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

Gurashish Brar
Microsoft Corporation
Redmond, Washington, USA
Gurashish.Brar@microsoft.com

Douglas M. Blough
Georgia Institute of Technology
Atlanta, Georgia USA
doug.blough@ece.gatech.edu

Paolo Santi
IIT-CNR
Pisa, Italy
paolo.santi@iit.cnr.it

Abstract

It is known that CSMA/CA channel access schemes are not well suited 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. 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.

1 Introduction

In wireless mesh networks, wireless backbone nodes (also called *wireless routers* in the following) must convey 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 practical relevance. The two main interference models relevant to wireless mesh networks are the *protocol* and *physical* interference models [11].

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 traffic demands of wireless mesh networks. One reason is that CSMA/CA is a conservative mechanism: due to the combination of carrier sensing and collision avoidance techniques, many network nodes are silenced when a certain communication takes place. Accordingly, 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 [18, 23].

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 exponential time centralized and/or distributed protocols [7, 8, 9, 13, 21], which are clearly infeasible. Other works dealing with the physical interference model consider only a simplified scenario with unit traffic demand on each link [6, 15, 16], which is not representative of real-world

*This research was supported in part by the National Science Foundation under Awards ENG-0225417, INTL-0405157, and CNS-0721596.

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

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 for physical interference with a proven approximation factor. In fact, to our knowledge, the heuristic algorithm of [10] is the only existing distributed scheduling algorithm that accounts for physical interference.

In this paper, we take a novel approach to distributed scheduling, which is based on a global primitive that we refer to as a SCREAM. The SCREAM primitive resembles the busy tone mechanism which has been proposed to improve the performance of CSMA-CA schemes [12, 22]. However, in contrast with the busy tone mechanism, SCREAM is a multi-hop, network-wide primitive, i.e., nodes relay a detected SCREAM in order to propagate it throughout the network.

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 particular 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 terms of running time* and maintains the *proven approximation bound* of [4]. We also present a variant of our scheduling algorithm that has slightly higher schedule length, i.e. lower throughput, but runs substantially faster. For the proposed algorithms, we present detailed simulation results that evaluate their running times, schedule lengths, and throughputs, and demonstrate that both algorithms are able to outperform 802.11 in terms of throughput for scenarios representative of wireless mesh networks.

2 Network Model and Assumptions

We consider a wireless mesh network composed of n nodes (wireless routers). Links among nodes are represented by a *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. We do not assume any specific radio propagation model, nor that all 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 [17]. 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 dynamic 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 [11] introduced in [4], which we summarize for completeness. In contrast with [11], 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 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 v can detect some activity on the channel when u is transmitting, and all 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.

Traffic generated at each node in the mesh is conveyed to pre-defined gateway nodes. We assume that traffic is routed

to the gateways along reverse trees rooted at the gateways, which thus form a routing forest RF. A node that is not a gateway decides which tree to join using a simple criterion, 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 is the sum of the demands generated at the nodes belonging to the subtree rooted at u.

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 $SyncSlotBoundary()$ to wait for the next time period which is globally synchronized. In Section 6, we investigate the effect of clock skew on the protocols' performance.

3 Distributed Scheduling Protocols

In this section, we present two distributed protocols based on the centralized greedy scheduling algorithm of [4]: (1) Partially Randomized Protocol (PRP) and (2) Fully Deterministic Protocol (FDP).

The algorithms share a common approach, proceeding in rounds and scheduling one slot per round. In each round, a node with unsatisfied demand is selected as the control node for the slot through leader election. The algorithms guarantee that at least one link associated with the controller is scheduled in the slot. This guarantees that the algorithm will terminate and all demand will be satisfied. The schedule is augmented in a greedy fashion, by adding links to the slot in steps. Whenever links are added that cause previously allocated links to lose packets due to the additional interference that is generated, this is detected and the SCREAM primitive is used to notify all nodes that the last scheduled links should be removed from the slot. Full details of the protocols are given later in this section. We begin by presenting the SCREAM primitive and a leader election protocol based on it.

