A Medium Access Control Framework for Wireless ATM Networks

Conference Paper · September 1996

CITATIONS
6

READS
8

4 authors:

Nikos Passas
National and Kapodistrian University of Athens
150 PUBLICATIONS **2,049** CITATIONS
SEE PROFILE

Stamatis Outsios
Athens University of Economics and Business
5 PUBLICATIONS **6** CITATIONS
SEE PROFILE

Dimitris Skyrianoglou
National and Kapodistrian University of Athens
25 PUBLICATIONS **252** CITATIONS
SEE PROFILE

Lazaros Merakos
National and Kapodistrian University of Athens
311 PUBLICATIONS **3,240** CITATIONS
SEE PROFILE

A MEDIUM ACCESS CONTROL FRAMEWORK FOR WIRELESS ATM NETWORKS

Nikos I. Passas, Stamatis Outsios, Dimitris Skyrianoglou and Lazaros Merakos
{passas | soutsi | grad0104 | merakos}@di.uoa.gr

Communication Networks Laboratory
Department of Informatics, University of Athens
Panepistimiopolis, Zografou - TYPA Buildings
157 84 Athens, Greece

ABSTRACT

In this paper, we consider a multiservice wireless ATM system where key issues related to scheduling of transmitted traffic and collision resolution in the radio interface are discussed. Considering scheduling, a technique that combines priority classes and the leaky bucket regulator is proposed. The scheduler aims at treating fairly all active connections, according to their contracts. The collision resolution algorithm is based on the general concept of stack algorithms and its main objective is to reduce the delay of each packet transmitted in the contention periods. A performance analysis, based on simulations, shows that the proposed algorithm attains much lower delays, compared to slotted-ALOHA, with a small increase in the length of the contention period.

I. INTRODUCTION

Wireless ATM is used as a term to refer to a system that combines the advantages of wireless operation and freedom of mobility, with the services and quality of service (QoS) guarantees of fixed ATM networks. The main challenge of such a system is to harmonise the development of broadband wireless systems with fibre-optics-based networks and offer similar advanced multimedia services with QoS guarantees [1]. An important system design issue for the wireless ATM network is the construction of an efficient medium access control (MAC) protocol for the radio interface. This protocol must be able to support all or a useful subset of ATM services with often conflicting requirements, and guarantee a QoS for every connection. For this purpose, usage parameter control (UPC) techniques proposed for fixed ATM networks, should be properly embedded in the MAC protocol.

In this paper, we consider a system consisting of ATM mobile terminals (AMTs), "connected" to an ATM premises switch through access points (APs). The ATM switch is considered as a standard ATM switch, probably connected to a fixed ATM network, with embedded functionality to support mobility of the AMTs (Figure 1). We propose a MAC protocol framework for the radio interface, and focus on a specific bandwidth allocation technique for the contention-free periods of the protocol, and a collision resolution algorithm for the contention-based periods.

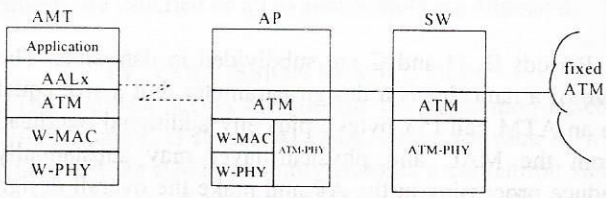

Figure 1: Generic wireless ATM architecture

The paper is organised as follows. Section II describes a time division multiple access (TDMA) based protocol framework for the radio interface of a wireless ATM network. In Section III, a dynamic bandwidth allocation technique, based on priorities and the leaky bucket regulator [2] is proposed. In Section IV we describe a collision resolution algorithm for the contention-based periods of the proposed protocol, and compare it with slotted-ALOHA. Finally Section V contains our conclusion.

