Consiglio Nazionale delle Ricerche

The SCREAM Approach for Efficient Distributed Scheduling with Physical Interference in Wireless Mesh Networks

G. Brar, D. Blough, P. Santi

IIT TR-08/2006

Technical report

Agosto 2006

Istituto di Informatica e Telematica

The SCREAM Approach for Efficient Distributed Scheduling with Physical Interference in Wireless Mesh Networks

Gurashish Brar
School of ECE
Georgia Inst. of Technology
Atlanta, GA

Douglas M. Blough
School of ECE
Georgia Inst. of Technology
Atlanta, GA

Paolo Santi
Istituto di Informatica
e Telematica del CNR
Pisa, Italy

***Abstract*— It is known that CSMA/CA channel access schemes are not suitable to meet the high traffic demand of wireless mesh networks. One possible way to increase traffic carrying capacity is to use a spatial TDMA (STDMA) approach in conjunction with the physical interference model, which allows more aggressive scheduling than the protocol interference model on which CSMA/CA is based. However, a major difficulty in using STDMA with physical interference is the inherent complexity of this interference model. While an efficient, centralized solution for STDMA with physical interference has been recently proposed, no satisfactory distributed approaches have been introduced so far. In this paper, we first prove that no localized distributed algorithm can solve the problem of building a feasible schedule under the physical interference model. Motivated by this, we design a global primitive, called SCREAM, which is used to verify the feasibility of a schedule during an iterative distributed scheduling procedure. Based on this primitive, we present two distributed protocols for efficient, distributed scheduling under the physical interference model, and we prove an approximation bound for one of the protocols. We also present extensive packet-level simulation results, which show that our protocols achieve schedule lengths very close to those of the centralized algorithm and have running times that are practical for mesh networks.**

I. Introduction

In wireless mesh networks, wireless backbone nodes must convey a large amount of traffic generated by wireless clients to a few nodes that act as gateways to the Internet. For these networks, the main design concern is increasing the traffic carrying capacity of the wireless backbone as much as possible. The main factor that limits capacity in mesh networks is *interference*, which is a consequence of using a shared communication medium. Hence, an accurate modeling of interference is fundamental in order to derive theoretical and/or simulation-based results of some practical relevance. In the literature, two main interference models have been proposed [10]: the *protocol* and the *physical* interference models.

In the protocol model, a communication from node u to node v is successful if no other node within a certain *interference range* from v is simultaneously transmitting. Due to its simplicity, and to the fact that this model can be used to mimic the behavior of CSMA/CA networks such as IEEE 802.11 [1], the protocol interference model has been mostly used in the literature. In the physical interference model, a communication between nodes u and v is successful if the SINR (Signal to Interference and Noise Ratio) at v (the receiver) is above a certain threshold, whose value depends on the desired channel characteristics (e.g., data rate). This model is less restrictive than the protocol interference model and higher network capacity can in general be achieved by applying the physical interference model. [1]

Recent research indicates that CSMA/CA is not suitable to meet the high traffic demand of wireless mesh networks. The reason for this is that CSMA/CA is a very conservative mechanism: due to the combination of carrier sensing and collision avoidance techniques, many network nodes are silenced when a certain communication takes place. This is the reason why existing implementations of 802.11-based mesh networks disable the collision avoidance mechanism (i.e., the RTS/CTS message exchange) [3], or completely new TDMA-like MAC protocols are proposed for mesh networks [16], [19].

The above discussion motivates use of the physical interference model in investigations of the capacity of wireless mesh networks. A major difficulty lies in the complexity of handling physical interference. In fact, most of the related work on scheduling with physical interference presents centralized algorithms that are either exponential time or for which time complexities are not given [7], [8], [11], [18]. None of these algorithms have proven approximation bounds. Other works dealing with the physical interference model consider only a simplified scenario with unit traffic demand on each link [13], [14], which is not representative of real-world wireless mesh network deployments. In a recent paper [4], we published the first *centralized* scheduling algorithm for the physical interference model that runs in polynomial time and has a proven approximation factor relative to the optimal schedule. Currently, there is no known distributed algorithm with a proven approximation factor. In fact, to our knowledge, [9] is the only existing distributed scheduling algorithm that accounts for physical interference. [9] uses a localized approach where only interferers within a short network distance of a receiver are considered. We prove in this paper that localized algorithms can not guarantee to produce a feasible schedule under the physical interference model.

In this paper, we take a novel approach to distributed

[1] The physical interference model is representative of a scenario that *does not* use CSMA techniques; instead, transmissions should be carefully scheduled, using TDMA-like channel access schemes, so that only sender/receiver pairs that do not conflict with each other transmit simultaneously.

scheduling, which is based on a global primitive that we refer to as a SCREAM. In our approach, nodes iteratively build a feasible schedule one slot at a time, and the SCREAM primitive is used to quickly determine whether communications being attempted in a slot are feasible. In this way and using ideas from our centralized scheduling algorithm [4], we are able to design a distributed scheduler that is efficient in running time and maintains the proven approximation bound of [4]. We also present a variant of our scheduling algorithm that has slightly reduced performance (in terms of schedule length) but runs substantially faster. For each of the proposed algorithms, we present detailed simulation results that demonstrate that their schedule lengths are very close to the centralized algorithm, while their execution times are practical for typical mesh network sizes.

II. Network model and assumptions

We consider a wireless mesh network composed of n nodes (wireless routers). The links among nodes are represented by the *communication graph* $G = (V, E)$, where V is the set of nodes, and edge $e = (u, v) \in E$ if and only if a link between nodes u and v exists in absence of interference from any other network node. We do not assume any specific radio propagation model, nor that all the nodes use the same transmission power. Hence, unidirectional links can be present in the physical communication graph. However, to provide fair comparisons against 802.11, we assume that link-layer reliability (using ACKs) is employed even for STDMA [15]. We, therefore, assume that unidirectional links are not used even if they are present and, hence, we ignore them in G.