3.1 The SCREAM Primitive

A significant advantage of the SCREAM primitive is that it is resilient to collisions in the wireless channel, i.e., it is guaranteed to work independently of whether collisions occur or not. Given this property, leader election using SCREAM has a *deterministic* execution time, which provides a foundation for upper bounding execution time of the higher-level protocols that use it. This is significant because most wireless protocols either ignore collisions or deal with them using probabilistic means.

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 $V = var(1) \vee var(2) \vee \cdots \vee var(n)$. The SCREAM primitive runs through K SCREAM_SLOTS, where $K \geq ID(G_S)$, and returns V as a result. The primitive uses two functions:

1. $Scream()$: Transmits SM_Bytes 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:

SubRoutine $bool : SCREAM(var)$
$relay = var$
for $sslot = 1 \rightarrow K$ **do**
  $SyncSlotBoundary()$
  if ($relay = true$) **then** $Scream()$
  else $relay = Listen()$
end for
return $relay$

It is important for each node to participate in the above SCREAM subroutine even if it does not have a variable value to contribute. These nodes participate passively and simply relay the scream.

SCREAM 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, i.e., a node can successfully detect a carrier even if multiple nodes are concurrently transmitting. In Section 5, we present results from experiments performed on Mica 2 motes, which demonstrate the feasibility of the SCREAM primitive.

3.2 Leader Election

Our election algorithm assumes that every node has a unique id. Let id_bits be the number of bits in an id. The following algorithm performs leader election by selecting the node with the highest id. $LeaderElect(ID_i)$ takes as input the id of the invoking node u_i, and returns *true* if u_i is the leader, and *false* otherwise.

SubRoutine $bool : LeaderElect(ID_i)$
$votedout = false$
for $j = (id\_bits - 1) \rightarrow 0$ **do**
  $SyncSlotBoundary()$
  if ($ID_i(j) \wedge \neg votedout$) **then** $SCREAM(true)$
  else $votedout = SCREAM(false) \vee votedout$
end for
return NOT $votedout$

In the above algorithm, $ID_i(j)$ refers to the j-th bit of ID_i. The algorithm iterates through the bits in the 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. At the end of $LeaderElect$'s execution, only the node with highest ID has not been voted out, and wins the election.

Due to the use of STDMA, which requires loose clock synchronization, a SyncSlotBoundary() can be accomplished without communication among the nodes. A node

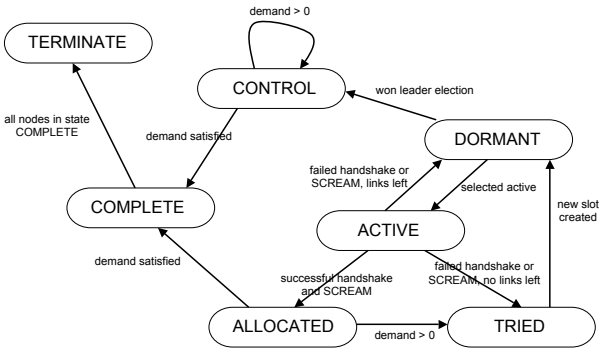

Figure 1. PRP/FDP State Transition Diagram

only needs to wait long enough to ensure that every other node has finished the last SCREAM and it can then proceed with the next round. Thus, a single SyncSlotBoundary() and SCREAM together take $O(K)$ time and the leader election protocol has $O(K \cdot \log n)$ time complexity, assuming that $id\_bits = \log_2 n$.