II. MAC PROTOCOL FRAMEWORK

ATM services have different and often conflicting QoS requirements. The multiple access technique used in the radio interface of a wireless ATM network should be flexible enough to fulfil these requirements. TDMA seems to be the most suitable technique, basically due to its ability to adjust a user's bit rate by allocating more or fewer time slots, according to current needs and traffic load. For the sharing of the radio medium between uplink and downlink traffic, the choice is usually between frequency division duplex (FDD) and time division duplex (TDD). The scarcity of available frequencies and

the difficulty in implementing sharp filters in GHz frequencies, where wireless ATM systems are expected to operate, lead to the choice of a dynamic TDD scheme, where the portion of time allocated to each direction is dynamically determined, based on several factors, such as traffic load, service classes, and priorities.

For the dynamic TDMA/TDD protocol that we describe, time is divided into variable length time frames. Each frame is subdivided into five periods: Frame Header (FH), Uplink Request period (R), Downlink Contention-free period (D), Uplink Contention-free period (U), and, Uplink Contention-based period (C) (Figure 2).

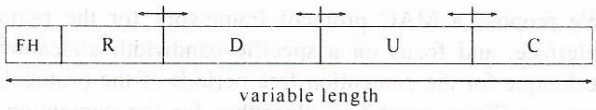

Figure 2: Frame structure

Periods D, U and C are subdivided in data slots. The size of a data slot is a design parameter, but a size equal to an ATM cell (53 bytes), plus any additional overhead from the MAC and physical layer may substantially reduce processing at the AP and make the overall design simpler. Period R is subdivided in request slots, used from the uplink connections to declare their needs in data slots. This is necessary for the implementation of usage parameter control (UPC) algorithms. UPC, already known from fixed ATM networks, is required in the radio interface to preclude malicious or unintentional misbehaviour of some connections, which can affect QoS of other conforming connections. The natural position of such a control mechanism is the crossing point of all connections, i.e. the AP. A special entity at the AP, called *"the scheduler"*, decides on the allocation of the data slots of the contention-free periods, based on traffic characteristics and QoS requirements, declared during connection setup, and current bandwidth needs of each connection. A more detailed description of the scheduler can be found in Section III. Downlink bandwidth needs are implicitly derived through ATM cells arriving at the AP, while uplink connections use the request slots. Depending on the allocation method, a request slot may contain a connection's identity (e.g., VPI/VCI, or radio specific), and the number of the requested slots. The number of request slots can be variable, depending on the number and the kind of the sources, and the access method used. Beside the simple slotted-ALOHA method, other access methods, such as stack-based algorithms, can be used to reduce collisions. A stack-based collision resolution algorithm is described in Section IV. To ensure that requests will pass to the AP in a reasonable tim collision probability must be kept considerably low.

As already mentioned, period C is used for contention-based data transmission and it is accessible by all uplink data connections. Although such a kind of transmission is not recommended for ATM networks, because unpredictable delays can lead to violation of QoS agreements, there are cases where such a technique can improve system performance. Suppose that in a certain frame the scheduler can satisfy all the requests issued by the active connections, and there is still free space. This can be declared as a contention-based period, referred to as period C, meaning that uplink connections can access it directly without previous allocation. It can be used as a general use period for transmission of data or control information with strict delay requirements that cannot wait for a request/allocation cycle. The access methods that can be used here are similar to those of period R. Period C can be considered as a low priority period. Its length depends mainly on the traffic load. Under light traffic conditions, the available bandwidth can lead to extension of period C. If the traffic is heavy, period C must be reduced to the minimum, or even omitted, to leave more bandwidth for contention-free traffic.

The frame header contains allocation information for the data slots of each frame and for the length of the contention period. Its structure is strongly related to the policy of the scheduler. If data slots for each connection are allocated consecutively, the frame header can contain a "slot-table" consisting of entries of the form (Connection-ID, Number-of-Slots). Connection-ID can be the VPI/VCI, or a more compact, radio specific identifier. If cell delay or cell delay variation constraints force to the spreading of the allocated slots, the exact position of each allocation should be somehow mentioned in the frame header.

