http://ansinet.com/itj

ITJ

ISSN 1812-5638

INFORMATION TECHNOLOGY JOURNAL

Information Technology Journal 4 (4): 409-415, 2005
ISSN 1812-5638

Performance Analysis of Collision Resolution Protocol-CICRMA for Wireless Environment

S. Malarvizhi and M. Meenakshi
Anna University, Chennai, India

Abstract: In this study a new MAC protocol for wireless environment, on the lines of the Incremental Collision Resolution Multiple Access (ICRMA) protocol using spreading codes, called Coded Incremental Collision Resolution Multiple Access (CICRMA) protocol was proposed. In CICRMA, the channel is divided into cycles of variable length; each cycle consists of a contention period and a queue transmission period. The queue transmission period corresponds to a variable-length train of packets, which are transmitted by stations that have been added to the data transmission queue by successfully completing a collision-resolution round in the previous contention period. During the contention period, stations with packets to send compete for the right to be added to the data-transmission queue using N-ary tree-splitting algorithm, where N is the number of spreading codes used. An analytical model for the above protocol was proposed. The proposed protocol was simulated; its throughput performance was analyzed and verified with the analytical model. Jakes fading model was used for the simulation of fading channel.

Key words: Collision resolution, spreading codes, Medium Access Control Protocol, tree splitting

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. A sender sends a Request-to-Send (RTS) to the receiver, who in turn sends Clear-to-Send (CTS). If the sender receives the RTS free of errors, only then it can can transmit a data packet. These protocols solve collisions by backing off and rescheduling RTS transmissions. This procedure gives good results if the RTS traffic is low, but it is unstable when the RTS transmission rate increases. The proposed protocol CICRMA resolves collisions among RTS by using tree-splitting procedure and it works well for both low and high RTS traffic.

In CICRMA, each transmitter is dynamically assigned a spreading code, which it uses to send RTS. Each successful RTS causes the transmitter to be added to the transmission queue with an assigned spreading code for data transmission. In this protocol the channel access time is divided into cycles, each cycle consisting of a small, dynamically sized contention period and a dynamically sized queue transmission period as shown in Fig. 1. Each of these cycles is synchronized for all the N spreading codes. Stations have to monitor the state of the channel while they are not transmitting. Stations having packets for transmission can send RTS during the contention period, using their assigned spreading code. The queue transmission period is a variable length train of packets from stations that have been added to the transmission queue by successfully completing a collision resolution round in the previous contention period. The queue transmission period consists of packet transmissions from the transmission queue and each station transmits packets using its assigned spreading code. The spacing between packets is fixed at 2 times the delay spread of the channel; it is to avoid interference between successive packets, which are transmitted using the same code. It is shown as T in the Fig. 1. The transmission queue can be allocated a maximum size to provide delay guarantees. Once the transmission queue reaches its maximum length, stations are not allowed to REQUEST for addition to the queue, until at least one member of the queue ends transmitting its packets and leave the queue.

Stations follow a non-persistent CSMA strategy for the transmission of RTSs. The sender of each RTS to an intended destination listens to the channel for one maximum round trip time plus the time needed for the destination to send CTS maximum round trip time is estimated based on the coverage area considered. If the CTS is not corrupted and is received within the time limit, the station is added to the transmission

Corresponding Author: S. Malarvizhi, Anna University, Chennai, India

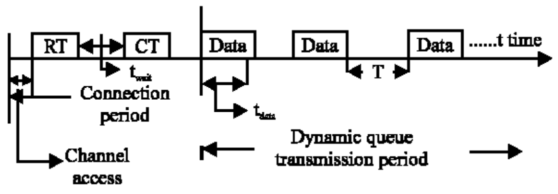

Fig. 1: CICRMA channel diagram

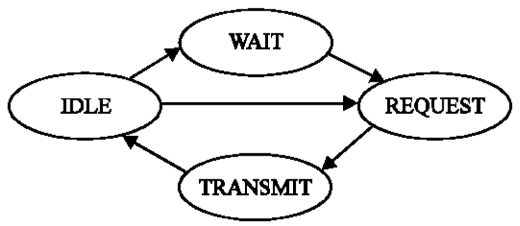

Fig. 2: State representation of station

queue and allowed to start transmission in the next queue transmission period. Each successful RTS causes the transmitter to be added to the transmission queue with an assigned spreading code for data transmission. Collisions of RTSs may occur when more than one station sends an RTS during a contention step. Based on its knowledge of the state of the transmission queue and the current step of the collision-resolution algorithm a station can be in one of the following states:

TRANSMIT: The station is part of the transmission queue.

