A Decentralized Scheduling Algorithm for Time Synchronized Channel Hopping

Andrew Tinka[1], Thomas Watteyne[2,3], Kris Pister[2], and Alexandre M. Bayen[4]

[1]Electrical Engineering and Computer Sciences, University of California, Berkeley, CA
[2]Berkeley Sensor & Actuator Center, University of California, Berkeley, CA [3]Currently with Dust Networks, Inc.
[4]Systems Engineering, Department of Civil and Environmental Engineering, University of California, Berkeley, CA
tinka@berkeley.edu, twatteyne@dustnetworks.com, pister@eecs.berkeley.edu, bayen@berkeley.edu

Abstract. Time Synchronized Channel Hopping (TSCH) is an existing Medium Access Control scheme which enables robust communication through channel hopping and high data rates through synchronization. It is based on a time-slotted architecture, and its correct functioning depends on a schedule which is typically computed by a central node. This paper presents, to our knowledge, the first scheduling algorithm for TSCH networks which both is distributed and which copes with mobile nodes.
Two variations on scheduling algorithms are presented. Aloha-based scheduling allocates one channel for broadcasting advertisements for new neighbors. Reservation-based scheduling augments Aloha-based scheduling with a dedicated timeslot for targeted advertisements based on gossip information. A mobile ad-hoc motorized sensor network with frequent connectivity changes is studied, and the performance of the two proposed algorithms is assessed. This performance analysis uses both simulation results and the results of a field deployment of floating wireless sensors in an estuarial canal environment. Reservation-based scheduling performs significantly better than Aloha-based scheduling, suggesting that the improved network reactivity is worth the increased algorithmic complexity and resource consumption.

Keywords: time-synchronized channel hopping, mobile ad-hoc networks, decentralized scheduling, simulation.

1 Introduction

The *Floating Sensor Network* (FSN) project built by UC Berkeley [1] includes autonomous, motorized floating sensor packages for deployments in rivers and estuaries (see Figure 1). The floating sensors (or "drifters", in the terminology of the hydrodynamic community) are untethered; once deployed in the river, they are carried by the current, and can modify their trajectory using limited actuation (differential drive propellers) to control their positioning. The Berkeley FSN drifters carry two communication systems: a GSM module for transmissions to a central server, and a low-power, low-range IEEE802.15.4-2006 [2] radio for communication between nodes.

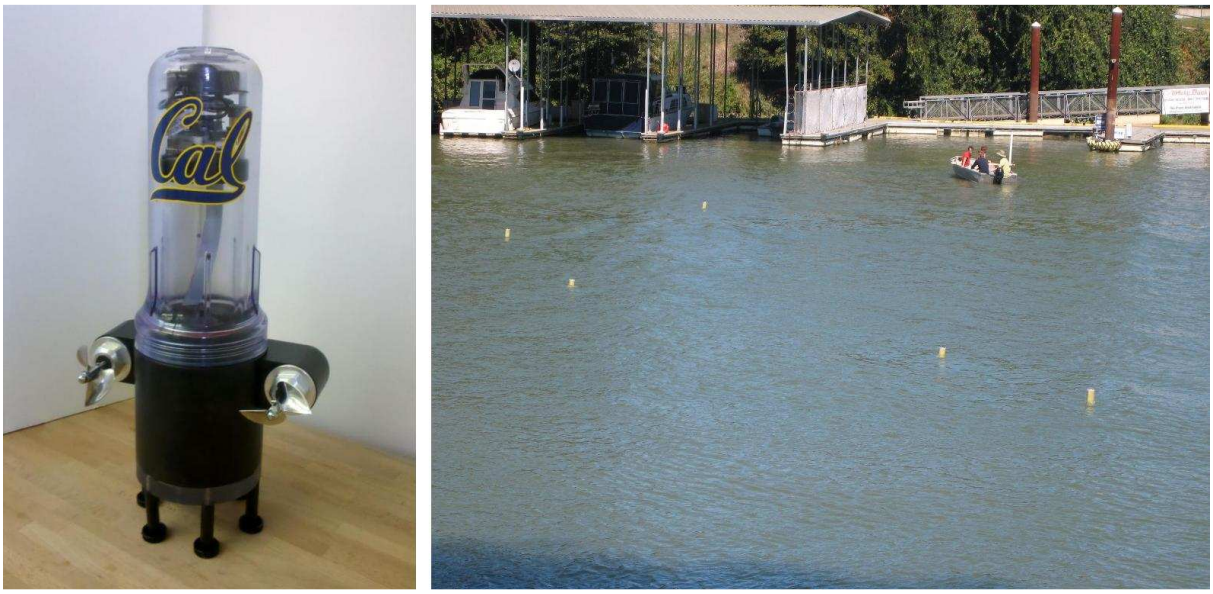

Figure 1: Prototype of a motorized drifter (*left*). Five passive drifters in a river (*right*).

