UC Santa Cruz

UC Santa Cruz Previously Published Works

Title

NOMAD: Deterministic Collision-Free Channel Access with Channel Reuse in Wireless Networks

Permalink

<https://escholarship.org/uc/item/3500s6kw>

Author

Garcia-Luna-Aceves, J.J.

Publication Date

2011-06-27

Peer reviewed

NOMAD: Deterministic Collision-Free Channel Access with Channel Reuse in Wireless Networks

J.J. Garcia-Luna-Aceves and Ashok N Masilamani

Computer Engineering Department

University of California, Santa Cruz

Santa Cruz, CA 95064

Email: jj, ashok@soe.ucsc.edu

Abstract—The Neighborhood Ordering for Medium Access with Determinism (NOMAD) protocol is introduced. NOMAD defines collision-free transmission schedules dynamically and with no need for a predefined number of time slots per transmission frame by coordinating circular permutations of the identifier of nodes in the neighborhoods shared among nodes. NOMAD is shown to attain feasible transmission schedules within a short finite time and to provide channel access intervals of negligible variance. The performance of the NOMAD is compared with the performance of 802.11 DCF and the node activation multiple access (NAMA) protocol, which is representative of distributed transmission scheduling based on probabilistic elections. NOMAD is shown to attain higher throughput than 802.11 and NAMA in static and dynamic ad hoc networks, and to eliminate the large variances in channel access times present in contention-based schemes and prior transmission-scheduling schemes that do not use reservations.

I. INTRODUCTION

Contention-based MAC protocols, including IEEE 802.11 distributed coordination function (DCF) [1], can be viewed as attempting to emulate “Ethernets in the sky” in that the MAC protocol is assumed to operate over a single multiple-access link, at most one transmission is allowed to reach a receiver, and senders are forced to back off on a packet-by-packet basis in the presence of multiple access interference (MAI). On the other hand, as our summary of related work presented in Section II indicates, MAC protocols based on the establishment of transmission schedules dynamically, typically organize the channel into transmission frames consisting of a fixed number of time slots assigned to nodes through elections or reservations. While these protocols eliminate the problems associated with back-offs after noise or multiple access interference (MAI) is detected, their use of probabilistic methods for electing or reserving time slots from a fixed-length frame lead to long delays in the establishment of collision-free transmission schedules in every node, and to long variances of the time between two consecutive transmissions by the same node in the absence of reservations.

We introduce a simple approach to collision-free channel access that attains dynamic channel reuse together with deterministic channel access delays, and which is much more efficient than contention-based access or transmission scheduling based on probabilistic schemes. Section III presents the NOMAD protocol (*Neighborhood Ordering for Medium Access with Determinism*). NOMAD attains distributed and

deterministic transmission scheduling with channel reuse by the dynamic coordination of circular permutations of node identifiers known to each node. According to NOMAD, the channel is organized into time slots but without using fixed transmission frames; the number of time slots that form the transmission frame assumed by each node is determined dynamically based on the number of identifiers in the neighborhood of the node. To ensure that the time of transmission assigned to a node is the same in its entire neighborhood, each node communicates to its immediate neighbors a circular permutation of node identifiers in a control packet transmitted at the beginning of a time slot. Each node computes its circular permutation based on the circular permutations received from its neighbors by following a few simple rules to order node identifiers in a way that two nodes that are two hops away from one another are not allowed to transmit during the same time slot (position in a permutation). To account for differences in the size of node neighborhoods and to allow channel reuse among nodes more than two hops away from each other, nodes insert empty node identifiers in some positions of their circular permutations to represent transmission turns from two-hop neighbors or transmission turns that are not utilized. To simplify the task of ensuring that the circular permutations assumed by nodes induce collision-free transmissions, the lengths of circular permutations are powers of two.

Section IV shows that NOMAD provides collision-free access to nodes for data packets within a finite time and shows the importance of attaining zero variance in channel access delays when all nodes have up-to-date neighborhood information. Section V presents the results of simulation experiments comparing the performance of NOMAD with the performance of 802.11DCF and NAMA [2]; the simulation results show that NOMAD outperforms the other two protocols in terms of throughput, that NOMAD provides far smaller channel access delays at the same time, and the variances in channel access delays at any given node in a dynamic network are very small.

II. PRIOR WORK

Many medium access control (MAC) protocols have been proposed to control access to a common wireless channel using contention and distributed scheduling schemes. Because of space limitations and the known performance limitations

of contention-based schemes, we focus our summary of prior work on schemes aimed at distributed transmission scheduling. Collision-free channel access using fixed TDMA schedules results in very poor channel utilization. As a result, many approaches have been proposed in the past to attain transmission scheduling dynamically in order to increase channel reuse. What is striking about all of these schemes is that they organize the communication channel into fixed transmission frames consisting of a pre-defined number of time slots used to access the channel.

Dynamic transmission scheduling schemes can be categorized into topology-independent and topology-dependent schemes. Topology independent schemes (e.g., [3], [4]) establish transmission schedules dynamically corresponding to a unique code. Such schedules are independent of the topology of the network. This topology independence makes the transmission schemes very robust in the presence of node mobility; however, Kunz and Rentel [5] have shown that this approach has similar performance to that of slotted ALOHA.