REQUEST: The station is trying to acquire a position in the transmission queue and is participating in the collision-resolution algorithm.

WAIT: The station has local packets to send and is waiting for a backoff time to complete while the collision resolution of other stations is in progress.

IDLE: the station is not part of the transmission queue and has no local packets to send. Mobile state representation is shown in Fig. 2.

COLLISION RESOLUTION METHODOLOGY

Each station is assigned a unique identifier (ID) and knows the maximum number of stations allowed in the network and the maximum propagation delay. It maintains a stack and n sets of variables $(LID, HID)_i$, where n is the number of spreading codes used, LID, is the lowest ID number that is allowed to send an RTS using spreading code i and HID, is the highest ID number that is allowed to send an RTS using spreading code i. $(LID, HID)_i$ constitute the allowed ID-number interval that can send RTSs using spreading code i. If the ID of a station is not within this interval, it cannot send its RTS. The stack is simply a storage mechanism for ID-intervals that are waiting for permission to send an RTS. The values of (LID, HID) at the beginning of each collision resolution process are assigned as follows:

$$\begin{aligned} &\text{For } i = 1 \text{ to } n \\ &LID_i = (i-1).n_T/n + 1 \\ &HID_i = i.n_T/n \end{aligned} \qquad (1)$$

where, n_T is the total number of stations in the network and n is the number of spreading codes used.

Each station also maintains the state of the transmission queue, i.e., it knows the members in the transmission queue, their position in the queue and the beginning of queue-transmission period for each spreading code. RTSs and CTSs specify the IDs of the sender and the intended receiver, as well as the allowed ID-number interval known to the sending station. Including the allowed ID interval in the control packet allows listening stations to update themselves on the state of the collision resolution process. A data packet contains: (a) the user data and destination of the packet; (b) the allowed ID-number intervals known to the sending station; (c) the state of the transmission queue, such as the number of stations in the queue, the position of the sending station in the queue and its spreading code and (d) the state of transmission for the sending station, such as the length of the current packet and whether the station is leaving the transmission queue. This information helps listening stations to quickly update the state of the channel.

Stations REQUEST are added to the transmission queue only during contention periods. If the maximum length of the transmission-queue is reached, the contention steps are skipped, until space is available by one or multiple stations leaving the queue. Stations are allowed to send RTSs only during the beginning of a contention period.

If an IDLE station receives local packets to send, it immediately enters the WAIT state. It then waits for the current collision resolution process to complete before it enters the REQUEST state. It can then send its RTS using the assigned spreading code determined by the algorithm explained subsequently. The sender then waits and listens to the channel for one maximum round-trip time plus the time needed for the destination to send CTS. When the sender receives the CTS from the destination, it is added to the data transmission queue by updating the state of the transmission queues and it enters the TRANSMIT state. The station can start sending collision-free packets in the following queue-transmission period using the spreading code determined by the algorithm.

If the sender of an RTS does not receive the corresponding CTS within the allocated time, this station and all other stations in the network assume that a collision has occurred for that spreading code. As soon as a collision is detected, the collision resolution process based on the tree splitting algorithm is put into action, as explained below.

