

# A Near Optimum MAC Protocol Based on the Incremental Collision Resolution Multiple Access Protocol for CDMA based Communications System

**S.Malarvizhi,**  
SRMEC,  
Kattankulathur, India  
[malarvizhi@rediffmail.com](mailto:malarvizhi@rediffmail.com)

**Karthik Ramani**  
SRMEC,  
Kattankulathur, India  
[electrokar@hotmail.com](mailto:electrokar@hotmail.com)

**Hariharan  
Rajasekaran**  
SRMEC, India  
[electrohari@rediffmail.com](mailto:electrohari@rediffmail.com)

**Dr.M.Meenakshi,**  
Anna University,  
India  
[meens68@yahoo.com](mailto:meens68@yahoo.com)

## Abstract

*This paper presents and analyses a new near optimum medium access control (MAC) protocol. The proposed access scheme is suitable for a CDMA based communication system and keeps under control and upper bounded the number of simultaneous transmissions. The protocol's throughput performance approaches that of an ideal channel access protocol based on a distributed transmission queue and incremental collision resolution.*

*The proposed protocol extends the ICRMA protocol to a CDMA environment. ICRMA dynamically divides the channel into cycles of variable length. Each cycle consists of a contention period and a queue transmission period. ICRMA resolves collisions in lesser number of steps than the DQRAP protocol, which is one of the best protocols based on collision resolution proposed to date.*

## Keywords

Incremental Collision Resolution Multiple Access, Code division multiple access, Throughput, Delay, Collision.

## I. INTRODUCTION

Many of the medium access control (MAC) protocols for wireless LANs proposed to date are based on a collision avoidance dialogue between senders and receivers, e.g., [1]. A sender sends a request-to-send (RTS) to the receiver, who in turn sends a clear-to-send (CTS) if it receives the RTS free of errors; only then can the sender transmit a data packet. These protocols solve collisions by backing off and rescheduling RTS transmissions. As with CSMA protocols, this procedure yields good results if the RTS traffic is low, but is inherently unstable. As the RTS transmission rate increases, the constant RTS collisions can cause the channel to collapse, bringing the flow of data packets to a halt when no new data transmission queues can be started.

A way to stabilize the system is by increasing the retransmission delays; however, a more efficient way can be devised by using collision resolution. Several stable MAC protocols have been proposed in the past based on tree-splitting algorithms for collision resolution (e.g., [2]).

Those protocols in which data packets are used to resolve collisions achieve throughput below 0.6 [3]. Several MAC protocols have been proposed that implement collision resolution using either control packets that are much smaller than data packets, or are based on the ability of the transmitter to abort transmission rapidly after detecting collision (e.g. [4]). Among those stable MAC protocols that achieve high throughput, some build a separate queue for the transmission of data packets, in addition to the stack or queue of the control packets used for collision resolution.

The ICRMA protocol [8], proposed by Rodrigo Garcés and J.J. Garcia-Luna-Aceves operates with a collision-resolution stack for control packets and a distributed queue for data packets. ICRMA does not require time slotting and minislots to operate like prior similar protocols did (e.g. DQRAP [5], and the announced arrival protocol [6]) and does not require base stations to understand simultaneous transmissions (e.g., TRAMA [7]). ICRMA builds a distributed transmission queue dynamically using a deterministic tree-splitting algorithm.

ICRMA builds a distributed transmission queue dynamically using a deterministic tree-splitting algorithm. A station attempts to join the transmission queue during contention intervals by sending an RTS to any intended

**CTS**  
queue. RTSs are sent according to a tree-splitting algorithm that resolves collisions. CTSs are sent to the queue that arrive during the contention period. Access time to the channel is shared among all members of the queue. Transmissions for all members of the queue are allowed, which we call a queue-transmission. Short contention periods during which we call a queue-transmission. The queue is a variable-length train of packets. Packets are added to the transmission queue until completing a collision-resolution period. A single step of collision resolution is successful, and idle step or a collision is allowed in each contention period. Packets are used in each contention period. Packets.

In this paper, we propose a near-optimum medium access protocol that modifies and extends the ICRMA techniques for use in a CDMA environment. For this purpose, the idea of spreading codes is introduced. As in ICRMA the channel is divided into cycles of variable length. Each cycle consists of a contention period and a queue transmission period. The stations access the channel using a particular spreading code, which is determined by a counter maintained by all the stations in the system. The modified protocol was simulated with Throughput and Delay as the main performance parameters. The results obtained show a significant improvement in the system's delay and throughput performance.