The GSM communication channel is both expensive (both monetarily and in terms of energy consumption) and unreliable (due to variable GSM coverage, particularly on the water). One strategy for delivering data from individual nodes to a remote server is to have one or several nodes with good GSM connection act as *ad hoc* sink nodes. Nodes connected by IEEE802.15.4-2006 links that do not have their own GSM connections available can send their data to one of the sinks, which retransmits it via GSM to the server. Since it is not known a priori which nodes have GSM connectivity, the design objective for the IEEE802.15.4-2006 network must be to maximize point-to-point connectivity.

We define the *physical connectivity graph* to be the ensemble of wireless links "good enough" to be used for communication at a given instant in time. We define the *logical connectivity graph* to be the set of links scheduled to be used at the same instant.

Due to the water currents, the mobility of the nodes means that the physical connectivity between nodes changes significantly over time. Global connectivity is not guaranteed. Therefore, centralized schemes for determining a communication schedule are poor fits for the problem. Our goal is to develop an algorithm which schedules intermittent bi-directional links between neighboring nodes as these links become available. We assess candidate schemes by evaluating how close the logical connectivity gets to the physical connectivity; that is, how many of the possible links are actually scheduled by the protocol.

Time Synchronized Channel Hopping (TSCH) is a *Medium Access Control* (MAC) scheme which enables robust communication through channel hopping and high data rates through synchronization. It is based on a time-slotted architecture, where a schedule indicates to the nodes on which timeslot and on which channel to transmit/receive data to/from which neighbor. TSCH is being standardized by the IEEE802.15.4e Working Group [3], and expected to be included in the next revision of the IEEE802.15.4 standard. In this paper, the terms "timeslot" and "slot" are used interchangeably.

TSCH only defines the mechanism, and makes no recommendation on how the schedule should be built. Typically, nodes report their communication needs (expressed in terms on throughput, reliability and latency) to a central scheduler, which computes a schedule and injects this into the network. This technique has proven perfectly adequate for static networks such as industrial control *Wireless Sensor Networks* (WSNs). A distributed solution seems more appropriate for mobile networks. In those types of networks, each topological change would have to be reported to the central scheduler, which would have to re-compute a schedule and inform the nodes about the change. This is sometimes infeasible since this central scheduler may be disconnected from parts of the network.

This article presents two related distributed scheduling algorithms to be used on top of a TSCH MAC protocol. These algorithms are designed for the scheduling needs of IEEE802.15.4-2006 radios in applications with high mobility. In particular, these algorithms are purely decentralized. The first algorithm, "Aloha-based scheduling", uses advertisements on a specific channel to discover neighbors and initiate schedule negotiations. The second algorithm, "Reservation-based scheduling", augments the Aloha-based algorithm with a gossip mechanism that distributes the scheduling information to more nodes, speeding up the negotiation of a common schedule. In order to assess the performance of the scheduling algorithms, we present two metrics: "relative connectivity", a static metric which evaluates how many feasible neighbors from the physical connectivity graph have been added to the schedule; and "link duration", a dynamic metric that evaluates the lifetime of a link in the logical connectivity graph compared to its lifetime in the physical connectivity graph. We have evaluated the two algorithms in both a simulated environment and with a field experiment. Our field experiment features an interleaved implementation of the two algorithms, which allows us to compare their performance directly, without having to replicate the physical connectivity in separate experiments. By comparing the performance of the algorithms under different network density conditions, we can infer the importance of the different features of the two approaches, which gives insight into the design of future protocols.

The remainder of this article is organized as follows. Section 2 provides a comprehensive overview of MAC protocol approaches and standardization activities, and highlights the need for a distributed scheduling algorithm for TSCH. Section 3 then details the two scheduling algorithms proposed in this article, called "Aloha-based scheduling" and "reservation-based scheduling". A simulation environment is described in Section 4, and an implementation and field experiment described in Section 5. Performance of the two algorithms in simulated and real environments is explored in Section 6. Finally, Section 7 concludes this article and presents directions for future work.

2 Time Synchronized Channel Hopping

There are two main approaches for regulating access to a shared wireless medium: contention-based and reservation-based approaches. Any derived MAC protocol is based on one of those two approaches, or a combination thereof.