III. TIME SLOT ALLOCATION

The time slots of D and U periods are allocated by the scheduler to specific connections, according to traffic characteristics, QoS agreements, and short term requests. Since both directions share the same radio channel and traffic is usually unbalanced, a single procedure should be used, handling both directions equally. In this respect, periods D and U can be considered as one period, referred to as DU. Depending on the turnaround overhead, interleaving of uplink and downlink allocations can be considered, but it is usually preferable to have all downlink allocations first, followed by the uplink allocations, to reduce the overhead, and result in higher bandwidth utilisation.

In order to get advantage of the statistical multiplexing gain, and ensure fair bandwidth allocation, the scheduler has to know the current bandwidth needs of each connection. As already mentioned, the uplink connections declare their needs through the request slots of period R,

while downlink needs are expressed directly through the ATM cells arriving at the AP. For the scheduler purposes, both uplink requests and downlink ATM cells are referred to as "requests". Since the requests can exceed the available data slots in DU period, a priority mechanism is required to determine which requests will be satisfied immediately, in the next frame, and which will be queued for later or discarded if they are expired. We propose a mechanism that is based on both ATM service classes and traffic characteristics of each connection. ATM cells belonging to real time classes have strict delay constraints and must be transmitted as soon as possible, while ATM cells belonging to non-real time classes can accept a delay in service from the scheduler, as long as mean delay and loss stay within the acceptable limits. The scheduler can give a priority to each connection, based on its service class:

Priority number	Service class
5	CBR
4	rt-VBR
3	nrt-VBR
2	ABR
1	UBR

The greater the priority number, the greater the priority of a connection for allocating slots.

If only the priority mechanism is used, the scheduler cannot determine if a connection is exceeding its declarations or not. For example, a rt-VBR connection requesting more bandwidth than negotiated can result in degradation of the QoS offered to a conforming nrt-VBR connection, only because it is higher in the priority list. Accordingly, an additional mechanism is required that will ensure fair treatment of all connections, based on their contracts. As already proposed in [3], a mechanism similar to the leaky bucket, referred to as *"the leaky-bucket scheduler"*, can be used. Simulation models used in [3], showed that this technique can attain fair treatment of all connections, according to their contracts.

According to the leaky-bucket scheduler, a token pool, located inside the scheduler entity, is introduced for every connection. Tokens are generated at a fixed rate corresponding to the guaranteed bandwidth, agreed during connection setup. The size of the pool is equal to the burstiness of the connection, also agreed during connection setup. For every slot allocated to a connection a token is removed from the appropriate pool. Thus, the state of each pool gives an indication on the declared bandwidth the corresponding connection has consumed at any instance of time. Each pool can be readily implemented as an integer variable, which is incremented by one on token generation and decremented by one on slot allocation.

The operation of the algorithm is as follows. Starting from priority 5 (CBR), and down to priority 2 (ABR), the scheduler satisfies requests of the connections of each service class, as long as tokens are available. UBR connections have no guaranteed bandwidth, thus no token pool corresponding to it. At every priority class, it is very possible to have more than one connections requesting slots. In that case, the scheduler gradually allocates one slot at a time to the connection (or connections) with the most tokens (i.e., highest token variable), decreasing the token variable by one. The rationale behind this is that the connection with the most tokens has consumed less bandwidth than declared, and has higher priority of allocating slots.

When the satisfaction of "conforming" requests is completed, and if there are still available slots, the scheduler tries to satisfy "exceeding" requests. At this state, the token variables of all connections requesting slots are less than or equal to zero. The scheduler follows the same procedure as before, starting from priority 5 (CBR), down to priority 1 (UBR). As before, if more that one connections belonging to the same priority class, request slots, one slot at a time is allocated to the connection with the highest token variable. Since this is exceeding traffic, decreasing will result in negative values for the token variables. The procedure stops when all requests are satisfied or all available slots are allocated.

IV. COLLISSION RESOLUTION

The simplest access method for the contention-based periods R and C is slotted-ALOHA, where for each ATM cell/request, an AMT randomly chooses a contention slot and transmits it. If another AMT transmits in the same slot, the ATM cell/request will collide and the AMT will have to retransmit it in a subsequent frame. The acknowledgement of a collision or not can be included in the FH of the next frame. These periods must be large enough to guarantee a collision probability that is acceptable by all connections. Since the number and the kind of the connections can vary, the length of these periods must vary as well.

