Differentiated QoS over Wireless TDMA Channels

Hongjun Zhang & Pawel Gburzynski
University of Alberta, Department of Computing Science
Edmonton, Alberta, Canada T6G 2E8

Abstract

We propose an approach to channel allocation in wireless TDMA systems aimed at accommodating sessions with diverse QoS requirements. The idea of our proposed solution can be stressed in two points: 1. every session receives a single slot within the frame, the size of this slot being tailored to the session's bandwidth requirements (with the granularity of one ATM cell); 2. bandwidth scheduling is delayed until the latest possible moment before the frame is due. We demonstrate the feasibility, robustness, and flexibility of our approach by illustrating its behavior on a synthetic load consisting of three traffic types with different QoS expectations.

Keywords: TDMA, QoS, wireless ATM, personal communication systems

1 Introduction

With the rapid growth of personal networking, the demand for non-voice services in mobile environments has been growing much more rapidly than those services could be deployed—because of bandwidth limitations and the restrictive nature of traditional inflexible solutions. New protocols for personal mobile communication must account for the presence of several different classes of traffic with diverse QoS requirements, and make sure that those classes coexist within the framework of limited bandwidth and flexibility of the mobile environment.

We introduce a new TDMA-based bandwidth allocation scheme for mobile applications, aimed at accommodating ATM cells belonging to traffic streams with different QoS requirements. Whereas other TDMA-based protocols discussed in the literature (including PRMA [9], DPRMA [6], C-PRMA [2], DRMA [11], D-TDMA [7, 12], RAMA [1], and DQRUMA[10]) use fixed-size slots within a frame and try to fit the offered traffic to that size, our protocol fits the slot size to the requirements of the mobile station. This solution offers better flexibility of bandwidth allocation than traditional methods, which directly translates into a lower overhead and better fulfillment of the QoS expectations of the diverse traffic types occurring in modern applications, including Personal Communication Systems.

2 The TDMA model

We consider a star shaped network consisting of a single base and a number of mobile stations. The mobile stations do not communicate directly with each other. All traffic in the network is carried between the base station and the mobile stations. The population of mobile stations may grow and shrink; their traffic patterns and bandwidth requirements may vary dynamically.

With TDMA, the same *uplink channel* is shared by all mobile stations trying to transmit packets to the base station, while the *downlink channel* is used exclusively by the base station to send to the mobile stations packets and control information. The uplink channel is organized into fixed-length frames and time-divided among the multiple mobile stations that try to reach the base at the same time. Depending on the details of the access protocol, the frame may include additional components, e.g., synchronizing signals, acknowledgements, contention slots.

All the protocols mentioned in the introduction operate within the above model. They divide every frame into a number of equal-length transmission slots and, possibly, a number of smaller minislots used for access contention. This approach is efficient if all the mobile stations have the same traffic patterns and bandwidth requirements. In a scenario with diverse applications involving VBR and data traffic, some protocols (e.g., D-TDMA and RAMA) have the flexibility of assigning two or more transmission slots to one mobile station, however, there are two problems with this solution. First, the granularity of bandwidth assignment is constrained by the slot size or, as in DPRMA, by some multiples of slot size. Since the slot length is usually tailored to efficiently accommodate CBR traffic (i.e., voice), its length may not fit very well the requirements of non-CBR traffic. Second, each slot requires a guard time at the beginning and end equal to the maximum propagation delay across the cell, as well as a synchronizing preamble preceding the actual data. Therefore, if multiple slots are assigned to the same mobile in one frame, some bandwidth is wasted on multiple guards within a *de facto* single transmission unit. This overhead tends to increase linearly with the transmission rate and cell size.

All the TDMA protocols mentioned in the introduction have a similar channel structure. Time on the uplink channel is divided into frames, and each frame is divided into a number of equal length transmission slots and a number of possibly smaller minislots used for contention resolution. This approach is efficient if all mobiles have the same traffic patterns and bandwidth requirements, e.g., as in a cellular voice system. In a scenario with diverse applications involving VBR and data traffic, some protocols (e.g., D-TDMA and RAMA) have the flexibility to assign two or more transmission slots to one mobile station, as needed to satisfy its dynamic bandwidth requirements. One problem with this solution is that the granularity of bandwidth assignment is constrained by the slot size or, as in DPRMA, by some multiples of slot size. Since the slot length is usually tailored to efficiently accommodate one traffic type, i.e., voice, it may not fit very well the requirements of non-CBR traffic.