3.3 Partially Randomized Protocol

During PRP'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 PRP's state transition diagram. For each slot, one node is chosen as a control node through the leader election protocol from the previous subsection. Other nodes that still have demand to be satisfied and do not already have a link allocated in this slot are in the dormant state and randomly choose to become active in the slot with a certain probability. Each active, allocated, or control node tries to transmit at the same time and receive an acknowledgement back from the receiver. Any active node whose transmission fails does not allocate that link in the current slot. If any transmission fails for an allocated or control node, the failed transmitter will initiate a SCREAM indicating the failure. In this case, all active nodes hear the SCREAM and either move to state TRIED if all of their links have been tried in the slot or move back to state DORMANT if they still have untried links. If there is no transmission failure, all active nodes whose transmissions succeeded move to the allocated state. In either case (transmission failure or not), a new set of active nodes is randomly chosen and tried. This process continues until there are no active nodes, at which time a new slot is created and the process repeats with a new control node. Note that the control node for a slot always schedules a transmission in the slot, meaning there are no empty slots in the final schedule. The pseudocode in the remainder of the subsection describes PRP in further detail.

The following is the top level of the pseudocode in which a leader is chosen for each slot and termination is checked.

Input: K, upper bound on network interference diameter
Output: $Slot$: set of reserved slots
$scheduleLength$: the length of the schedule
$Slot \leftarrow \phi$
$curSlot \leftarrow 0$
$curDemand \leftarrow 0$
$state \leftarrow DORMANT$
$Released \leftarrow true$
repeat
  if $Released$ **then**
    if $state \neq COMPLETE$ **then**
      if $LeaderElect(ID_i) = true$ **then**
        $SyncSlotBoundary()$
        $state \leftarrow CONTROL$
        $SCREAM(true)$
      else
        $SyncSlotBoundary()$
        if $SCREAM(false) = false$ **then**
          $state \leftarrow TERMINATE$
        end if
      end if
    else
      $LeaderElect(0)$ *# Passive participation*
    end if
  end if
  if $state \neq TERMINATE$ **then**
    $GreedyScheduleSlot(curSlot, state)$
    $Released = CheckControlRelease(curDemand, state)$
  end if
  $curSlot \leftarrow curSlot + 1$
  if $state = TRIED$ **then**
    $state \leftarrow DORMANT$
  end if
until $state = TERMINATE$
$scheduleLength \leftarrow curSlot$

GreedyScheduleSlot() schedules a slot once its control node is elected.

SubRoutine: GreedyScheduleSlot($curslot, state$)
repeat
  if $state = DORMANT$ **then**
    if $SelectActive() = true$ **then**
      $state \leftarrow ACTIVE$
    end if
  end if
  $SyncSlotBoundary()$
  {*# Handshake time step*}

```
HSfail ← false
if state = ACTIVE|ALLOCATED|CONTROL then
  HSfail ← DoHandShake()
end if
{# Verification time step}
if state = ALLOCATED|CONTROL then
  HSfail ← SCREAM(HSfail) #Veto Power
else
  HSfail ← SCREAM(false)
end if
stillActives ← false
if state = ACTIVE then
  stillActives ← true
  if HSfail = false then
    state ← ALLOCATED
    assign link to curSlot
    if demand satisfied then
      state ← COMPLETE
    else
      state ← TRIED
    end if
  else
    if untried links left then
      state ← DORMANT
    else
      state ← TRIED
    end if
  end if
end if
SyncSlotBoundary()
stillActives ← SCREAM(stillActives)
until stillActives = false
```

The $SelectActive()$ function chooses the active nodes. In PRP, 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. Nodes that become active choose one of their untried links with unsatisfied demand at random and attempt to communicate on that link.

The $DoHandShake()$ function performs a two-way handshake on the communication link $l = (u, v)$ being scheduled, which is associated with the head node. The head node sends a data packet in the first sub-slot to the tail node. If 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 $true$. Note that $DoHandShake()$ is performed by all links that are tentatively scheduled in the current slot at the same time, so that if all handshakes succeed, this implies that the tentative schedule for the slot is valid.

Procedure $CheckControlRelease()$ determines whether the demand of the control node has been satisfied at the end of a round. To do this, a network wide OR is performed using the $SCREAM$ primitive with only the control node having a $true$ value if its demand is satisfied. If the demand of the control node has been satisfied, the node makes a transition to state $COMPLETE$, otherwise it remains in state $CONTROL$. After possible state transition, the result of the SCREAM operation is returned to the invoking procedure. In this way, all the nodes know whether control has been released and a new leader must be elected for the next slot.