Topology-dependent transmission scheduling protocols establish transmission schedules taking into account the connectivity of the network and in some cases the traffic at each node. The assignment of time slots to nodes is based either on the election of entities competing for the data time slots (nodes or links), or reservation requests for data time slots following some predefined rules. Some schemes require an initial topology-independent schedule, followed by some negotiation among network nodes used to obtain a final schedule [6], [7], [8]. In schemes based on reservations, the channel is divided into frames consisting of a fixed number of time slots, and each time slot is divided into several mini-slots dedicated for the contention and reservation of the time slots as well as the transmission of data in the time slot (e.g., [9], [10]).

There are many examples of topology-dependent transmission scheduling protocols based on the election of transmission schedules in a distributed manner. To elect transmission schedules, each node knows the identities of all other nodes one and two hops away from itself, and the present time in the network [11], [2]. Each node builds and maintains a list of contending entities (nodes or links) and applies a permutation function on the list of contending entities to select a winning node from the list of nodes for each time slot of the transmission frame. Some protocols also allow for nodes that win the election of a time slot to reserve the time slot. The main limitation of all the above protocols is that the selection of winning nodes for each time slot does not provide any performance guarantees on a short-term basis and can render long channel-access delays for some nodes, because the main purpose of the permutation functions used in these protocols is to randomize the winners of elections.

Considerable work has also been reported on the establishment of efficient transmission schedules taking into account the nodal traffic demands and attempting to limit the overhead incurred in the establishment of schedules that approach the optimum (e.g., [12], [13]). However, the approaches reported

to date incur extensive signaling overhead and are not practical in MANETs.

III. NOMAD

The design of NOMAD is based on very few assumptions, which are similar to those used in prior MAC protocols based on transmission scheduling, namely: (a) the radios used in the network are half-duplex and can tune to only one channel at a time; (b) each node in the network is assigned a unique node identifier (nid); (c) the radio links used for communication in the network are bidirectional; (d) channel access time is slotted and time slots have a fixed duration; and (e) any pair of nodes can be synchronized at the time-slot level, such that the nodes within a two-hop neighborhood assume the same time for the start and end of a given time slot and have the same time slot number. The time slotting needed in NOMAD and all prior MAC protocols based on transmission scheduling can be attained in practice using such distributed clock synchronization schemes as those demonstrated in the past by Rentel and Kunz [17] and Djukic and Mohapatra [18]. As such, the rest of this paper simply assumes that time slotting is available.

In contrast to most prior MAC protocols based on transmission scheduling, NOMAD does not require a predefined number of time slots per transmission frame to organize channel access. For simplicity of exposition, the rest of this paper assumes that multiple access interference (MAI) occurs only among one and two-hop neighbors, which is also assumed in most MAC protocols based on transmission scheduling. Accordingly, the neighborhood of a node is assumed to consist of those nodes whose transmissions the node can decode, which we call one-hop neighbors, and the one-hop neighbors of those nodes, or two-hop neighbors. In practice, MAI may occur due to nodes more than two hops away, but this can be taken into account in NOMAD by using three-hop neighborhood information, for example.

A. Information Stored and Exchanged

The data structures used by each node are

- Reported Transmission List(RTL): The list that is transmitted as updates by the node.
- Ordered Transmission List(OTL): Calculated from the RTLs received and it establishes the transmission positions of nodes in the two-hop neighborhood.
- Neighborhood Transmission List(NTL): The list of RTLs from one-hop neighbors.
- Neighborhood Transmission Schedule(NTS): Defines the transmission schedule assumed by a node.

Node k maintains a *neighborhood transmission schedule* (NTS_k) that defines the time slots assigned to the nodes in the neighborhood of node k . It consists of a row for each node in its one-hop neighborhood and a column for each time slot required in the schedule. NTS_k is obtained from the *ordered transmission list* (OTL_k) of node k , which states a permutation of node identifiers that establishes permutation positions assigned for the transmissions of nodes in its two-hop

neighborhood, taking into account differences in the neighborhood sizes of nodes and channel reuse (see Section III-B). OTL_k is generated from the list of *reported transmission lists* (RTL) that node k receives from its neighbors and is stored in the *Neighborhood Transmission List* (NLT_k). Node k in turn generates its own *reported transmission list* (RTL_k) from OTL_k .

Node k transmits RTL_k periodically every $RTL_Interval$ seconds. RTL_k consists of a list of tuples that includes node k itself and each of its known one-hop neighbors. Each tuple in RTL_k specifies the time slot position in the transmission schedule and the corresponding node identifier value. The last tuple in RTL_k with value r_max may specify a node identifier equal to Φ , and r_max may be larger than the value of the time-slot position value of the last tuple with a non-empty node identifier value. This value determines the rate at which the schedule repeats for that node in NLT_k .

In addition to RTL_k , node k transmits a *Join* message. A *Join* message consists of three fields: a node identifier, a time-slot number chosen for transmission and the rank of the node (described in Section III-C). *Join* messages are transmitted in "participation slots" used for nodes to join a neighborhood in the network (see Section III-E). *Join* messages are retransmitted by the one-hop neighbors of the origin of the *Join* message in their respective time slots of the transmission schedule. As such, they traverse two hops.