Even the flexible schemes that can allocate multiple slots to a single source suffer from a bandwidth wastage resulting from the fact that each of the multiple slots requires individual "framing." This framing consists of a guard time at the beginning and end of the slot equal to the maximum propagation delay across the cell, as well as a synchronizing preamble preceding the actual data. Therefore, if multiple slots are assigned to the same mobile in one frame, some bandwidth is wasted on multiple slot boundaries within a *de facto* single transmission unit (clearly visible if the multiple slots are adjacent, e.g., as in D-TDMA). This overhead tends to increase linearly with the transmission rate and cell size.

2.1 Silent periods in CBR traffic

Voice traffic, which is the most common representative of the CBR class, is known to be intermittent with a duty factor of approximately 3/8 [8]. Some TDMA protocols assume that individual talkspurts in a CBR session are separate "sessions", which have to individually request bandwidth as they occur. With this approach, it may be possible to transmit non-CBR traffic during silent periods of a CBR session [4]. One problem with such protocols is that they admit the possibility of a collision when the voice source resumes its talkspurt, and the station may lose its reservation in such a case.

An alternative approach is to keep the CBR slot reserved for the entire duration of the actual voice session, including the silent periods. This approach is more wasteful because the bandwidth used to sustain a silent CBR session cannot be used for anything else. Unless the protocol explicitly accounts for the presence of CBR sessions that have temporarily become silent (and treats them in a special way), the choice between the two options need not belong to the protocol specification.

To be able to reuse the bandwidth temporarily relaxed by a silent CBR session, in a way that will make that bandwidth available at the next burst, the base station must receive advance notifications about the status of the current burst and be able to accommodate the new burst, preferably without excessive and nondeterministic contention, when it shows up. The concept of piggybacking seems to be an essential prerequisite for this capability. For example, if the only way for the base station to learn that the current burst has ended is to receive an empty CBR slot (like in PRMA or D-TDMA), the empty slot is irreparably wasted. On the other hand, in DQRUMA, as stations piggyback their advance bandwidth requirements onto transmitted (full) slots, a CBR source can indicate that the burst is about to end early enough for its next slot to be reassigned to another mobile.

This feature must be combined with a way of reverting a silent CBR session to its active state at the beginning of the next burst. Although the bandwidth scheduler at the base station can freely preempt less important sessions to make room for the new burst, it must be able to learn that the silent session has become active. Ideally, the mobile should keep a tiny portion of its original bandwidth while in the silent state—just enough to be able to "piggyback" the new status of its session. With the rigid organization of the frame (where bandwidth comes in fixed-size slots) this approach is not feasilble.

3 DS-TDMA/CP

Our proposed protocol is dubbed DS-TDMA/CP, for *Dynamically Slotted TDMA with Contention Permission*. Assume that the traffic in the system consists of ATM cells. The frame length is fixed and set to coincide with the packet arrival rate of the CBR traffic. The granularity of bandwidth allocation within a frame is one ATM cell.

The frame is dynamically divided into two sections. The first (contention) section consists of a sequence of minislots. The second (transmission) section is partitioned into a number of variable-length transmission slots. An access request packet sent by a mobile in a minislot includes the service type, the mobile ID, the requested length of the transmission slot, and the deadline for the request.

The sequence of ATM cells transmitted by a mobile within its slot starts with a slot header, which is used to piggyback the station's further requirements for bandwidth. In particular, if the station has been transmitting CBR packets and its burst has run out, it will indicate in the slot header that its slot should be released in the next frame. Similarly, a station sending VBR traffic will indicate in the slot header the requested size of the next slot and the due date for this request.