3.4 Fully Deterministic Protocol

FDP follows the same procedure as PRP, except for the functioning of $SelectActive()$. Instead of randomly choosing whether to become active or not at a given step of a given slot, FDP deterministically chooses a single node to become active at each step using the leader election protocol among all dormant nodes. Each active node successively tries all of its links on which it has unsatisfied demand until it finds a link that is successful in the slot or it runs out of unsatisfied links. At that point, the node either moves to state ALLOCATED (if it was successful) or state TRIED and a new node is elected to become active. Once there are no more dormant nodes, a new slot is initiated.

4 Analysis

4.1 Impossibility of Localized Scheduling

We first give some definitions that are used in our proof that no localized distributed algorithm can be used to compute a feasible schedule under the physical interference model.

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

Definition 4 (Link k-neighborhood). *The k-hop neighborhood of any link l is the set of links with hop distance from l at most k.*

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 is an arbitrary constant ($k \in O(1)$).*

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

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 this theorem can be extended to a weaker notion of locality, in which nodes can communicate up to hop-distance $f(n)$, with $f(n) \in o(n)$. Note also the physical distance between links l and l' in the proof sketch can be quite small (an endpoint of l' can be as close as $\Delta_{\max} + \epsilon$ from an endpoint of l, where $\Delta_{\max}$ is the maximum transmission range). Hence, the interference ignored by localized algorithms can be quite large in practice.

4.2 Interference Diameter Analysis

The SCREAM primitive is a fundamental building block of our scheduling algorithms. For SCREAM to work properly, an upper bound on the network's interference diameter must be known. Recall that SCREAM is invoked with a parameter K, and we must have $K \geq ID(G_S)$. This leads to the problem of estimating (an upper bound to) the interference diameter of a certain sensitivity graph.

In the following, we 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 these assumptions, 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 is the interference diameter. Since we are interested in an *upper bound* on the interference diameter, we can consider the minimum possible meaningful value of r_{CS} with respect to r_c, i.e. $r_{CS} = r_c$, and we denote this common range by r. Note that, under this condition, 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.

We now introduce the *neighbor density*, which will be used to classify the different scenarios we consider.

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 two different scenarios, with increasing neighbor density: square grid deployments ($\rho(G) = \Theta(1)$) and random uniform deployments ($\rho(G) = \Theta(\log n)$). The analysis of these scenarios seems 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 is quite interesting, because it indicates that a high network density, which is often considered detrimental in many respects (e.g., capacity limitation), helps to reduce the interference diameter of the network, and hence speeds up scheduling computation.

4.2.1 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.*

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*

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

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.

Figure 2 shows a square grid augmentation of a line segment and the corresponding upper and lower paths.

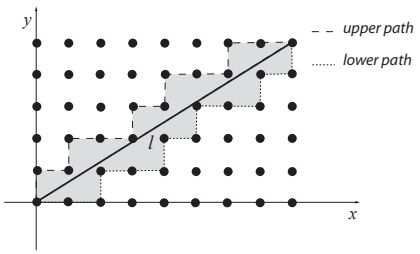

Figure 2. Square grid augmentation of line segment l

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.

Figure 3 shows an example of square grid diameter. We are now ready to prove the main result of this section.

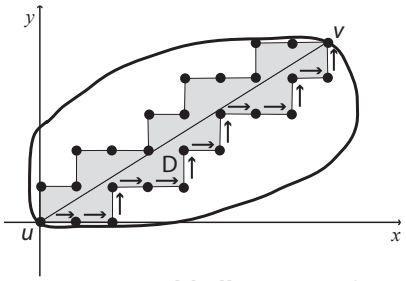

Figure 3. Square grid diameter of a square grid convex region

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

Note that the bound stated in Theorem 2 is tight. If R is a square perfectly aligned with the lattice, then $diam(R) = r\sqrt{2n}$, 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 $\Theta(\sqrt{n}) = \Theta\left(\sqrt{\frac{n}{\rho(G)}}\right)$.