Every contending station PUSHes the corresponding ID-interval to the stack. Once the transmission period in that cycle has passed, all the stations POPs the ID interval (LID_P, HID_P) from the stack and the new allowed ID intervals are assigned as follows:

$$\begin{aligned}&\text{For } i = 1 \text{ to } n\\ &LID_i = LID_P + (i-1).n_{ID}/n \text{ and}\\ &HID_i = LID_P + i.n_{ID}/n - 1\end{aligned} \tag{2}$$

where, n_{ID} is the dynamically varying size of the ID interval and n is the number of spreading codes used.

$$\begin{aligned}&\text{If however, } n_{ID}<(n*2-1) \text{ then,}\\ &\text{For } i = 1 \text{ to } m1\\ &LID_i = LID_P + (i-1).n_{ID}/m1 \text{ and}\\ &HID_i = LID_P + i.n_{ID}/m1 - 1\end{aligned} \tag{3}$$

where, $m1 = n_{ID}/n$. and there are some free spreading codes. Another POP operation is performed (LID_Q, HID_Q) from the stack and the remaining spreading codes are assigned as follows:

$$\begin{aligned}&\text{For } i = m1+1 \text{ to } n\\ &LID_i = LID_Q + (i-1).n_{ID}/m2 \text{ and}\\ &HID_i = LID_Q + i.n_{ID}/m2 -1\\ &\text{Where, } m2 = n - m1.\end{aligned} \tag{4}$$

If in the previous step, the number of free codes, m2, is 1 then:
$(LID_n, HID_n) = ((LID_Q + HID_Q)/2, HID_Q)$ and the ID interval $(LID_Q, (LID_Q + HID_Q)/2 - 1)$ is PUSHed into the stack once again.

This procedure is repeated each time a collision is detected. Hence, a station knows that collisions are currently being resolved when its local stack is not empty and the accumulation of all the allowed ID intervals does not equal the entire ID interval.

If a station is in IDLE state and receives one or more packets to send, it goes immediately to the WAIT state. Only those stations that are in the REQUEST state participate in the current round of the collision-resolution algorithm for addition to the transmission queue. All stations in the allowed ID-intervals that are in the REQUEST state transmit an RTS in the next contention period. After every step of the collision-resolution algorithm, the state of the contention period for spreading code can be one of the following four cases:

Case 1--Idle: There is no station in the REQUEST state who's ID is within the allowed ID interval for that spreading code. Therefore, the channel must remain idle and no new member is assigned to the transmission queue for that spreading code. An idle step of collision resolution lasts for a maximum round-trip time.

Case 2--Success: There is a single station with an RTS to send whose ID lies within the allowed ID interval for that spreading code. In this case, this station is able to complete an RTS/CTS handshake successfully and enter the TRANSMIT state so it is added to the transmission queue. The duration of a successful collision resolution step lasts for two RTS lengths and two channel propagation delays.

Case 3--Collision: There are two or more stations with RTSs to send whose IDs are within the allowed ID interval; therefore, each such station sends an RTS creating a collision. A collision period can last no more than an RTS plus a maximum round-trip time. The allowed ID interval is PUSHed into the stack by each of the stations.

The tree-splitting algorithm executes a collision resolution (idle, success, or collision), until no station remains in the REQUEST state and all requests for the queue have been resolved. Again, all stations know when the tree-splitting algorithm terminates its current round, i.e., when the stack is empty. Once termination is detected, the allowed ID interval is reinitialized. To ensure fairness within the splitting algorithm, the position of the stations in the tree i.e. their ID numbers can vary after each tree-contention period. For example, the ID numbers of the stations can rotate cyclically. Each station increases its ID number by one and the last station takes the ID number of the previous first station.

Deletions from the transmission queue occurs when a station in the TRANSMIT state ends transmitting a packet train and gets out of the queue to enter the IDLE state. In this case, a data packet notifies the stations when the sending station is sending its last packet before leaving the queue.

WORKING EXAMPLE OF THE CICRMA

The following example, illustrates the operation of CICRMA using four spreading codes as given in Fig. 3.

The total number of stations is taken to be 32. The stations receive packets to transmit in the following order of IDs: 4, 1, 7, 9, 24, 13, 31, 2, 16, 29, 18, 3, 26, 30, 23, 12, 32, 8, 5.