We assume that no transmit power control technique is used, i.e. all the nodes send packets using a fixed transmit power level (which, however, can be different for every node). We model interference using a variation of the physical interference model [10] introduced in [4], which we summarize for completeness. Differently from [10], the model of [4] accounts for link-layer reliability. In particular, it is assumed that a packet sent by node u is correctly received by node v if and only if the packet is successfully received by v, and the ACK sent by node v is correctly received by node u. Furthermore, for a transmission from node x to node y that is concurrent with the packet on (u, v), the model accounts for the interference both from node x's data packet *and* from node y's ACK. In this paper, we consider a minor variation of the model of [4], where slots are divided into two sub-slots, one for data packet transmission and one for ACK transmission, so that data packets and ACKs do not overlap. Thus, a packet sent from u to v is correctly received if and only if:

$$\frac{P_v(u)}{N + \sum_{x \in V'} P_v(x)} \geq \beta \quad \text{and} \quad \frac{P_u(v)}{N + \sum_{y \in V''} P_u(y)} \geq \beta \,,$$

where V' contains all nodes that are transmitting data packets in the same slot as u, V'' contains the corresponding nodes that send ACKs to the nodes in V' in that slot, $P_r(t)$ denotes the received power at r of the signal transmitted by node t, N is the background noise, and β is a constant that depends on the desired data rate, modulation scheme, etc. Based on this interference model, we say that a set $E' \subset E$ of transmissions is *feasible* if and only if all of them can be scheduled concurrently and correctly received.

We now introduce the concept of *interference diameter*, which is used in the definition of our protocols.

Definition 1 (Sensitivity Graph): For a given communication graph $G = (V, E)$, the sensitivity graph $G_S = (V, E_S)$ is defined on the same node set V. Directed edge $(u, v) \in E_S$ if and only if node v can detect some activity on the channel when node u is transmitting, and all the other nodes remain silent.

It is easy to see that the sensitivity graph is a super-graph of the communication graph $G = (V, E)$.

Definition 2 (Interference Diameter): The interference diameter of a network represented by the sensitivity graph $G_S = (V, E_S)$ is defined as the maximum hop distance between any two nodes in G_S. Formally,

$$ID(G_S) = \max_{u,v \in V} d_{G_S}(u, v) \ ,$$

where $d_{G_S}(u, v)$ is the hop length of the minimum length directed path connecting u to v in G_S. If G_S is not strongly connected, we define $ID(G_S) = \infty$.

Since we can assume that the communication graph of a wireless mesh is strongly connected, and the sensitivity graph is a super-graph of the communication graph, from now on we assume that G_S is also strongly connected, i.e. the interference diameter is finite.

The traffic generated at each node in the mesh is conveyed to one or more pre-defined gateway nodes, which provide access to the Internet. In this paper, we assume that traffic is routed to the gateways along reverse trees RT rooted at the gateways, which thus form a routing forest RF. A node that is not a gateway decides which tree to join depending on a simple criteria, i.e. minimum hop distance to the root, breaking ties randomly. Note that the set of edges forming RF must be a subset of E. In the following, we use the term edge to refer to an edge in RF only. Each node has some traffic demand associated with it. Since each node u is part of exactly one tree, the aggregated demand on the link connecting node u with its parent in RT equals the sum of the demands generated at the nodes belonging to the subtree rooted at u.

The protocols described in this paper allocate $\mid demand(e) \mid$ slots for each edge $e \in RF$. For each edge $e = (u, v)$, one of the nodes u or v (whichever is at higher depth in the RT) is in charge of allocating slots so as to satisfy the demand on e. Since we have established a one-to-one mapping between nodes and edges (except the root nodes – i.e., gateways –, which are not associated with any edge), from now on we use the terms edge or node interchangeably. Note that, up to straightforward modifications, the protocols presented in this paper can be used to schedule an arbitrary link set (not necessarily a forest).

We assume all nodes have their clocks synchronized to a global time, within a reasonable degree of accuracy.

In subsequent algorithm descriptions we use the function $GlobalSync()$ to wait for the next time period which is globally synchronized. In Section VI, we investigate the effect of clock skew on the protocols' performance.

III. Distributed Scheduling Protocols

In this section, we present two distributed protocols based on the centralized greedy scheduling algorithm presented in [4]:

1) Partially Deterministic Distributed Protocol (PDD);
2) Fully Deterministic Distributed Protocol (FDD).

The algorithms proceed in rounds, scheduling one slot in each round. At the beginning of each round, a node is selected as the controlling node for the slot through leader election. The algorithms guarantee that at least the edge associated with the controller is scheduled in the slot, which implies algorithm termination. The set of scheduled links is augmented in a greedy fashion, by adding nodes to the slot in steps. In a given step, there is some set of edges that are already scheduled (call these the "previously scheduled edges"), and a new set of edges (called "active edges") are computed in a distributed fashion and tentatively added to the slot. For all previously scheduled edges and active edges, a two-way handshake is performed. If any previously scheduled edge fails its handshake, it initiates a SCREAM (see Section III-A), and each active edge removes itself from the slot when it hears the SCREAM. If all the handshakes between previously scheduled edges are successful (indicated by the absence of a SCREAM), the active edges whose handshakes were successful are included in the slot. The other active edges (whose handshakes failed) are discarded. Active edges that are not added to the schedule can be re-scheduled only in the next round. When no new set of active edges can be selected, the slot is sealed, and the next round is initiated. This process continues, adding one slot to the schedule at each round, until all node demands are satisfied.

The only difference between PDD and FDD is in the way the set of active nodes is selected: according to a randomized strategy in case of PDD, and through a network-wide leader election algorithm in case of FDD. While the fully deterministic allocation strategy used in FDD allows proving an important approximation result about the computed schedule quality, PDD is considerably faster than FDD, while at the same time providing a scheduling performance close to the one provided by PDD (see Section VI).

Both FDD and PDD rely on a network-wide leader election algorithm. The cost of leader election is dramatically reduced, compared to existing schemes, using a new distributed primitive called SCREAM, which we describe next. A significant advantage of SCREAM is that it is resilient to collisions and, therefore, has a deterministic execution time. This provides a foundation for bounding execution time of the higher-level protocols that use it.

A. The SCREAM primitive

Let $var(i)$ be a Boolean variable stored at node u_i. The SCREAM primitive provides a network-wide OR operation on the Boolean variables stored at each node in the network. That is, after the SCREAM primitive is run, each node holds the result of $var(1) \vee var(2) \vee \cdots \vee var(n)$.

The SCREAM primitive runs through K $SCREAM\_SLOTS$, where $K \geq ID(G_S)$. The primitive uses two functions:

1) $Scream()$: Transmits $SMBytes$ on radio interface;
2) $Listen()$: Listens for Activity on radio interface. Returns $true$ if Activity detected, $false$ otherwise.

The SCREAM primitive is built from the above two functions as follows:

```
1: SubRoutine bool : SCREAM(var)
2: relay = var
3: for sslot = 1 → K do
4:    GlobalSync()
5:    if relay then
6:       Scream()
7:    else
8:       relay = Listen()
9:    end if
10: end for
11: return relay
```

It is important for every node in the network to participate in the above SCREAM subroutine even if they do not have a variable var value to contribute. These nodes participate passively and simply relay the scream.