Fig. 1. Network

Fig. 2. RTLs and OTL at Node D

Figure 1 shows an example network. Nodes b, c, d, e and f form the two-hop neighborhood of node d . Node d receives RTLs from its one-hop neighbors c and e , denoted by RTL_{dc} and RTL_{de} in Figure 2. The RTLs that are to be sent by these nodes to the one-hop neighbors and the corresponding form in which they are sent are listed in Figure 3. The maximum slot number in the RTL of a node is r_max . The tuple $(8, \Phi)$ in RTL_e and RTL_b indicate that the rank of these nodes is 8.

Node	RTL	RTL(in Tuples)	Rank
b	- a c b - - - -	[(2,a),(3,c),(4,b), (8, Φ)]	8
c	d - c b	[(1,d),(3,c),(4,b)]	4
d	d e c -	[(1,d),(2,e),(3,c)]	4
e	d e f - - - - -	[(1,d),(2,e),(3,f), (8, Φ)]	8
f	- e f - - - g -	[(2,e),(3,f),(7,g)]	8

Fig. 3. RTL list

B. Updating RTLs and OTLs

Node k updates RTL_{kn} in its NLT_k after receiving the reported transmission list RTL_n from one-hop neighbor n or after detecting the loss of radio connectivity with one-hop neighbor n .

As shown in Figure 2 the RTLs sent by nodes c and e to node d are $RTL_c = [(1,d)(3,c)(4,b)]$ and $RTL_e = [(1,d)(2,e)(3,f)(8,\Phi)]$. From this information, node b stores $RTL_{dc} = [(1,d)(2,\Phi)(3,c)(4,b)]$ and $RTL_{de} = [(1,d)(2,e)(3,f)(4,\Phi)(5,\Phi)(6,\Phi)(7,\Phi)(8,\Phi)]$.

Node k updates OTL_k in two phases each time it updates NLT_k . During the first phase, node k selects time-slot positions assigned to one-hop or two-hop neighbors in OTL_k and time-slot positions with an empty node identifier to accommodate the length of the transmission lists of its one-hop neighbors. During the second phase, node k selects a time-slot position for itself.

Let $nid_{kn}[r]$ denote the node identifier value of the tuple at position r in RTL_{kn} , let $nid_k[r]$ denote the node identifier of the tuple at position r in OTL_k , and assume that a non-existing entry of RTL_k has an empty node identifier.

If $nid_k[r] = x$ in OTL_k AND $nid_{kn}[r] = y$ in RTL_{kn} , then

- If $x = \Phi$, then $nid_k[r] = y$.
- If x and y are one-hop neighbors of node k , and $y > x$, then $nid_k[r] = y$.
- If x and y are two-hop neighbors of node k , and $y > x$, then $nid_k[r] = y$.
- If x is a one-hop neighbor of node k , and y is a two-hop neighbor of node k , then $nid_k[r] = x$.
- If x is a two-hop neighbor of k , and y is a one-hop neighbor of node k , then $nid_k[r] = y$.

After node k updates OTL_k as described above, it selects for itself the time-slot position with the smallest slot-position that does not have a node identifier. This slot-position is called $Slot_Number_k$. If no time-slot position can be assigned to node k from the existing time-slot positions, node k adds a time-slot position for itself at the end of OTL_k . It updates its RTL_{kk} by first copying the content of OTL in RTL and then setting to empty the node identifiers of two-hop neighbors. Node k also prepares RTL_k to be sent to its neighbors, which is a list of tuples as described in III-A. Node k determines the loss of connectivity with a one-hop neighbor when it fails

to receive three consecutive RTLs from that neighbor, after which it resets all the node identifiers in the tuples of RTL_{kn} to empty and updates OTL_k as described.

C. Updating The Neighborhood Transmission Schedule

A node updates its NTS from the RTLs of its one-hop neighborhood. The sequence of node identifiers contained in RTLs is copied one or multiple times into the NTS. The number of times that the RTL is copied into the NTS is such that the latter starts with the first time-slot position of all RTLs of the one-hop neighborhood and ends at the time slot prior to the next time slot for which all the first time-slot positions of all RTLs would be repeated. Based on the information in NTS_k , node k can select those time slots in NTS_k for which its one-hop neighbors have no conflict with a transmission from node k .

For any node, the rank of the node is defined as the smallest power of base number 2 that is greater than or equal to the maximum occupied slot number in the OTL of the node. But if this rank is equal to the slot number that is chosen by the node itself, then the rank is multiplied by two and all slots between the new and old rank are kept empty. Algorithm 1 is the algorithm for computing the rank of a node. The objective of using powers of two is to make the number of time slots of any NTS equal to the rank of the longest RTL in the one-hop neighborhood of the node computing the NTS. $Max_slot_number_a$ is the maximum occupied slot number in the OTL of node a . Lines 6 and 7 in the algorithm guarantee that there is at least one free time slot every two transmission schedules for nodes to broadcast new RTLs, and this is explained in Section III-E.