3.1 Contention resolution

The contention resolution part of DS-TDMA/CP can be implemented independently of the remaining elements of the protocol. For simplicity, we opt here for a variant of slotted ALOHA inspired by [5, 10]. When a mobile station is ready to issue a request, it randomly selects a contention minislot and transmits the reservation packet. If the request makes it to the base station, the mobile will hear a scheduling message with its own ID on the downlink channel. If this doesn't happen within the current frame, the station will assume that a collision has occurred and make another try in the next frame with the probability

$$P_n = \min\left\{\frac{S_n}{S_c} \times \frac{P_c}{P_c+1}, 1\right\} \qquad (1)$$

where P_c is the retransmission probability used for the previous request (set to 1 for the first attempt), S_n is the total number of contention minislots available in the upcoming frame, and S_c is the number of contention minislots in the last frame. This algorithm resembles the "harmonic" backoff described in [4, 10], and additionally accounts for the dynamic structure of the contention section.[1] Thus, if the number of contention minislots in the upcoming frame is reduced, the retransmission probability will decrease faster than with the straightforward harmonic algorithm. On the other hand, when more contention minislots become available, the retransmission probability will decrease by less or, possibly, increase.

Other collision resolution protocols, e.g., deterministic ones based on the Binary Tree Algorithm [3, 10], may be good alternatives to ALOHA. Although the performance of DS-TDMA/CP may vary depending on the collision resolution scheme used, the protocol may still be compared with other similar solutions, as long as they all use similar methods of resolving contention. Besides, as other collision resolution schemes are more deterministic (and offer lower delays) than ALOHA, the ALOHA-based version of DS-TDMA/CP can be viewed as the worst case.

[1] Our backoff function reduces to the harmonic backoff, if the length of the contention section remains fixed across frames.

3.2 Transmission scheduling

We consider the following traffic classes (in the decreasing order of their priorities): CBR, RT-VBR, NRT-VBR, ABR, UBR. These classes are indexed by numbers from 0 (CBR) to 4 (UBR). Each of them has its specific QoS requirements; therefore, the access requests arriving at the base station are stored in different queues directly corresponding to the traffic classes. Requests in each queue are stored (and processed) in the nondecreasing order of their deadlines. The scheduler examines the request queues in the order of their priorities. While doing this, the scheduler generates scheduling messages, one message per each request that has been granted.

New requests, i.e., ones that arrive in contention minislots, are always acknowledged, even if they cannot be granted immediately. In response to such a request, the scheduler sends a scheduling message with zero slot length. This way the mobile will know that its request has made it to the base, and it will not have to be re-issued until its due date. A request that cannot be fulfilled by its due date is dropped by the base station.

A non-CBR source that cannot finish its transmission in the current frame (which fact is signaled by a piggybacked request in the header of the currently transmitted slot) is put back into the corresponding access request queue. On the other hand, once a CBR source is admitted by the scheduler, it will be receiving a guaranteed slot in subsequent frames until the end of its session. Although in principle the size of this slot is fixed, it may temporary shrink to the bare header during silent periods.

The amount of time available to the scheduler to complete its task is limited by the size of the last slot in the current frame. To make a good use of this limited time, the scheduler broadcasts the layout packet on-the-fly, while it is performing the bandwidth allocation. Whenever a slot space is assigned to a mobile, the scheduler emits a short message identifying the mobile and specifying the size of the allocated slot as a number of ATM cells. The allocation loop of the scheduler can be exited in one of two ways. First, it may happen that all the available bandwidth has been allocated or there are no more pending requests. Second, an internal timer may go off indicating that the scheduler must finish immediately because the frame is due. Intentionally, the second case should occur very seldom and possibly never. We admit it because the protocol can operate sensibly in such circumstances.[2] The worst that can happen is that some (low-priority) requests may not be fulfilled, even if enough bandwidth remains available.

Following the bandwidth allocation stage, the scheduler broadcasts to the mobile stations the number of minislots in the next frame and the contention permission flags (CPF) indicating which traffic classes are allowed to contend for bandwidth. The role of CPF is to selectively restrain traffic classes when the bandwidth is scarce.

The scheduling algorithm

We assume the following notation for the constants and variables used by the algorithm:
Constants: B–the total amount of bandwidth available in an empty frame; B_u–the amount of bandwidth needed to sustain a silent CBR session (i.e., the bare slot header); B_s–the amount of bandwidth needed for one contention minislot; C_{max}–the maximum index of a traffic class; S_{max}–the maximum number of minislots in a frame; B_v–an array indexed by traffic classes indicating the (average) amount of bandwidth[3] needed by a session in a given class; S_{incr}–an array indexed by traffic classes used to determine the number of extra minislots to be included in the contention section of the next frame.
Variables: S–the calculated number of minislots in the upcoming frame; CPF–the calculated setting of the contention permission flags; B_q–the amount of bandwidth temporarily released by silent CBR sessions; B_a–the calculated amount of bandwidth still available within the frame; B_{asg}–an array indexed by traffic classes indicating how much bandwidth has been allocated to the given traffic class.