The SCREAM primitive is based on the carrier sensing mechanism to detect activity in the medium. We rely on the basic assumption that carrier sensing is resilient to collisions. In Section V, we present results from experiments performed on Mica 2 motes, which demonstrate the feasibility of the SCREAM primitive.

B. Leader Election

This algorithm assumes that every node has a unique id (e.g., its MAC address). Let id_bits be the number of bits required for representing the id. The following algorithm performs leader election by selecting the node with the highest unique id. The $LeaderElect(ID_i)$ function takes as input the unique id ID_i of the invoking node u_i, and returns $true$ if u_i is the leader, and $false$ otherwise.

```
1: SubRoutine bool : LeaderElect(ID_i)
2: votedout = false
3: for j = (id_bits − 1) → 0 do
4:    GlobalSync()
5:    if ID_i(j) ∧ ¬votedout then
6:       SCREAM(true)
7:    else
8:       votedout = SCREAM(false) ∨ votedout
9:    end if
10: end for
11: return votedout
```

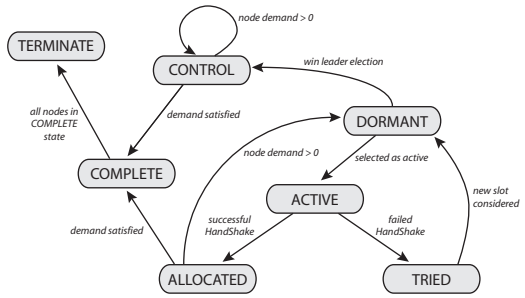

Fig. 1. State transition diagram of the PDD and FDD algorithms.

In the above algorithm, $ID_i(j)$ refers to the j-th bit of ID_i. The algorithm iterates through the bits in the unique id, starting from the most significant bit. During each iteration, a network wide OR is performed on the bit values of all unique ids at the corresponding index. If the OR result does not match the local bit value, the node is voted out and for the rest of the iterations contributes with a 0 bit value. It is immediate to see that, at the end of $LeaderElect$'s execution, only the node with highest ID has not been voted out, and wins the election. It is also immediate to see that the above algorithm has $O(K \cdot \ln n)$ time complexity, assuming $id\_bits = \ln n$.

We are now ready to describe the PDD and FDD protocols.

C. Partially Randomized Distributed Protocol (PDD)

During PDD's execution each node can be in one of the following mutually exclusive states:

1) CONTROL: controller of the current slot;
2) ALLOCATED: allocated to the current slot;
3) ACTIVE: active node, whose edge is tentatively included in the current slot;
4) TRIED: an active node which could not join the current slot because of failed handshake;
5) DORMANT: a node which has not been picked up yet in any of the active subsets;
6) COMPLETE: a node whose demand has been satisfied;
7) TERMINATE: the algorithm has terminated

Figure 1 shows PDD's state transition diagram. The following algorithm describes PDD in further detail:

Input: K, upper bound on network interference diameter
Output: $Slot$: set of reserved slots
$scheduleLength$: the length of the schedule

$Slot \leftarrow \phi$
$scheduleLength \leftarrow 0$
$curSlot \leftarrow 0$
$curDemand \leftarrow 0$
$state \leftarrow DORMANT$
$Released \leftarrow true$
repeat
  if $Released$ **then**
    if $state \neq COMPLETE$ **then**
      if $LeaderElect(ID_i) = true$ **then**
        $GlobalSync()$
        $state \leftarrow CONTROL$
        $result \leftarrow SCREAM(true)$
      else
        $GlobalSync()$
        if $SCREAM(false) = false$ **then**
          $state = TERMINATE$
        end if
      end if
    else
      $LeaderElect(0)$ *# Passive participation*
    end if
  end if
  if $state \neq TERMINATE$ **then**
    $GreedyScheduleSlot(curSlot, state)$
    $Released = CheckControlRelease(state)$
  end if
  $curSlot \leftarrow curSlot + 1$
  $scheduleLength \leftarrow scheduleLength + 1$
until $state \neq TERMINATE$

The following pseudo code, describes scheduling of a single slot with a $CONTROL$ node already elected. All nodes which are not in the $COMPLETE$ or $CONTROL$ state are returned to the $DORMANT$ state. The $SelectActive$ function is used to build the $Active$ set of nodes. The $ACTIVE|CONTROL|ALLOCATED$ nodes then perform the two-way-handshake on each currently allocated link, including the tentatively allocated ones. Following this, if none of the $CONTROL|ALLOCATED$ nodes veto, the successful $ACTIVE$ nodes are included in the set of allocated edges for the slot. The unsucessful $ACTIVE$ nodes change their state to $TRIED$. The subroutine terminates when no more active nodes can be selected, i.e. when the $stillActives$ variable becomes false.

SubRoutine: GreedyScheduleSlot($curslot, state$)
if $state \neq COMPLETE|CONTROL$ **then**
  $state \leftarrow DORMANT$
end if
repeat
  if $state = DORMANT$*# Build Active set* **then**
    if $SelectActive() = true$ **then**
      $state \leftarrow ACTIVE$
    end if
  end if
  $GlobalSync()$
  {*# Handshake time step*}
  $HSfail \leftarrow false$
  if $state = ACTIVE|ALLOCATED|CONTROL$ **then**
    $HSfail \leftarrow DoHandShake()$
  end if
  {*# Verification time step*}
  if $state = ALLOCATED|CONTROL$ **then**
    $HSfail \leftarrow SCREAM(HSfail)$ *#Veto Power*
  else

$HSfail \leftarrow SCREAM(false)$
end if
$stillActives \leftarrow false$ *# Check if active set empty*
if $state = ACTIVE$ **then**
$stillActives \leftarrow true$
if $HSfail = false$ **then**
$state \leftarrow ALLOCATED$
else
$state \leftarrow TRIED$
end if
end if
$GlobalSync()$
$stillActives \leftarrow SCREAM(noMoreActive)$
until $stillActives = false$

The $SelectActive()$ function chooses the set of active nodes. In PDD, we use a probabilistic approach to determine active nodes, i.e. nodes that are in the DORMANT state become ACTIVE with a certain probability p.

The $DoHandShake()$ function performs the two-way handshake on the communication edge $e = (u, v)$ being scheduled, which is associated with the head node. The head of the edge sends a data packet in the first sub-slot to the tail node. In case the tail node correctly receives the packet, it sends back to the head an ACK packet in the next sub-slot. Upon correct reception of the ACK, the head node declares the two-way handshake successful, and the function returns $false$. In case of unsuccessful handshake, the function returns $true$.