Algorithm 1 Rank computation

- 1: For node a ,
- 2: $Rank_a = 1$
- 3: **while** $Rank_a < max_slot_number_a$ **do**
- 4: $Rank_a \leftarrow Rank_a \times 2$
- 5: **end while**
- 6: **if** $Rank_a == Slot_Number_a$ **then**
- 7: $Rank_a = Rank_a \times 2$
- 8: **end if**

In Figure 2, the ranks of nodes c and e are 4 and 8, respectively, whereas the rank of node d is 4. The NTS built at node d from these RTLs is shown in Figure 4. The RTL of node e is copied once and the RTLs of nodes c and d are copied twice in the NTS. The rank of the nodes is listed in Figure 3 along with the respective RTLs. Figure 4 also shows the NTSs built at nodes c , e and f . From the RTLs and the ranks of the nodes, each node build its own NTS. HR_k is the highest rank of all nodes in the one-hop neighborhood of k . Algorithm 2 is used for building the NTS in a node k .

D. Transmission Algorithm

Let OHN_k be the one-hop neighborhood of node k . $NTS_k[n, t]$ is the node ID in the NTS of node k in the row corresponding to the one-hop neighbor n and in the slot-number t .

		1	2	3	4	5	6	7	8
NTS at c	b	-	a	c	b	-	-	-	-
	d	d	e	c	-	d	e	c	-
	c	d	-	c	b	d	-	c	b
NTS at d	e	d	e	f	-	-	-	-	-
	c	d	-	c	b	d	-	c	b
	d	d	e	c	-	d	e	c	-
NTS at e	d	d	e	c	-	d	e	c	-
	f	-	e	f	-	-	-	g	-
	e	d	e	f	-	-	-	-	-
NTS at f	e	d	e	f	-	-	-	-	-
	g	j	h	f	l	k	i	g	-
	f	-	e	f	-	-	-	g	-

Fig. 4. NTS of nodes

Algorithm 2 Computing NTS

- 1: **for all** n in OHN_k **do**
- 2: $NTS_k[n] = \Phi$; $count = 0$;
- 3: Compute $cycle_{kn} = HR_k / rank[RTL_{kn}]$
- 4: **while** $count < cycle_{kn}$ **do**
 $NTS_k[n] = NTS_k[n] \oplus nid[RTL_{kn}]$ $count = count + 1$
- 5: **end while**
- 6: **end for**

Algorithm 3 is the transmission algorithm that all nodes follow in order to decide if it should transmit or receive in a slot. For each slot, the node calculates $t_n = SlotID \% rank$ value, where $SlotID$ is the time slot number of the synchronized network. It goes to *Transmit* state only if $NTS_k[n, t_n]$ is k and there is no other one-hop neighbor with a node ID other than k at t_n in its NTS.

In Figure 4, it can be seen that slots 1 and 5 in the NTS of node d are the slots in which node d would transmit. Similarly, we observe from Figure 4 that the transmitting slots for other nodes are: 3 and 7 for node c ; 2 for node e ; and 3 for node f . The transmission slots for the nodes in the neighborhood of interest are shown in Figure 5

E. Joining a Network and Handling Mobility

Let max_rank_a be the maximum rank of the one-hop neighborhood nodes of a . In the NTS of a , if there is

Algorithm 3 Transmission Algorithm

```

1: Let  $t_n = \text{SlotID} \% \text{rank}(n)$ 
2: if  $\text{NTS}_k[k, t_n] == k$  then
3:   for all  $n$  in  $\text{OHN}_k$  do
4:     if  $\text{NTS}_k[n, t_n] \neq k$  OR  $\text{NTS}_k[n, t_n] \neq \Phi$  then
5:        $\text{State}_k \leftarrow \text{Receive}$ .
6:     return .
7:   end if
8: end for
9:  $\text{State}_k \leftarrow \text{Transmit}$ .
10: else
11:    $\text{State}_k \leftarrow \text{Receive}$ .
12: end if

```

Fig. 5. Transmission Slots

no two-hop node specified in the column corresponding to max_rank_a , then max_rank_a is called the *Participation Slot* for that node. If there is a two-hop node, then *Participation Slot* is $\text{max_rank}_a \times 2$. No node in the two hop neighborhood is allowed to transmit data in the participation slot. Thus, there is at least one empty slot per $2 \times \text{max_rank}_a$ slots in any neighborhood of a . Note that there cannot be any one-hop neighbor node specified in the column corresponding to max_rank_a because of the way the rank of a node is computed. This empty slot is used for new nodes joining the network and nodes that lose their schedule because of mobility to transmit *Join* messages. The participation slot is made up of several sub-slots of fixed duration. Each subslot is long enough to transmit a single *Join* message, the size of which is constant. The number of subslots available for a node to transmit its *Joins* depends on the slot duration chosen for the network.

When a new node wants to join a network, it listens to the RTLs of nodes in the neighborhood for a time period equal to one *RTL_Interval*, after which the node starts to execute the mobility algorithm. If due to mobility, a node receives a new RTL with a different node identifier in its transmission slot, it chooses a new slot for transmission as explained in section III-B, calculates its rank, builds a *Join* packet and executes a mobility algorithm similar to the neighbor discovery algorithm proposed in [14]. The execution of the mobility algorithm proceeds in phases, each phase consisting of one or more sub-slots in the participation slot. In phase i , each node transmits with probability $1/2^i$, where each phase i lasts $2^i e(\ln 2^i + c)$ time slots, where c is a positive constant. The basic idea is that nodes geometrically reduce their transmission probabilities until they enter the desired phase of execution, when the chances of transmission without collision is maximized. A node listens to other *Joins* in the sub-slots in which it is not

transmitting. If it receives another *Join* with the chosen slot number, the node chooses a new slot and builds a different *Join* packet.