4.2.2 Random Uniform Deployments

Analysis of the random uniform 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 [14]), it is known that by setting r and C as above every cell contains at least

[3] In this paper, w.h.p. means probability $\to 1$ as $n \to \infty$.

one node w.h.p. Hence, an upper bound on the interference diameter of the network is given by 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 $r(n) = \sqrt{\frac{\ln n}{n}}$ are distributed uniformly at random in $R = [0,1]^2$. The interference diameter of the resulting communication graph G is $\Theta(\sqrt{\frac{n}{\log n}}) = \Theta(\sqrt{\frac{n}{\rho(G)}})$.*

4.3 Approximation Bound

In this section, we prove that the length of the schedule computed by FDP 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 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_{FDP} be the length of the schedule computed by FDP. Then, $\frac{T_{FDP}}{T_{opt}} \in O(n^{1-\frac{2}{\psi(\alpha)+\epsilon}}(\log n)^{\frac{2}{\psi(\alpha)+\epsilon}})$, for any arbitrarily small constant $\epsilon > 0$, w.h.p.*

Proof. We prove that the schedule computed by FDP 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 FDP 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 FDP is the same scheduling as the one computed by the variation of GreedyPhysical described above. □

The approximation bounds obtained for some values of α are reported in Table 1.

α	bound
2.1	$n^{0.714}(\ln n)^{0.286}$
2.5	$n^{0.460}(\ln n)^{0.540}$
3	$n^{0.333}(\ln n)^{0.666}$
4	$n^{0.219}(\ln n)^{0.780}$
5	$n^{0.164}(\ln n)^{0.836}$
6	$n^{0.131}(\ln n)^{0.868}$

Table 1. Approximation bounds for various α

4.4 Computational Complexity

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

Theorem 5 (Time complexity). *Algorithm FDP 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. FDP 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 big-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, FDP'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 6, we do a detailed evaluation and show that the running times are quite feasible for networks in this size range.

5 Mote-based SCREAM Analysis

In Section 3, we described the SCREAM primitive that is used extensively in the algorithms. This primitive is based on transmission of a small number of bytes (SCREAM), 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 the SCREAM primitive works in presence of collisions provided an adequate number of bytes (SM_Bytes) are transmitted.

5.1 Experimental Setup

We used the Crossbow Mica2 motes to implement the SCREAM primitive. The code was written in nesC [5] 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 transmitting SM_Bytes. 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 $Initiator$.

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 the percentage of measured SCREAM intervals outside of $\pm 5\%$ of the expected interval (100 ms).

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

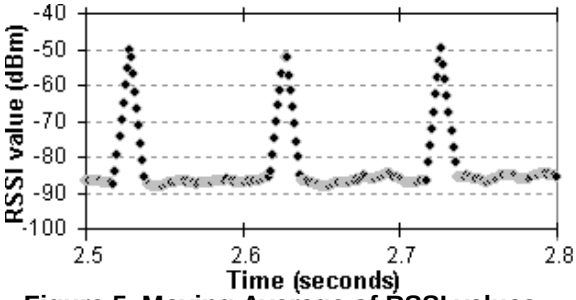

Figure 5. Moving Average of RSSI values.

5.2 Results

Figure 4 shows the percentage error in detecting SCREAM vs. SCREAM size (SM_Bytes). For large SCREAM sizes (above 20 bytes), the percentage error is negligible. However, the percentage error increases rapidly for SCREAM sizes below 10. Figure 5 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: about $30ms$ after a SCREAM is sent by the $Initiator$ (at $2.5s$, $2.6s$, etc.), the $Monitor$ correctly detects the SCREAM.