Contention-based protocols are fairly simple, mainly because neither global synchronization nor topological knowledge is required. In a contention-based approach, nodes compete for the use of the wireless medium and only the winner of this competition is allowed to access the channel and transmit. Aloha and *Carrier Sense Multiple Access* (CSMA) are canonical

representative schemes of contention-based approaches. They do not rely on a central entity and are robust to node mobility, which makes it intuitively a good candidate for dynamic mobile networks.

Preamble-sampling is a low-power version of contention-based medium access, widely popular in WSNs. All nodes in the network periodically sample the channel for a short amount of time (at most a few milliseconds) to check whether a transmission is ongoing. Nodes do not need to be synchronized, but all use the same check interval. To ensure all neighbors are listening, a sender pre-pends a preamble which is at least as long as the check interval. Upon hearing the preamble, nodes keep listening for the data that follows. The optimal check interval, which minimizes the total energy expenditure, is a function of the average network degree and the load of the network. A check interval of 100 ms is typical. Numerous efforts have proposed ways to optimize the sampling [4], reducing the preamble length by packetization [5] or by synchronizing the nodes [6].

Despite their success, contention-based protocols suffer from degraded performance in terms of throughput when the traffic load increases. In addition, the uncoordinated nature of their resource allocation prevents them from achieving the same efficiency as ideal reservation-based protocols. Finally, frequency agility is hard to achieve by such protocols, as nodes are not synchronized.

Reservation-based protocols require the knowledge of the network topology to establish a schedule that allows each node to access the channel and communicate with other nodes. The schedule may have various goals such as ensuring fairness among nodes, or reducing collisions by preventing nodes from transmitting at the same time. *Time Division Multiple Access* (TDMA) is a representative example for such a reservation-based approach.

In TDMA, time is divided into slots which are grouped into superframes which repeat over time. A schedule is used to indicate to each node when it has to transmit or receive, to/from which neighbor. Provided the schedule is correctly built, transmissions do not suffer from collisions, which guarantees finite and predictable scheduling delays and also increases the overall throughput in highly loaded networks.

Many approaches to MAC for wireless sensor networks combine some elements of contention-based protocols, especially for neighbor discovery or other startup tasks, with reservation-based scheduling for improved performance once neighbors are known. For example, in the PEDAMACS protocol [7], nodes transmit randomly using CSMA in order to discover the network topology and collect the topology information at a central node, which then computes all schedule information for all nodes in the network and distributes it. After this centralized schedule has been distributed, communication is governed by the schedule. In the TRAMA protocol [8], each TDMA superframe contains “random-access” frames, where neighbors are discovered and local topological information is shared, and “scheduled-access” frames, where nodes determine which of their two-hop neighbors has priority using a hash of frame number and node ID. In the Dozer protocol [9], new nodes use CSMA-like arbitration to respond to the beacon packets transmitted by nodes that have already joined the network; authority for setting the schedule is based on the tree hierarchy that emerges as “child” nodes associate with the older “parent” nodes. The SMACS protocol [10] uses a contention-based exchange of “invitation” and “response” packets to establish links between neighbors and to negotiate a transmit/receive schedule for that link for future communications. These four examples show the variety of approaches to scheduling that have been explored, from centralized (PEDAMACS) to purely decentralized (SMACS and TRAMA). The two algorithms presented in this article belong to the family of purely decentralized scheduling algorithms, and are designed specifically for the scheduling requirements of TSCH networks, and in particular the challenges of scheduling on a mobile network with connectivity that changes frequently.

The reliability of a wireless link is mainly challenged by external interference and multi-path fading. Previous works [11,12] show how channel hopping combats both of these, respectively. If a transmission fails, the sender retransmits the packet on a different frequency channel. Because this frequency change causes the wireless environment to be different, the retransmission has a higher probability of being successful than if it were retransmitted on the same channel.

Channel hopping was first applied to WSNs in a proprietary protocol called *Time Synchronized Mesh Protocol* (TSMP) [13]. In TSMP, nodes in the network are synchronized on a slotted time base. An individual timeslot is long enough for a sender to send a data frame, and for a receiver to acknowledge correct reception (a timeslot of 10 ms is common). *L* consecutive timeslots form a superframe, which repeats over time. A schedule of length *L* timeslots indicates, for each timeslot, whether the node is supposed to transmit or receive, to/from which neighbor and on which channel. TSMP runs on IEEE802.15.4-2006 [2] compliant radios, which offer 16 frequency channels in the 2.4GHz ISM band. A central scheduler is used to compute a schedule, which is then injected and used in the network.