It has been shown [14] that the number of sub-slots W required for a node to transmit the packet without any other node transmitting in the same slot in the two-hop neighborhood of size n is $W \leq 2ne(\ln n + c)$, where c is a positive constant. For example, the 802.11b PHY Model with data rate 11 Mbps and a time-slot duration of NOMAD set to 1 ms results in about 300 sub-slots in a single participation slot, whereas a 54 Mbps 802.11g model gives more than 1200 sub-slots. Hence, when a close group of nodes of size 15 enters a network due to mobility and starts building new schedules after hearing conflicting RTLs, the nodes are able to transmit their *Joins* without collision within one participation slot, and hence within one schedule even with the 802.11b model. If there happens to be any conflicts at all, they get resolved and the nodes get a collision-free schedule within a maximum time period of one *RTL_Interval* and a few repetitions of the transmission schedule. The slot duration value that is set will depend on the packet delivery delay desired and synchronization capabilities of the network.

When a node k with a scheduled transmission slot receives a *Join* message from a node without a schedule in the participation slot, it forwards the message in its slot. If the slot specified in the *Join* belongs to node k , then it transmits its own *Join* in its scheduled transmission slot. If two or more *Joins* are received for the same slot it forwards the *Join* of the node with the lowest Node-ID. The participating node transmits an RTL in the new slot if it receives its *Join* message from atleast one one-hop neighbor and if there is no other conflicting node identifier reported or conflicting *Join* message forwarded for that slot. Conflicts in the chosen slot among two-hop neighbors is resolved as soon as an RTL is received from a common one-hop neighbor in the neighborhood with the conflicting node reported in the slot.

For example, consider the same network of Figure 1 and focus on the part of the network shown in Figure 6. Let a new node x joins the network from near node d . The node waits for a time period equal to one RTL interval, during which it hears the RTLs of nodes c , d and e . The node transmits *Joins* in sub-slots in slot 8 with probabilities as described above. Note that though NTS at d shows slot 8 as occupied by node b , it is clear from RTL of b that slot 8 is empty. If two nodes x and y try to join the network near node d , if the nodes are in range of each other, they would hear the *Joins* and choose different slots for transmission. If the nodes join two hops away and happen to choose the same slot, the common neighbor forwards only the *Join* of the smallest node ID, thereby forcing the node with greater node IDs to choose a different slot.

When a node chooses a new slot after receiving a different or new RTL from a neighbor, the node does not send the new RTL right away in the newly chosen slot. If there is no conflicting node specification in its old scheduled slot, the node transmits the new RTL in the old slot and then switches to transmitting in the new slot when there is consensus as per

Fig. 6.

algorithm 3. If there is a conflicting node in the old schedule, then the node builds a *Join* message and executes the mobility algorithm.

It may be noted that when a slot position x becomes vacant due to a node leaving a neighborhood, all nodes in the one-hop neighborhood which have chosen a higher slot number for its schedule will now choose slot x . But the node that transmits in slot $x+1$ as per the old schedule will transmit the new RTL before the other nodes and hence its new schedule would be to transmit at slot x and $x+1$ becomes vacant. This is true for subsequent slots. The slot positions occupied by nodes greater than x are moved to the left by one, vacating the slots towards the end of the schedule. Hence, no delay is introduced in the transmission due to nodes executing the mobility algorithm when nodes move away from a neighborhood.

When a new node arrives in a neighborhood, from the new RTLs received, the node detects a conflict with its old schedule. The node chooses the new slot for transmission, builds the *Join* packet and executes the mobility algorithm. To avoid cases where there is a conflict in schedule that goes undetected due to lack of any common neighbors, each node selects a subset of the time slots it selects for itself and remains silent during those time slots. This enables the node to receive conflicting RTLs or data packets in its transmission slot and hence build a new schedule.

IV. CORRECTNESS OF NOMAD

Once the nodes of the ad-hoc network have consistent knowledge of their two-hop neighborhood, NOMAD achieves the following goals:

- 1) Avoids collisions from simultaneous transmissions.
- 2) Provides fair sharing of the network bandwidth which avoids the resource starvation problem present in contention-based schemes.
- 3) Allows constant bandwidth utilization, even under heavy traffic load, and hence keeps data transmission live at all times.
- 4) Provides service to nodes at consistent intervals that depend on neighborhood size, so that the variance of the media access delay for individual nodes is zero

A collision occurs if two nodes in two hop neighborhood, a and b , transmit in the same slot. Let a_1 be the slot number that is chosen and broadcasted by node a for transmission. Let r_a be the rank of node a . Node a transmits every r_a slot after a_1 since the transmission list for a repeats itself every r_a slots.

Similarly, let b_1 and r_b be the slot number and rank of node b .

The rank r_a is always greater than or equal to the slot numbers chosen by all the nodes in the two hop neighborhood. Similarly, this is true for rank r_b of any node b in the two hop neighborhood of a with respect to the slot number chosen by node a .

