme called Remote-Queueing Multiple Access (RQMA) that supports QoS guarantees over wireless links. The novel ideas in RQMA lie in how real-time (and CBR) sessions are supported, and how it deals with errors. For real-time sessions, RQMA uses a backlog field so that allocation of link bandwidth is flexible and efficient. Regarding errors, some minimum portion of the link bandwidth is reserved for the retransmission of packets that are not received correctly but whose deadlines are still not violated.

We presented the basic algorithms of RQMA and presented a performance evaluation of RQMA through simulation experiments. RQMA is capable of delivering real-time traffic with a small average packet loss rate (on the order of  $10^{-5}$ ) for fast or slow fading channels with an average error rate of  $10^{-3}$  or lower. Even with an average error rate of  $10^{-2}$ , the retransmission scheme of RQMA is still able to reduce the average packet loss rate to  $10^{-5}$  for a fast fading channel. However, for a slow fading channel with an average error rate of  $10^{-2}$ , neither FEC nor retransmission of packets are able to effectively reduce the average packet loss rate. This is because the slow fading channel fades for long periods of time, and real-time packets are dropped (because their deadlines expire) before they can be successfully transmitted.

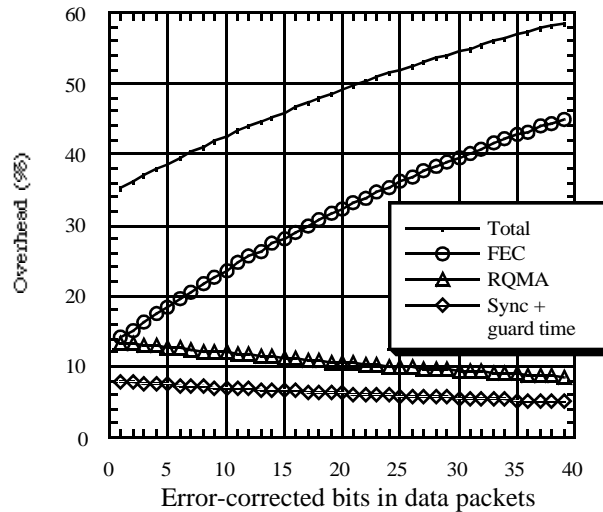
Based on these results, we believe that RQMA is a feasible solution for integrated wireless/wired networks that carry real-time traffic.

## References

- [1] G. Bianchi, F. Borgonovo, L. Fratta, L. Musumeci, M. Zorzi, "C-PRMA: The Centralized Packet Reservation Multiple Access for Local Wireless Communications," *Proc. IEEE GLOBECOM '94*, Vol. 3, pp. 1340-1345, November 1994.
- [2] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An Architecture for Differentiated Services", IETF RFC 2475, December 1998.
- [3] E. O. Elliot, "Estimates of Error Rates for Codes on Burst-Noise Channels," *Bell System Technical Journal*, Vol. 42, pp. 1977-1997, September 1963.
- [4] D. Ferrari and D. Verma, "A Scheme for Real-Time Channel Establishment in Wide-Area Networks," *IEEE Journal on Selected Areas in Communications*, Vol. 8, No. 4, pp. 368-379, April 1990.
- [5] N. R. Figueira and J. Pasquale, "An Upper Bound on Delay for the VirtualClock service Discipline," *IEEE/ACM Transactions on Networking*, Vol. 3, No. 4, pp. 399-408, August 1995.
- [6] N. R. Figueira and J. Pasquale, "Leave-in-Time: A New Service Discipline for Control of Real-Time Communications in a Packet-Switching Network," *Proc. ACM SIGCOMM '95*, pp. 207-218, August 1995.
- [7] N. R. Figueira and J. Pasquale, "A Schedulability Condition for Deadline-Ordered Service Disciplines," *IEEE/ACM Transactions on Networking*, Vol. 5, No. 2, pp. 232-244, April 1997.
- [8] N. R. Figueira and J. Pasquale, "Rate-Function Scheduling," *Proceedings IEEE INFOCOM '97*, pp. 1065-1074, April 1997.
- [9] E. N. Gilbert, "Capacity of a Burst Noise Channel," *Bell System Technical Journal*, Vol. 39, pp. 1253-1265, September 1960.
- [10] D. J. Goodman, R. A. Valenzuela, K. T. Gayliard, and B. Bamamurthi, "Packet Reservation Multiple Access for Local Wireless Communications," *IEEE Transactions on Communications*, Vol. 37, pp. 885-890, August 1989.
- [11] N. Guo and S. D. Morgera, "Frequency-Hopped ARQ for Wireless Network Data Services," *IEEE Journal on Selected Areas in Communications*, Vol. 12, No. 8, pp. 1324-1337, October 1994.
- [12] V. Jacobson, K. Nichols, and K. Poduri, "An Expedited Forwarding PHB," IETF RFC 2598, June 1990.
- [13] M. J. Karol, Z. Liu, and K. Y. Eng, "Distributed-Queueing Request Update Multiple Access (DQRUMA) for Wireless Packet (ATM) Networks," *Proc. International Conference on Communications*, pp. 1224-1231, June 1995.
- [14] S. Lin, "An Introduction to Error-Correcting Codes," Englewood Cliffs, Prentice-Hall, New Jersey, 1970.
- [15] P. Narasimhan and R. D. Yates, "A New Protocol for the Integration of Voice and Data over PRMA," *IEEE Journal on Selected Areas in Communications*, Vol. 14, No. 4, pp. 623-631, May 1996.
- [16] D. Raychaudhuri and N. D. Wilson, "ATM-Based Transport Architecture for Multiservices Wireless Personal Communication Networks," *IEEE Journal on Selected Areas in Communications*, Vol. 12, No. 8, pp. 1401-1414, October 1994.
- [17] D. Verma, H. Zhang, and D. Ferrari, "Delay Jitter Control for Real-Time Communication in a Packet Switching Network," *Proc. IEEE TriCom '91*, pp. 35-43, April 1991.
- [18] D. Yates, J. Kurose, D. Towsley, and M. G. Hluchyj, "On Per-session End-to-End Delay Distributions and the Call Admission Problem for Real-Time Applications with QoS Requirements," *Proc. ACM SIGCOMM '93*, pp. 2-12, September 1993.
- [19] L. Zhang, "VirtualClock: A New Traffic Control Algorithm for Packet Switching Networks," *ACM Transactions on Computer Systems*, Vol. 9, No. 2, pp. 101-124, May 1991.
- [20] L. Zhang, et al., "Resource reSerVation Protocol", IETF RFC 2205.