TSMP makes a subtle difference between channel and frequency. The former is used in the schedule: node *A* schedules a link to node *B* on a given timeslot, and a given channel. This means that every superframe, node *A* will have the opportunity to use that link. The latter is the frequency nodes *A* and *B* communicate on. Nodes use the Absolute Slot Number (ASN) to keep track of which timeslot they are in. It is an ever-increasing number which is incremented at each timeslot, and which is shared by all nodes in the network as part of the synchronization procedure. TSMP uses the following function to obtain the frequency used for transmission from the channel in the schedule and the ASN. ‘%’ is the modulo operator; 16 indicates that there are 16 available channels.

$$\text{frequency} = (\text{channel}+\text{ASN})\%16$$

As a consequence, even when a link always appears at the same channel in the schedule, the operation described above ensures that communication happens in a channel hopping manner, thereby increasing the reliability of the link.

TSMP, which combines time synchronization and frequency agility, has been shown to achieve end-to-end reliabilities larger than 99.999% [14]. Its core idea has been standardized for industrial applications in WirelessHART [11][12] [17] and ISA100.11a [14]. In 2009, it has been introduced in the draft standard IEEE802.15.4e under the name Time Synchronized Channel Hopping (TSCH). This draft standard will replace the current IEEE802.15.4-2006 standard in its next revision.

All of the above standards rely on a central controller to compute a schedule for the network to use. The goal of this paper is to propose a distributed alternative, targeted at mobile nodes.

3 Distributed Scheduling Algorithms

3.1 Goal and Metrics

The goal of the proposed schedule is full connectivity, which is achieved when each node in the network has established a bidirectional link to each of its physical neighbors. A bidirectional link is established between nodes *A* and *B* when, in the superframe, there is at least one slot scheduled from *A* to *B*, and one from *B* to *A*. The unreliability of the wireless link and the movement of the nodes are challenges the scheduling algorithm needs to cope with.

If a link is present in the physical graph, it is *feasible*; if a link is present in the physical but not in the logical graph, it is said to be *unscheduled*; a link which still appears in the logical graph after it has disappeared from the physical graph is called *stale*. We use the ratio between the scheduled and feasible links as a metric for the static goodness of the scheduling algorithm.

Node mobility causes links to come and go. A link therefore has a finite lifetime, or *link duration*. To take advantage of a link, the scheduling algorithm needs to establish a logical link as soon as the physical link appears, and unscheduled it as soon as it disappears from the physical graph. We quantify the dynamic goodness of the scheduling algorithm by comparing the link duration between the physical and logical graphs.

Results presented in Section 6 are normalized against the optimal case, i.e. the physical connectivity graph. The variables to be used in this article are listed in Table 1.

Table 1. Variables used in this article.

Variable	Description
c	a channel
i, j, k, n	slot numbers
L	number of slots in a superframe
$S = \{S_0, S_1, \ldots S_{L-1}\}$	state for each slot
$C = \{C_0, C_1, \ldots C_{L-1}\}$	data channel for each slot
$N = \{N_0, N_1, \ldots N_{L-1}\}$	neighbor for each slot (can be NULL)
$P = \{(r,c)_1, (r,c)_2 \ldots\}$	list of potential neighbors (id and channel)
$D = \{(r,c)_1, (r,c)_2 \ldots\}$	list of neighbors self is connected to

To be able to communicate, two nodes need to schedule a slot to one another. They hence need to communicate to agree which slot in the superframe to use, and which channel. We present two variants of the proposed scheduling mechanism. Aloha-based scheduling (Section 3.2) is a simple, canonical algorithm, in which neighbor nodes opportunistically discover each other and establish links. Reservation-based scheduling (Section 3.3) builds upon that. By adding an explicit reservation channel, nodes discover each other faster, which is desirable in the presence of mobile nodes.

3.2 Aloha-Based Scheduling

For each of the L slots in the superframe, the algorithm maintains a state S_i, a channel C_i, and a neighbor N_i. There are five states: "*Aloha*", "*Transmit Connection Request*", "*Receive Connection Request"*, "*Transmit Data*", and "*Receive Data*". A slot is assigned a channel C_i and a neighbor N_i only in the latter four states.