Hence,

$$b_1 \leq r_a, \text{ and } b_1 \leq r_b \quad (1)$$

Similarly,

$$a_1 \leq r_a, \text{ and } a_1 \leq r_b \quad (2)$$

$$a_1 \neq b_1 \text{ and } b_1 > 0, a_1 > 0 \quad (3)$$

For a collision to occur, for some integers $m \geq 0$ and $n \geq 0$, we must have

$$a_1 + n.r_a = b_1 + m.r_b, \quad (4)$$

The ranks r_a and r_b are powers of two. Let k and l be integers such that,

$$r_a = 2^k \text{ and } r_b = 2^l$$

Then Eq. (4) becomes,

$$a_1 + n.2^k = b_1 + m.2^l, \quad (5)$$

$$(b_1 - a_1) = [m.2^l] - [n.2^k] \quad (6)$$

In Eq. (6), always, $LHS < \min(2^l, 2^k)$ (from Eqs. (1) - (3)) and $RHS \geq \min(2^l, 2^k)$

However, this makes Eq. (4) impossible. Hence, the transmissions of two nodes in the network cannot collide with each other as long as the nodes have consistent neighborhood information.

V. NOMAD PERFORMANCE

A. Channel Access Delays

Let X_k be the channel access delay for a node k in the network. For the NOMAD protocol, not taking the participation slot into account for simplification, for a given two-hop neighborhood of size N_k , X_k is $2^{\lceil \log_2 N_k \rceil}$. The mean and the second moments of the random variable X_k are:

$$\bar{X}_k = 2^{\lceil \log_2 N_k \rceil}$$

$$\bar{X}_k^2 = 2^{2\lceil \log_2 N_k \rceil}$$

In a fully connected network of size N_k , the NAMA protocol[2] has mean and second moments of the channel access delay as follows:

$$\bar{X}_k = 1/q_k$$

$$\bar{X}_k^2 = \frac{q_k^2 - 2q_k + 2}{q_k^2}$$

where $q_k = \frac{1}{N_k}$, N_k being the two-hop neighborhood size of node k .

Figure 8 shows the average channel access delay for a node for different two-hop neighborhood sizes. It can be seen

that the access delay for NOMAD is slightly more than that of NAMA. However, this is true only for a fully connected network. When the network is not fully connected, the two-hop neighborhoods are different at different nodes. In NAMA, when a node loses the election to a node one or two hops away, which in turn loses the election to another node in its neighborhood (and not in the neighborhood of the first node), it results in only one node transmitting in the entire three or four hop region. This results in sub-optimal utility of bandwidth and increased channel access delay in NAMA compared to NOMAD.

For example, consider the network shown in Figure 7, which is a small part of a wider grid network. Let N_i be the two-hop neighborhood of any node i and let N_a , the two-hop neighborhood of node a , be the region of interest in the network. The probability that the communication channel in the region N_a remains vacant during a time-slot is given by the probability that a node i in N_a wins the slot at node a , but loses to another node in $(N_i \setminus N_a)$. $(N_i \setminus N_a)$ has the value of 5 for 8 of the nodes in N_a and 8 for 4 of the nodes in N_a . Hence for a grid network of node degree 4, the probability that no node in N_a transmits in a slot is given by,

$$Pr(Empty_slot) = 8\left(\frac{1}{13} \times \frac{5}{13}\right) + 4\left(\frac{1}{13} \times \frac{8}{13}\right) = \frac{72}{169}$$

Therefore, the mean access delay (number of slots) for a node i using NAMA for a grid network is given by,

$$\bar{X}_i = 13 \times \frac{169}{87} = 25.25$$

The mean delay is 16 slots for NOMAD. This is shown in Figure 8.

Fig. 7. Grid Network

The variance of the channel access delay for NAMA is given by

$$Var(X) = \frac{q_k^2 - 2q_k + 2}{q_k^2} - 1/q_k^2$$

where $q_k = \frac{1}{N_k}$, N_k being the two-hop neighborhood size of node k . For NOMAD, this variance is zero. Figure 9 shows the standard deviation of the channel access delay for a node in NAMA and NOMAD for increasing neighborhood sizes. It is obvious that NOMAD is the preferred protocol under any type of node density.

Fig. 8. Mean Channel Access Interval

Fig. 9. Standard Deviation of Channel Access Interval

B. Simulation Results

We present simulation results comparing NOMAD with IEEE 802.11 DCF and NAMA. All the protocols use the 802.11b physical layer with two-ray pathloss model and no fading. We use goodput and channel access delay as our performance metrics. The goodput is the number of useful data bytes sent across the transmission link between nodes measured at higher layers in the protocol stack. This is to avoid counting the control packets and retransmitted data packets that might occur in throughput calculations at the MAC. The channel access delay is an indication of the fairness and the coupon-collector's problem in the protocols. The simulation was done for different terrain dimensions in order to change node density and number of collision domains. The nodes have a transmission range of 250m.

We employ a combination of random waypoint and group mobility [16] models as our mobility model. In our simulations, the members of a group move following the group mo-

Fig. 10. Goodput : Static Networks

Fig. 11. Goodput : Dynamic Networks

bility model whereas nodes inside the group move according to the random waypoint mobility model within the group area. We believe that this combined mobility model depicts more accurately common situations where a few members of the same team tend to move close by.