## Appendix

Consider how increasing the number of error-corrected bits in data packets affects performance. Here we assume the same number of error-corrected bits as presented in Table 2, except for those used for data packets (contained in the Transmit Data subfield), which we vary from 1 to 39. Roughly, each error-corrected data bit requires 9-10 FEC bits. While increasing the number of error-corrected bits reduces errors, the resulting number of FEC bits increases overhead. Figure 6 shows the overhead (for an entire RQMA frame as defined in Table 2) versus the number of error-corrected bits in data packets. With 39 error-corrected bits, just FEC overhead is 45% of the link capacity.



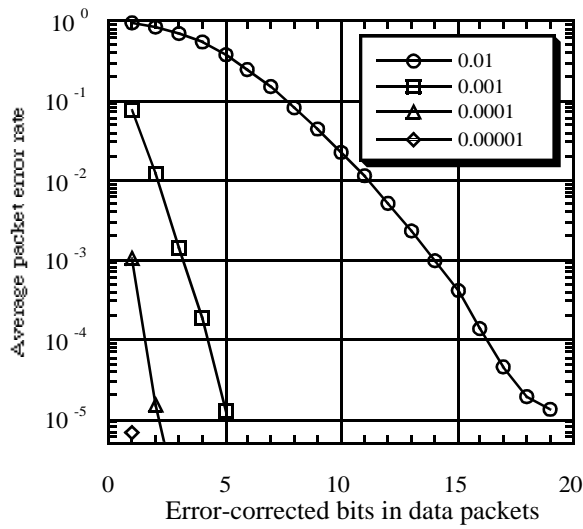
**Figure 6:** Total overhead of FEC and RQMA versus the number of error-corrected bits of data packets. The overhead also depends (inversely) on the frame length, which increases with the number error-corrected bits.

In the following experiments, the number of mobiles, real-time sessions, and best-effort sessions is the same for all experiments (see Table 3). We vary the number of mobiles to compensate for the varying amount of available bandwidth due to FEC overhead.

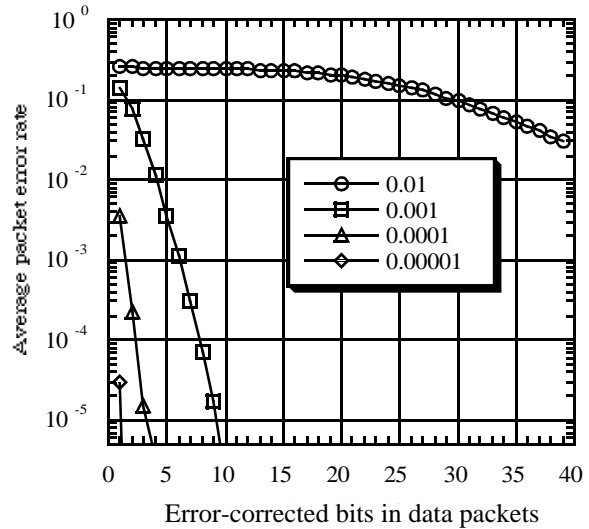
**Table 3: Error-corrected bits versus the number of mobiles.**