The *Aloha* state is the default. When establishing a unidirectional link from *A* to *B*, the scheduling algorithm causes a slot in *A*'s schedule to transition from *Aloha* to *Transmit Connection Request*, to *Transmit Data*. Similarly, the same slot in *B*'s schedule transitions from *Aloha* to *Receive Connection Request*, to *Receive Data*. When both *A* and *B*'s slots are in the

Transmit Data and *Receive Data* state, respectively, data packets can be transmitted from *A* to *B*, once per superframe if exactly one slot is scheduled in the superframe. While communicating, *A* monitors whether its data packets are acknowledged; *B* monitors whether it receives data at all. If for 5 consecutive superframes no data is successfully transmitted, the slot returns to the *Aloha* state; the connection is then lost. To ensure these statistics are up-to-date, if a sender has no data to send on a given slot, it sends an empty "keep-alive" message

Three types of packets move through the network:

- *Advertisement* packets contain a list of *Receive Connection Request* slots of the sender node. This can be used by neighbors to know where it can be reached to establish a link. Each entry is a tuple *(s,c)* of slot and associated channel. Advertisements are broadcast and always exchanged on channel 0;
- *Connection Request* packets are sent in response to Advertisements; they are unicast on one of the slots announced in the Advertisement (at the announced channel, always different from channel 0);
- *Data* packets flow over the slot when a link is established. Their content is determined by the application, but their successful transmission is monitored by the scheduling algorithm to detect stale links. An empty data packet is used as a keep-alive. Data packets are always sent on a channel different from channel 0.

Note that there are *L* slots in a superframe, each of which can be used for an independent link. That is, an independent state machine is running for each slot. IEEE802.15.4-2006 compliant radios can transmit on 16 independent frequency channels. We dedicate channel 0 exclusively to Advertisements, and channels 1-15 exclusively to Connection Requests and Data packets.

Pseudocode listings for the two proposed algorithms are given below. The Aloha-based algorithm is described in Algorithm 1. The reservation-based algorithm has different behaviors during time slot 0 and other slots; its slot 0 behavior is given in Algorithm 2, while the behavior at other times is given in Algorithm 3.

Algorithm **1** in the Appendix presents Aloha-based scheduling in pseudo-code. It is executed by every node in the network. Upon startup (lines 1-5), all the slots are set to the *Aloha* state. The main loop (lines 6-51) iterates at each slot; different actions are taken according to the state of the slot. When in an *Aloha* slot, a node listens for Advertisements 90% of the time (on channel 0, lines 17-26), while 10% of the time it transmits an Advertisement (lines 10-15).

Sending Advertisement packets. The idea of sending an Advertisement is for a node to announce different rendezvous slot/channel tuples so that interested nodes can establish a link to it. When sending an Advertisement, a node converts all of its *Aloha* slots to the *Receive Connection Request* state, and assigns each of those a random channel other than channel 0 (lines 10-14). It puts that list in an Advertisement which it sends on channel 0. It then waits to be contacted on one of the *Receive Connection Request* slots it just announced.

Receiving Connection Request packets. When reaching a slot in the *Receive Connection Request* state (lines 29-36), a node listens to the channel it has previously randomly picked and announced in its Advertisement. If it does not receive anything (line 35), it converts that slot back to *Aloha* state. If it does receive a Connection Request (lines 31-33), it converts that slot to *Receive Data* state and records the identifier of the requester.

Receiving Advertisement packets. When receiving an Advertisement (lines 19-25), a node learns about the presence of a neighbor and is given the opportunity to contact it. If it has no slot scheduled to that neighbor, it picks one of the slots announced in the Advertisement where itself is in the *Aloha* state, i.e. it picks a rendezvous slot and channel. In case there are multiple slots which satisfy these requirements, it picks one of them randomly. It changes the state of that slot in its schedule to *Transmit Connection Request* (line 22), records the channel announced in the Advertisement (line 23), and the sender of that packet (line 24).

Transmitting Connection Request packets. When reaching a slot *i* in the *Transmit Connection Request* state (lines 38-44) a node sends a Connection Request to the neighbor recorded in N_i, at the channel recorded in C_i (line 38). If it receives an acknowledgment, it puts that slot in the *Transmit Data* state, and the logical link is established. If the Connection Request is not established (for example, due to a collision or nodes moving apart), the slot is reset to the *Aloha* state.

3.3 Reservation-Based Scheduling

The reservation-based scheduling protocol behaves like the Aloha-based protocol, with the following additions:

- Slot 0 is a permanent rendezvous slot, i.e. only Advertisements can be exchanged. Unlike other slots, Advertisements can be exchanged on any of the 16 available channels, in slot 0. Each node picks a channel on which it listens for Advertisements. Using slot 0 as a reservation slot gives nodes more opportunities to establish links to one another.
- In their Advertisements, nodes also include the list of the neighbors they are connected to, and the channel those neighbors are listening on in slot 0. This means that nodes learn about their two-hop neighbors.
- Each node maintains a list *P* of potential neighbors and the channel they are listening on in slot 0. This information is obtained by listening to Advertisements. Each node also maintains a list *D* of neighbors it is currently connected to. The scheduling algorithm tries to get as many nodes from *P* to *D*.

