A Unified Wireless LAN Architecture for Real-Time and Non-Real-Time Communication Services

Sunghyun Choi, *Member, IEEE* and Kang G. Shin, *Fellow, IEEE*

***Abstract*—This paper addresses how to support *both* real-time and non-real-time communication services in a wireless LAN with dynamic time-division duplexed (D-TDD) transmission. With D-TDD, a frequency channel is time-shared for both downlink and uplink transmissions under the dynamic access control of the base station. The base station 1) handles uplink transmissions by polling mobiles in certain order determined on a per-connection (per-message) basis for transmitting real-time (non-real-time) traffic from mobiles and 2) schedules the transmission of downlink packets. To handle location-dependent, time-varying, and bursty errors, we adopt the channel-state prediction, transmission deferment, and retransmission. We consider the problems of scheduling and multiplexing downlink packet transmissions, and polling mobiles for uplink transmissions depending on the channel state. We also establish conditions necessary to admit each new real-time connection by checking if the connection's delivery-delay bound can be guaranteed as long as the channel stays in good condition without compromising any of the existing guarantees. Last, the performance of the proposed protocol is evaluated to demonstrate how the protocol works and to study the effects of various parameters of the protocol.**

***Index Terms*—Admission tests, dynamic time-division duplexing (D-TDD), location-dependent errors, MAC protocol, polling, priority scheduling, QoS-sensitive communication, wireless LAN.**

I. Introduction

WIRELESS LAN's are emerging as an attractive alternative, or complement, to wired LAN's [2], [25] because they enable us to set up and reconfigure LAN's easily without incurring the cost of wiring. WLAN's are characterized as high-speed wireless systems that cover relatively small geographical areas, as compared to other wireless systems such as cellular, PCS, and mobile data radio systems. WLAN's are expected to be the solution to the problem of meeting the growing demand for mobile clients to have access to the existing high-speed wired networks. As the need for broadband multimedia communications involving digital audio and video grows, it is increasingly important for communication systems to support various traffic with quality-of-service (QoS) guarantees.

Depending on its distinct characteristics and QoS requirements, diverse traffic is categorized into two classes as summarized in Table I: 1) class-I real-time traffic such as voice and video that requires bounded delays, but is usually tolerant of some packet losses; and 2) class-II traffic like the conventional data services that requires loss-free transmission, but requires no bounded delay. Class II can be divided further into two subclasses: a) class II-A, which is delay-sensitive like FTP and remote log-in; and b) class II-B, which is delay-tolerant like paging and e-mail. Class II-A is given priority over class II-B. Real-time communication service deals with the first traffic class, in which the packet delivery delay is bounded at the cost of some packet losses depending on the time-varying channel condition, while non-real-time communication service is for the second traffic class. In this paper, we consider a unified WLAN architecture, composed of a base station (BS) and a number of mobiles, which provides both real-time and non-real-time communication services.

TABLE I
CLASSIFICATION OF HETEROGENEOUS TRAFFIC

	Class I	Class II-A	Class II-B
Name	real-time	non-real-time	
Examples	voice & video	remote login	e-mail
Delay	bounded	sensitive	tolerable
Loss	loss-tolerant	zero-loss	

The integrated communication services considered here can be divided into four types: 1) downlink real-time, 2) uplink real-time, 3) downlink non-real-time, and 4) uplink non-real-time services, where the downlink (uplink) is for transmitting BS-to-mobile (mobile-to-BS) traffic. To guarantee the timely delivery of packets, real-time traffic is handled by a connection-oriented service, while non-real-time traffic is handled by a connection-less service. Dynamic time-division duplexed (D-TDD) transmission is used, and hence, the same frequency channel is time-shared for downlink and uplink transmissions under the dynamic access control of the BS.

Due to the different nature of uplink and downlink transmissions, they are treated differently. For the downlink, the BS schedules the transmission of those packets that had already arrived at the BS. On the other hand, the exact status of uplink packets in mobiles is not known to the BS. Thus, mobiles request the BS for permission to transmit their uplink packets, which, upon reception of the requests, schedules uplink transmission permissions. These requests are made on a *per-connection* basis for the real-time service and on a *per-message* basis for the non-real-time service, where a message is composed of a number of packets. The BS can give a mobile a packet-transmission permission by *polling* the mobile. Real-time traffic is given priority over non-real-time traffic for its timely delivery.

Manuscript received October 9, 1998; revised May 25, 1999; recommended by IEEE/ACM TRANSACTIONS ON NETWORKING Editor I. F. Akyildiz. This work was supported in part by the U.S. Department of Transportation under Grant DTFH61-93-X-00017.

S. Choi was with The University of Michigan, Ann Arbor, MI 48109 USA. He is now with Philips Research, Briarcliff Manor, NY 10510 USA (e-mail: sunghyun.choi@philips.com).

K. G. Shin is with the Real-Time Computing Laboratory, Department of Electrical Engineering and Computer Science, The University of Michigan, Ann Arbor, MI 48109 USA (e-mail: kgshin@eecs.umich.edu).

Publisher Item Identifier S 1063-6692(00)01438-2.

The WLAN system should provide:

1) timely delivery of real-time packets while minimizing packet losses;
2) virtually error-free transmission of non-real-time packets;
3) good average delay and throughput performance of non-real-time traffic utilizing the bandwidth left unused by real-time traffic;
4) fair usage of a channel for non-real-time traffic among mobiles;
5) low latencies for non-real-time packet transmissions and real-time connection setup, and in handling handoff requests.

We design a medium access control (MAC) protocol that can support these requirements, the scheduling of uplink and downlink packet transmissions, and the admission control of newly requested real-time connections in a dynamic environment with location-dependent and bursty errors.

This paper is organized as follows. Section II states the specification and assumptions of the wireless network under consideration. In Section III, we describe the MAC protocol to support both real-time and non-real-time traffic. Section IV presents the run-time scheduling of real-time and non-real-time traffic, and the admission-control scheme for real-time connection requests. The proposed protocol is evaluated in Section V. Section VI discusses related work, putting our protocol in a comparative perspective. This paper concludes with Section VII.

II. Assumptions and System Specifications

Before presenting our protocols, we state the assumptions and specification of the WLAN under consideration.

A. Network Specification

The WLAN of our interest is based on an infrastructure that is composed of a wired backbone network and a (possibly large) number of base stations, which cover the whole geographic area in service. Each BS handles an area, called a *cell*, where all the mobiles in the cell communicate through the BS. In this system, the downlink and uplink in a cell can respectively be thought as the end-most and front-most links of a multihop end-to-end connection. Since wireless links usually have much less bandwidth than the wired counterpart, the former might become the bottleneck of the end-to-end communication performance. We will, in this paper, focus on the uplink and downlink transmissions within a single cell.

Each mobile is assumed to transmit packets with some limited transmission power so as to reach the BS only, which is located at the "center" of the cell. So our system differs from peer-to-peer communication systems, which do not require any infrastructure, e.g., *ad hoc* networks. The lower the transmission power, the better, since 1) mobile devices operate with batteries and 2) the transmitted signals in a cell are nothing but inter-cell interferences to other cells. In this sense, it is desirable to transmit packets with as low a power as possible, which may, unfortunately, cause the *hidden* terminal problem [2]. In Fig. 1, the transmission ranges of mobiles 1 and 2 do not allow them to hear each other, but both can be heard by the BS in between. Mobiles 1 and 2 are hidden terminals to each other. So,

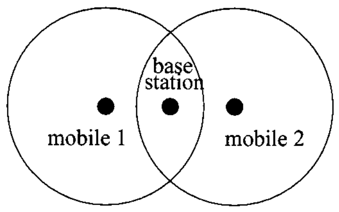


Fig. 1. The hidden terminal problem.

in a cell, the uplink (mobile-to-BS) is not a broadcast channel while the downlink (BS-to-mobile) is. Hence, mobiles are assumed not to listen directly to one another even if some can, depending on their relative locations. When a mobile wants to send a packet, regardless whether the packet is destined for another mobile in the same cell or a mobile or static node outside the cell, it first sends the packet to its BS. The BS will then forward the packet to its final destination, sometimes via a wired backbone network, i.e., when the packet is destined for a node outside the cell. When the BS transmits packets through the broadcast downlink, all but the destination mobile in the cell ignore them. Note that only the BS can determine if a collision has occurred in the uplink channel. In summary, the system under consideration can be considered as a network with the *star topology* whose center is the BS.

B. Dynamic Time-Division Duplexed Transmission

Dynamic time-division duplexed transmission is used in the system, i.e., a wireless channel over a frequency slot is time-shared for both downlink and uplink transmissions under the BS's dynamic access control. We could instead use: 1) frequency-division duplexed (FDD) transmission as in most cellular systems, in which two different frequency channels are allocated for uplink and downlink or 2) static TDD in which a portion (usually a half) of each time frame is allocated for the uplink and the remaining portion is for the downlink. Even though these static duplexing transmissions are simpler, and adopted in most currently available cellular systems, D-TDD is shown to offer more efficient link utilization in case of unbalanced and time-varying uplink and downlink traffic [8]. Considering the growing number of applications involved with unbalanced two-way traffic (e.g., web browsing), D-TDD is a very promising and important design choice.

C. Location-Dependent Errors and Their Control

It is well known that a wireless channel is inherently unreliable due mainly to noises, interferences, fadings, etc. We use a *hybrid* of forward-error correction (FEC) and automatic repeat request (ARQ). That is, the receiver tries to correct errors first, and if unsuccessful, a decoding failure is declared, implying the detection of a packet error. A selective-repeat ARQ is used, so the receiver sends an ACK or NAK for each received packet according to the decoding result, and only NAKed packets will be retransmitted by the sender. However, real-time packets can be retransmitted only for a limited time due to their timing constraints or deadlines, which should be met by appropriately scheduling packet transmissions. Wireless channels, in many cases, are known to experience

TABLE II
COMPARISON OF REAL-TIME AND NON-REAL-TIME COMMUNICATION SERVICES IN THE PROPOSED PROTOCOL

	Real-Time	Non-Real-Time
Connection	Connection-oriented	Connection-less
Request	Per-connection based	Per-message based
Admission Test	Yes	No
Scheduling	Deadline-driven	Round-robin-variant

location-dependent, time-varying, and bursty errors, switching back and forth between good and bad states over time [12], [13], [29], [30]. A *channel* is defined between each mobile and the BS. Channel probe and transmission deferment are adopted to handle the above-mentioned types of errors, i.e., before transmitting each real-time packet over a channel, the channel condition is probed through a hand-shaking mechanism, and if the condition is bad, the transmission is deferred, and a different packet is scheduled for transmission. This in turn reduces the need of retransmissions significantly. Since the prediction of channel condition/state cannot be perfect, a packet transmission may fail. Deferred or erroneously transmitted real-time packets must be scheduled for (re)transmission before their deadline in a best effort manner so that they will not compromise the other undeferred packets' deadline guarantees. We assume that certain channel coding is used for the receiver to detect and correct errors in packets it received. However, the MAC protocol should provide the means of channel probing and feedback from the receiver to the sender for possible retransmissions.

Throughout the rest of this paper, we will ignore the packet-propagation delay, since it is usually small relative to the other delay components like queueing and transmission delays in a cell[1] Last, all packets, like ATM cells, are assumed to be of the same fixed size. So, a long message is fragmented into a number of packets of identical size at the transmitter. With a selective-repeat ARQ scheme, fragmentation could be effective. Otherwise, a long message received in error could result in the retransmission of the entire message, wasting a significant amount of network resources.

III. PROTOCOL DESCRIPTION

This section describes the proposed protocol that supports both real-time and non-real-time communication services. In this protocol, due to their different QoS requirements, real-time and non-real-time packets are treated differently. Table II summarizes how the two communication services are differentiated and supported. Throughout this and the next sections, we give a detailed account of each entry in the table. We will first outline the protocol and then present in the next section the details of both the scheduling and admission-control schemes.