Error-corrected bits	Number of mobiles
1-16	33
17-22	30
23-31	28
32-38	26
39	25

Figure 7 shows the average packet error rate (over all kinds of packets) versus the number of error-corrected bits in data packets for fast fading and slow fading channels, respectively, for channel average error rates varying from  $10^{-2}$  to  $10^{-5}$  (which are obtained by varying  $p_1$  from  $3.9997 \times 10^{-2}$  to  $3.7 \times 10^{-5}$ , respectively). For both kinds of channels, 10 error-corrected bits are sufficient to achieve an average packet error rate of  $10^{-5}$  for channel average error rates ranging from  $10^{-3}$  to  $10^{-5}$ . For a channel with an average error rate of  $10^{-2}$ , the number of error-corrected bits must be significantly higher to reduce the average packet error rate. For a fast fading channel, at least 16 error-corrected bits are necessary to achieve an average packet error rate below  $10^{-4}$ .



(a) Fast fading

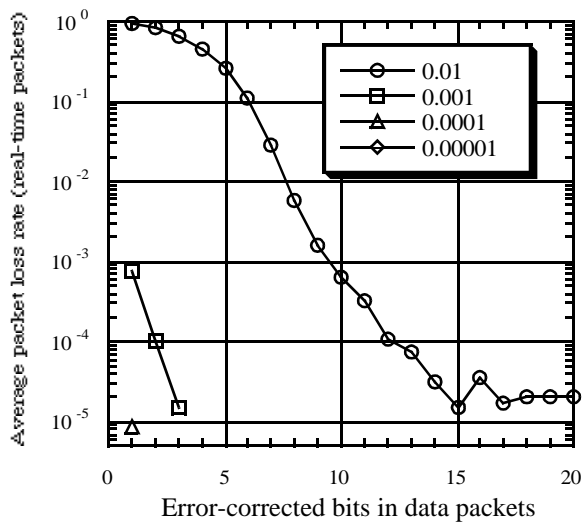


(b) Slow fading

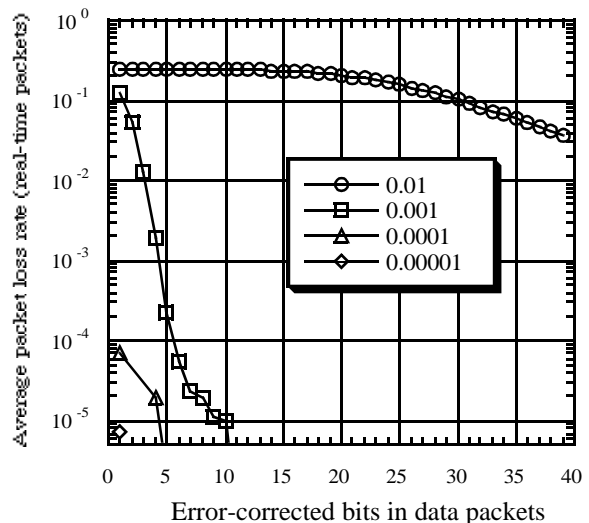
**Figure 7:** Packet error rate vs. number of error-corrected bits in data packets for (a) a fast fading and (b) a slow fading channel, for channel average error rates ranging from  $10^{-2}$  to  $10^{-5}$ .

For a slow fading channel, even 39 error-corrected bits are not sufficient to reduce the average packet error rate below  $10^{-2}$ . The problem with slow fading channels is that fades are relatively long and concentrate many bit errors in single packets, thus requiring many error-corrected bits per data packet to recover all the errors. The fast fading channel is less sensitive to this problem because fades are shorter and do not concentrate as many bit errors in a single packet.

Figure 8 shows the average packet loss rate of *real-time* packets versus the number of error-corrected bits in data packets for a fast fading channel and a slow fading channel with average channel error rates ranging from  $10^{-2}$  to  $10^{-5}$ . This shows how real-time packets are affected by the transmission errors of the channel. For either a fast fading or slow fading channel with average channel error rates below  $10^{-3}$ , RQMA with a small number of error-corrected data bits performs well. For a slow fading channel, the same is true except for an average channel error rate of  $10^{-2}$ , where even large numbers of error-corrected bits cannot reduce the packet loss rate.



(a) Fast fading



(b) Slow fading

**Figure 8:** Average packet loss rate of real-time packets versus the number of error-corrected bits in data packets for (a) a fast fading channel and (b) a slow fading channel, with channel average error rates varying from  $10^{-2}$  to  $10^{-5}$ .