Here we propose another algorithm for random access in the contention-based periods, that aims at reducing the mean resolution delay of each collision and the mean length of the contention periods, resulting in better throughput. It is based on the general concept of the stack algorithm, initially presented in [4], and it was influenced by ideas found in [5]. Below, we refer to both requests of period R and ATM cells of period C as "packets".

The operation of the algorithm is as follows. At any time, the first M slots of the contention period are exclusively used for new arrivals, while the rest for resolving collisions occurred in previous frames (the exact value of M is a design parameter). In an initial state, where no packets for transmission exist, the length of the contention period is M. Let $F_k(j)$ denote the feedback corresponding on the contention slot j of frame k. This feedback will be on the FH of frame k+1. The value of $F_k(j)$ can be one bit:

$F_k(j)$= C ,if slot j contained a collision
NC ,if slot j contained no collision (empty or successful transmission)

Let a packet x arrive at a user. The user utilises the feedback on the FH and implements the algorithm according to the following rules:

1) Assignment: At the beginning of each frame k the user assigns to packet x a non-negative integer $I_k(x)$.
2) Initialisation: For a packet x arrived during frame k, the user sets $I_k(x)=U_k(x)$, where $U_k(x)$ is a random variable uniformly distributed on {1,2,...,M}.
3) Transmission: In frame k, packet x is only transmitted in contention slot $I_k(x)$.
4) Updating: Every slot that a collision occurred is replaced by *n* new slots in the next frame and the collided packets are uniformly distributed to these slots. The exact value of *n* is a design decision and can be evaluated by simulation or analysis. If C_k is the number of contention slots of frame k, Cont_length(k) the length of period C in frame k, and $V_k(x)$ a random variable uniformly distributed on {1,2,...,*n*}, then updating for frame k+1 is performed according to the following algorithm:

```
Cont_Length(k+1)=M
for i=1 to C_k
  if F_k(i)=C
    foreach packet x where I_k(x)==i
      I_{k+1}(x) = Cont_Length(k+1) + V_{k+1}(x)
    endfor
    Cont_Length(k+1) = Cont_Length(k+1) + n
  endif
endfor
```

The algorithm can be implemented at the AMTs, to calculate the values of $I_{k+1}(x)$ or only at the AP and the results ($I_{k+1}(x)$) can pass to the AMTs through the FH. At the end of the procedure, the variable Cont_Length(k+1) contains the length (in data slots) of period C of frame k+1. An example of the method is shown in Figure 3.

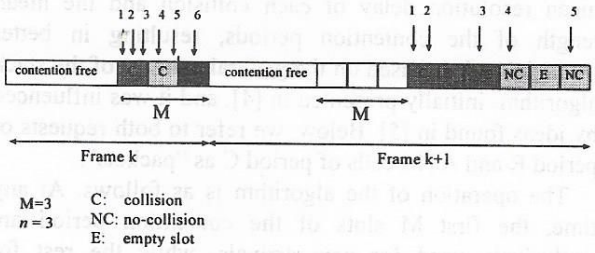

Figure 3: Example of the stack algorithm

To reveal the effectiveness of the algorithm, we used simulation methods to compare it with slotted-ALOHA. The simulation tool used was OPNET. In our model, we considered a radio channel with raw capacity of 20Mb/sec. The frame structure was the one described in Section II. We concentrated on the C period, so we considered constant length for periods R and DU, with a total length of 50 slots. M was set equal to 5 slots, and *n* equal to 3. The traffic destined for period C was considered Poisson, with variable mean rate. In Figure 4, we can see the improvement, in terms of successful transmission delay of a packet, that the stack algorithm can attain. As we can see, the improvement is considerably large for heavy traffic loads, and delay remains predictable even when slotted-ALOHA is saturated. Figure 5 presents the overhead of the algorithm, i.e., the extension of the contention period. As we can see, starting with a contention period length of M=5, the overhead is considerably low (less than a slot in most cases), compared to the improvement in successful transmission delay.