A. Downlink Versus Uplink

A BS has full control of the transmission of all downlink and uplink packets within its cell. For the downlink, the BS just schedules the transmission of the packets, which had already arrived at the BS. In contrast, the exact number of pending uplink packets in each mobile is not known to the BS. Mobiles request the BS's permission to transmit uplink packets, and upon reception of these requests, the BS schedules uplink packet transmissions. These requests are made on a per-connection basis for the real-time service, and on a per-message[2] basis for the non-real-time service as explained below. The BS gives a mobile packet-transmission permission by *polling* the mobile, i.e., checking if it has a packet to transmit. That is, the BS schedules the order of both polling and downlink packet transmissions.

B. Real-Time Versus Non-Real-Time

The real-time communication service is provided through connection-oriented communication. Real-time traffic like audio and video usually arrives regularly or periodically and a real-time session/connection usually lasts for a long time, as compared to randomly arriving non-real-time traffic. Moreover, it is desired to guarantee the requested delivery-delay bound for *each* real-time packet, and hence, we need to set up a real-time connection. For a connection to be admitted, a request should be made to the BS with the connection specification including the requested delay bound. Then, the BS performs an admission test to decide if it is possible to guarantee the requested delay bound without violating the existing connections' guarantees. At run-time, the BS performs priority-based scheduling on packets to guarantee the required delay bounds for all admitted real-time connections provided their channels stay in good condition. A connection is defined to be *unidirectional*, i.e., downlink or uplink. For the sake of notational simplicity, it is assumed that an *active* mobile has only one connection even though it is usually expected to have at least two connections (i.e., one for uplink and the other for downlink) at a time. Active mobile i with downlink (uplink) real-time connection i is called downlink (uplink) real-time mobile i. non-real-time communication service, on the other hand, is connectionless (or best effort). Whenever a non-real-time packet arrives at the BS, its transmission is scheduled by the BS. If an uplink message arrives, or is generated at a mobile, the mobile should request the BS's permission to transmit the message; the BS then schedules the transmission permission for that mobile. Some form of scheduling is also needed to give each mobile fair access to the wireless link. Use of the wireless link is prioritized; when there is a pending schedule to poll mobiles for uplink real-time packets or there are "eligible" (to be defined later) downlink real-time packets, the BS will continue to poll mobiles for uplink real-time packets or transmit downlink real-time packets. However, whenever there is no such a polling schedule nor any downlink real-time packet to transmit, the BS will transmit downlink non-real-time packets or poll mobiles for the transmission of uplink non-real-time packets. As will be described later, the prioritized channel usage becomes more complicated when location-dependent errors must be handled efficiently.

C. Slotted Channels

A frequency channel is slotted along the time axis with three types of slots: *packet-transmission slot, transmission-request*

[1]A cell in this paper refers to a *microcell*, which has coverage of a few hundred meters, or a *picocell*, which covers small indoor areas [25].

[2]A message, in this paper, is referred to as a number of non-real-time packets, which arrive at the same time.

slot, and *control mini-slot*. For simplicity, it is assumed that both packet-transmission and transmission-request slots have the same size T_s while the control mini-slot has the size $T_{\rm ms}(= T_s/K)$, where K is an even number much larger than one. First, a (fixed-size) downlink or uplink packet is transmitted in a packet-transmission slot. Within a packet, the following information can be piggybacked:

- information within a downlink packet to give a mobile permission to transmit uplink packets;
- ACK/NAK for a previously received packet within a downlink or uplink packet;
- a message-transmission or connection-setup request by a mobile within an uplink packet.

Second, the transmission-request slot is used by mobiles to convey a request to the BS. Specifically, the usage is threefold.

- A non-real-time message transmission can be requested with (mobile ID, Class II-A/B ID, the number of packets in the message).
- A real-time connection-setup request can be made with (mobile ID, connection specification).
- A handoff request can be made by a mobile, which is communicating with an adjacent BS in the overlapping region of two cells, with (previous cell ID, mobile ID, connection specification).

Last, using the control mini-slot, a control packet is transmitted by the BS or by a mobile. A control packet can be used for the following cases:

- for the BS to poll a mobile for transmission of an uplink packet;
- for the BS to issue a transmission-request slot by announcing that the next slot is a transmission-request slot;
- for the BS to announce the admission-test result for a newly requested or handed-off real-time connection;
- for a real-time mobile to return the transmission permission to the BS without transmitting any packet;
- for the BS or a mobile to send an ACK/NAK for each received packet;
- for the BS to send or receive a channel-probing control packet to predict the channel condition a mobile is experiencing.

More than one of the above can be piggybacked in a control packet, and only the last three cases use uplink control packets.

D. Transmission Request Access

As shown in Fig. 2, a transmission-request slot is divided into two parts: 1) the first half is the set of $K/2$ *request mini-slots* used by mobiles and 2) the second half is the set of $K/2$ *result-announcing mini-slots* corresponding to the previous request mini-slots. Then, the first $K_{\rm ho}$ request mini-slots are reserved for real-time handoff requests. The request mini-slots are used by a slotted ALOHA-like random access protocol: when the BS issues a transmission-request slot, each mobile with pending requests decides whether to make a request with probability p_r or not (i.e., in a p_r-persistent manner). If it decided to send the request, the mobile randomly chooses one of the last $(K/2 - K_{\rm ho})$ request mini-slots, then sends it in that chosen mini-slot. One difference with real-time handoff requests is that they are made through one of the first $K_{\rm ho}$ request mini-slots. The result of each of $K/2$ mini-slots can be success[3], or collision, or empty/unused. Using each of the next $K/2$ downlink mini-slots, the result of the corresponding request mini-slot is announced.

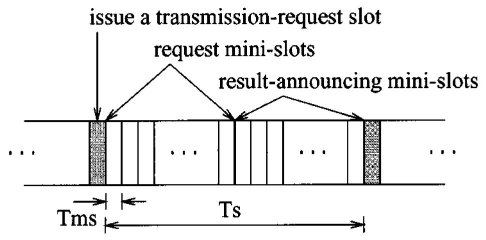

issue a transmission-request slot
request mini-slots
result-announcing mini-slots
Tms
Ts

Fig. 2. The structure of a transmission-request slot.

The reason why $K_{\rm ho}$ mini-slots are reserved for real-time handoff requests is because without this reservation, the handoff requests, which should be made in a timely manner to guarantee the connection's deadline, might be swamped by many requests for non-real-time traffic transmissions. If any of $K_{\rm ho}$ reserved mini-slots results in collision, all request mini-slots in the following transmission-request slot are dedicated to real-time handoff requests (i.e., $K_{\rm ho} = K/2$) so that the collided handoff requests can be made successfully with a much higher probability. The initial value of $K_{\rm ho}$ can be adaptively set according to the handoff rates. For example, if there was a collision in a reserved request mini-slot, the size can be increased by one (and this new value is used after the request slot with $K_{\rm ho} = K/2$ as described above), then if there is no such collision for a certain time period, it can be decremented by one.

The request access probability p_r is determined by each mobile independently of others. For the first attempt of a message-transmission request, p_r is set to 1. If the request becomes unsuccessful, then p_r is decreased according to

$$p_r := \frac{p_r}{p_r + 1} \tag{1}$$

thus following the sequence 1, 1/2, 1/3, ... (i.e., the "harmonic" backoff) [10]. Each unsuccessful request should be retransmitted p_r-persistently until it is successfully transmitted. Note that there are two ways to transmit a request: one through a transmission-request slot and the other by piggybacking it in an uplink packet. Which of the two ways should be used depends on the request type. First, for a non-real-time message-transmission request, if a mobile has not yet transmitted all of its pending uplink packets of which transmission requests had already been made to the BS, then, upon arrival of another message, the mobile need not use a contention-based transmission-request slot. Instead, the request can be piggybacked in an uplink packet. Second, for a real-time connection-setup request, the mobile transmits the request using either of two ways, whichever becomes available first. Third, a handoff request can be made only through a transmission-request slot since the BS is unaware of the existence of that mobile. In this case, the mobile will access transmission-request slots in a 1-persistent manner. Due to the limited packet size, one may not be able to piggyback all pending requests in a single uplink packet.

[3]Due to the *capture effects* [2], a request can be transmitted successfully even in the presence of concurrent requests from other mobiles.

In such a case, priority is given in the order of: 1) real-time connection-setup request; 2) class II-A message-transmission request; and 3) class II-B message-transmission request.

E. Error-Handling Mechanisms

To handle location-dependent, time-varying, and bursty channel errors, the channel state can be predicted via channel probing before a packet is transmitted. That is, before polling a mobile or transmitting a downlink packet to the mobile, the BS transmits a probing control packet to the mobile, which then returns the control packet to the BS. If the BS does not receive the probing control packet correctly from the mobile, the channel is considered bad, and the polling or transmission is deferred. The channel condition is estimated before each real-time packet transmission in order to reduce the need for retransmitting real-time packets, since retransmission can be really harmful in meeting deadlines while it is optional for the non-real-time case (to reduce its significant overhead).

The ACK/NAK is sent from the receiver upon packet reception.[4] The most desirable way to send an ACK/NAK is to piggyback it in another packet. ACK/NAK for an uplink packet can always be piggybacked since after each uplink packet transmission, a downlink packet (or a control packet) will always be transmitted from the BS. Note that a downlink (control) packet and the piggybacked ACK/NAK may have different destination mobiles. The packet transmitted after a downlink packet cannot be a regular uplink packet from the destination mobile of the previous downlink packet because the destination mobile transmits an ACK/NAK via an uplink control packet right after it received a downlink packet. If the destination mobile of a downlink packet does not know the packet's destination due to an error in the packet, then the mobile will not transmit NAK either. If the BS does not receive an ACK/NAK within T_{ms} after transmitting a downlink packet, the packet will be assumed lost and hence retransmitted later. In summary, 1) ACK/NAK for each packet is transmitted right after its reception, 2) ACK/NAK for an uplink packet is always piggybacked in a downlink (control) packet, and 3) after a downlink packet transmission, a control packet containing ACK/NAK is transmitted from the destination mobile unless an uplink packet transmission from that mobile is scheduled next.

When a mobile is polled for a real-time (non-real-time) packet transmission, if it does not have any real-time (non-real-time) uplink packet to transmit,[5] the mobile transmits an uplink control packet, saying that it has no packets to transmit. So, if the BS does not receive any (control) packet within the mini-slot period T_{ms}, the polling control packet had not been received by the mobile due to an error, implying that the mobile's channel is bad, and hence the polling is deferred. The issue of handling this deferment is addressed next.

[4]Actually, right after determining if the received packet can be corrected by the channel decoder.

[5]Even though this should not happen with the non-real-time case if everything went correctly because the polling is based on transmission requests from mobiles

F. Real-Time Communication Service

Real-time connection i is specified by a triplet (M_i, T_i, D_i), where:

- (M_i, T_i): M_i is the maximum number of packets that can arrive in an interval of length T_i;
- D_i: the packet delivery delay (i.e., from the mobile to the BS, or from the BS to the mobile, hence a fraction of the target end-to-end delay) bound for real-time packets of connection i, which has the following relationship with T_i:

$$D_i \geq D_i^{\min} = \begin{cases} T_i, & \text{for downlink} \\ 2T_i, & \text{for uplink.} \end{cases} \tag{2}$$

As will be clear later, the BS will admit new connections and schedule packet-transmissions and channel-polling based on the minimum delay bound $D_i^{\min}$ so that each packet's delay is bounded by $D_i^{\min}$ as long as the channel condition continues to be good. Once a packet transmission is deferred or fails, the BS will attempt to deliver the packet within D_i in a best effort manner. Note that the larger D_i, the more likely packets will be delivered in time under a time-varying channel condition.