- A node only announces the *even* slots in its Advertisement. When the state of even slot *i* becomes *Transmit Data* (resp. *Receive Data*), the state of odd slot *i+1* is implicitly changed to *Receive Data* (resp. *Transmit Data*). This means that links are scheduled in pairs, one in each direction, establishing only bidirectional links.

Algorithm 2 and Algorithm 3 in the Appendix present reservation-based scheduling in pseudo-code. Algorithm 2 contains initialization, the main loop, and the behavior for Slot 0, while Algorithm 3 contains the behavior for all other slots.

4 Simulation Environment

We use a Python-based simulator[1] to model the mobility and RF propagation characteristics for a fleet of 25 mobile nodes. The superframe size was chosen to be 17 slots. The size must be co-prime with 16 in order to gain the benefits of the channel offset scheme; a relatively small superframe size was chosen to ensure that the scheduling constraints would be significant.

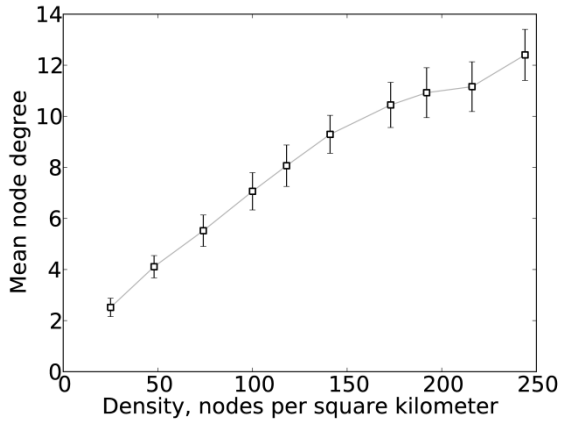

Figure 2: Mean node degree vs density of nodes in simulated environment.

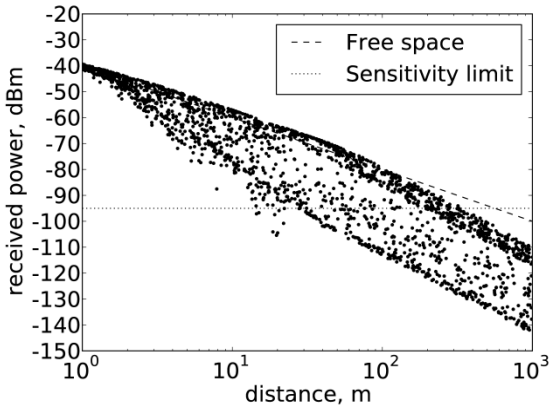

Figure 3: Received power from randomly chosen locations in simulated environment.

4.1 Propagation Model

The design objective for the RF propagation model is to create a deterministic model which captures the variance of the distance-to-received-power relationship observed in empirical studies of static spatial configurations [4], while also providing plausible spatial correlation of link strength. Approximately 30% of the simulated environment is covered with obstacles. The radiated power from a transmitting antenna is attenuated by an inverse square law as it moves through "obstacle-free" space, but is attenuated by an inverse fourth power law as it moves through "obstacle" space. This "higher power attenuation" scheme is inspired by empirical models of the effect of foliage on line-of-sight transmission [15]. The foliage model and density of obstacles is intended to represent an outdoor estuarial environment similar to that encountered by the Floating Sensor Network project. The multipath effect of the signal reflecting off the ground is modeled. The reflection is assumed to result in a 180° phase change and no attenuation.

The size of the simulated environment is modified as needed to yield desired node densities. The minimum and maximum densities are 25 and 250 nodes per square kilometer. Figure 2 shows the mean node degree (number of neighbors in the physical connectivity graph) for the different simulated densities. The bars represent the 95% confidence interval for the estimate of the mean.

4.2 Co-channel Interference Model

The interfering effect of two nodes transmitting on the same channel at the same time (usually called a "collision") is one of the main constraints on the decentralized schedule.

The IEEE802.15.4-2006 standard specifies required jamming resistance for interference coming from an adjacent channel (1 channel away) or an alternate channel (2 channels away), but does not specify a required resistance to

[1] As an on-line addition to this paper, the source code of the simulator is made freely available at http://float.berkeley.edu.

interference on the same channel. The Texas Instruments CC2420 2.4 GHz IEEE802.15.4-2006 compliant transceiver [16] has a specified co-channel rejection of -3 dB; in other words, if node *A* receives a transmission from node *B* with *p* dBm power, and a simultaneous transmission from node *C* with *(p-3)* dBm power, the transmission for *B* will be received correctly and the transmission from *C* rejected. We use this model for our simulation. Adjacent and alternate channel interference are not modeled in this simulation.