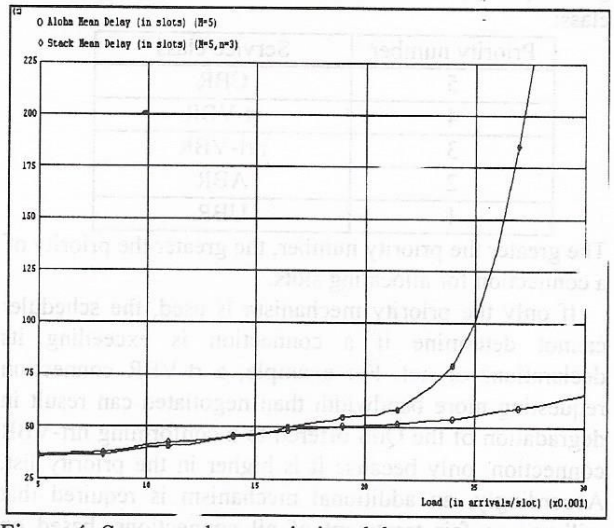

Figure 4: Successful transmission delay versus traffic load both stack algorithm and slotted-ALOHA

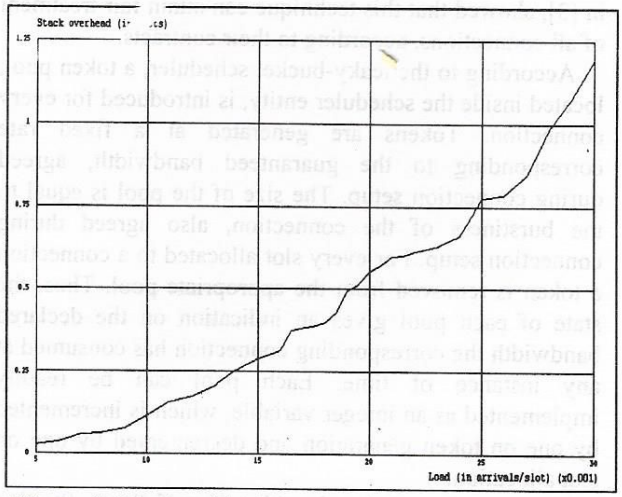

Figure 5: Overhead in the contention period when using the stack Algorithm

V. CONCLUSIONS

We have presented a TDMA/TDD MAC protocol framework for the radio interface of a wireless ATM network, and proposed a scheduler for the contention-free periods and a collision resolution algorithm for the contention-based periods.

The scheduler is a combination of a priority mechanism and a leaky-bucket-based technique, and aims at ensuring fair treatment of all connections, according to their contracts. The collision resolution algorithm belongs to the family of stack-based algorithms, and can be used to avoid unpredictable delays for packets transmitted in the contention-based periods. Simulation models showed that the proposed algorithm results in much lower delays than slotted-ALOHA, especially in heavy loads, with a small increase of the length of the contention period.

REFERENCES

[1] D. Raychaudhuri and N.D. Wilson, "ATM-Based Transport Architecture for Multiservices Wireless Personal Communication Networks", IEEE Journal on Sel. Areas in Commun., vol. 12, no. 8, pp. 1401-1414, Oct. 1994.

[2] J.S. Turner, "New Directions in Communications (or Which way to the Information Age?)", IEEE Communications Magazine, vol. 25, no. 10, pp. 8-15, Oct. 1986.

[3] N. Passas, N. Loukas, and L. Merakos, "A Leaky-Bucket Based Scheduling Technique for Wireless Personal Communication Networks", in Proc. International Conference in Telecommunications, pp. 306-312, Istanbul, April 1996.

[4] B.S. Tsybakov and N.D. Vvedenskaya, "Random multiple-access stack algorithm", Probl. Peredach. Inform., vol.16, no.3, pp.80-94, July-Sept. 1980.

[5] L.F. Merakos and C.Bisdikian, "Delay Analysis on the n-Ary Stack Random-Access Algorithm", IEEE Trans. on Inform. Theory, vol.34, no.5, Sept. 1988.