To set up a real-time connection, the source node (which is a mobile if it is an uplink connection) will request the BS to set up the connection with its specification triplet. The BS will then perform the admission test given in Section IV-B. Depending on the test result, the new connection will be admitted or rejected. This test result is conveyed to the mobile through a downlink control packet. If it is admitted, a connection is set up: if it is an uplink connection, the BS starts to schedule the polling for that mobile, or if it is a downlink connection, it starts to schedule downlink real-time packets arriving at the BS. Even though it was described in the context of a new connection request, the same procedure can be used for a handoff connection request. For an uplink connection, the BS just polls mobiles assuming the contracted traffic characteristics because the BS does not know the exact status of a mobile's pending real-time packets. The control packet for polling a mobile contains the mobile ID. When a mobile is polled, it can transmit one pending real-time packet through the uplink. Fig. 3 shows how the channel probing and deferment work for both downlink and uplink transmissions, where the label 'X' represents loss of a control packet. The BS will attempt to transmit each deferred polling or downlink packet later so that its delivery bound can be met if the channel condition improves before the deadline expires without compromising other undeferred packet transmissions as described in Section IV-A. The channel-state probing can be used adaptively; that is, a real-time connection can be serviced without this process initially, but later if this connection is determined to suffer from time-varying errors, this process can be invoked. Especially, a mobile that stays close to the BS will be experiencing a very good channel, so it will not need this process at all. Note that this channel probing imposes bandwidth overhead; $2T_{\mathrm{ms}}$ is consumed for each channel probing. As shown in Fig. 3, the transmission of a real-time packet (both downlink and uplink) consumes $3T_{\mathrm{ms}} + T_s$. Note that the ACK/NAK of an uplink packet can always be piggybacked in a downlink (control) packet, so it was not counted.

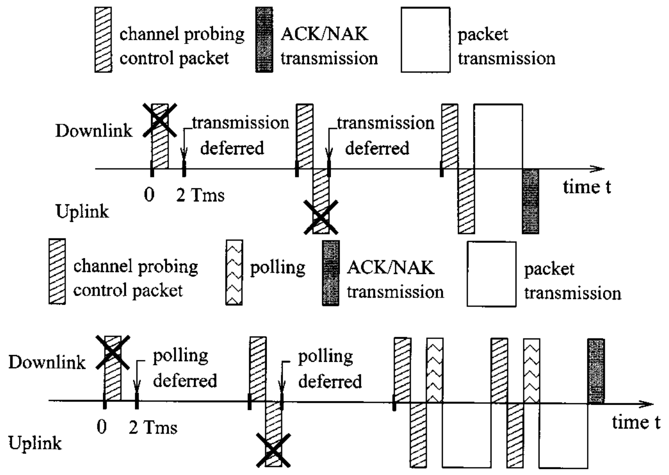


Fig. 3. Real-time communication slot structures with channel probing.

G. Non-Real-Time Communication Service

The BS schedules the transmission of downlink non-real-time packets or the polling for uplink non-real-time packet-transmission permissions with lower priority than the scheduled real-time traffic. Important concerns about non-real-time communication service include:

- virtually error-free transmission via our retransmission-based error control mechanism;
- (long-term) fair access to the link of mobiles: a round-robin-based packet/polling scheduling is used for fairness;
- maximizing aggregate throughput: our non-real-time scheduling is geared toward reduction of control packet overheads associated with ACKs/NAK's and polling. Moreover, channel probing is not used as long as a mobile's channel continues to be good.

IV. Run-Time Scheduling and Admission Control

As mentioned earlier, D-TDD requires uplink and downlink real-time packets to be multiplexed and scheduled over the same frequency channel. We need two different packet/polling scheduling policies for real-time and non-real-time services due to their distinct requirements. To set up a real-time connection, the BS performs an admission test to check if it can guarantee the timely delivery of every real-time packet from the new channel without compromising other existing guarantees.

A. Real-Time Traffic Scheduling

The conventional first-in-first-out (FIFO) scheduling is unable to bound the packet-delivery delay. So, we adopt a nonpreemptive earliest deadline first (EDF) scheduling algorithm—the earlier the deadline the higher the priority. Liu and Layland [20] proved that the deadline-driven scheduling is optimal among all dynamic-priority scheduling policies when the deadline of each task is equal to the end of its period. Our real-time scheduling and admission control are based on the real-time channel protocol in [17].

1) Downlink Packet Scheduling: The *logical arrival time* for the nth packet, arrived at the BS at time $t_i(n)$, of real-time connection i is defined as

$$l_i(n) = \begin{cases} t_i(n), & \text{for } 1 \le n \le M_i, \\ \max\{l_i((\lfloor n/M_i \rfloor - 1)M_i + 1) + T_i, t_i(n)\}, & \\ \qquad \text{for } n > M_i. \end{cases} \quad (3)$$

The *logical deadline* of this packet is then $l_i(n) + D_i^{\min}$. A packet with $t_i(n) < l_i(n)$ (i.e., arrived too early) is not transmitted until it becomes 'current' since its immediate transmission might result in buffer overflow at the destination mobile. By using this logical deadline, one can protect well-behaving connections from misbehaving ones. For example, much more than M_i packets could arrive during T_i from a misbehaving connection with contract (M_i, T_i, D_i), which can cause violation of the bounded-delivery guarantees of packets belonging to other well-behaving connections if their logical deadlines are not enforced. Note that $D_i^{\min}$ is added to the logical arrival time instead of D_i in the calculation of the logical deadline. In this way, the BS attempts to deliver each real-time packet within its minimum delay bound as long as the destination mobile's channel remains in good condition. A deferred or erroneously transmitted packet will be (re)transmitted when the channel is predicted and its actual deadline D_i can be met.

When the nth packet, $p_i(n)$, of real-time connection i, C_i, arrives at the BS from the wired network at time $t_i(n)$, it is placed in a connection-specific FIFO queue $Q[i]$, and its logical deadline is calculated. Then, the packet's connection ID is fed into queue **R** (meaning 'ready') if $t_i(n) \ge l_i(n)$, and the entries of this queue are sorted according to the corresponding packets' deadlines. On the other hand, if $t_i(n) < l_i(n)$, this connection ID will be held in another queue **H** (meaning 'hold'), and will later be fed into **R** at time $t = l_i(n)$ (i.e., when it becomes current). When **R** is selected for service, the connection, say, C_j, whose ID is at the head of **R**, is served, and then the connection ID is removed from **R**. To serve C_j, the BS: 1) drops all those packets in queue $Q[j]$ whose deadlines cannot be met; 2) probes the destination mobile's channel; and 3) transmits the packet at the head of queue $Q[j]$ if the channel condition is predicted to be good; else it defers its transmission and places the connection ID in a FIFO queue **D** (meaning 'deferred').

If the packet transmission upon prediction of a good channel condition results in a NAK, the connection ID is placed into a FIFO queue **B** (meaning 'back-logged'). Both queues **D** and **B** are served similarly to **R**. For convenience, the process of serving queue **X** is referred to as a "queue **X** service," where **X** is **R** or **D** or **B**. Note that the (re)transmission of an once-deferred or unsuccessfully tried packet should not cause other undeferred packets to miss their deadlines. To achieve this, we use a bookkeeping counter, called the *credit counter* (CC), which records the time both queues **D** and **B** can be served continually without delaying queue **R** service. The CC is updated as follows:

- $CC := CC + 5T_{\text{ms}} + T_s$, if a queue **R** service results in a deferment;
- $CC := CC + 2T_{\text{ms}}$, if a queue **R** service results in a packet transmission;

- $CC := CC - 2T_{\rm ms}$, if a channel is probed for a queue **D** or **B** service;
- $CC := CC - (T_{\rm ms} + T_s)$, if a packet is transmitted via a queue **D** or **B** service;
- $CC := CC - T_{\rm tr}$, if the link is used for non-real-time traffic or transmission requests for the duration of $T_{\rm tr}$;
- $CC := 0$, if $CC < 0$ after any of the above updates.

As will be described in Section IV-B, $5T_{\rm ms} + T_s$ is assumed necessary for each packet transmission during the admission-control phase, because packet transmission without deferment takes $3T_{\rm ms} + T_s$ and additional $2T_{\rm ms}$ is reserved for one extra channel probing for each packet. This is why $5T_{\rm ms} + T_s$ $(2T_{\rm ms})$ is added when a queue **R** service results in a deferment (packet transmission).

When $CC \geq 3T_{\rm ms} + T_s$, queue **D** is given priority over queue **R**. Then, **B** is given next priority after **D**. Basically, the channel usage is prioritized in the order of **D**, **B**, and **R** if $CC \geq 3T_{\rm ms} + T_s$, and **R**, **D**, and **B** if $CC < 3T_{\rm ms} + T_s$. Whenever the wireless link becomes free (after completing an on-going transmission), the BS will choose the nonempty highest priority queue for the next channel usage. When queue **D** is chosen, the connection specified by the $D_{\rm index}$th entry of **D** is served. Unless the queue **D** service results in a deferment, the $D_{\rm index}$th entry is removed, and all the following entries are shifted toward the head within **D**, i.e., the $(i+1)$th entry becomes the ith entry for all $i \geq D_{\rm index}$. The index $D_{\rm index}$ is updated as follows, with the initial value $D_{\rm index} = 0$:

- $D_{\rm index} := 0$, if **D** becomes empty;
- $D_{\rm index} := D_{\rm index} + 1$, when a queue **D** service results in a deferment;
- $D_{\rm index} := 1$, when: 1) an entry in queue **R** is fed into the empty queue **D**; 2) the last entry in queue **D** is served; or 3) CC becomes less than $3T_{\rm ms} + T_s$.

In fact, if $D_{\rm index} = 1$, there is another condition—that is, the queue **D** service flag $D_{\rm flag} = 1$—to serve queue **D**. Otherwise, the link is used to serve the queue with the next priority. The value of $D_{\rm flag}$ is updated as follows, with the initial value $D_{\rm flag} = 0$:

- $D_{\rm flag} := 1$, if any packet (other than control packets) is transmitted over the channel;
- $D_{\rm flag} := 0$, when an entry in queue **R** is fed into the empty queue **D**, or a queue **D** service results in a deferment when $D_{\rm index} = 1$.

Note that by using $D_{\rm flag}$, once a packet transmission is deferred, at least one packet is transmitted before serving this packet again. This is reasonable since trying to continuously serve a deferred packet can waste bandwidth significantly due to continuous channel probes without its successful transmission. The same rule is used for queue **B** by replacing $D_{\rm index}$ and $D_{\rm flag}$ with $B_{\rm index}$ and $B_{\rm flag}$, respectively.

Note that the entries in real-time queues **R**, **D**, and **B** specify connection ID's, not packet ID's, and the packet with the smallest deadline in each connection—irrespective of which queue specifies this connection for a service—is transmitted. In this way, we can maximize the chance to deliver packets before their deadlines. Fig. 6 summarizes how to determine which queue to serve, including the non-real-time queues that will be covered later.

2) Polling Order: As mentioned earlier, the polling order for uplink real-time mobiles is determined based on the contracted traffic characteristics as follows.

1) The BS generates a polling request for uplink real-time connection i with (M_i, T_i, D_i), periodically once every T_i.
2) The request generated at time t_i is sorted in queue **R** according to the deadline $t_i + D_i'$ along with downlink connection ID's, where $D_i' = D_i^{\min} - T_i = T_i$. Conceptually, a polling request for connection i is considered as a packet of size $M_i(3T_{\rm ms} + T_s)$ including the channel probing overhead.
3) When the request is at the head of queue **R**, the BS will serve[6] uplink real-time mobile i up to M_i times consecutively until the channel for this mobile is predicted to be bad, or this mobile does not have any more pending packets to transmit.