The allowed ID intervals are initially assigned as:

	LID	HID
Code 1	1	8
Code 2	9	16
Code 3	17	24
Code 4	25	32

Initially, the stations 4, 1, 7, 9, 24, 13 and 31, wish to transmit packets. They enter the REQUEST state. They use the spreading codes determined by the allowed ID interval to which their IDs belong. Thus; 1, 4, 7 use spreading code 1; 9, 13 use code 2; 24 uses code 3 and 31 uses code 4. Successful RTS/CTS exchange takes place in codes 3 and 4, while collision takes place in codes 1 and 2. Stations 24 and 31 enter the TRANSMIT state and are added to the transmission queue. Allowed ID intervals (1, 8) and (9, 16) are PUSHed into the stack.

Stations 24 and 31 transmit one packet each in the queue transmission period using spreading codes assigned by dividing the length of the transmission queue among the spreading codes. Thus station 24 uses code1 and station 31 uses code 2. Stations 2, 16 meanwhile, receive packets to transmit and enter the WAIT state. The ID interval (9, 16) is POPed from the stack and the new allowed ID intervals now become:

	LID	HID
Code 1	9	10
Code 2	11	12
Code 3	13	14
Code 4	15	16

Station 9 uses spreading code 1 and station 13 uses code 3. Successful RTS/CTS exchange takes place in codes 1 and 3, while codes 1 and 2 are idle. Stations 9 and 13 enter the TRANSMIT state and are added to the transmission queue.

Stations 24, 31, 9 and 13 transmit one packet each in the queue transmission period.

Stations 29, 18 meanwhile, receive packets to transmit and enter the WAIT state. An ID interval (1, 8) is POPed from the stack and the new allowed ID intervals become:

	LID	HID
Code 1	1	2
Code 2	3	4
Code 3	5	6
Code 4	7	8

Station 1 uses spreading code 1, station 4 uses code 2 and station 7 uses code 4. Successful RTS/CTS exchange takes place in codes 1, 2 and 4, while code 3 is idle. Stations 1, 4 and 7 enter the TRANSMIT state and are added to the transmission queue.

Stations 24, 31, 9, 13, 1, 4 and 7 transmit one packet each in the queue transmission period. Stations 3, 26, 30 and 23, meanwhile receive packets to transmit. Since the stack is empty, it denotes the end of one complete collision resolution process. All the stations in the WAIT state (2, 3, 16, 18, 23, 26, 29, 30) now enter the REQUEST state and the allowed ID intervals are reinitialized to:

	LID	HID
Code 1	1	8
Code 2	9	16
Code 3	17	24
Code 4	25	32

Stations 2 and 3 use code 1 to send RTS; 16 uses code 2; 18 and 23 use code 3 and 26, 29 and 30 use code 4. Successful RTS/CTS exchange occurs in code 2 and station 16 enters the TRANSMIT state. Collisions occur in codes 1, 3 and 4 and the ID intervals (1,8), (17, 24) and (25, 32) are PUSHed into the stack.

Stations 24, 31, 9, 13, 1, 4, 7 and 16 transmit one packet each in the queue transmission period. Stations 12 and 32 meanwhile, receive packets to transmit and enter the WAIT state. An ID interval (25, 32) is POPed from the stack and the allowed ID intervals are assigned as follows:

	LID	HID
Code 1	25	26
Code 2	27	28
Code 3	29	30
Code 4	31	32

Station 26 uses spreading code 26 to transmit RTS and 29 and 30 use code 3. Codes 2 and 4 are unused. Successful RTS/CTS exchange takes place in code 1 and station 26 enters the TRANSMIT state and is added to the transmission queue. The special case of success occurs in code 3 and stations 29 and 30 also enter the TRANSMIT state and enter the transmission queue.

Stations 24, 31, 9, 13, 1, 4, 7, 26, 29 and 30 transmit one packet each in the queue transmission period. The collision resolution process continues in this manner.