The algorithm

1. Set $S := 0$, $B_a := B$, $B_q := 0$, $B_{asg}[0,\ldots,C_{max}] := 0$, CPF$[0,\ldots,C_{max}] := 1$.
2. For all CBR sessions in progress, and then for all pending CBR requets, perform steps 3 through 5, with M_r indicating the mobile source.
3. If there are no more requests to process, proceed to 6. If $B - B_{asg}[0] \geq B_v[0]$, proceed to 4. Otherwise, if the request arrived in the last frame, issue the scheduling message $< M_r, 0 >$. Continue at 3 for the next request.
4. If the session is currently silent, set $B_q := B_q + (B_v[0] - B_u)$, $B_r := B_u$; else set $B_r := B_v[0]$. Set $B_a := B_a - B_r$ and $B_{asg}[0] := B_{asg}[0] + B_v[0]$.
5. Transmit the scheduling message $< M_r, B_r >$ and continue at 3 for the next CBR session or request.
6. If $B - B_{asg}[0] < B_v[0]$, set CPF$[0] = 0$. Otherwise, set $S_r = \lfloor (B - B_{asg}[0]) / B_v[0] \rfloor$, $B_a := B_a - S_r \times B_s$, $S := S + S_r$.
7. Process the remaining traffic queues according to their priorities. For each request, perform steps 8 through 12, with M_r indicating the the mobile source that issued the request, C_r indicating the traffic class, B_r indicating the requested bandwidth.
8. If the internal timer went off (the frame is due), continue at 13.
9. If there are more requests in the current class C_r, proceed to 10. Otherwise, if $B_a + B_{asg}[C_r] < B_v[C_r]$, set CPF$[C_r] = 0$. If $C_r = C_{max}$, proceed to 13. Otherwise, continue at 8 with $C_r := C_r + 1$.
10. If $C_r > 1$, calculate the number of additional contention minislots to be included in the frame: $S_r = \lfloor (B_{asg}[C_r] + B_r) / S_{incr}[C_r] \rfloor - \lfloor B_{asg}[C_r] / S_{incr}[C_r] \rfloor$, and the resulting frame space requested: $B_d := B_r + S_r \times B_s$.
11. If $B_d > B_a$, proceed to 12. Otherwise, update $B_{asg}[C_r] := B_{asg}[C_r] + B_r$, $B_a := B_a - B_d$, $S := S + S_r$. Transmit the scheduling message $< M_r, B_r >$ and continue at 8 for the next request.
12. Request denied. If this request arrived in the last frame, transmit the scheduling message $< M_r, 0 >$. Continue at 8 for the next request.
13. Calculate the final number of minislots: $S := min\{S + (B_a/B_s), S_{max}\}$. Transmit $< S, \text{CPF} >$ and terminate.

The scheduling algorithm accounts for the CBR sessions in progress, which may temporarily become silent and release some bandwidth. This extra bandwidth can be recycled by non-CBR traffic, but it cannot be allocated to new CBR sessions. The scheduler makes sure that all admitted CBR sessions can always be accommodated into the frame, even if they all become active simultaneously. Therefore, when a new CBR request is being scheduled, the available bandwidth B_a is decremented by B_q, i.e., the amount of bandwidth temporarily released by the silent CBR sessions. This extra bandwidth is included in B_a, and it can be reused by other (non-CBR) sources without problems, as they never make reservations for more than one frame.

[2]This means that lower-cost hardware can be used if needed, e.g., because of budgetary constraints.
[3]Note that for the CBR class, this value is exact.

Note that although CBR traffic preempts any other traffic class at the bandwidth reservation stage, other traffic classes may take advantage of the silent periods within CBR sessions, which in turn are unavailable to new CBR sessions. Therefore, assuming that the silent periods within CBR sessions are bound to occur, there will always be some spare bandwidth available to lower priority traffic.

After all CBR requests have been processed (step 6), the number of minislots in the new frame is incremented by the number of additional CBR requests that could be accommodated. If no more CBR sessions could be admitted, CPF[0] is cleared to inhibit new CBR requests. Owing to the short due dates of CBR sessions and their relatively long duration, it makes no sense to admit and store new CBR requests if the sessions in progress have used up the entire bandwidth.