The deadline-driven scheduling plus the admission control described in Section IV-B will ensure that the BS can start probing the channel for uplink connection i as late as $D_i' - M_i(3T_{\rm ms} + T_s)$ after generation of the last polling request. Then, for the kth polling request generation time G_k (where $G_{l+1} - G_l = T_i$ for all l), all of M_i packets generated during the interval of $[G_{k-1}, G_k]$ will be transmitted by time G_{k+1} as long as the channel condition continues to be good. So, the worst case delay will be $G_{k+1} - G_{k-1}$, which is $2T_i$ or $D_i^{\min}$. If the BS sends a channel probing control packet to a mobile when this mobile does not have any pending real-time packet, the mobile will specify that it does not have any real-time packet via the returning control packet so that the BS need not poll the mobile. On the other hand, if a mobile completes the transmission of all of its pending real-time packets after being polled N_i times (where $N_i < M_i$), the BS will be informed of this via the N_ith packet so that the BS need not poll the mobile again. When polling for connection i in queue **R** is deferred after polling it N_i times, where $0 \leq N_i < M_i$, the polling request is queued up in **D** with $P_i = M_i - N_i$, which is the number of polls to be taken, as well as the connection ID. When this request is to be served later, mobile i will be polled up to P_i times consecutively. The credit counter is updated with polling as follows:

- $CC := CC + 2T_{\rm ms}N_i + (3T_{\rm ms} + T_s) + (M_i - N_i - 1)(5T_{\rm ms} + T_s)$, if polling for connection i in queue **R** is deferred after it is polled N_i times;
- $CC := CC - 2T_{\rm ms} + M_i(5T_{\rm ms} + T_s)$, if mobile i had no pending packet to transmit;
- $CC := CC + 2T_{\rm ms}N_i + (M_i - N_i)(5T_{\rm ms} + T_s)$, if mobile i completed the transmission of its pending packets after being polled N_i times (where $0 < N_i \leq M_i$), i.e., it had N_i pending packets;

The other rules associated with serving a polling request are the same as those for real-time downlink packet transmissions. For

[6]By 'serving' an uplink real-time mobile, we imply that this mobile's channel is probed, and the mobile is polled if the channel is predicted to be good; otherwise, the polling is deferred.

each erroneously transmitted uplink packet, a polling request is queued up in **B**.

3) Transmission-Request Slot Scheduling: To achieve low latencies for message transmissions or connection setup/handoff requests, it is necessary to issue transmission-request slots regularly and frequently. When it is desirable to have a transmission-request slot once every $T_{\rm req}$ time units, this slot is issued with the same scheduling used for polling with $T_i = T_{\rm req}$, $M_i = 1$, and $D_i = D_i^{\min} = 2T_{\rm req}$. Note that no channel probe is made before issuing a transmission-request slot since there is no specific destination mobile in this case. For the rest of this paper, we "imagine" there exists a virtual uplink connection with $(M_i, T_i, D_i) = (1, T_{\rm req}, 2T_{\rm req})$, which is equivalent to transmission-request slots. However, the scheduling discussed so far is valid only when the system is busy, i.e., there always exist some downlink packets or uplink pollings scheduled continuously. If not, the BS basically has nothing to do. Then, the BS continues to issue transmission-request slots, thus reducing the transmission-request access latency as much as possible. With the above scheduling, during the system busy period, a transmission-request slot will appear, on average, once every $T_{\rm req}$ time units, and the time span between two consecutive transmission-request slots is bounded by $2T_{\rm req}$. Note that the packet scheduling itself cannot guarantee a bounded delivery delay. A proper admission control should be performed to determine if a set of connections can be guaranteed to have their required delivery-delay bounds using the above scheduling mechanisms.

B. Admission Test

We now consider the admission test for a new connection request. If the other communicating party of a mobile is outside of the cell, the entire communication path will be divided into one or two wireless links and a route within the wired network. The admission test should consider end-to-end communication. However, here we consider the admission test for a wireless link only since the transmission over the wired links can be handled by other known schemes like the one in [17]. (Admission tests for the wired-network part have been studied by many other researchers, and depend on the scheduling policy used in the intermediate nodes.) For a new connection to be established, the results from admission tests on all components (on the communication path) should be positive.

As mentioned already, the admission control is applied so that the delay of each packet in connection i for all i is bounded by $D_i^{\min}$ as long as mobile i's channel stays in good condition. Let us consider a set of connections $\{C_i = (M_i, T_i, D_i) | i = 1, 2, \ldots, N_r\}$. The admission test is composed of two phases, i.e., a bandwidth test phase, and a delay-bound test phase. For the newly requested connection *new*, the bandwidth test is to check if the sum of reserved bandwidth for each connection including overheads satisfies the following condition:

$$(T_s + 5T_{\rm ms})\left(\sum_{i=1}^{N_r} M_i/T_i + M_{\rm new}/T_{\rm new}\right) \le 1 - \Delta_r \quad (4)$$

where Δ_r is the portion of the link capacity reserved for real-time packet retransmissions and non-real-time traffic. Note that $(5T_{\rm ms} + T_s)$ instead of $(3T_{\rm ms} + T_s)$ is needed for a packet transmission to reserve additional $2T_{\rm ms}$ for an extra channel probing. The larger Δ_r, the more packets will make their deadlines even under bad channel conditions. The value of Δ_r can be adapted depending on the channel conditions of the existing connections and the observed packet-dropping probability.

If a new connection does not satisfy the first condition, it is rejected while it is considered for the second phase delay-bound test only if it satisfies the first condition. The delay-bound test is set up as follows. First, suppose the elements of set $\{C_i\}$ are arranged in ascending order of D_i', where

$$D_i' = \begin{cases} D_i^{\min}, & \text{if downlink} \\ D_i^{\min} - T_i, & \text{if uplink.} \end{cases} \quad (5)$$

In fact, $D_i' = T_i$ for both uplink and downlink packets. Note that when a packet (or polling request) from each of N_r connections arrives at the same time, the packet (or polling request) from a smaller numbered connection will have higher priority. To establish a real-time connection, the BS needs to check its schedulability with the connection's worst case delay. We can derive an admission test similar to that in [17]. For downlink connection i, M_i packets are assumed to arrive periodically at the beginning of each virtual packet-arrival period T_i, which represents the worst case in terms of the delivery delay of these packets. Then, the last of these M_i packets should finish the transmission within D_i' as long as the channel continues to be good. If this works correctly, uplink packets are also transmitted by the minimum delay bound, as explained in Section IV-A2.

Now, for the schedulability test, we consider the transmission time of the last of M_i packets of downlink connection i. The worst case packet-delivery delay occurs when its arrival coincides with all other higher priority packet arrivals. This "critical" instant is denoted as time $t = 0$. The transmission of the packet can be delayed by all instances of higher priority packets (and pollings) at $t = 0$ and by the subsequent instances of these packets (and polling requests). Arrival times of these instances within the delay bound D_i' are given as

$$\begin{aligned} A_i = &\{D_i'\} \cup \{kT_j | j = 1, 2, \ldots, i-1; \\ &k = 1, 2, \ldots, \lfloor (D_i'/T_j) \rfloor\}. \end{aligned} \quad (6)$$

The total time required for transmitting this packet and others with priority equal to, or higher than, this packet (or polling) is given by

$$\begin{aligned} W_i(t) = &T_{\max\_\rm poll} + M_i(5T_{\rm ms} + T_s) \\ &+ \sum_{j=1}^{i-1} \{M_j(5T_{\rm ms} + T_s)\}\lceil t/T_j \rceil. \end{aligned} \quad (7)$$

Note that the first and second terms of (7) denote, respectively, an upper bound of the time to complete in-progress packet transmission or polling and the time to transmit M_i packets of real-time connection i. The third term denotes the time to transmit all packets with the same or higher priority. Note that $(5T_{\rm ms} + T_s)$ instead of $(3T_{\rm ms} + T_s)$ was again assumed necessary for a packet transmission. In fact, $T_{\max\_\rm poll}$ is affected by uplink connections since the time to complete an in-progress downlink packet transmission will be just $3T_{\rm ms} + T_s$. An uplink real-time mobile i will be polled up to M_i times consecutively once it is polled

$$T_{max\_poll} = \max\{2T_s, \max_{i \in \mathbf{S}_u} \{M_i(3T_{\rm ms} + T_s)\}\} \tag{8}$$

where $\mathbf{S}_u$ is the set of uplink connections' indexes. By definition, T_{max_poll} will be at least $2T_s$ even when there is no uplink real-time connection. This follows from the maximum possible non-real-time traffic transmission without Throughput versus non-real-time offered load: $K = 20$ and $T_{\rm req} = 200$.interrupting real-time packet transmissions as explained in Section IV-C. A set of connections $\{C_i\}$ is said to be *schedulable* (i.e., all packet minimum delivery-delay constraints will be met as long as the channel conditions continue to be good) if the following condition holds:

$$W_i(t) \le t, \qquad \text{for some } t \in A_i \qquad \text{for all } i = 1, 2, \ldots, N_r \tag{9}$$

where N_r is the number of connections and A_i is given as in (6). From this schedulability condition, we now derive the second phase delay-bound test of a new connection when there already exists a set of connections $\{C_i\}$ in the system. Note that the schedulability condition in (9) already holds for the set of existing connections $\{C_i\}$.

- Upon arrival of a new connection request, the BS inserts the newly requested connection *new* in the set of connections $\{C_i\}$ according to the ascending order of $D'_{\rm new}$.
- If *new* is a downlink connection, the admission test is to check if the following condition holds:

$$W_i(t) \le t \quad \text{for some } t \in A_i \quad \text{for } i = new, new+1, \ldots, N_r. \tag{10}$$

- If *new* is an uplink connection, $T_{\rm max\_poll}$ is recalculated using (8).
  - If $T_{\rm max\_poll}$ has not changed, the admission test is given by (10).
  - If $T_{\rm max\_poll}$ has changed, the admission test is given by (9).

Note that if $T_{\rm max\_poll}$ has not changed, the performance guarantees of connections with smaller D'_i's are not affected by the admission of connection *new*.

C. Non-Real-Time Traffic Scheduling

The transmission of non-real-time traffic is based on a modified round-robin scheduling policy with a set of two prioritized queues. Using the round-robin scheduling, it is possible to give each mobile fair channel access as long as each mobile's channel condition continues to be good. When there are M_o mobiles in the cell, the BS is equipped with the following two service queues (in that order of priority): 1) a class II-A round-robin queue (with M_o entries), called queue **RR.A**, and 2) a class II-B round-robin queue (with M_o entries), called queue **RR.B**. The ith entry of a round-robin queue, corresponding to mobile i, is composed of two parallel queues for the downlink and uplink, respectively. The uplink queue has a number of uplink packets requested for transmission permissions, which is the sum of the number of packets in transmission permission-requested messages. On the other hand, the downlink queue buffers the downlink packets in a FIFO manner, which had arrived at the BS. Fig. 4 shows a round-robin queue for M_o mobiles in the cell.

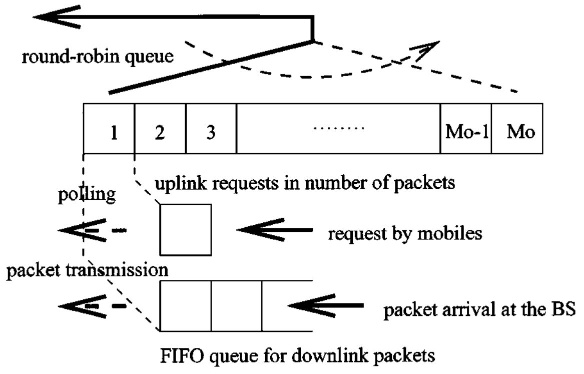


Fig. 4. A round-robin queue for non-real-time communications.