6 Simulation Results

6.1 Simulation Setup

We implemented the PRP and FDP algorithms on the Georgia Tech Network Simulator (GTNetS) [20], which is a packet level simulator with a complete 802.11 MAC model. The log-normal propagation model was used with a path loss exponent of 3 and $\sigma = 6dB$. We consider two topologies, planned (grid layout) with homogeneous transmission power and unplanned (uniform distribution) with heterogeneous transmission power. The results of PRP and FDP were compared against the centralized GreedyPhysical algorithm of [4]. We also introduced bounded clock skew, where the clocks of all nodes differ from each other by a bounded amount. The protocol implementations compensate for the clock skew among the nodes.

All simulations were done with 64 nodes, with 4 nodes

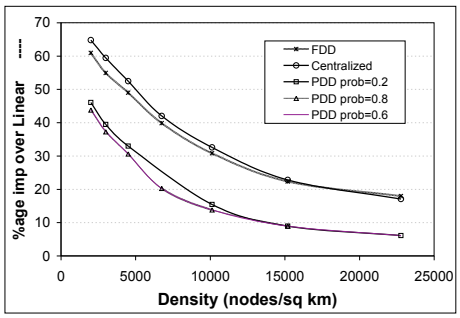

Figure 6. Schedule Length Improvement, Grid

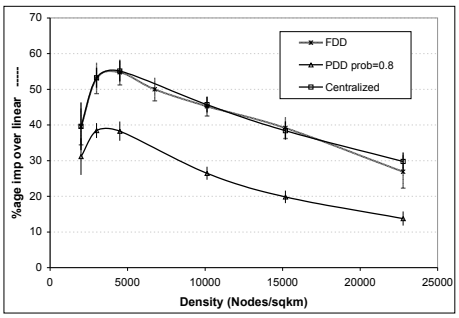

Figure 7. Schedule Length Improvement, Uniform Random Placement

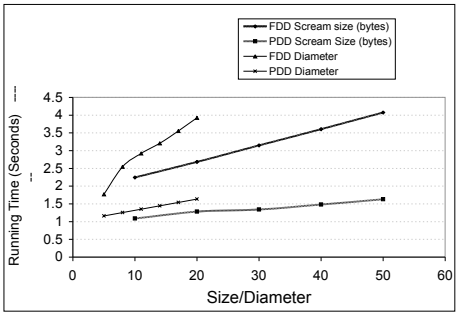

Figure 8. Execution Time vs. SCREAM size and Interference Diameter

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 densities, where the density was changed by varying the area and keeping the number of nodes fixed. All results are computed with 95% confidence intervals. We use a SCREAM size of 15 bytes and an interference diameter of 5.

6.2 Schedule Length Results

Figure 6 compares the schedule lengths of PRP and FDP 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 FDP protocol closely follows the centeralized GreedyPhysical algorithm results, as expected, since the FDP algorithm mimics the GreedyPhysical Algorithm in a distributed manner. In some cases, FDP provides marginal improvements with respect to GreedyPhysical. This is due to the fact that the link order used to build the schedule is different (ID-based in FDP, according to an interference metric in GreedyPhysical). For PRP, 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 PRP protocol. We notice that for a low probability value of 0.2, PRP does marginally better than with the higher probability values. On average, the PRP's performance is about 10 percentage points worse than the centralized GreedyPhysical Algorithm.

Figure 7 shows the schedule lengths using the same setup but with unplanned deployments. Again, FDP does as well as the GreedyPhysical algorithm but, in this case, PRP with a high probability value performs about 15 percentage points worse on average than GreedyPhysical.

6.3 Execution Time

Figure 8 shows execution times of PRP and FDP vs. SCREAM size and interference diameter, based on a complete implementation of the protocols in GTNetS. 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 of once per minute. As expected, PRP's execution times are significantly lower than FDP's.