We used the discrete event simulator Qualnet [15] version 4.5, which provides a realistic simulation of the physical layer, and a well-tuned version of IEEE802.11DCF. Each simulation was run for ten different seed values. The slot duration for both NAMA and NOMAD was set to 1 ms, with both protocols capable of transmitting multiple data or control packets during the same slot and the *RTT_L* Interval set to 300 milliseconds.

Table. 1 lists the details of the simulation environment.

TABLE I
SIMULATION PARAMETERS

Total nodes	150	Node Placement	Random
		Simulation time	150s
Phy Model	802.11b	Tx Rate	11000000bps
Data Source	CBR		
Mobility model	Random waypoint	Pause-time	10s
		Min-Max Vel.	1-5 m/s
	Grp Mobility	Grp pause-time	10s
No. of Groups	5	Grp min-max Vel	1-5 m/s

C. Goodput

In this experiment, the traffic load of the network is increased continuously. Unicast data packets are delivered to a

one-hop neighbor that is chosen at random, and the average number of packets received in the network is measured. The experiment was done for three different terrain dimensions, and hence three different node densities, for both static networks (Fig 10) and dynamic networks (Fig 11). It can be seen that 802.11, NAMA and NOMAD deliver almost all the packets when the network load is low. However, as the number of nodes generating traffic and the traffic generated at each node is increased, NAMA and NOMAD perform better. When the traffic load in the network is high, NOMAD outperforms NAMA. This is because NOMAD provides better channel reuse, converges to feasible schedules faster, and has less channel-access delay variance, and therefore provides better bandwidth utilization than NAMA. NOMAD also adapts better to mobility than NAMA and outperforms 802.11 even in dense networks.

D. Mean and Standard Deviation of Access Delay

Access delay is the time delay between successive instants of time at which a node accesses the media for transmission. In this experiment, the average access delay is compared for NAMA and NOMAD for different node densities. The simulation for 150 nodes is run for different terrain dimensions (i.e. node densities), for both static and dynamic networks and the mean and standard deviation values are tabulated in Table II. It can be seen that the mean access delay for NOMAD without reservation is worse than that of NAMA especially in case of high node density (small terrain dimensions) because

TABLE II
MEAN AND STANDARD DEVIATION OF ACCESS INTERVAL(IN MS)

Static Networks 4000x4000			
Data Pkts Tx/Node	802.11	NAMA	NOMAD
1000	28.92 ± 12.2	4.7 ± 2.1	4.2 ± 0.068
2500	21.11 ± 12.5	4.7 ± 2.1	4.2 ± 0.068
5000	15.97 ± 9.2	4.7 ± 2.1	4.2 ± 0.068
7500	12.53 ± 8.9	4.7 ± 2.1	4.2 ± 0.068
10000	15.99 ± 11.3	4.7 ± 2.1	4.2 ± 0.068
2500x2500			
Data Pkts Tx/Node	802.11	NAMA	NOMAD
1000	34.66 ± 26.91	14.01 ± 12.13	12.98 ± 0.67
2500	42.86 ± 36.66	14.01 ± 12.13	12.98 ± 0.67
5000	48.33 ± 45.44	14.01 ± 12.13	12.98 ± 0.67
7500	49.93 ± 57.93	14.01 ± 12.13	12.98 ± 0.67
10000	55.24 ± 69.80	14.01 ± 12.13	12.98 ± 0.67
1000x1000			
Data Pkts Tx/Node	802.11	NAMA	NOMAD
1000	60.66 ± 48.92	36.64 ± 31.73	31.99 ± 5.23
2500	64.64 ± 58.23	36.64 ± 31.73	31.99 ± 5.23
5000	62.52 ± 72.39	36.64 ± 31.73	31.99 ± 5.23
7500	69.50 ± 81.78	36.64 ± 31.73	31.99 ± 5.23
10000	75.85 ± 87.28	36.64 ± 31.73	31.99 ± 5.23
Dynamic Networks 4000x4000			
Data Pkts Tx/Node	802.11	NAMA	NOMAD
1000	38.34 ± 19.56	5.81 ± 3.22	4.62 ± 0.96
2500	30.09 ± 16.78	5.81 ± 3.22	4.62 ± 0.96
5000	18.11 ± 10.11	5.81 ± 3.22	4.62 ± 0.96
7500	18.86 ± 14.21	5.81 ± 3.22	4.62 ± 0.96
10000	25.55 ± 15.35	5.81 ± 3.22	4.62 ± 0.96
2500x2500			
Data Pkts Tx/Node	802.11	NAMA	NOMAD
1000	37.3 ± 27.42	19.12 ± 15.89	20.98 ± 1.98
2500	47.6 ± 35.63	19.12 ± 15.89	20.98 ± 1.98
5000	58.22 ± 46.13	19.12 ± 15.89	20.98 ± 1.98
7500	59.45 ± 62.11	19.12 ± 15.89	20.98 ± 1.98
10000	70.97 ± 71.33	19.12 ± 15.89	20.98 ± 1.98
1000x1000			
Data Pkts Tx/Node	802.11	NAMA	NOMAD
1000	61.65 ± 51.92	58.97 ± 37.87	50.39 ± 10.11
2500	65.64 ± 57.32	58.97 ± 37.87	50.39 ± 10.11
5000	67.51 ± 75.44	58.97 ± 37.87	50.39 ± 10.11
7500	72.50 ± 85.18	58.97 ± 37.87	50.39 ± 10.11
10000	77.85 ± 84.32	58.97 ± 37.87	50.39 ± 10.11

more nodes in the neighborhood mean longer schedules, which results in many empty unused slots. The mean access delay is lower when reservations are enabled and is inversely proportional to the packet delivery as expected.