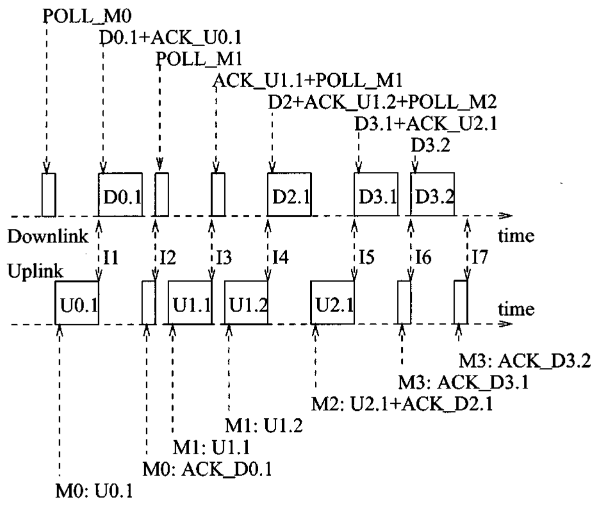


Fig. 5. An example of non-real-time transmissions.

We first describe how the round-robin queues work assuming that there is no channel error, then address how to handle channel errors. For the ith entry's turn, up to two packets for mobile i are served. To reduce the ACK/NAK-transmission and polling overheads, it is desirable to sequentially serve one downlink and one uplink packets. In this case, a downlink packet destined for mobile i is transmitted first with up to two different pieces of piggybacked information: 1) ACK/NAK for a previously transmitted uplink packet if the previous packet was uplink and 2) the transmission permission for mobile i to transmit an uplink packet (i.e., a polling to mobile i). Then, mobile i will transmit an uplink packet with ACK/NAK for the previously received downlink packet. Note that through this transmission order, two mini-slots were saved, i.e., for one downlink control packet to poll mobile i and for one uplink control packet for the ACK from mobile i. If the ith entry does not have both downlink and uplink requests, up to two downlink or uplink packets for mobile i can be transmitted consecutively using two mini-slots for polling or ACK transmissions. An uplink packet and a downlink packet can also be transmitted sequentially if a downlink packet arrives during the transmission of an uplink packet.

An example of this non-real-time scheduling is shown in Fig. 5, where a square and a small rectangle represent a

```
01. whenever (the link becomes free) {
02.   if ($C \geq 3T_{ms} + T_s$ and ($D_{index} > 1$ or ($D_{index} = 1$ and $D_{flag} = 1$))) serve queue D;
03.   else if ($C \geq 3T_{ms} + T_s$ and ($B_{index} > 1$ or ($B_{index} = 1$ and $B_{flag} = 1$))) serve queue B;
04.   else if (queue R is non-empty) serve queue R;
05.   else if ($D_{index} > 1$ or ($D_{index} = 1$ and $D_{flag} = 1$)) serve queue D;
06.   else if ($B_{index} > 1$ or ($B_{index} = 1$ and $B_{flag} = 1$)) serve queue B;
07.   else if (queue RR.A is non-empty and $RR^A_{flag} = 1$) serve queue RR.A;
08.   else if (queue RR.B is non-empty and $RR^B_{flag} = 1$) serve queue RR.B;
09.   else issue a transmission request slot;
10. }
```

Fig. 6. A pseudocode to determine which queue to serve when the wireless link becomes free.

packet transmission slot and a mini-slot, respectively. The notation used is: $U_{i.j}$ is the jth uplink packet transmission of mobile i; $D_{i.j}$ is the jth downlink packet transmission to mobile i; $POLL\_M_i$ is a polling to mobile i; $ACK\_\{x\}$ is the ACK/NAK for the previous packet transmission x; M_i represents the transmission from mobile i, and '+' represents piggybacking of the following information in the same (control) packet. Four different cases are shown in chronological order: 1) an uplink packet from, and a downlink packet sequentially to, mobile 0; (2) two uplink packets from mobile 1; (3) a downlink packet to, and an uplink packet sequentially from, mobile 2; and (4) two downlink packets to mobile 1. Note that two control packets are used to transmit two packets from/to mobiles 0, 1, and 3. However, no control packet is used for mobile 2 to receive one downlink and transmit one uplink packet.

Note that real-time traffic or higher priority non-real-time traffic (such as class II-A traffic if the traffic considered is class II-B) can take over the channel according to the priority rule when the wireless link becomes free after 1) the completion of the ACK transmission for a non-real-time downlink packet or 2) the completion of a non-real-time uplink packet transmission. For the latter case, the BS needs to piggyback the ACK/NAK within the following (control) downlink packet. So, the BS can switch to the real-time traffic service or a higher priority non-real-time traffic service at the marked (with I_n) moments in Fig. 5. The worst case delay before switching to the real-time traffic service is shown in the figure, i.e., from I_4 to I_5. To interrupt at I_5, there is a delay of up to $2T_s$ time units. This is why T_{max_poll} should be at least $2T_s$ in (8). Even though it was not shown in the figure, the request for an uplink message transmission can also be piggybacked in an uplink packet.

Now, let us assume that there was a NAK associated with a packet transmission from/to mobile i. Then, the entry for this mobile in the particular round-robin queue is marked "back-logged," indicating that the mobile's channel condition is bad. For this entry's next service turn, the mobile's channel is probed first. If the channel is predicted to be good, the mobile's packets are served and the entry is marked "active." Otherwise, the entry stays back-logged without the mobile's being served at all. For the turn of the ith entry, which is back-logged, the ith *non-real-time compensation counter* $NCC.(*)[i]$ is increased by the number of packets to be served if the entry was active, where $(*)$ is A (B) if the queue is for class II-A (class II-B). This counter is increased when an entry becomes back-logged as well. For example, if uplink class II-A packet $U_{0.1}$ in Fig. 5 is erroneously transmitted, the zeroth entry of queue **RR.A** is marked back-logged, and $NCC.A[0]$ is increased by two since both $U_{0.1}$ and $D_{0.1}$ could be transmitted if the channel was good. For the turn of the ith entry which becomes active, as many as $NCC.(*)[i] + 2$ packets of mobile i are served continually. Then, $NCC.(*)[i]$ is reset to zero. In this way, we can achieve long-term fairness among mobiles. Note also that before serving the first entry in a round-robin queue, if all the entries are marked back-logged, the round-robin queue service flag $RR^{(*)}_{\text{flag}}$ is set to zero, where $(*)$ is again A (B) if the queue is for class II-A (class II-B). $RR^{(*)}_{\text{flag}}$'s are set to 1 whenever a packet is transmitted over the channel. A round-robin queue can be served only when its flag is 1. This way we can reduce the overhead associated with the channel probing for non-real-time traffic while being able to serve class II-B traffic when all mobiles with class II-A traffic are experiencing bad channel conditions. Note that the channel is probed for back-logged connections only in order to reduce unnecessary bandwidth waste for channel probing.

An ACK/NAK can be lost due to errors. Because the feedback should be received right after the packet transmission, the sender assumes, in case of ACK/NAK loss, that the packet transmission was not successful and retransmits the previous packet later. The receiver, in turn, knows that the previous ACK/NAK was lost when it receives a duplicate packet. If the receiver is the BS, the BS needs to update a round-robin queue's entry by increasing the number of uplink packet requests by one since it received the same packet again. Last, recall that the above-mentioned scheduling is applied whenever 1) all the real-time queues **R**, **D**, and **B** are empty or 2) queue **R** is empty, $D_{\text{flag}} = 0$, $D_{\text{index}} = 1$, $B_{\text{flag}} = 0$, and $B_{\text{index}} = 1$. Fig. 6 provides a pseudocode to determine which queue to be served whenever the wireless link becomes free.

V. Experimental Results and Discussion

This section evaluates and discusses the performance of the proposed scheme. We first state the assumptions and the system specification used in the evaluation study.

A. Assumptions and System Specification

We assume that there are N_u $(= 10)$ mobiles in the cell, and the duration T_{ms} of a mini-slot is used as a basic *time unit*. Only one request can be transmitted in a transmission-request slot or piggybacked in an uplink packet by a mobile. The wireless channel is modeled as follows.

C1: The channel between the BS and a mobile is modeled by a Markov chain as shown in Fig. 7. The channels of two different mobiles are assumed to be independent.

C2: A transition between the good and bad states can happen at mini-slot boundaries with the transition probability

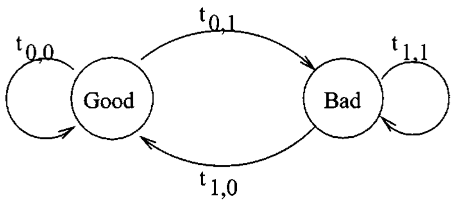


Fig. 7. Two-state Markov chain modeled channel.

$p_{G,B}$ $(p_{B,G})$ from the good to bad states (from bad to good states).

C3: For a given average time duration T_G (T_B) at the good (bad) state, $p_{G,B} = 1/T_G$ $(p_{B,G} = 1/T_B)$. We assume that $T_G = 2000$, and $T_B = 100$.

C4: Each request could be received in error 1) when it is corrupted due to channel errors or 2) when it collides with another request within the same request mini-slot.

C5: If a packet transmission duration overlaps with a time in the bad state, the packet is received in error. The packet is received correctly otherwise.

C6: Every packet received in error is assumed to be detected by the channel decoder.

The real-time traffic of interest is specified as follows.

R1: Four types of real-time connections are considered with the connection specification (M_i, T_i, D_i): 1) type-1 downlink (1, 200, 300); 2) type-1 uplink (1, 200, 500); (3) type-2 downlink (1, 500, 600); and (4) type-2 uplink (1, 500, 1100). Note that type-1 (or 2) downlink and uplink connections have the same M_i and T_i.

R2: Real-time connections arrive according to a Poisson process with rate $\lambda_{\rm rt}$. Half of them are handoff connections, which arrive from adjacent cells.

R3: The lifetime of each connection i is distributed geometrically with mean $T_{\rm life}$ $(= 50T_i)$.

Last, the assumptions on non-real-time traffic are as follows.

N1: Downlink (uplink) messages arrive according to a Poisson process with rate λ_d (λ_u) (messages/mini-slot). The overall message-arrival rate $\lambda_{\rm nrt}$ is defined to be $\lambda_{\rm nrt} = \lambda_d + \lambda_u$.

N2: The downlink class II-A (II-B) messages, destined for each of N_u mobiles, arrive independently at the BS with rate $0.9\lambda_d/N_u$ $(0.1\lambda_d/N_u)$.

N3: The uplink class II-A (II-B) messages arrive independently with rate $0.9\lambda_u/N_u$ $(0.1\lambda_u/N_u)$ at each of N_u mobiles.

N4: The number of packets in a class II-A (II-B) message is geometrically-distributed with average 2 (18).

A frequently used parameter, the *offered load,* is defined as the overall expected number of the packets to be transmitted within a slot. For the overall non-real-time message-arrival rate $\lambda_{\rm nrt}$ (messages/mini-slot), the offered load from non-real-time traffic, $L_{\rm nrt}$, is given by

$$\begin{aligned} L_{\rm nrt} &= (0.9 \cdot 2 + 0.1 \cdot 18)K\lambda_{\rm nrt} \\ &= 3.6K\lambda_{\rm nrt} \quad \text{(packets/slot)} \end{aligned} \tag{11}$$

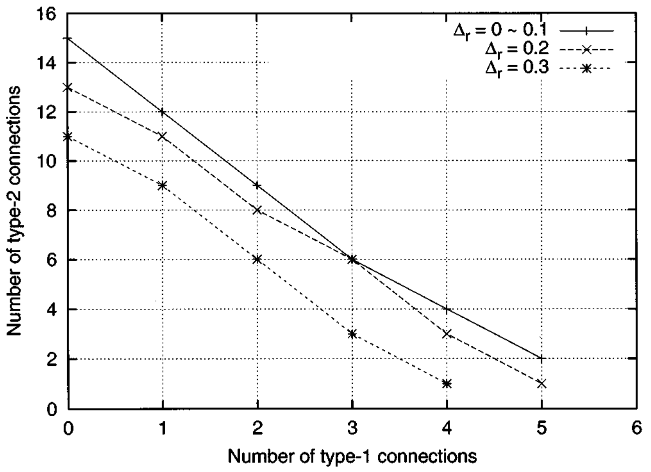


Fig. 8. The admission region for real-time connections: $K = 20$ and $T_{\rm req} = 200$.

of which half is for class II-A and the other half is for class II-B. On the other hand, the offered load from real-time traffic $L_{\rm rt}$, is given by

$$\begin{aligned} L_{\rm rt} &= \lambda_{\rm rt} T_{\rm life} E\left\{\frac{M_i T_s}{T_i}\right\} \\ &= \lambda_{\rm rt} T_{\rm life} \sum_{j=1}^{2} \alpha_j \frac{\hat{M}_j T_s}{\hat{T}_j} \quad \text{(packets/slot)} \end{aligned} \tag{12}$$

where $E\{\cdot\}$ represents the averaging operation, α_j is the fraction of type-j (both downlink and uplink) connections out of the total real-time connection arrivals, $\hat{M}_j$ is the M_i of type-j connection i, and $\hat{T}_j$ is the T_i of type-j connection i..