To determine whether the demand of the $CONTROL$ node has been satisfied at the end of a round, a network wide OR is performed using the $SCREAM$ primitive with only the $CONTROL$ node having a $true$ value if its demand is satisfied. In this way, all the nodes know whether the control has been released and a new leader for the next slot to be scheduled must be elected.

D. Fully Deterministic Distributed Protocol (FDD)

The FDD protocol follows the exact procedure as PDD. The only difference between the two protocols is in the $SelectActive()$ function. In FDD, a single new node is selected as ACTIVE at each step, where the active node is determined through network-wide leader election.

IV. Analysis

A. Impossibility of localized distributed scheduling

In this section, we prove that no localized distributed algorithm can be used to compute a feasible schedule under the physical interference model. We first need some definitions.

Definition 3 (Link hop distance): Given any two links l_1, l_2 in the network, their hop distance is the minimum hop distance between their endpoints in the communication graph.

Definition 4 (Link k-neighborhood): Given any link l, the k-hop neighborhood of l is the set of all links at most k hops away from l.

Definition 5 (Locality): A distributed scheduling algorithm is localized if and only if it computes a schedule under the chosen interference model by taking a decision on whether a certain link l can be scheduled in a certain slot t_i based only on the information regarding links in the k-hop neighborhood of l, where k in an arbitrary constant ($k \in O(1)$).

Theorem 1 (Impossibility result): No localized distributed algorithm can compute a feasible schedule under the physical interference model for the general case of networks with arbitrary node distribution and arbitrary radio propagation model.

Proof: [Sketch] Assume A is a localized distributed algorithm for computing a feasible scheduled under the physical interference model. By assumption, when considering a specific link l to be scheduled, the decision on whether link l can be scheduled in slot t_i is taken only by considering information regarding links in the k-hop neighborhood of l. Consider a certain link l' outside the k-hop neighborhood of l. Note that, given the assumption of arbitrary node distribution, we can always build an example in which the hop diameter of the network is $\Theta(n)$ (e.g., nodes along a line), and choose l and l' such that their hop distance is $\Theta(n)$ (e.g., links at the opposite sides of a line). Since by assumption $k = O(1)$, we have that such a l, l' link pair always exists. Given then the link pair l, l', and given the locality assumption on algorithm A, we have that the decision on whether l should be scheduled in slot t_i is oblivious to whether l' is also scheduled in the same slot. Assume now that slot t_i is feasible if link l is scheduled concurrently to the currently scheduled links E_i for t_i, but it is infeasible if both l and l' are scheduled concurrently to links in E_i. Note that, given the assumption of arbitrary node distribution and radio propagation model, an example in which this situation occurs can always be built. Due to the locality assumption, algorithm A has no possibility to know whether l' is also scheduled in t_i when taking the decision about link l, possibly leading to the construction of an infeasible schedule. ■

Note that the above theorem can be easily extended to a weaker notion of locality, in which nodes can communicate up to hop-distance $f(n)$, with $f(n) \in o(n)$.

B. Characterization of the interference diameter

The SCREAM primitive constitutes a fundamental building block of the distributed scheduling algorithms. In order for this primitive to work properly (i.e., to implement a network-wide OR operation), we need to upper bound the interference diameter of the network. We recall that the SCREAM primitive is invoked with a parameter K, which determines the duration of the primitive expressed as the number of slots. In order for SCREAM to correctly implement network-wide OR, we must have $K \geq ID(G_S)$ (the proof of this fact is straightforward). To reduce time complexity, we must ideally set $K = ID(G_S)$. This leads to the problem of estimating (an upper bound to) the interference diameter of a certain sensitivity graph.

In the following, we present upper bounds on the interference diameter of the sensitivity graph in different scenarios, under the following assumption:

– *CS range*: we assume that $(u, v) \in E_S$ if and only v is within the carrier sensing range r_{CS} of node u.

For simplicity, we also assume that all the nodes in the network have the same CS range. With this assumption, G_S can be regarded as an undirected graph.

In general the CS range r_{CS} is at least as large as the communication range[2] r_c. The larger r_{CS} is with respect to r_c, the denser is the sensitivity graph G_S with respect to the communication graph G, and the lower the interference diameter. Since we are interested in providing an *upper bound* on the interference diameter, we should consider the minimum possible meaningful value of r_{CS} with respect to r_c, i.e. $r_{CS} = r_c$. For this reason, from now on we assume $r_{CS} = r_c$, and we use the simpler notation r to denote both the CS and the communication range. Note that, under the above assumption, the sensitivity graph G_S coincides with the communication graph G. Thus, from now on the concept of interference diameter will be applied to the communication graph G.

Let us now introduce the concept of *neighbor density*, which will be used to classify the different scenarios considered in the following.

Definition 6: Let $G = (V, E)$ be the communication graph of a network composed of n nodes with communication range $r = r(n)$. The neighbor density $\rho(G)$ of G is the average node degree in G, i.e. the average number of 1-hop neighbors of a network node.

In the next subsection, we consider three different scenarios, with increasing neighbor density: square grid deployments ($\rho(G) = \Theta(1)$), random uniform deployments ($\rho(G) = \Theta(\log n)$), and infinite density deployments ($\rho(G) = \Theta(n)$). The analyses of these scenarios seem to indicate (we have no formal proof of this fact, though) that the following relation between the neighbor density and the interference diameter occurs:

$$ID(G) = O\left(\sqrt{\frac{n}{\rho(G)}}\right) ,$$

i.e. the higher the neighbor density, the lower the interference diameter. This fact is quite interesting, since it indicates that a high network density, which is often considered as detrimental under many respects (e.g., capacity limitation), is advantageous to reduce the interference diameter of the network, and hence speeds up scheduling computation.

1) Square grid deployments: Square grid deployments can be considered as a minimal neighbor density scenario: by properly choosing the communication range and the grid step, neighbor density can be made as low as $O(1)$. In particular, in the following we assume a square grid deployment, with the communication range exactly set to the value of the grid step. With this configuration, each node in the network S has exactly four neighbors, independently of the number of network nodes, i.e. $\rho(S) = \Theta(1)$.

Before proving the main result of this section, we need some preliminary definitions.

Definition 7 (Square grid augmentation): Assume a square lattice of arbitrary step $s > 0$ is super-imposed on the two-dimensional Euclidean plane, and define a cell as a single square which is part of the lattice. For any line segment l in the plane, the square grid augmentation $Augm(l)$ of l is defined as the region of the plane obtained as the union of the cells which are traversed by l.