The standard deviation of the access delay is much lower in NOMAD than in NAMA or 802.11. This is because a consistent schedule results in nodes accessing the medium at intervals closer to the mean.

VI. CONCLUSION

NOMAD is self-stabilizing and allows each node to start computing its transmission schedule without having any information from its one-hop neighbors. NOMAD establishes deterministic transmission schedules dynamically without requiring the channel to be organized into fixed transmission frames consisting of a pre-defined number of transmission slots and without multiple rounds of negotiations among nodes. It was shown through simulations that NOMAD outperforms 802.11

and NAMA in terms of packet delivery and also provides deterministic delays between two consecutive transmission slots assigned to the same node. Our work on NOMAD opens up a new area of research in distributed and deterministic transmission scheduling.

REFERENCES

- [1] B.P. Crow, I. Widja ja, L.G. Kim, and P.T. Sakai. "IEEE 802.11 Wireless Local Area Networks". *IEEE Communications Magazine*, 35(9):11626, Sept 1997.
- [2] L. Bao and J.J. Garcia-Luna-Aceves, "Distributed Channel Access Scheduling for Ad Hoc Networks," Chapter 4, Handbook of Algorithms for Wireless Networking and Mobile Computing (A. Boukerche, Ed.), Chapman and Hall/CRC, 2006.
- [3] I. Chlamtac, A. Farago, and H. Zhang, "Time-spread multiple-access (TSMA) protocols for multihop mobile radio networks," *IEEE/ACM Transactions on Networking*, Vol. 6 No. 5, pp. 804–12, Dec. 1997
- [4] J. Ju and V. Li, "An optimal topology-transparent scheduling method in multihop packet radio networks," *IEEE/ACM Transactions on Networking*, Vol. 6, No. 3, pp. 298-306, Jun. 1998
- [5] Carlos H. Rentel, and Thomas Kunz, "On the Average-Throughput Performance of Code-based Scheduling Protocols for Wireless Ad Hoc Networks," *6th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, Urbana-Champaign, IL, May 2005.
- [6] I. Chlamtac, "Fair Algorithms for Maximal Link Activation in Multihop Radio Networks," *IEEE Transactions on Communications*, Vol. COM-35, no. 7, July, 1987.
- [7] A. Ephremides, T. Truong, "Scheduling Broadcasts in Multihop Radio Networks," *IEEE Transactions on Communications*, Vol. COM-38, No. 4, April, 1990.
- [8] C. D. Young, "USAP: A Unifying Dynamic Distributed Multichannel TDMA Slot Assignment Protocol," *Proc. IEEE MILCOM 96* October 1996.
- [9] Z. Tang and J.J. Garcia-Luna-Aceves, "A Protocol for Topology-Dependent Transmission Scheduling," *Proc. IEEE WCNC 99*, New Orleans, Louisiana, Sept. 21-24, 1999.
- [10] Z. Tang and J.J. Garcia-Luna-Aceves, "Hop reservation Multiple Access for Multichannel Packet Radio Networks," *Computer Communications*, Elsevier, 2000.
- [11] R. Rozovsky and P. R. Kumar, "SEEDEX: A MAC protocol for ad hoc networks." *Proceedings of The ACM Symposium on Mobile Ad Hoc Networking and Computing*, MobiHoc 2001, pp. 67-75, Long Beach, Oct. 4-6, 2001.
- [12] L. Bui, S. Sanghavi, and R. Shirkant, "Distributed Link Scheduling with Constant Overhead," *IEEE/ACM Trans. Networking*, Vol. 17, No. 5, Oct. 2009.
- [13] C. Joo and N. Shroff, "Performance of Random Access Scheduling Schemes in Multi-hop Wireless Networks," *IEEE/ACM Transactions on Networking*, Vol.17, No. 5, October 2009.
- [14] S.Vasudhevan, D.Towsley, D.Goeckel, and R.Khalili, Neighbor Discovery in Wireless Networks and the Coupon Collectors Problem, *Proceedings of the 15th annual international conference on Mobile computing and networking*, pp181-192.
- [15] Qualnet4.5, ScalableNetworkTechnologies: <http://www.scalablenetworks.com>
- [16] X. Hong, M. Gerla, G. Pei, and C.-C. Chiang, A group mobility-model for ad hoc wireless networks. In *Proc. of ACM/IEEE MSWiM99*, pages 5360, 1999.
- [17] C. H. Rentel and T. Kunz, "A mutual network synchronization method for wireless ad hoc and sensor networks," *IEEE Transactions on Mobile Computing*, Vol. 7, No. 5, pages 633-646, May 2008.
- [18] P. Djukic and P. Mohapatra, "Soft-TDMAC: Software TDMA-based MAC over Commodity 802.11 hardware," *IEEE INFOCOM 2009*.