When allocating bandwidth to a non-CBR traffic class, the scheduler increases the number of minislots in the frame proportionally to the amount of bandwidth allocated to the class (step 10), to account for the fact that there exist higher priority classes that should be allowed to compete for that bandwidth. This does not happen when bandwidth is assigned to class number 1 (RT-VBR), because the extra minislots for the single class preceding it (CBR) are handled separately (in step 6).

The size of the contention section is determined by a simple heuristics. We assume that we know the average amount of bandwidth $B_v[i]$ allocated to a session in class i. Consider a situation in which some bandwidth B is allocated to an NRT-VBR session (class number 2) From the viewpoint of the RT-VBR class, this bandwidth is available; therefore, it makes sense to increase the size of the contention section by $B/B_v[1]$ minislots—to accommodate that many new RT-VBR contenders. If the same amount of bandwidth B is allocated to a session in class $i > 2$, we should account for the fact that there are several classes considered more important than i, and each of them has its specific average bandwidth requirements. Thus, we use the following harmonic formula:

$$S_{incr}[i] = \frac{1}{\sum_{j=1}^{i-1} \frac{1}{B_v[j]}} \ , \ 2 \le i \le C_{max}$$

which has the intuitively desirable property that $S_{incr}[2] = B_v[1]$ and $S_{incr}[i] < S_{incr}[i-1]$ for $2 < i \le C_{max}$. Note that, instead of being constants, the values $B_v[i]$, $1 \le i \le C_{max}$, can be updated on the fly, e.g., using an exponential averaging formula, depending on the measured actual amount of bandwidth allocated to sessions in a given class.

3.3 Processing at the mobile station

The information broadcast by the base station in the frame announcement packet must be consistently preprocessed by the mobile stations. This processing is straightforward and very inexpensive.

Consider a single selected mobile station receiving the announcement packet. The sequence of scheduling messages $< M_r, B_r >$ arrives before the number of minislots, so the station must wait until the very end of the packet before it can know the exact positions of slots. As a matter of fact, the station does not care about the positions of slots other than its own, which considerably simplifies the processing. The following algorithm lets the station acquire this knowledge.

Scheduling algorithm at the mobile

1. Set $P := 0$ (the current position within the frame), $P_s := -1$ (the position of your slot), $B_{max} := 0$ (the maximum length of a slot seen so far).
2. For all subsequent scheduling messages $< M_r, B_r >$, perform steps 3 through 5, and when done proceed to 6.
3. If M_r indicates this station, set $P_s := P$, $B_s := B_r$.
4. If $B_r > B_{max}$, set $B_{max} := B_r$, $P_{max} := P$.
5. Set $P := P + B_r$ and continue for the next scheduling message.
6. Read $< S, \text{CPF} >$ and calculate $P_m = S \times B_s$, where B_s is the amount of frame space needed for one minislot. P_m determines the length of the contention section.
7. If $P_s = -1$, terminate. No bandwidth has been allocated to this station. If the request was previously accepted (see step 8), the station must continue waiting until the due date and reissue the request if it isn't accepted by then. Otherwise, if the request was issued in the previous frame, it didn't make it to the base and must be reissued as soon as possible.
8. If $B_s = 0$, terminate. The request has made it to the base, but it hasn't been granted in this frame.
9. The longest slot is moved to the end. If $P_s > P_{max}$, set $P_s := P_s - B_{max}$; else if $P_s = P_{max}$, set $P_s := P - B_{max}$.
10. Offset the slot pointer by the amount of space used by the minislots: $P_s := P_s + P_m$.

If all mobile stations execute the same algorithm, they will all come up with the same layout of the new frame, including the location of the longest slot, which is implicitly moved to the very end of the frame.

3.4 Scheduling UBR traffic

For a traffic class other than UBR, the *requested bandwidth* specified by the source in its request packet is treated by the scheduler as the exact amount to be allocated. The scheduler assumes that a UBR source can sensibly use any bandwidth whatsoever, and the specification arriving in the request packet is viewed as the maximum.