[2]Implicit in this discussion is the fact that we are assuming a deterministic radio propagation model, such as the log-distance path model.

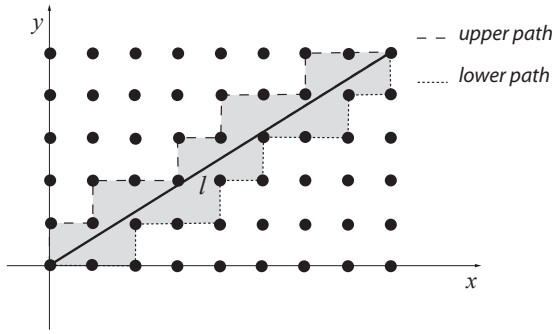

Fig. 2. Square grid augmentation of line segment l (shaded region), and corresponding upper and lower paths.

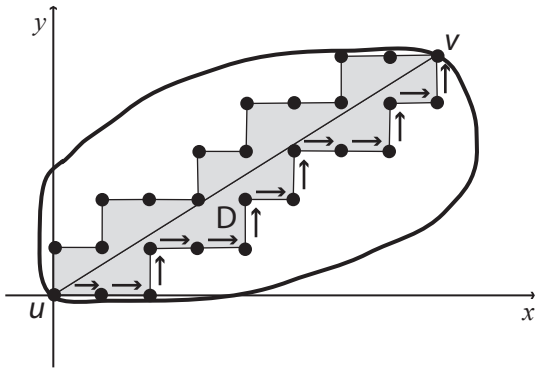

Fig. 3. Square grid diameter of a square grid convex region.

Definition 8 (Lattice paths): Let u, v be two arbitrary points of the above defined lattice, and let l be the line segment connecting u and v. The upper lattice path of l is defined by connecting all the lattice points in $Augm(l)$ whose y coordinate is above segment l, and the lower lattice path of l is defined by connecting all the lattice points in $Augm(l)$ whose y coordinate is below segment l. If segment l is parallel to the y axis, we arbitrarily define the upper lattice path the one on the left of l, and the lower path the one on the right of l.

An example square grid augmentation of a line segment and the corresponding upper and lower paths is reported in Figure **??**.

Definition 9 (Square grid interior): Let R be an arbitrary closed region of the two-dimensional plane. The square grid interior $Int(R)$ of R is defined as the set of points in the above defined lattice lying in the interior of R.

Definition 10 (Square grid convexity): Let R be an arbitrary closed region of the two-dimensional plane. R is said

to be square grid convex if an only if, for any two points u, v in $Int(R)$, we have that at least one of the lattice paths of the segment $\overline{uv}$ is contained in the interior of R.

Definition 11 (Diameter): Let R be any closed region of the two-dimensional Euclidean plane. The diameter of R is defined as the maximum length of a line segment connecting two points in R. Formally,

$$diam(R) = \max_{u,v \in R} d(u, v) \ ,$$

where $d()$ is the Euclidean distance.

An example of square grid diameter is reported in Figure **??**.

We are now ready to prove the main result of this section.

Theorem 2: Assume n nodes are deployed in a square grid lattice of step r (the communication range) inside a certain two-dimensional closed region R. Let us denote with $G = (V, E)$ the resulting communication graph. If R is square grid convex, then $ID(G) \leq \sqrt{2} \cdot \frac{diam(R)}{r}$.

Proof: Let us consider an arbitrary line segment l whose both endpoints are in $Int(R)$. Since R is square grid convex, at least one of the lattice paths associated to l in entirely included in $Int(R)$. Hence, for any pair of nodes u, v in G, an upper bound on their 'interference distance' can be derived by upper bounding their hop distance in one of the lattice paths associated to the line segment l connecting them. Denote with β the angle between line l and the x axis. Without loss of generality, assume $0 \leq \beta \leq \frac{\pi}{2}$ (the proof for the other cases is the same, up to symmetries). It is easy to see that the hop length in the square grid of both lattice paths associated to line l of length $\bar{l}$ equals

$$\frac{\bar{l}}{r} \cdot \sin\beta + \frac{\bar{l}}{r} \cos\beta = \frac{\bar{l}}{r} \cdot (\sin\beta + \cos\beta) \ .$$

The upper bound on the interference diameter easily follows by observing that $\bar{l} \leq diam(R)$ and that $\sin\beta + \cos\beta \leq \sqrt{2}$. ■

Observe that the bound stated in Theorem 2 is tight: in fact, if R is a square perfectly aligned with the lattice, we have that $diam(R) = r\sqrt{2n}$, and $ID(G) = 2\sqrt{n}$. Hence, the interference diameter of a square grid network with n nodes deployed in a square region R is $\Theta(\sqrt{n}) = \Theta\left(\sqrt{\frac{n}{\rho(G)}}\right)$.

2) Random uniform deployments: The analysis of the random uniform deployment scenario is based on the well-known subdivision of the deployment region into equally sized cells, and on the use of occupancy theory.

In particular, similarly to [4], we assume the following:

- the deployment region R is the unit square $[0, 1]^2$.
- n nodes with communication range $r = r(n) = \sqrt{\frac{\ln n}{\pi n}}$ are distributed uniformly at random in R.

With the above assumption, it is known that the resulting network is connected w.h.p.[3], and the communication range $r(n)$ is the minimum possible value (in asymptotic terms) of the communication range which is necessary for connectivity w.h.p. It is easy to see that the neighbor degree of the resulting network is $\Theta(\log n)$ w.h.p., and this is the minimum possible node degree needed for connectivity w.h.p. in random uniform networks.

Let us subdivide R into $C = \frac{8}{r^2}$ square cells of equal side $\frac{r}{2\sqrt{2}}$. The cell side is set such that any two nodes in adjacent cells (horizontal, vertical, and diagonal adjacency) are within each other communication range. Using standard arguments from occupancy theory (see [12]), it is known that by setting r and C as above every cell contains at least one node w.h.p. Hence, an upper bound on the interference diameter of the network is given the number of cells traversed by one of the diameters of R (i.e., a diagonal connecting two opposite corners of R). It is easy to see that this number equals $2\sqrt{\frac{2\pi n}{\ln n}}$. This bound is tight, since *every* cell contains at least one node w.h.p. when the minimal density for connectivity is achieved. We have thus proved the following theorem:

Theorem 3: Assume n nodes with communication range $r = r(n) = \sqrt{\frac{\ln n}{n}}$ are distributed uniformly at random in $R = [0, 1]^2$. The interference diameter of the resulting communication graph G is $ID(G) = \Theta(\sqrt{\frac{n}{\log n}}) = \Theta(\sqrt{\frac{n}{\rho(G)}})$.