The paper is organized as follows. The protocol description is detailed in Section II. Section III shows the working of the protocol. Section IV is devoted to simulation results for the protocol and Section V is devoted to conclusion.

## II. PROTOCOL DESCRIPTION

Let us consider  $N$  data terminals, which share a CDMA channel with  $K$  available spreading codes to communicate with a base station. The time axis is divided into cycles, with each such cycle consisting of a small, dynamically sized contention period and a dynamically sized queue transmission period. Stations are assumed to monitor the state of the channel while they are not transmitting, and know the current state of the channel. A counter maintained distributedly in all the stations keeps track of the spreading codes used in the system. Initially this counter is set to zero and its maximum value is  $K$ . The processing delays for the stations are assumed to be very negligible in this protocol.

Stations compete with each other to enter a transmission queue during the contention periods between queue transmissions periods in order to transmit their messages. Stations follow a non-persistent CSMA strategy for the transmission of **RTS**s with which they request to be added to the transmission queue.

The protocol uses a deterministic tree splitting algorithm to resolve collisions of **RTS**s. A step in the contention period can be a **RTS** collision period, a successful **RTS** exchange or an idle period. Once the collision resolution algorithm is started by the collision of **RTS**s from a set of stations, other stations that want to be added to the transmission queue make use of the different spreading codes to request for transmission based on their availability. The stations know when this occurs by means of a stack that is maintained distributedly.

### A. Information maintained and exchanged

Each station in the system is assigned a unique identifier, knows the maximum number of stations allowed in the network, the maximum propagation delay and maintains a stack and two variables (Low ID, (LID) and High ID,

(HID)) and an integer counter ( $C$ , set to zero whenever the stack is empty) for the spreading codes used.

LID is initially set to 1 and denotes the lowest id number that is allowed to send an **RTS**. HID is the highest ID number that is allowed to send an **RTS** and is initially set to the largest number of stations allowed in the network. LID and HID constitute the allowed ID number interval that can send **RTS**s. Other stations whose ID is not within the interval or is in the transmission queue cannot send its **RTS**.

$C$  is the integer counter, which is initially set to zero, and any new **RTS** arising in the system uses the code  $(C+1)$  for transmission. If  $C$  is equal to  $K$ , where  $K$  is the total number of spreading codes used, then stations that arrive newly in the system have to wait till the stack is empty. The control and data packets hold information necessary for the other stations to know the current status of the channel such as the  $C$  value, Stack etc.

### B. Adding Members to the Transmission Queue

Stations are allowed to send **RTS**s to be added to the transmission queue only during the first few seconds of a contention period called the access period. If the sender of a **RTS** does not receive the corresponding **CTS** within the allocated time, then a collision is assumed to have occurred and the value of the counter ( $C$ ) is increased by one. Each contention period of collision resolution can be one of the following cases.

#### Case 1: Collision

The collision resolution algorithm starts and each station divides the ID interval (LID, HID) into two ID intervals. The first ID interval, called the backoff interval,  $(LID, ((HID+LID)/2)-1)$ , is pushed into the stack by executing a PUSH stack command in every station. After this is done the station updates LID and HID with the values from the other interval, called the allowed ID interval,  $((HID+LID)/2, HID)$ . If the new stations that have arrived during this interval, then the code counter  $C$  is incremented by one. The collided stations always use the original code in which the collision has occurred to repeat their request.

#### Case 2: Idle

It lasts for a maximum round trip time after which each station executes a POP command and the new ID interval becomes the new HID and LID

#### Case 3: Success

The single station, which has requested successfully, is added to the transmission queue. Each station executes a POP command and the new ID interval becomes the new HID and LID only if all the requests that are using the different spreading codes are successful. If the new stations that have arrived during this interval, then the code counter  $C$  is incremented by one. The counter value  $C$  is made equal to zero when the stack is empty.

The queue transmission period follows the contention period and the stations that are present in the transmission queue are divided into groups consisting of  $\min(p, K)$  stations, where  $p$  is the total number of stations in the transmission queue. Each station in the group transmits its packet in the spreading code, which is equal to its position in the group. Each group transmits a single packet and after an interval of twice the propagation delay the next group transmits its packet. The queue transmission period

ends when all the stations have finished transmitting one of their data packets.