	Collision Resolution Process 1					
Cycle	Cycle 1		Cycle 2		Cycle 3	
Step	CS1	TX1	CS2	TX2	CS3	TX3
Time	$2Tctl^1+2p^2$	Tx^3+p^2	$2Tctrl^1+2p^2$	Tx^3+p^2	$2Tctr1^1+2p^2$	$2Tx^3+2p^2$

a. Details of collision and transmission cycle. 1.Time required for transmission of RTS/CTS, 2. Propagation time, 3.Time required for transmission of data packets in the transmission queue

	Lid, Hid	RTS	CTS	TransQ	Lid, Hid	RTS	CTS	TransQ	Lid, Hid	RTS	CTS	TransQ
Code (1)	1, 8	1,4,7	X	24_1	9, 10	9	9	24_2	1, 2	1	1	1_1 24_3
Code (2)	9, 16	9,13	X	31_1	11, 12	X	X	31_2	3, 4	4	4	4_1 31_3
Code (3)	17, 24	24	24	X	13, 14	13	13	9_1	5, 6	X	X	7_1 9_2
Code (4)	25, 32	31	31	X	15, 16	X	X	13_1	7, 8	7	7	13_2

b. Details of allowed intervals for each code, stations in the transmission queue and RTS/CTS

Stack	(1, 8), (9, 16)	(1,8)	(1,8), (17, 24), (25, 32)	……

c. Details of stack

IDLE	3, 5, 6, 8, 10, 11, 12, 14, 15, 17, 18, 19, 20 21, 22, 23, 25, 26, 27, 17, 28, 29, 30, 32	3, 5, 6, 8, 10, 11, 12, 14, 15, 28, 30, 32 27, 28, 32, 22, 23, 25, 26,	2, 3, 16, 18, 23, 26, 29, 30
WAIT	2, 16	2, 18, 16, 29	1, 4, 7
REQ	1, 4, 7, 9,13, 24,31	1, 4, 7, 9, 13	9, 13, 24, 31
TRANS		24, 31	

d. Details of stations in WAIT state, REQUEST state, TRANSMISSION state and in IDLE state

Fig. 3: Example illustrating CICRMA operation

ANALYTICAL MODEL AND SIMULATION OF CICRMA

Throughput analysis is carried out for CICRMA, following analysis presented[1].The channel load is considered to be a Poisson source, with an infinite number of stations and each station having at most one RTS to transmit at any time. The mean arrival rate is λ RTS per unit time. The average number of RTS arrivals in a time interval of length t is λt. The duration of a data packet in the equation is denoted as t_{data} and control packets are denoted as t_{cntrl} seconds.

The average channel throughput is equal to

$$T = \frac{\overline{durn_{cu}}}{\overline{durn_{idle}} + \overline{durn_q} + \overline{durn_{con}}} \tag{5}$$

Where, $\overline{durn_{cu}}$ is the average utilization time of channel during which the channel is being used to transmit message packets, $\overline{durn_{idle}}$ is the average idle period i.e., average interval between two consecutive busy periods, $\overline{durn_{con}}$ is the average duration of contention period and $\overline{durn_q}$ is the average duration of queue transmission period.

The $\overline{durn_{cu}}$ depends on the average number of stations in the transmission queue M_q

$$\overline{durn_{cu}} = M_q * t_{data} \tag{6}$$

The average queue transmission period, $\overline{durn_q}$ is equal to the average number of stations in the transmission queue M_q and the transmission period for one packet. The transmission period for one packet is the sum of time taken for accessing the channel and the duration of a data packet.

Therefore,

$$\overline{durn_q} = M_q \text{. (Transmission period)} = Mq^*(\tau_a + t_{data}) \tag{7}$$

Where, τ_a is the access time.

The average idle period can be found out by considering the transmission queue in an empty state and in non empty state. If the transmission queue is empty, the length of idle period includes the next RTS arrival time into the channel plus a waiting period of $2\tau_{prop}$ sec. The inter arrival of RTS is given by $1/\lambda$. When the transmission queue is not empty, the length of the idle period is extends till the start of the next transmission period which is 2τ seconds.

$$\text{Therefore } \overline{durn_{idle}} = \Pi_0{*}(1/\lambda + 2\tau) + (1 - \Pi_0){*}\, 2\tau$$
$$= 1/\lambda{*}\Pi_0 + 2\tau \tag{8}$$

Where, Π_0 is the probability that the transmission queue is empty.