Proof: Due to lack of space, the proof of this theorem is reported in the full version of the paper [5].

3) Infinite density deployments: In infinite density deployments, it is assumed that, for any node u, and for any distance d within communication range and any direction β in the deployment region, there exists a node located at distance d and direction β from u. This model, which is scarcely relevant in practice (except for modeling extremely dense networks), is interesting to evaluate best-case or worst-case scenarios for wireless networks (see, for instance, [2]).

Assume a network G is deployed in a convex region R with infinite density, and let r be the nodes' communication range. A tight upper bound to the interference diameter of G is given by $\frac{diam(R)}{r}$. Let us set r such that the node neighbor degree is $\Theta(n)$, which corresponds to the highest possible neighbor density (in asymptotic terms). Since in the infinite density scenario the number of nodes within range r from a certain node u is given by πr^2 (assuming u is far enough from the border of R), in order to have a neighbor density of $\Theta(n)$ we must have $r = O(1)$, i.e. the node communication range must be independent of n. This implies that also the interference diameter of G does not depend on n, i.e. $ID(G) = \Theta(1)$, which again equals $\Theta\left(\sqrt{\frac{n}{\rho(G)}}\right)$.

C. Approximation bound

In this section, we prove that the length of the schedule computed by FDD is at most a factor $O(n^{1-\frac{2}{\psi(\alpha)+\epsilon}}(\log n)^{\frac{2}{\psi(\alpha)+\epsilon}})$ away from the optimal schedule, where $\epsilon > 0$ is an arbitrarily small positive constant and $\psi(\alpha)$ is a constant which depends on α. Similarly to [4], this result holds under the assumption that network nodes are distributed uniformly at random in a

[3] In this paper, w.h.p. means with probability converging to 1 as n goes to infinity.

square of unit area, and that radio signal propagation obeys the log-distance path model with path loss exponent $\alpha > 2$.

Theorem 4 (Approximation bound): Let G be a communication graph with given demands on the nodes. Let T_{opt} be the minimum possible value of T such that a schedule of length T is feasible for G under the physical interference model, and let T_{FDD} be the length of the schedule computed by FDD. Then, $\frac{T_{FDD}}{T_{opt}} \in O(n^{1-\frac{2}{\psi(\alpha)+\epsilon}}(\log n)^{\frac{2}{\psi(\alpha)+\epsilon}})$, for any arbitrarily small constant $\epsilon > 0$, w.h.p.

Proof: We prove that the schedule computed by FDD is the same as the one computed by the centralized GreedyPhysical algorithm of [4], for which the above approximation bound has been proved in [4]. Indeed, we consider a variation of GreedyPhysical, in which the edges to be scheduled are ordered according to decreasing order of the IDs of their heads. This edge ordering is different from the one used in GreedyPhysical but, as observed in [4], the approximation bound holds independently of the initial edge ordering.

GreedyPhysical is a simple greedy algorithm which sequentially considers edges in decreasing order, and greedly allocates the new edge e to the first slot in the current schedule such that including e in the slot does not make it infeasible. This process is iterated until the demand on e is satisfied, and then a new edge is considered. GreedyPhysical terminates when the demand on each edge is satisfied.

It is easy to see that FDD re-creates this exact greedy procedure in a fully distributed way. Initially, the node with highest ID, corresponding to the first edge e_1 in the ordering, gets control of the first slot, and allocates its edge in the first $demand(e_1)$ slots. When a certain slot is considered, new edges are tried sequentially (i.e., selected as the unique active node) in decreasing order of their head's ID. If the link associated with an active node u does not conflict with currently scheduled nodes (whose ID can only be higher than u's ID), it is included in the current slot. This implies that every edge e_i is included in the first $demand(e_i)$ slots such that e_i does not conflict with the currently scheduled edges, where currently scheduled edges e_js precede e_i in the edge ordering. I.e., the scheduling computed by FDD is the same scheduling as the one computed by the variation of GreedyPhysical described above. ■

D. Computational complexity

We now prove that FDD has polynomial time complexity. A similar proof, which is not shown due to lack of space, can be provided also for the PDD algorithm.

Theorem 5 (Time complexity): Algorithm FDD has $O(TD \cdot ID(G) \cdot n \log n)$ time complexity, where TD is the total traffic demand in the network and $ID(G)$ is the interference diameter of the communication graph G.

Proof: FDD proceeds in rounds, allocating a slot in each round. In the worst case, all links have to be scheduled sequentially, and a total of TD rounds are needed to compute the schedule. This gives the TD term in the $O()$ notation. For each round, a leader election protocol (with complexity $O(ID(G) \cdot \log n)$) is run initially to determine the controller u of the slot. Then, each link with some pending demand is tried concurrently with the link governed by u to check for feasibility. In the worst case, there are $O(n)$ such links (we are routing along a routing forest RF), and for each such link we must first run the leader election algorithm. This implies that the time complexity of each round is $O(ID(G) \cdot n \cdot \log n)$, and the theorem follows. ■

Observe that $ID(G) \in O(\sqrt{n})$ in case of square grid deployments, and $ID(G) \in O(\sqrt{\frac{n}{\log n}})$ in case of random uniform deployments. Then, computational complexity becomes $O(TD \cdot n^{3/2} \log n)$ and $O(TD \cdot n^{3/2}(\log n)^{1/2})$, respectively. Furthermore, under the assumption that the ratio between the maximum and minimum demand generated by a node is upper bounded by a constant, and observing that we are routing along a forest RF, we have that $TD \in O(n^2)$, implying time complexities of $O(n^{7/2} \log n)$ and $O(n^{7/2}(\log n)^{1/2})$, respectively. If the routing trees are balanced, there are $O(\log n)$ levels in each tree, and the aggregated traffic at each level is $O(n)$, implying $TD \in O(n \log n)$. Thus, FDD's time complexities become $O(n^{5/2}(\log n)^2)$ and $O(n^{5/2}(\log n)^{3/2})$, respectively.

Note that the scheduling algorithms are applied only to the backbone nodes in a wireless mesh network and this number is expected to be in the tens to around a hundred for most mesh network scenarios. In Section VI, we do a detailed evaluation and show that the running times are quite feasible for networks in this size range.