4.3 Node Mobility Model

Each node is modeled as a mobile device moving at a constant speed in the environment described above. The speed of each node is drawn from a uniform distribution over [0.8,1.2] m/s. Each node transmits at 0 dBm (1 mW) using an isotropic antenna. The height of the antenna from the ground (used for the multipath calculations) is drawn from a uniform distribution over [0.7,1.3] m for each node. Node motion is controlled by a random waypoint procedure: nodes select a cardinal direction randomly, then a distance to move in that direction. When they reach their destination, they repeat the selection process. The nodes are confined to a square area with dimensions determined by the desired node density.

Figure 3 shows the received power for randomly located transmitter and receiver nodes in the simulated environment.

5 Experimental Setup

On November 19, 2010, an implementation of the TSCH algorithms presented in Section 3 was tested using ten Berkeley FSN drifters in the Grant Line Canal near Tracy, California.

The algorithms were implemented on Texas Instruments eZ430-RF2500 platforms, which consist of an MSP430 16-bit 16-MHz micro-controller and a CC2500 radio chip. The radio chip was programmed to communicate on the frequencies of the IEEE802.15.4-2006 standard, on the 2.4GHz frequency band.

Each drifter was equipped with an eZ430-RF2500 platform. Distributed synchronization of those nodes was facilitated by a *pulse per second* (PPS) signal generated by the GPS unit on board the drifter, which provides a 1 Hz synchronization pulse with 25 ns jitter. The memory footprint of the implemented algorithms is 6 kB of flash memory and 500 B of RAM memory[2].

Both synchronization algorithms as well as a physical connectivity discovery mechanism were executed concurrently by the nodes by using a “master” superframe of 100 frames, and scheduling various operations within that framework, as shown in Table 2. The idea is to gather baseline physical connectivity data and to run both algorithms simultaneously to allow for fair comparison of their performance. Each slot is 10ms long; the superframe repeats every second.

Table 2. Superframe structure.

Slot	Function
0-2	ASN synchronization
3-58	Physical graph discovery
63-79	Aloha algorithm
82-98	Reservation algorithm

- The *physical graph discovery* phase consists of each node deterministically broadcasting on each channel in sequence (i.e. there are no collisions). When not transmitting, a node listens for its peers and record which node was heard, on what slot, and on what frequency channel. Because there are 10 drifters and 16 channels, it takes 160 physical graph discovery slots to completely survey the connectivity. With 56 slots per superframe dedicated to physical discovery, we obtain a full image of the physical connectivity every 3 superframes, i.e. every 3 seconds.
- During the *Aloha algorithm* phase, the nodes execute the scheduling algorithm presented in Section 3.2. During the *reservation algorithm* phase, the nodes execute the scheduling algorithm presented in Section 3.3. These algorithms are executed independently from each other and from the physical graph discovery phase. As in the simulation-based study, both phases are 17 slots long.

We use the results of the physical graph discovery as an estimate of the instantaneous connectivity in order to evaluate the algorithmic performance of the Aloha and reservation algorithms.

[2] The firmware source code is available at http://wsn.eecs.berkeley.edu/svn/ezwsn/.

The results of the discovery phase, and the state/neighbor/channel tables from each TSCH algorithm, were output from the eZ430-RF2500 motes and recorded using the data logging capabilities of the FSN drifter. Seven hours of data were recorded, resulting in over 250000 records of connectivity and algorithm state[3].

The Berkeley FSN drifters acted as passive floating sensors, being carried by the water current at approximately 0.3 m/s. Variations in the channel velocity profile caused their relative positions to change during the experiment. Overall, connectivity was not as highly dynamic as the simulation environment.

6 Results

6.1 Static Metric: Relative Connectivity

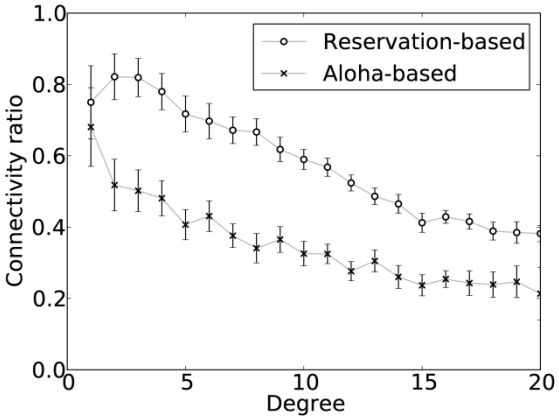

Figure 4: Mean connectivity ratio by degree for *unidirectional* links in simulated environment.