Contention period can be an idle period, successful RTS/CTS exchange period or a collision period. The duration of the idle contention period is $T_i=2\tau$, the duration of the successful contention period is $T_s=2t_{cntrl}+2\tau$ and the duration of the RTS collision period is $T_f < t_{cntrl}+2\tau$. Therefore, the duration of a contention period is at most:

$$[\, 2\tau{*}P_{idle} + T_s{*}(P_{rts1} + P_{col})]{*}(1-\Pi_h) \tag{9}$$

Where, P_{idle} is the probability that a contention period is idle is equal to the probability that no packets arrive during the access period τ, which can be expressed as:

$$P_{idle} = e^{-\lambda\tau} \qquad (10)$$

For a station to be added to the data transmission queue, the RTS/CTS exchange must be successful, the RTS must be the only one in the channel during its access period and the data transmission queue must not be full. The probability P_{rts1} of this happening equals the probability that only one RTS arrives during the τ seconds of the access period. If there are already 'h' queue members or stations in the queue, no new member can be added .Where, h is the storage capacity of the transmission queue. In such a case the contention would be skipped, until a space in the data transmission queue is available. Therefore P_{rts1} can be expressed as:

$$P_{rts1} = \lambda\tau e^{-\lambda\tau} \qquad (11)$$

The probability P_{idle} and P_{rts1} are the fraction of a cycle that has zero or one RTS during the access period. Therefore $(1 - P_{idle} - P_{rts1})$ is the fraction of the cycle that has more than one RTS during the access period. When there is more than one RTS in a code in the same access period, collision is bound to occur. The probability of collision P_{col} can be specified by:

$$P_{col} = 1 - P_{idle} - P_{rts1} \qquad (12)$$
$$= 1 - \lambda\tau e^{-\lambda\tau} - e^{-\lambda\tau} \qquad (13)$$

The contention period will be skipped, if there are already 'h' stations in the queue The probability that a new member is added to the transmission queue per contention period is denoted as W and the probability that no one is added is denoted as V.

Markov chain can be used to represent the states of the queue transmission period. The states in the Markov chain represent the probability Π_k that k members are in the data transmission queue. The number of members in the data transmission queue can only increase by one, but can decrease by up to k. This means that, only one new member can be added to the data transmission queue per contention period, but any number of members can be released from the transmission queue during the queue transmission period.

Figure 4 shows the Markov Chain representation of stations in the transmission queue. The values of Π_1, Π_2, $\Pi_3 \ldots \Pi_h$ are found out and used in equation 8 and 9. The average throughput for CICRMA with is obtained for different offered loads using the Eq. 5.

The CICRMA protocol is simulated using MATLAB. The simulation parameters and numerical values considered for simulation are based[1] and IEEE 802.11 are listed below:

Transmission rate	= 1Mbps
Number of stations	= 256
Length of a data packet	= 1296 bits
Length of a control packet	= 272 bits

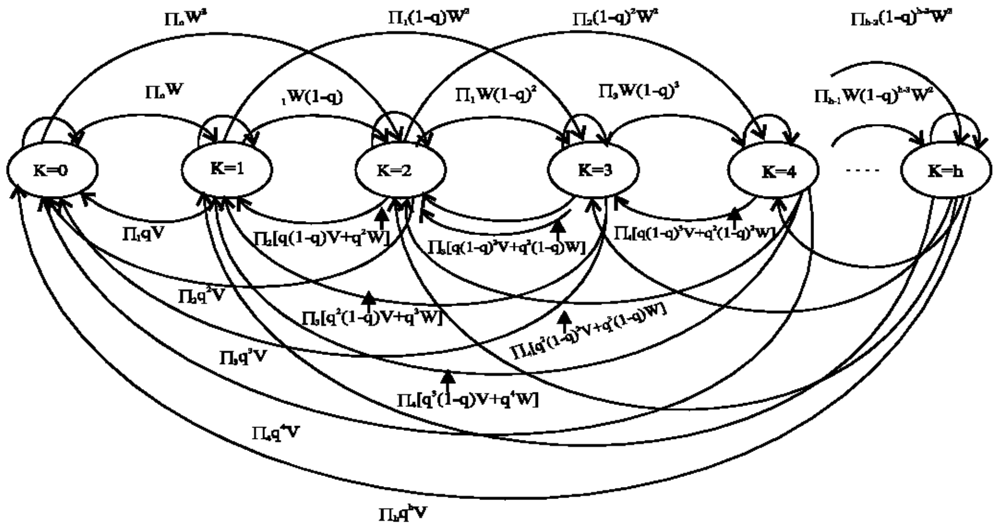

Fig. 4: Markov Chain of stations in transmission queue

Diameter of the network = 16090 m
Number of spreading codes = 4
Fading Model: JAKES SIMULATOR
f_{max}=30 Hz
Number of tabs =8
Number of low frequency oscillators.