Execution times of the algorithms are quite sensitive to clock skew, however. Figure 9 shows the running time of FDP and PRP vs the clock skew bound. Both axes are on log scale. Assuming that the schedule must be recomputed once per minute, PRP can compute the schedule with less than 5% overhead for a clock skew up to 100 microseconds. FDP 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 should be achievable even with distributed synchronization algorithms for

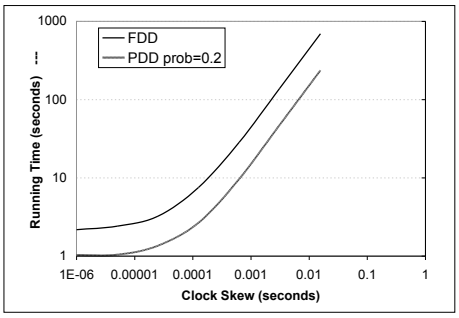

Figure 9. Execution Time vs. Clock Skew

typical mesh network sizes, e.g. less than 100 nodes.

7 Discussion and Conclusions

Our simulations showed that PRP and FDP can be executed in a few seconds in typical mesh network scenarios. Scheduling overhead depends on the frequency of schedule recomputation, which depends on factors such as changes in traffic demands, network topology, environmental conditions, etc. Our results indicate that PRP and FDP generate a reasonable overhead in scenarios in which the schedule need not be recomputed too often (say, every few minutes). In cases where more frequent schedule computation is needed, different solutions should be designed, e.g. protocols that adapt the existing schedule in response to network variations, rather than recomputing it from scratch.

Concerning comparison with CSMA/CA, our results show that the length of the schedule computed by our protocols is comparable to Algorithm GreedyPhysical [4]. Hence, our algorithms should provide improvements with respect to 802.11 comparable to those reported for GreedyPhysical (up to three-fold improvements).

Our results also show that a relatively tight node synchronization is needed to keep the execution times of our protocols low. External sources, such as GPS or the NIST atomic clock disseminated by AM radio, can be used to achieve the desired degree of synchronization with virtually no overhead. For situations where external signals are not available for synchronization, the overheads of synchronization must be evaluated and included in the analyses.

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 this approach, we believe this paper constitutes a promising step in this direction. We also believe that the SCREAM primitive, described herein, may be applicable to other distributed problems for wireless networks. What other problems can be efficiently solved by this network-wide OR operation is currently an open question.

References

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

[2] A. Bader, E. Ekici, "Throughput and Delay Optimization in Interference-Limited Multihop Networks", *Proc. 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. 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. Mobicom*, pp. 2–13, 2006.

[5] D. Gay, et al., "The nesC Language: A Holistic Approach to Networked Embedded Systems", *Proc. PLDI*, 2003.

[6] O. Goussevskaia, Y. Oswald, R. Wattenhofer, "Complexity in Geometric SINR", *Proc. MobiHoc*, pp. 100-109, 2007.

[7] J. Gronkvist and A. Hansson, "Comparison Between Graph-Based and Interference-Based STDMA Scheduling", *Proc. 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. VTC*, pp. 2156–2160, 2004.

[9] J. Gronkvist, "Traffic Controlled Spatial Reuse TDMA in Multi-hop Radio Networks", *Proc. Int'l. Symp. on Personal, Indoor, and Mobile Radio Comm.*, pp. 1203–1207, 1998.

[10] 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.

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

[12] Z. Haas, J. Deng, "Dual Busy Tone Multiple Access DBTMA: A Multiple Access Control Scheme for Ad Hoc Networks", *IEEE Trans. on Communications*, Vol. 50, n. 6, pp. 975–985, 2002.

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

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

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

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

[17] 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.

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

[19] I. Rhee, et al., "DRAND: Distributed Randomized TDMA Scheduling for Wireless Ad Hoc Networks, *Proc. MobiHoc*, pp. 190–201, 2006.

[20] G. Riley, "The Georgia Tech Network Simulator," *Proc. MoMeTools Workshop*, 2003.

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

[22] F.A. Tobagi, L. Kleinrock, "Packet-Switching in Radio Channels: Part II - The Hidden Terminal Problem in Carrier Sense Multiple Access and the Busy Tone Solution", *IEEE Trans. on Communications*, Vol. 23, pp. 1417–1433, 1975.

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