B. Interworking of Real-Time Connections with Non-Real-Time Traffic

Throughout this subsection, the following parameters are used: $K = 20$ and $T_{\rm req} = 200$, so there are 10 $(= K/2)$ request mini-slots within a transmission-request slot. Three request mini-slots are reserved for handoff requests, i.e., $K_{\rm ho} = 3$. We first consider the admission region of type-1 and -2 connections depending on the reserved link capacity portion Δ_r for real-time packet retransmissions and non-real-time traffic. Using (4) and (9), each pair of the numbers of type-1 and -2 connections is tested; a pair (n_1, n_2) represents a set of n_1 type-1 connections and n_2 type-2 connections. Fig. 8 shows the curves representing the admission regions for three different Δ_r values. For each line, a set of connections for each pair under or on a curve can be admitted into the system. For example, six type-1 uplink connections and two type-2 uplink connections can coexist in the system at a time when $\Delta_r = 0 - 0.1$. We observe that the larger Δ_r, the smaller the admission region, as it should be. The admission regions for $\Delta_r = 0$ and $\Delta_r = 0.1$ are the same is due to the effect of T_{max_poll} in (7) for the delay-bound test. Note that the admission regions of both downlink and uplink connections are the same since both downlink and uplink connections of the same type have the same M_i and T_i. Now, let us consider the performance of real-time traffic. For further study, we assume that $\Delta_r = 0$, and only type-1 uplink connections were

TABLE III
PERFORMANCE OF REAL-TIME TRAFFIC WHEN NONREAL-TIME UPLINK TRAFFIC ARRIVES WITH $\lambda_U = 0.0005$: $K = 20$ AND $T_{\rm req} = 200$

Real-time arrival rate (λ_{rt})	0.0001	0.0005	0.001
Real-time offered load (L_{rt})	0.10	0.50	1.00
Real-time throughput	0.099	0.359	0.439
Conn.-blocking probability	0.0013	0.273	0.563
Average packet delay	164.55	181.76	185.09
Maximum packet delay	500	500	500
Packet-dropping probability	0.0031	0.0040	0.0042

generated, where each of them is a constant bit rate (CBR) connection, i.e., M_i packets are generated every T_i time units. We learned from the previous admission-region figure that up to five type-1 uplink connections can be admitted at a time. The reason why we consider only uplink connections is to examine the access latency of uplink connection requests, which is a very important performance measure. In the simulation, non-real-time uplink traffic arrives with rate $\lambda_u = 0.001$. Note that the performance of real-time traffic is almost unaffected by non-real-time traffic because real-time traffic is given priority over non-real-time traffic. (Some small effect can appear due to nonpreemptive transmission of non-real-time traffic, and packet deferments of real-time traffic.) Table III shows the real-time offered load $L_{\rm rt}$, real-time traffic throughput, connection-blocking probability, average and maximum packet delays, and packet-dropping probability (i.e., packet drops due to the deadline expirations) for three different arrival rates $\lambda_{\rm rt}$ of real-time connections. The blocking probability represents both new-connection blocks and handoff connection drops since a half of the entire real-time connection arrivals are assumed to be handoffs in our simulation. As the real-time connection arrival rate increases, more connections are blocked. We also observed that the larger $\lambda_{\rm rt}$, the larger the average packet delay and the packet-dropping probability, since the more real-time connections exist, the longer a scheduled polling is likely to be delayed. However, the maximum delay is observed to be always 500, which is the delay bound D_i. Note that this delay is bounded at the cost of some packet drops. We can easily expect that the larger Δ_r, the larger the connection-blocking probability, and the smaller packet-dropping probability.

Fig. 9 shows the average uplink class II-A message-transmission delay for four different real-time traffic-arrival rates. We observed that the larger $\lambda_{\rm rt}$, the larger the delay is for a given non-real-time offered load $L_{\rm nrt}$, because real-time traffic is given priority over non-real-time traffic. Note that a half of non-real-time offered load is from class II-B. This is the reason why the delay of class II-A could not be infinite even for the offered load is more than one. Even though it is not shown, the delay of class II-B is surely infinite for the offered load greater than one. The next subsection will elaborate on the performance of non-real-time communication service.

Next, we consider the request-access latency, which is defined as the time span from the arrival of a request to its successful reception by the BS. Recall that a request access is made through the transmission request slot or piggybacking for: 1) non-real-time (class II-A and II-B) message-transmission requests; 2) a new real-time connection request; and 3) a handoff request. The request-access latencies for four different cases are plotted in Fig. 10 when $\lambda_{\rm rt} = 0.001$, and only uplink non-real-time traffic exists. We observe that the request-access latency for real-time and handoff connections are reasonably low (less than 150) throughout the whole non-real-time offered load examined. Especially, the request-access latency for handoff connections are saturated by 120 beginning $L_{\rm nrt} \approx 0.4$. Note that the request-access latency of handoff connections should be kept low because otherwise, the handoff will not be smooth. To give a detailed account, we divide the entire non-real-time offered loads examined into three regions: 1) lightly loaded region: $L_{\rm nrt} = 0.0$ to 0.5; 2) moderately loaded region: $L_{\rm nrt} = 0.5$ to 0.8; and 3) heavily loaded region: $L_{\rm nrt} = 0.8$ to 1.8. First, we consider the request-access latency of non-real-time traffic. In the lightly loaded region, both the latencies of class II-A and II-B traffic increase monotonically. Class II-B traffic's access latency is larger since class II-A's requests are given priority. In the moderately loaded region, class II-B traffic's access latency starts to decrease due to the effect of piggybacked requests. Class II-A traffic's access latency, on the other hand, continues to increase further due to the effect of piggybacked requests. That is, in this region,

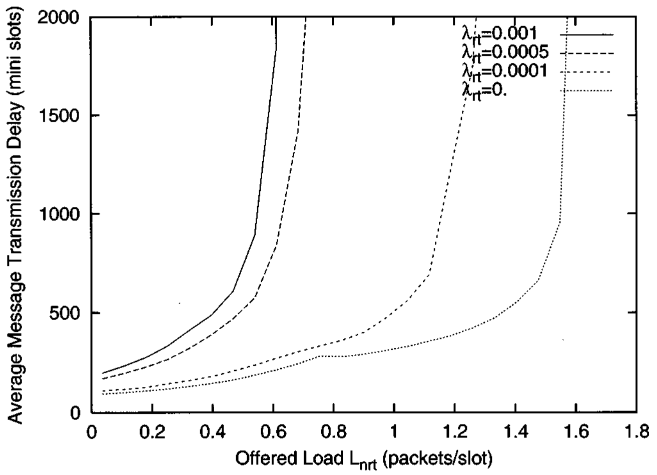


Fig. 9. Average uplink class II-A message-transmission delay versus non-real-time offered load: for different real-time connection-arrival rates $\lambda_{\rm rt}$, $K = 20$ and $T_{\rm req} = 200$.

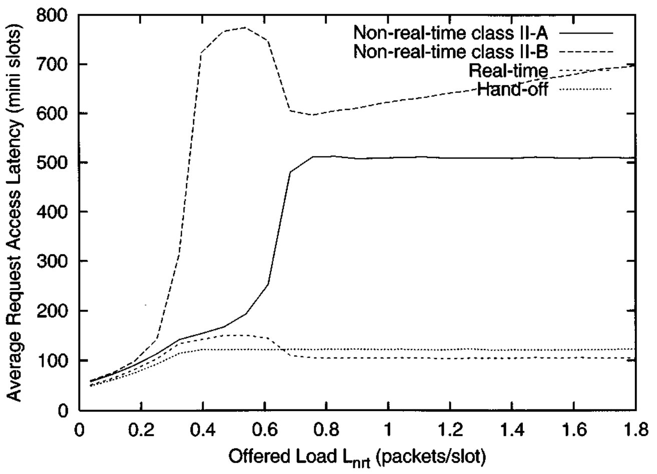


Fig. 10. Average request-access latency versus non-real-time offered load: $\lambda_{\rm rt} = 0.001$, $K = 20$, and $T_{\rm req} = 200$.

as the offered load increases, more requests will be made via piggybacked requests. The piggybacked request-access delay will increase since the time span between two consecutive services of a mobile will increase due to the increasing number of mobiles with pending requests in round-robin queues.

In the heavily loaded region, class II-A request-access latency is saturated at around 500 while class II-B request-access latency increases slowly beginning at around 600. In this region, non-real-time requests are likely to be made only though the piggybacked requests since the non-real-time round-robin service queues in the BS are unlikely to be empty. In this case, a mobile is supposed to have a chance to transmit two uplink packets once every 440 time units when there is no real-time traffic since there are ten mobiles and two packets (plus two control packets) can be transmitted at each round-robin turn. So, the request-access latency would be about 220 when there is no real-time traffic. Due to real-time connections, these chances are delayed more, depending on the real-time arrival rate. It is found to be about 500 for class II-A traffic when $\lambda_{\rm rt} = 0.001$. The request access of class II-B traffic is likely to be delayed more since only one request for a traffic class can be piggybacked in a packet.

The access latency of real-time and handoff connection requests is found to be lower than that of non-real-time traffic over the entire offered load region. This is because real-time connection requests are given priority over non-real-time transmission requests, and $K_{\rm ho}$ (which is three in our simulations) request mini-slots are reserved for handoff requests. For the very lightly loaded region (say $L_{\rm nrt} \leq 0.2$), the latencies for all types of requests are about the same since almost every request is made through a transmission-request slot and is mostly successful because collisions rarely occur. As the offered load increases, all the latencies increase while the handoff request latency is saturated. In the moderately loaded region, the access latency for real-time connections starts to decrease. This is again because some of transmission requests are piggybacked. Then, in the heavily loaded region, real-time and handoff request-access latencies are saturated at around 100 and 120, respectively. The real-time request-access latency is smaller than the handoff request-access latency since a handoff request can only be made through the transmission-request slot, while a new connection request can be made through either a transmission-request slot or piggybacking. They are about 100 since the period of issuing the transmission-request slot $T_{\rm req} = 200$, while they are about the same since the effect of the piggybacked request is negligible. The exact request-access latencies will depend on the real-time connection-arrival rate $\lambda_{\rm rt}$, but the overall tendency observed remains the same irrespective of the rate.