Several fair scheduling strategies for UBR traffic are possible. With the simplest strategy, the outstanding UBR requests are processed in a round-robin fashion, and each of them receives as much bandwidth as available within the currently scheduled frame, up to the requested maximum. This approach, dubbed the *least effort policy* (LEP), minimizes the fragmentation of UBR packets into slots, and thus maximizes the throughput, but incurs the longest delays. Depending on the nature of the UBR applications, and the number of active UBR mobiles, this delay may be acceptable or not.

On the other end of the spectrum is the *maximum population policy* (MPP), which maximizes the number of mobiles that can be serviced within one frame. With this approach, UBR slots will tend to be shorter (incurring more bandwidth overhead), but the delays perceived by individual users will be shorter as well.

The entire spectrum can be described by a single policy parameterized by the *preferred granularity* of bandwidth allocation denoted by G_{ubr}. The idea is to allocate the UBR bandwidth in chunks of G_{ubr} cells, up to the amount requested by the source. Any leftovers are first spread among the sources that have already received bandwidth (but still need more), and then assigned to the remaining stations with the granularity as close to G_{ubr} as possible. If $G_{ubr} = \infty$, the resulting policy is LEP, if $G_{ubr} = B_{min}$, where B_{min} corresponds to the amount of bandwidth needed to accommodate a single ATM cell, we get MPP.

4 Sample results

Figure 1 best illustrates the behavior of our protocol regarding the coexistence of traffic patterns with different priorities. For clarity, we only consider three traffic classes: CBR, VBR, and UBR. The figure shows the fraction of the network's effective bandwidth used by each of the three traffic classes under increasing load conditions. The *load factor* of 0.8 indicates the average fraction of all mobiles (their total population is marked on the x-axis) involved in traffic sessions of the indicated type.

When the population of mobiles is small, there

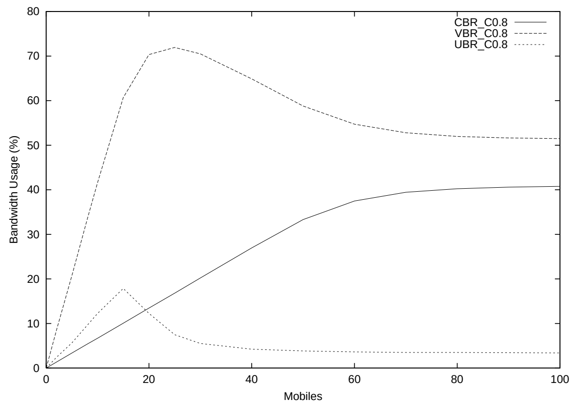

Figure 1: Bandwidth utilization in DS-TDMA/CP.

is enough bandwidth to accommodate all sessions without preemption. Then, the relative positions of the curves indicate the individual contributions of the three traffic types to the total offered load.[4] When the load reaches a certain threshold (about 88% capacity), the service received by UBR sessions begins to decline—to make room for CBR and VBR traffic. Note, however, that the system does not provide its maximum capacity at this point. Because all VBR and CBR sessions are satisfied, some bandwidth is set aside for additional contention minislots. This mode of operation is sustained for as long as all VBR requests are still satisfied. Having a higher priority than UBR requests, they are first to reuse the silent periods in the admitted CBR sessions. The system reaches its maximum capacity around 25 mobiles, when some VBR sessions begin to be rejected.

The amount of CBR traffic accommodated by the network increases with the load until its specific saturation threshold. The other two classes yield to CBR, with UBR additionally yielding to VBR, although neither of them dies out. In fact, VBR stabilizes at a level that is considerably higher than the saturation threshold of CBR. This is because CBR traffic is incapable of using more than a certain fraction of the entire useful bandwidth, roughly equal to $l_t/(l_t + l_s)$, where l_t and l_s are the average lengths of the talkspurt and silence periods. Similarly, UBR traffic is not completely eliminated by VBR. This is because VBR packets have much stricter bandwidth requirements than UBR packets, which can use practically any leftovers. Thus there is always some bandwidth unusable by VBR sources but still allocatable to the less picky UBR sessions.

In our numerous experiments, not reported in

[4]As the traffic generation processes are not memoryless, it is impossible to maintain specific fixed proportions of offered load.