RESULTS AND DISCUSSION

The throughput and delay performance of CICRMA and ICRMA[1] are analyzed using the mathematical and simulation model. From Fig. 5 obtained results shows that throughput performance of CICRMA performs better compared with CICRMA. It is observed from Fig. 5, the CICRMA performance is better than ICRMA in faded channel. The throughput of CICRMA is found to be higher than that of ICRMA under faded conditions. In the case of CICRMA, the impact of fading is reduced by the usage of spreading codes. Though the effective number of bits transmitted gets reduced due to spreading, there is improvement in throughput due to the reduction of MAI, achieved by the orthogonality of spreading codes. The faded channel has a serious impact on ICRMA compared to CICRMA.

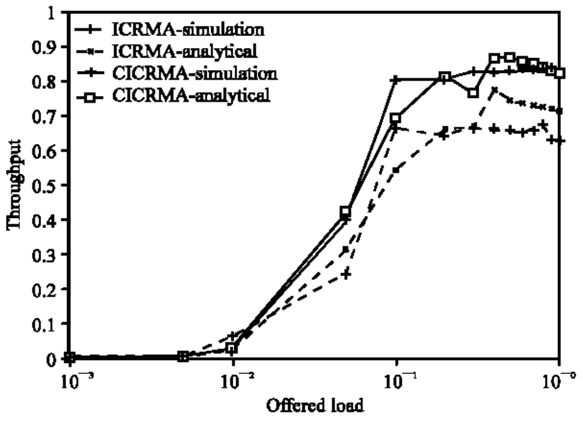

Fig. 5: Throughput Vs offered load (ICRMA and CICRMA under ideal channel)

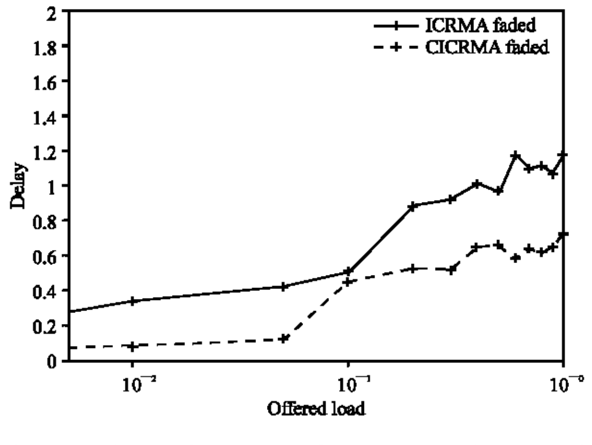

Fig. 6: Delay performances of CICRMA and ICRMA in faded channel

Figure 6 compares the delay performance of ICRMA and CICRMA under faded channel conditions. The delay performance is also seen to be much better for CICRMA, since the channel time is efficiently used for packet transmission.

CONCLUSIONS

In this study, the Coded Incremental Collision Resolution Multiple Access (CICRMA) Protocol is presented and analyzed by incorporating spreading codes. The performance of CICRMA is compared with ICRMA protocol. It is observed from mathematical and simulation analyses under ideal and faded channel conditions, the performance of CICRMA protocol performs better than ICRMA, because of the usage of N-ary tree-splitting and usage of spreading codes. The data rate reduction due to the use of spreading codes is compensated by the increased speed of the collision resolution process using N-ary tree algorithm.

REFERENCES

1. Rodrigo Garce`s and J.J. Garcia-Luna-Aceves, 1998. A near-optimum channel assess protocol based on incremental collision resolution and distributed transmission queues. Proceeding of IEE INFOCOM 98, San Francisco, California, March 29-April 2.