C. Non-Real-Time Communication Service

Here we examine in detail the performance of non-real-time communication service. To understand the non-real-time communication service better, we generated only non-real-time traffic for the simulations, i.e., $\lambda_{\rm rt} = 0$. Fig. 11 shows the average message-transmission delays for both class II-A and II-B for two extreme cases: one with downlink traffic arrivals only (marked with 'down') and the other with uplink traffic arrivals only (marked with 'up'). We observe that the delays with downlink traffic arrivals are a bit smaller than those with

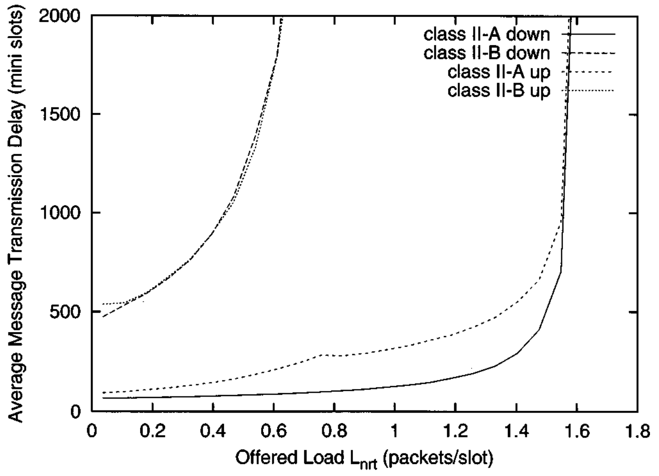


Fig. 11. Average message-transmission delay versus non-real-time offered load when downlink or uplink traffic exists: $K = 20$ and $T_{\rm req} = 200$.

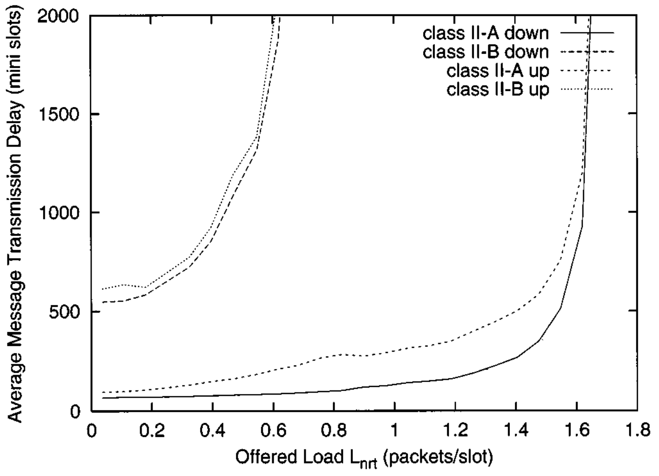


Fig. 12. Average message-transmission delay versus non-real-time offered load when downlink and uplink traffic arrive with the same rate, i.e., $\lambda_d = \lambda_u$: $K = 20$ and $T_{\rm req} = 200$.

uplink traffic arrivals. This indicates the effect of the uplink transmission-request access latency. For both cases, class II-B's delay goes to infinity at around $L_{\rm nrt} = 0.7$ while class II-A's delay goes to infinity at around $L_{\rm nrt} = 1.6$. This means that class II-B traffic is rarely transmitted for $L > 0.7$. This is reasonable since a half of the offered load is from class II-B, and the maximum achievable throughput is less than one. The fact that class II-A's delay goes to infinity at around $L_{\rm nrt} = 1.6$ means that the maximum achievable throughput is about 0.8. Fig. 12 shows the average message-transmission delays for both class II-A and II-B when a half of arrived messages are downlink and the other half are uplink, i.e., $\lambda_d = \lambda_u$. The general tendency is the same as in the previous figure, but the maximum achievable throughput is found to increase since the delay of class II-A is still about 2000 (i.e., within the figure) even for $L_{\rm nrt} = 1.65$. This shows the effect of sequential transmission of a downlink packet and an uplink packet without any control packet at a mobile's turn in the BS round-robin queue.

These facts can be seen clearer in Fig. 13. First, beginning at $L_{\rm nrt} = 0.8$, the throughput of class II-B starts to decrease while

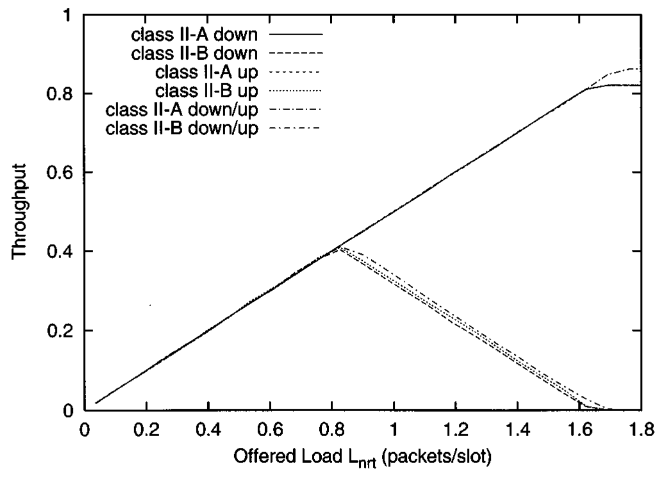


Fig. 13. Throughput versus non-real-time offered load: $K = 20$ and $T_{\text{req}} = 200$.

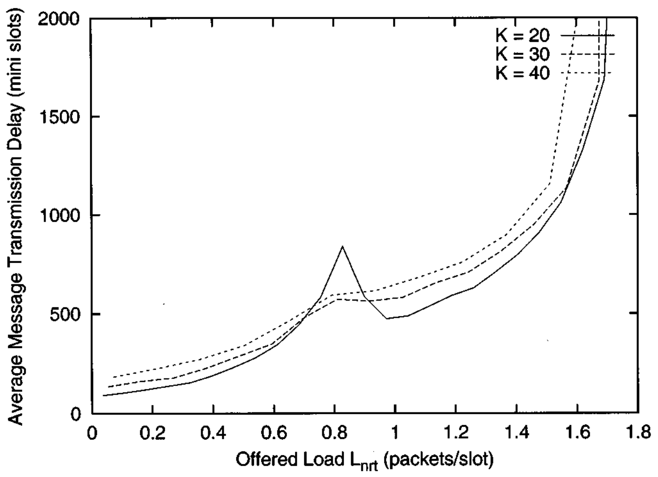


Fig. 15. Average uplink class II-A message-transmission delay versus non-real-time offered load: for $T_{\text{req}} = 600$ and with different values of K.

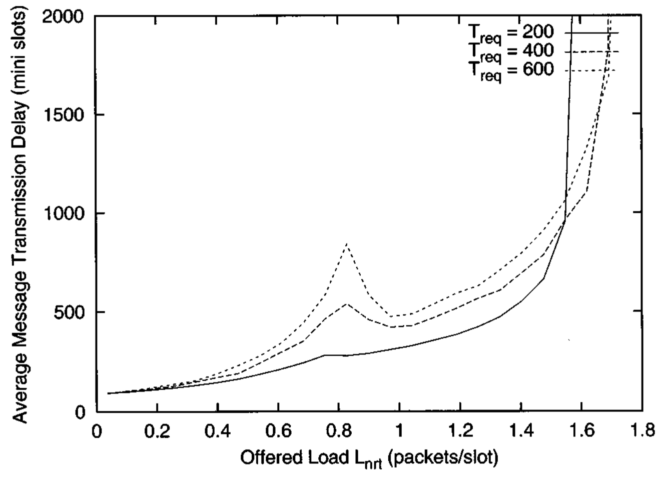


Fig. 14. Average uplink class II-A message transmission delay versus non-real-time offered load: with different values of T_{req}: $K = 20$.

that of class II-A continues to increase. When the maximum achievable throughput is reached, the throughput is saturated there, i.e., about 0.8 for both only downlink (marked with 'down') and only uplink (marked with 'up') arrival cases. Note that the throughputs for both downlink and uplink arrival case (marked with 'down/up') for a given offered load are larger than, or equal to, those for the other two extreme cases. The maximum achievable throughput for the latter case is found to be about 0.86. The reason why the maximum achievable throughput is less than one is due to the overhead of transmission-request slots, polling mini-slots, ACK/NAK-transmitting mini-slots, and channel probing processes.

So far, we used the value of $T_{\text{req}} = 200$, i.e., the average interval between two transmission-request slots is about 200. Fig. 14 shows the average transmission delays of uplink class II-A messages for three different T_{req} values. Note that the larger T_{req}, the larger the maximum achievable throughput could be. The delay is found to increase abruptly at around $L_{\text{nrt}} = 0.8$ for a large T_{req}. This is because the larger T_{req}, the more chance of a mobile having a message to request upon appearance of a transmission-request slot. So, the larger T_{req}, the more likely a request is to collide with others, and the larger the delay eventually. This effect is lessened dramatically after $L_{\text{nrt}} = 0.8$ due to the increases of piggybacked requests. Even though it is not shown here, the access latency of real-time connection requests also has a similar abrupt increase at some regions of offered load with large T_{req}. This observation implies that too large a T_{req} is not desirable.

Fig. 15 shows the effect of K values when $T_{\text{req}} = 600$. As K increases, the peak delay at $L_{\text{nrt}} = 0.8$ is found, on average, to decrease while the delay increases. A large K means a large packet size and an increased number of request mini-slots in a transmission-request slot. The decreased peak delay is due to the second fact, and the increased average delay is due to the first fact.

Note that too large a K is not good even though the peak delay is reduced because a large fixed-size packet is likely to be transmitted with some empty portion, thus lowering the system utilization. From the above two figures, one can conclude that the choice of K and T_{req} is closely related to each other, and hence, they should be considered together so that there is no undesirable peak in the average delay.

VI. Related Work

Recently, there have been significant research efforts to support QoS guarantees in wireless networks. A wireless MAC protocol in [9] can be considered as a polling scheme using stop-and-go queueing for real-time traffic, and FIFO queueing for non-real-time traffic. However, this scheme can support only a finite number of delay bounds due to the inherent limitation of stop-and-go queueing. The authors of [6] dealt with the scheduling and admission control for a TDMA system to support variable bit rate (VBR) connections with different packet-dropping probabilities. However, it deals with the uplink only, and connections are allowed to have only two different delay bounds. Both of the schemes do not provide proper means to handle wireless channel errors. Remote-queueing multiple access (RQMA) in [15] also addresses how to provide QoS for heterogeneous traffic using existing packet-scheduling algorithms designed for wired networks. This scheme also handles channel errors via a FEC/ARQ hybrid like ours. However, it did not consider how to handle location-dependent errors.

An emerging WLAN MAC standard, IEEE 802.11, provides both real-time and non-real-time communication services using polling and carrier sense multiple access/collision avoidance (CSMA/CA) [11]. The CSMA-based protocol is not suitable for a network environment with the star topology considered in this paper, since carrier sensing is not possible in such an environment. The IEEE 802.11 also does not specify the polling order, admission control, and how to handle channel errors which are all essential for real-time communication.

QoS-supporting wireless systems have recently been studied in the context of wireless ATM. The distributed queueing request update multiple access (DQRUMA) is used for BAHAMA as a MAC protocol [14], [18]. DQRUMA is a demand-assignment access scheme, which has some limited support for real-time traffic due to its contention-based access of transmission requests. In fact, the polling for uplink non-real-time transmission in our system can be classified as this demand-assignment access scheme because transmission requests are made on a per-message basis, and the permission is given via polling. The MAC protocol for a wireless ATM in [7] uses a polling scheme. However, it deals with uplink accesses only. the seamless wireless ATM network (SWAN) [1] also uses a polling scheme as a MAC protocol, but such details as polling scheduling and admission control were left unaddressed. WATMnet [28] uses a dynamic TDD/TDMA as the MAC to support heterogeneous traffic. The MAC protocol in WAND (MASCARA) [26] is another MAC protocol for a wireless ATM system (named WAND) using reservation and contention-based TDD/TDMA techniques. The authors of [19] proposed a wireless MAC protocol for wireless ATM, in which the access and scheduling procedures are distributed. Basically, none of the above wireless ATM schemes deals with location-dependent and bursty errors explicitly.