### C. Deleting Members from the Transmission Queue

A station leaves the transmission queue when it finishes sending all its packets.

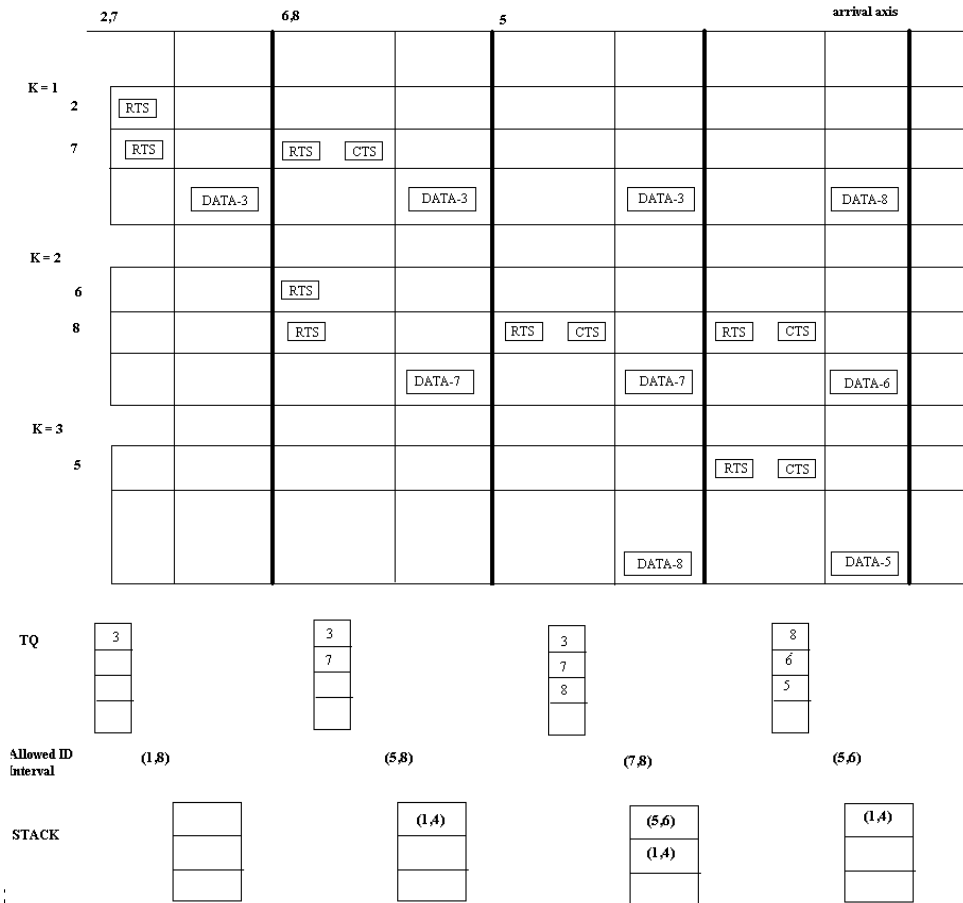


Figure 1. Working of the Protocol.

### III. EXAMPLE

The example shown in Fig. 1 illustrates the operation of the protocol with  $K=3$  and  $N=8$ . The messages generated by the terminals are assumed to be of variable length. In the example shown, station 3 is assumed to be in the transmission queue already and it still has five data packets to transmit. Initially LID is one and HID is eight and counter value is 0. Let stations 2 and 7 have two

packets each to transmit and hence they send their RTSs using code 1, i.e. code =  $C+1$ . A collision occurs here as a result of which the allowed ID interval is divided into two as (1,4) and (5,8). Here (1,4) is the back off interval and it is pushed into the distributed stack and the LID and HID are updated using the new allowed ID interval (5,8). The counter value is incremented by 1, i.e.  $C=1$ .

During the queue transmission period, station 3 transmits its data packet using code 1. Since there are no other stations in the queue we move to the next collision resolution interval. Let stations 6 and 8 arrive newly

during this interval. Stations 6 and 8 use code 2 for sending their RTSs while station 7 uses code 1 for its RTS. There is no collision in code 1 and station 7 is added to the transmission queue. Stations 6 and 8 collide in code 2 and hence the collision resolution steps again take over. The allowed ID interval is divided into two as (5,6) and (7,8).