Protocol	CBR = 32kbps			CBR = 64kbps		
	C	V	U	C	V	U
DS-TDMA/CP	9.9	7.1	5.1	10.7	6.8	4.8
D-TDMA	15.2	15.2	15.2	17.7	16.3	15.2
D-TDMA*	39.5	24.6	24.6	41.9	31.9	30.9
DQRUMA	17.4	17.4	17.4	17.4	17.4	17.4

Table 1: Bandwidth overhead percentage

this paper, we have compared the performance of our protocol with other TDMA solutions, notably D-TDMA, and DQRUMA. DS-TDMA/CP turns out to be consistently superior in handling prioritized traffic, and shows considerably better responsiveness to changing conditions in the network.

Table 1 compares the bandwidth overhead for three protocols under three traffic mixes (C–predominantly CBR, V–predominantly VBR, U–predominantly UBR) and and two CBR rates (constraining the frame size). The row labeled D-TDMA* includes the overhead caused by the silent periods in CBR traffic, which are unusable in D-TDMA. The overhead is calculated as the percentage of bandwidth of the uplink channel not used to transmit any data bits.

5 Conclusions

The TDMA protocol introduced in this paper offers differentiated quality of service to multiple traffic classes demanded by contemporary mobile applications. Owing to the variable and flexible slot size, our protocol incurs a lower bandwidth overhead than other solutions based on fixed size slots, especially when CBR traffic is not the dominant load in the network. By deferring scheduling decisions until the last possible moment and improving contention opportunities under light load, the proposed scheme is highly responsive to rapid changes in the traffic pattern. This feature also makes it possible to identify and reuse silent periods in voice sessions to accommodate other traffic, without forcing the voice sessions to contend for bandwidth again at the beginning of a new talkspurt.

One problem with our proposed solution is the relatively high demand on processing power at the base station. Although this demand appears to be well within the reach of contemporary hardware, its exact magnitude and dependence on the parameters of the network should be investigated and quantified. The implementation of the scheduling algorithm at the base station deserves some studies as well. These issues will be addressed in further research.

References

[1] N. Amitay. Distributed switching and control with fast resource assignment/handoff for personal communication systems. *IEEE Journal on Selected Areas in Communications*, 11:842–849, 1993.

[2] G. Bianchi, F. Borgonovo, L. Fratta, L. Musumeci, and M. Zorzi. C-PRMA: a centralized packet reservation multiple access for local wireless communications. *IEEE Transactions on Vehicular Technology*, 46:422–436, 1997.

[3] J. Capetanakis. Tree algorithms for packet broadcast channels. *IEEE Transactions on Information Theory*, 25:505–515, 1979.

[4] S. Choi and K. G. Shin. A cellular wireless local area network with QoS guarantees for heterogeneous traffic. *Mobile Networks and Applications*, 3(1):89–100, 1998.

[5] S. Choi and K. G. Shin. An uplink CDMA system architecture with diverse QoS guarantees for heterogeneous traffic. *IEEE/ACM Transactions on Networking*, 7(1):616–628, 1999.

[6] D. A. Dyson and Z. J. Haas. A dynamic packet reservation multiple access scheme for wireless ATM. In *Proceedings of IEEE MILCOM'97*, Monterey, CA, nov 1997.

[7] G. Falk, J. Groff, W. Milliken, M. Nodine, S. Blumenthal, and W. Edmond. Integration of voice and data in the wideband packet satellite network. *IEEE Journal on Selected Areas in Communications*, 1(6), 1983.

[8] K. S. Gilhousen *et al.* On the capacity of a cellular CDMA system. *IEEE Transactions on Vehicular Technology*, 40:303–312, 1991.

[9] D. J. Goodman *et al.* Packet reservation multiple access for local wireless communications. *IEEE Transactions on Communications*, 37(8):885–890, 1989.

[10] M. J. Karol, Z. Liu, and K. Y. Eng. Distributed-queueing request update multiple access (DQRUMA) for wireless packet (ATM) networks. In *Proceedings of IEEE International Communications Conference*, Seattle, WA, jun 1995.

[11] X. Qiu and V. O. K. Li. Dynamic reservation multiple access (DRMA): A new multiple access protocol for personal communication systems (PCS). *Wireless Networks*, 2(2), 1996.

[12] N. D. Wilson, R. Ganesh, K. Joseph, and D. Raychaudhuri. Packet CDMA versus dynamic TDMA for multiple access in an integrated voice/data PCN. *IEEE Journal on Selected Areas in Communications*, 11:870–884, 1993.