There have also been remarkable research efforts for QoS-provisioning CDMA systems, e.g., [3], [10]. While the CDMA technology is expected to dominate for the third-generation (3G) wireless systems, it is not suitable for typical WLAN environments since the data rate of an individual user can hardly be commensurate with the system bandwidth due to the spreading factor. The authors of [4] provide a good survey of MAC protocols covering both TDMA and CDMA-based schemes for wireless multimedia networks while a good survey of wideband local access schemes comparing both wireless LAN and wireless ATM technologies can be found in [24].

Recently, there have been significant research efforts on packet scheduling to handle location-dependent and bursty errors based on channel-prediction mechanisms and packet-transmission deferments [5], [16], [21]–[23], [27]. Most of the work has focused on how to provide (long-term) fair access to a wireless link in the presence of location-dependent errors [16], [21]–[23], [27]. Moreover, some of these have limitations, such as the issue of expediting the retransmission of packets was not addressed [16], [22], [23]. We adopted two different types of packet-scheduling policies to handle both real-time and non-real-time traffic, in which EDF scheduling is used for real-time traffic to bound packet-delivery delays at the cost of some possible packet drops, and round-robin scheduling is used for non-real-time traffic to achieve both error-free transmissions and long-term fairness among different mobiles.

Our scheduling for real-time communication guarantees that each connection receives the service it was promised as long as its channel continues to be good while a connection with deferred transmissions is served in a best effort manner. In that sense, our scheduling is different from those in [21]–[23], which compensate the bandwidth of connections which experienced deferments by reducing the bandwidth of other connections which received more bandwidth than promised before. Our non-real-time scheduling, however, is based on these scheduling algorithms. The scheduling policy in [27] does not have an explicit concept of compensation like our real-time scheduling, and it does not address how to bound packet delay bounds either. Last, our real-time scheduling is deadline-driven, so it can bound delays without addressing the fair access issue explicitly, but fairness is also a main concern in our non-real-time scheduling.

We adopted the polling scheme as in [1], [7], and [9] for both real-time and non-real-time traffic with different polling and scheduling strategies. The D-TDD technique in [1], [9], [15], [26], [28] was also used for more efficient and flexible utilization of a frequency channel. We showed how to support the distinct QoS requirements of heterogeneous traffic (i.e., bounded delivery-delay for real-time and virtually loss-free transmission for non-real-time traffic), including the transmission/polling scheduling and admission control for real-time connections, in the dynamic environment with location-dependent, time-varying, and bursty errors.

VII. Conclusion

This paper considered how to support both real-time and non-real-time communication services in a WLAN. We proposed a MAC protocol based on polling mobiles for their uplink accesses with slotted ALOHA-based and piggybacked transmission requests. A variation of EDF scheduling is used for real-time traffic to bound delivery delays at the cost of some packet losses depending on the channel conditions while a variation of round-robin scheduling is used for non-real-time traffic to achieve error-free transmissions and long-term fairness among mobiles. Also addressed are the problems of 1) multiplexing downlink and uplink traffic for D-TDD transmission and 2) handling location-dependent, time-varying, and bursty channel errors. An admission test for each new real-time connection request has been established, determining if the requested packet delivery-delay bound can be guaranteed without violating the existing guarantees.

An extensive evaluation study has shown the proposed scheme to work well and meet the design goals. We first considered the admission regions of real-time connections for four types of connections, and the average performance of real-time traffic. We also examined the interworking of real-time and non-real-time traffic while emphasizing the access delays of non-real-time messages, real-time connection requests, and handoff requests. Last, the performance of non-real-time communication service was evaluated in detail.

REFERENCES

[1] P. Agrawal , E. Hyden, P. Krzyzanowski, P. Mishra, M. B. Srivastava, and J. A. Trotter, "SWAN: A mobile multimedia wireless network," *IEEE Personal Commun. Mag.*, pp. 18–33, Apr. 1996.

[2] H. Ahmadi, A. Krishna, and R. O. LaMaire, "Design issues in wireless LANs," *J. High-Speed Networks*, vol. 5, no. 1, pp. 87–104, 1996.

[3] I. F. Akyildiz, D. Levine, and I. Joe, "A slotted CDMA protocol with BER scheduling for wireless multimedia networks," *IEEE/ACM Trans. Networking*, vol. 7, no. 2, pp. 146–159, Apr. 1999.

[4] I. F. Akyildiz , J. McNair, L. Carrasco, R. Puigjaner, and Y. Yesha, "Medium access control protocols for multimedia traffic in wireless networks," *IEEE Network Mag.*, vol. 13, no. 4, pp. 39–48, July–Aug. 1999.

[5] P. Bhagwat, P. Bhattacharya, A. Krishna, and S. K. Tripathi, "Using channel state dependent packet scheduling to improve TCP throughput over wireless LANs," *ACM Wireless Networks*, vol. 3, pp. 91–102, 1997.

[6] J. M. Capone and I. Stavrakakis, "Delivering diverse delay/dropping QoS requirements in a TDMA environment," in *Proc. ACM/IEEE MobiCom '97*, 1997, pp. 110–119.

[7] C.-S. Chang , K.-C. Chen, M.-Y. You, and J.-F. Chang, "Guaranteed quality-of-service wireless access to ATM networks," *IEEE J. Select. Areas Commun.*, vol. 15, pp. 106–118, Jan. 1997.

[8] S. Choi and K. G. Shin, "Centralized wireless MAC protocols using slotted ALOHA and dynamic TDD transmission," *Performance Eval.*, vol. 27 and 28, pp. 331–346, Oct. 1996.

[9] ——, "A cellular wireless local area network with QoS guarantees for heterogeneous traffic," *ACM Mobile Networks Applicat.*, vol. 3, no. 1, pp. 89–100, 1998.

[10] ——, "An uplink CDMA system architecture and diverse QoS guarantees for heterogeneous traffic," *IEEE/ACM Trans. Networking*, vol. 7, Oct. 1999.

[11] B. P. Chow, I. Widjaja, J. G. Lim, and P. T. Sakai, "IEEE 802.11 wireless local area networks," *IEEE Commun. Mag.*, pp. 126–133, Sept. 1997.

[12] D. Duchamp and N. Reynolds, "Measured performance of a wireless LAN," in *Proc. 17th Conf. Local Computer Networks*, Sept. 1992, pp. 494–499.

[13] D. Eckhardt and P Steenkiste, "Measurement and analysis of the error characteristics of an in-building wireless network," in *Proc. ACM SIGCOMM'96*, 1996, pp. 243–254.

[14] K. Y. Eng , M. J. Karol, M. Veeraraghavan, E. Ayanoglu, C. B. Woodworth, P. Pancha, and R. A. Valenzuela, "BAHAMA: A broadband ad-hoc wireless ATM local-area network," in *Proc. IEEE ICC'95*, 1995, pp. 1216–1223.

[15] N. R. Figueira and J. Pasquale, "Remote-queueing multiple access (RQMA): Providing quality of service for wireless communications," in *Proc. IEEE INFOCOM'98*, 1998, pp. 307–314.

[16] C. Fragouli, V. Sivaraman, and M. B. Srivastava, "Controlled multimedia wireless link sharing via enhanced class-based queuing with channel-state-dependent packet scheduling," in *Proc. IEEE INFOCOM'98*, Mar. 1998.

[17] D. D. Kandlur, K. G. Shin, and D. Ferrari, "Real-time communication in multi-hop networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 5, pp. 1044–1056, Oct. 1994.

[18] M. J. Karol, Z. Liu, and K. Y. Eng, "Distributed-queueing request update multiple access (DQRUMA) for wireless packet (ATM) networks," in *Proc. IEEE ICC'95*, 1995, pp. 1224–1231.

[19] M. Listanti, F. Mascitelli, and A. Mobilia, "D^{2}MA: A distributed access protocol for wireless ATM networks," in *Proc. IEEE INFOCOM'98*, 1998, pp. 315–321.

[20] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment," *J. ACM*, vol. 20, no. 1, pp. 46–61, Jan. 1973.

[21] S. Lu, T. Nandagopal, and V. Bharghavan, "A wireless fair service algorithm for packet cellular networks," in *Proc. ACM/IEEE MobiComm '98*, Oct. 1998, pp. 10–20.

[22] S. Lu, V. Bharghavan, and R. Srikant, "Fair scheduling in wireless packet networks," in *Proc. ACM SIGCOMM'97*, Sept. 1997, pp. 63–74.

[23] T. S. E. Ng, I. Stoica, and H. Zhang, "Packet fair queueing algorithms for wireless networks with location dependent errors," in *Proc. IEEE INFOCOM'98*, Mar. 1998, pp. 1103–1111.

[24] K. Pahlavan, A. Zahedi, and P. Krishnamurthy, "Wideband local access: Wireless LAN and wireless ATM," *IEEE Commun. Mag.*, pp. 34–40, Nov. 1997.

[25] K. Pahlavan and A. H. Levesque, *Wireless Information Networks*. New York: Wiley-Interscience, 1995.

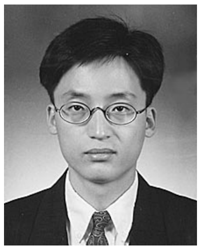
[26] N. Passas, S. Paskalis, D. Vali, and L. Merakos, "Quality-of-service-oriented medium access control for wirelss ATM networks," *IEEE Commun. Mag.*, pp. 42–50, Nov. 1997.

[27] P. Ramanathan and P. Agrawal, "Adapting packet fair algorithms to wireless networks," in *Proc. ACM/IEEE MobiCom'98*, Oct. 1998, pp. 1–9.

[28] D. Raychaudhuri *et al.*, "WATMnet: A prototype wireless ATM system for multimedia personal communication," *IEEE J. Select. Areas Commun.*, vol. 15, pp. 83–95, Jan. 1997.

[29] F. Swarts and H. Ferreira, "Markov characterization of digital fading mobile VHF channels," *IEEE Trans. Veh. Technol.*, vol. 43, pp. 977–985, Nov. 1994.

[30] H. Wang and N. Moayeri, "Finite state morkov channel—A useful model for radio communication channels," *IEEE Trans. Veh. Technol.*, pp. 163–171, Feb. 1995.



Sunghyun Choi (S'96–M'00) received the B.S. *(summa cum laude)* and M.S. degrees in electrical engineering from Korea Advanced Institute of Science and Technology in 1992 and 1994, respectively, and the Ph.D. degree from the Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, in 1999.

He is a Senior Member of Research Staff at Philips Research, Briarcliff Manor, NY. His research interests are in the area of wireless/mobile networks with emphasis on the QoS guarantee and adaptation, in-home multimedia networks, connection and mobility management, multi-media CDMA, and link layer protocols.

Dr. Choi received the Korea Foundation for Advanced Studies Scholarship and the Korean Government Overseas Scholarship during 1997–1999 and 1994–1997, respectively. He received the Humantech Thesis Prize from Samsung Electronics in 1997.

Kang G. Shin (S'75–M'78–SM'83–F'92) received the B.S. degree in electronics engineering from Seoul National University, Seoul, Korea, in 1970, and the M.S. and Ph.D degrees in electrical engineering from Cornell University, Ithaca, NY, in 1976 and 1978, respectively.

He is currently a Professor and the Director of the Real-Time Computing Laboratory, Department of Electrical Engineering and Computer Science, The University of Michigan, Ann Arbor. He has authored and coauthored approximately 600 technical papers and numerous book chapters in the areas of distributed real-time computing and control, computer networking, fault-tolerant computing, and intelligent manufacturing. He has also co-authored (with C. M. Krishna) *Real-Time Systems* (New York: McGraw Hill, 1997). His current research focuses on quality-of-service sensitive computing and networking with emphases on timeliness and dependability. He has also been applying the basic research results to telecommunication and multimedia systems, intelligent transportation systems, embedded systems, and manufacturing applications. He was an Area Editor of the *International Journal of Time-Critical Computing Systems*.

Dr. Shin was a Distinguished Visitor of the IEEE Computer Society and an Editor of the IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED COMPUTING.