The interval (5,6) is pushed into the stack and the allowed ID interval is updated using (7,8). The counter value C becomes 2. During the transmission period stations 3 and 7 transmit a single packet simultaneously using codes 1 and 2 respectively. During the next collision interval, let us assume that station 5 arrives at the system, but since its ID is outside the allowed ID interval it has to wait for its turn. Meanwhile station 8 sends an RTS for which it receives CTS due to which it moves into the transmission queue. A POP operation is performed on the stack and the new allowed ID interval becomes (5,6). During the next transmission interval, stations 6 and 5 make successful requests using different codes and the process continues.

#### IV. SIMULATION ANALYSIS:

Simulation was carried out for data packets of two different sizes (Large packets of 400 bytes and small packets of 53 bytes) using both low speed networks (9600 bps) and high-speed networks (1 Mbps). The diameter of the network was assumed to be 16090 meters, which yields a propagation delay of approximately 54 μs. For understanding, a step is defined as the combination of one collision resolution interval and one queue transmission interval. Simulation has been carried out for large number of steps and averages have been used to ensure convergence.

##### Delay performance:

The delay performance of ICRMA-CDMA is definitely better than that of ICRMA as seen from the figure 2.

The figure 1 shows the variation of per packet delay for small packets (53 bytes at a line speed of 9600 bps) as the arrival rate increases.

$$\text{Delay} = \text{processing delay} + \text{transmission delay}$$

The processing delay is the one that the packet experiences as soon as it arrives in the system till it starts sending its first packet. The waiting delay for packet i.e. the time elapsed between the arrival of the packet and the sending of RTS was not considered in this plot. The terminals are assumed to have data of size that are uniformly distributed between 1 and 10 packets. The results clearly show that ICRMA – CDMA provides a highly improved delay performance than that of ICRMA (nearly 33% faster for low arrival rates and 80% faster for very high loads).

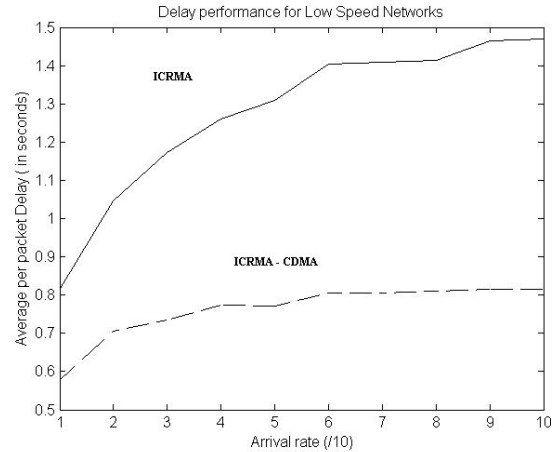


Figure 2. Delay performance for low speed networks for small data packets

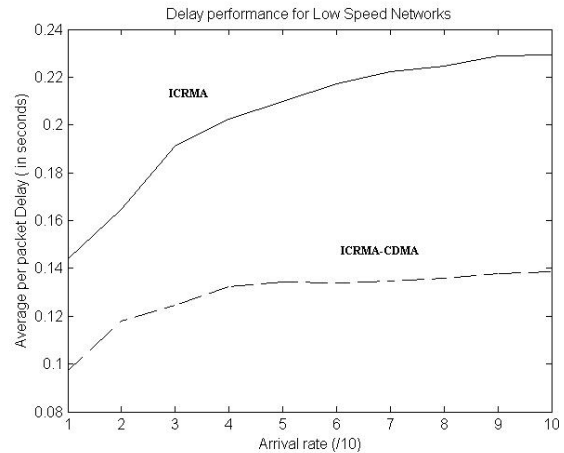


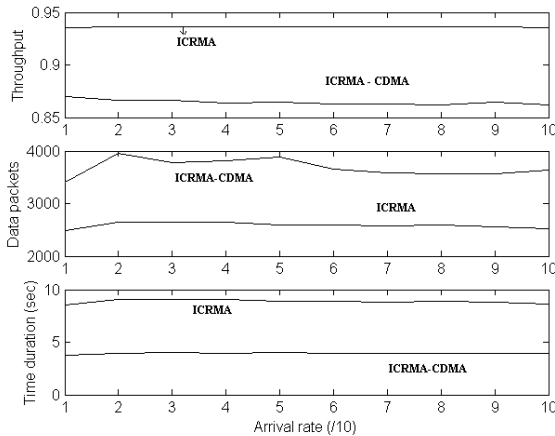
Figure 3. Delay performance of low speed networks for large data packets

The results clearly indicate that the ICRMA-CDMA protocol outperforms the original ICRMA by a large amount.