V. Mote-based SCREAM analysis

In Section III, we described the SCREAM primitive that is used extensively in the algorithms. This primitive is based on one hop transmission of a small number of bytes, and on the ability of neighboring nodes to detect activity (carrier sensing) on the medium as a direct result of the transmission. It is imperative that the carrier sensing mechanism employed by the radio interface be immune to failure due to collisions. In fact, to achieve a network wide OR operation, the SCREAM subroutine tends to generate a broadcast flood, resulting in many collisions. In this section we present the results from evaluation of a SCREAM implementation on Berkeley Motes.

The goal of this experiment is to show that SCREAM primitive works in presence of collisions provided an adequate number of bytes ($SMBytes$) are transmitted.

A. Experimental Setup

We used the Crossbow Mica2 motes to implement the SCREAM primitive. The code was written in nesC [6] and implemented on top of TinyOS. We define three types of nodes: an $Initiator$, a $Monitor$ and rest as $Relays$. The $Initiator$ periodically (100 ms) initiates a SCREAM by transmiting $SMBytes$. The $Relays$ continuously listens to the channel for any activity by comparing the RSSI values with a preconfigured threshold(-60dBm). $Relays$ transmit a SCREAM when they detect a SCREAM (activity). The $Monitor$ compares the moving average of the RSSI values received with the threshold (-60dBm). This ensures that $Monitor$ lags behind the $Relays$ in detecting the signal from the $Initiator$

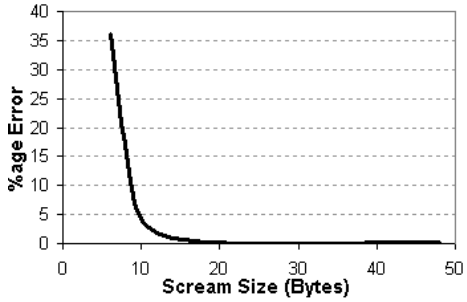

Fig. 4. Percentage Error in SCREAM detection vs SCREAM size (bytes).

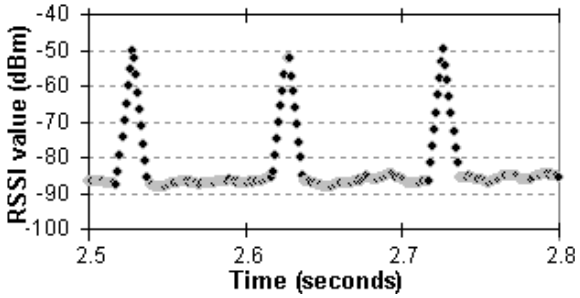

Fig. 5. Moving Average of RSSI values.

In the experiments, we used 8 motes (6 as $Relays$). To ensure collisions, we placed the $Monitor$ and $Relays$ in a clique formation and $Initiator$ was placed two hops away from the $Monitor$. Each experiment was run long enough to allow 2000 Screams. The error in scream detection is percentage of measured SCREAM intervals outside of $\pm 5\%$ of the expected interval (100 ms).

B. Results

Figure 2 shows the percentage error in detecting SCREAM vs. SCREAM size ($SMBytes$). For large SCREAM sizes (above 20 bytes), the percentage error is negligible. However, the percentage error increases rapidly for SCREAM sizes below 10. Figure 3 shows a snapshot of the moving average of RSSI signal values measured for the SCREAM size of 24 bytes. The moving average in this case was sampled after every 3 RSSI values owing to device and UART limitations. The figure shows the high quality of SCREAM detection with this SCREAM size.

VI. Simulation Results

A. Simulation Setup

We implemented the PDD, FDD and AFDD on the Georgia Tech Network Simulator [17], which is a packet level simulator with a complete 802.11 MAC model. Log-normal propgation model was used with path loss of 3. We consider two topologies, planned (grid layout) with homogeneous transmission power and unplanned (uniform node distribution) with heterogeneous transmission power. The results of PDD and FDD were compared against the centralized GreedyPhysical algorithm of [4]. We introduce bounded clock skew, where the clock of every single node is skewed from every other node within a fixed bound. The protocol implementations compensate for the clock skew among the nodes.

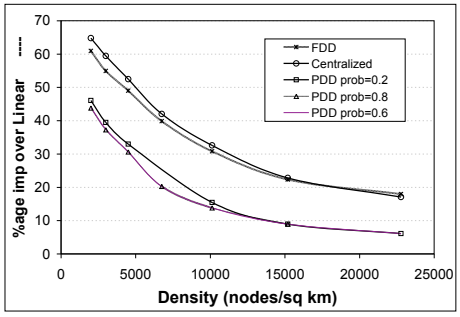

Fig. 6. Schedule Length Improvement for Grid

All simulations were done with 64 nodes, with 4 nodes acting as gateways. The demand on each node was taken from a uniform distribution in the interval $[1, 10]$. For each node, a shortest path to a nearest gateway is computed and the demand of the node is aggregated over the links on the route. We compute the schedule length for various density values, where the density was changed by varying the area and keeping the nodes fixed at 64. All results reported here are computed with 95% confidence intervals. We use a SCREAM size of 15 bytes and an interference diameter of 5.

B. Schedule Length Results

Figure 4 compares the schedule lengths of PDD and FDD to the schedule length of the centralized GreedyPhysical algorithm. We plot the percentage improvement of the computed schedules over the worst case serialized schedule. The results show that the FDD protocol closely follows the centeralized GreedyPhysical algorithm results, as expected, since the FDD protocol mimics the GreedyPhysical Algorithm in a distributed manner. For PDD, we plot the results for three different probability values. The probability here refers to the probability of a node joining the Active set in the PDD protocol. We notice that for low probability value of 0.2 PDD does marginally better than with the higher probability values. On average, the PDD's performance is about 10 percentage points worse than the performance of FDD and centralized GreedyPhysical Algorithm.

Figure5 shows the schedule lengths using the same setup as above but with the unplanned deployment. Again, FDD does as well as the GreedyPhysical algorithm but, in this case, PDD with high probability value performs about 15 percentage points worse on average than FDD and GreedyPhysical.