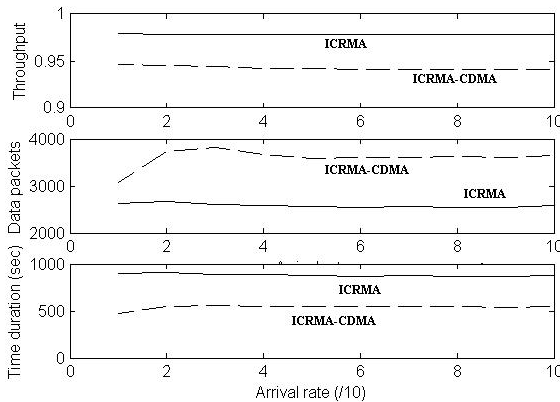
##### Throughput analysis:

Since ICRMA divides the channel access time into cycles consisting of small dynamically sized contention periods and dynamically sized queue-transmission periods, throughput as a measure of channel utilization time is used. ICRMA allows stations in the transmission queue to send their data packets in the queue order consecutively, due to which the channel utilization time is very high. The use of K different codes in a CDMA environment allows K consecutive stations in the transmission queue to transmit their data packets simultaneously, thereby reducing the total duration of every dynamically varying queue transmission period and transmitting a greater number of packets in a shorter period. From the above explanation, it is evident that we should expect a higher

throughput from ICRMA which uses a greater amount of time to transmit a comparatively fewer number of packets.



**Figure 4. Throughput performance of high speed networks for large data packets.**



**Figure 5. Throughput performance of low speed networks for large data packets.**

The throughput analysis for low speed networks also clearly show the advantage that ICRMA-CDMA has over the original ICRMA protocol.

#### IV. CONCLUSION

ICRMA-CDMA is a stable medium access protocol, which provides excellent delay and throughput performances for both high speed and low speed networks. It has been clearly shown by the simulation results that this protocol outperforms the ICRMA and is indeed very much adaptable to a CDMA environment.

#### REFERENCES

- [1] V. Bharghavan, A. Demers, S. Shenker, and L. Zhang, "MACAW: a media access protocol for wireless LANs," *Proc. ACM SIGCOMM '94*, pp. 212-25, ACM, 1994.
- [2] J.J. Capetanakis, "Tree algorithm for packet broadcasting channel," *IEEE Trans. Inform. Theory*, vol. IT-25, pp.505-515, Sept. 1979.
- [3] B.S.Tsybakov, and N.B.Likhanov,"Upper bound on the capacity of random multiple access system," *Problems of Information Transmission*, vol.23,no.3, pp.224-236, July-Sept.1987.
- [4] R.Garces and J.J.Garcia-Luna-Aceves,"Floor acquisition multiple access with collision resolution," *Proc. ACM/IEEE Mobile Computing and Networking '96*, Rye, NY, Nov.10-12, 1996.
- [5] W.Xu and G.Campbell, " A distributed queuing random access protocol for a broadcast channel," *Proc. ACM SIGCOMM '93*, San Francisco, CA,13-17 Sept. 1993.
- [6] T.Towsley, and P.O. Vales, "Announced arrival random access protocols," *IEEE Trans. Commun.* vol. COM-35, no.5,pp. 513-521,May 1987.
- [7] M.J. Karol and I.Chih-Lin,"A protocol for fast resource assignment in wireless PCS," *IEEE Transactions on Vehicular Technology*, vol.43,no.3,pp.727-32,IEEE,1994.
- [8] Rodrigo Garc'es and J.J. Garcia-Luna-Aceves, "A Near-Optimum Channel Access Protocol Based on Incremental Collision Resolution and Distributed Transmission Queues", *Proc. IEEE INFOCOM 98*, San Francisco, California, March 29--April 2, 1998