C. Execution Time

We study the execution time of PDD and FDD, based on the full implementation of the protocols in GTNetS. Figure 6 shows execution times of the two algorithms vs. SCREAM size and interference diameter. The plots show that the execution times are only a few seconds even for quite large values of these parameters, indicating that the algorithms incur very little overhead for schedule computations that are on the order

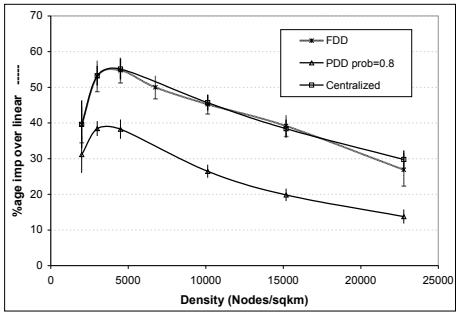

Fig. 7. Schedule Length Improvement for Uniform Random Placement

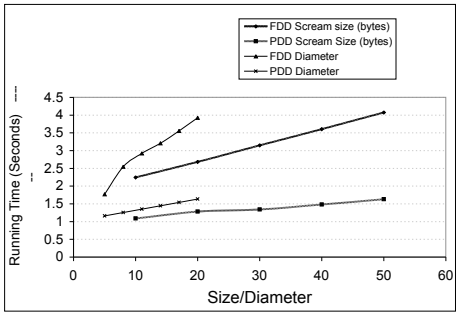

Fig. 8. Execution Time vs. SCREAM size and Interference Diameter

of once per minute. As expected, PDD's execution times are significantly lower than those of FDD.

Execution times of the algorithms are quite sensitive to clock skew, however. Figure 7 shows the running time of FDD and PDD vs the clock skew bound. Both axes are on log scale. Assuming that the schedule must be recomputed once per minute, PDD can compute the schedule with less than 5% overhead for a clock skew up to 100 microseconds. FDD is somewhat less tolerant of clock skew and should be used with a clock skew of no more than 10 microseconds for this frequency of schedule computation. Clock skews on this order are no problem for GPS-equipped devices and 100 microsecond clock skews are achievable even with distributed synchronization algorithms for typical mesh network sizes, e.g. less than 100 nodes.

VII. Conclusion

The algorithms of this paper represent the first efficient, distributed scheduling approach with a proven approximation bound for the physical interference model. While considerable work is still to be done to realize wireless mesh network implementations based on the physical interference model, we believe the approach presented in this paper constitutes a good starting point in this direction.

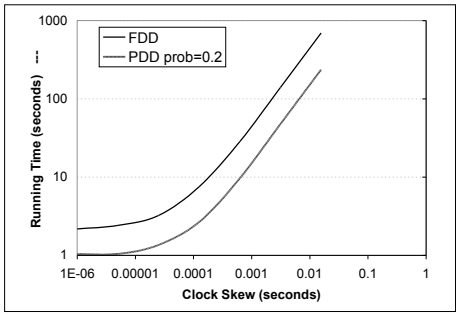

Fig. 9. Execution Time vs. Clock Skew

References

[1] M. Alicherry, R. Bathia, L. Li, "Joint Channel Assignment and Routing for Throughput Optimization in Multi-Radio Wireless Mesh Networks", *Proc. ACM Mobicom*, pp. 58–72, 2005.

[2] A. Bader, E. Ekici, "Throughput and Delay Optimization in Interference-Limited Multihop Networks", *Proc. ACM MobiHoc*, pp. 274–285, 2006.

[3] J. Bicket, D. Aguayo, S. Biswas, R. Morris, "Architecture and Evaluation of an Unplanned 802.11b Mesh Networks", *Proc. ACM Mobicom*, pp. 31–42, 2005.

[4] G. Brar, D. Blough, P. Santi, "Computationally Efficient Scheduling with the Physical Interference Model for Throughput Improvement in Wireless Mesh Networks", *Proc. ACM Mobicom (to appear)*, Sept. 2006.

[5] G. Brar, D. Blough, P. Santi, "The SCREAM Approach for Efficient Distributed Scheduling with Physical Interference in Wireless Mesh Networks", *Tech. Rep. IIT-TR12/2006*, Istituto di Informatica e Telematica del CNR, Pisa, Italy, August 2006.

[6] D. Gay, P. Levis, R. von Behren, M. Welsh, E. Brewer, D. Culler, "The nesC Language: A Holistic Approach to Networked Embedded Systems", *Proc. Programming Language Design and Implementation (PLDI)*, June 2003.

[7] J. Gronkvist and A. Hansson, "Comparison Between Graph-Based and Interference-Based STDMA Scheduling", *Proc. ACM MobiHoc*, pp. 255–258, 2001.

[8] J. Gronkvist, J. Nilsson, and D. Yuan, "Throughput of Optimal Spatial Reuse TDMA for Wireless Ad-Hoc Networks", *Proc. IEEE Vehicular Technology Conference*, pp. 2156–2160, 2004.

[9] J. Gronkvist, "Distributed Scheduling for Mobile Ad Hoc Networks - a Novel Approach," *Proc. Int'l. Symp. on Personal, Indoor, and Mobile Radio Communications*, pp. 964–968, 2004.

[10] P. Gupta and P.R. Kumar, "The Capacity of Wireless Networks," *IEEE Trans. Info. Theory*, Vol. 46, No. 2, pp. 388–404, 2000.

[11] K. Jain, J. Padhye, V. Padmanabhan, L. Qiu, "Impact of Interference on Multi-Hop Wireless Network Performance", *Proc. ACM Mobicom*, pp. 66–80, 2003.

[12] V.F. Kolchin, B.A. Sevast'yanov, V.P. Chistyakov, *Random Allocations*, V.H. Winston and Sons, Washington D.C., 1978.

[13] T. Moscibroda, R. Wattenhofer, "The Complexity of Connectivity in Wireless Networks", *Proc. IEEE Infocom 2006*.

[14] T. Moscibroda, R. Wattenhofer, A. Zollinger, "Topology Control Meets SINR: The Schedulin Complexity of Arbitrary Topologies", *Proc. ACM MobiHoc*, pp. 310–321, 2006.

[15] R. Nelson and L. Kleinrock, "Spatial-TDMA: A Collison-free Multihop Channel Access Protocol," *IEEE Trans. on Communication,* Vol. 33, pp. 934–944, Sept. 1985.

[16] B. Raman, K. Chebrolu, "Design and Evaluation of a new MAC Protocol for Long-Distance 802.11 Mesh Networks" *Proc. ACM Mobicom*, pp. 156–169, 2005.

[17] G. Riley, "The Georgia Tech Network Simulator," *ACM SIGCOMM MoMeTools Workshop,* 2003.

[18] O. Somarriba, *Multihop Packet Radio Systems in Rough Terrain,* Tech.lic. Thesis, Radio Communication Systems, Royal Institute of Technology, Stockholm, Sweden, Oct. 1995.

[19] Z. Wu and D. Raychaudhuri, "D-LSMA: Distributed Link Scheduling Multiple Access Protocol for QoS in Ad-hoc Networks," *Proc. IEEE Globecom*, pp. 1670–1675, 2004.