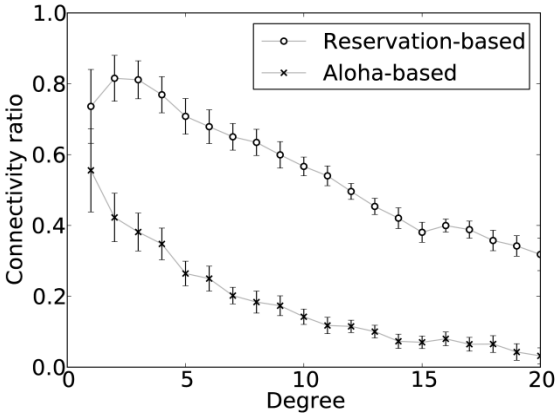

Figure 5: Mean connectivity ratio by degree for *bidirectional* links in simulated environment.

The static connectivity test in the simulated environment proceeds as follows:

1. Simulate 25 mobile nodes for 60 seconds;
2. pick a node and a superframe at random;
3. from the physical connectivity graph, count the number of unique edges incident to that node over the superframe (that is, the number of one-hop neighbors connected for at least 1 slot during the superframe); this is the degree of the node;
4. from the logical connectivity graph, find the number of outbound edges (for the unidirectional test), or find the number of neighbors with both an outbound and inbound edge (the bidirectional test);
5. the ratio of the logical connection count to the node degree is the *connectivity ratio* for the node. A connectivity ratio of 0.8 indicates that a logical link is present 80% of the cases a physical link is. A connectivity ratio of 1 is the best possible case.

To process the experimental results, the procedure was similar: a node and superframe were picked at random from the experimental logs, and the calculation of the connectivity ratio proceeded as in the simulation case.

Figure 4 and Figure 5 show the mean connectivity ratio vs. the node degree for 1250 simulations, for both unidirectional and bidirectional connections. Error bars represent the 95% confidence interval in the estimate of the mean.

In simulation, the reservation-based algorithm outperforms the Aloha-based algorithm at almost all node degrees (the confidence intervals overlap for degree 1). The reservation-based algorithm has more resources allocated to neighbor discovery, and a successful advertisement/connection request exchange results in a bidirectional connection. For both algorithms, increased node degree results in a decreased relative connectivity ratio. More local nodes means more collisions between Aloha advertisements, which reduces the effectiveness of neighbor discovery, and more cases of multiple nodes responding to an advertisement, resulting in collisions and lost connectivity. The superframes also fill up when more neighbors are present; since the superframe size is 17 slots, a node cannot have bidirectional links with more than 8

[3] The gathered traces are made freely available at http://wsn.eecs.berkeley.edu/connectivity/.

neighbors. The difference between the Aloha-based and reservation-based algorithm performance at high node degrees, however, demonstrates that both collisions and saturation must be significant.

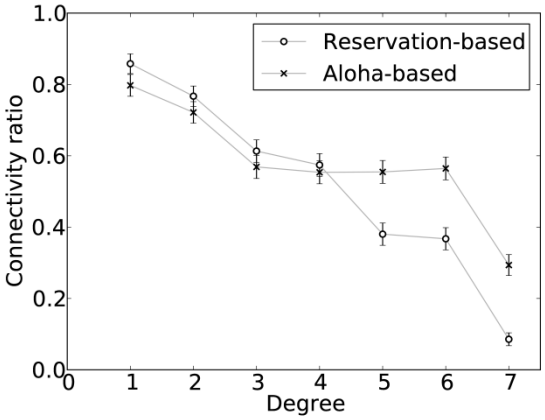

Figure 6: Mean connectivity ratio by degree for *unidirectional* links: experimental data.

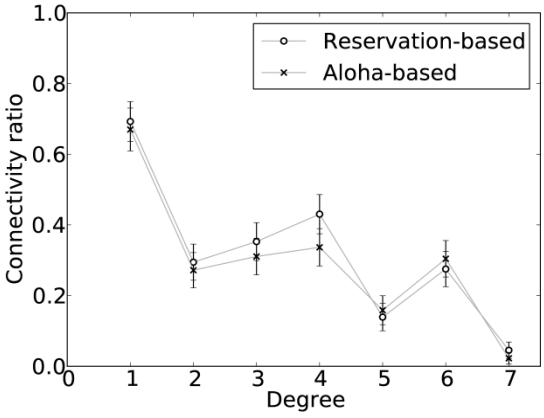

Figure 7: Mean connectivity ratio by degree for *bidirectional* links: experimental data.

In the experimental results, shown in Figure 8 and Figure 9, a different relationship between the Aloha and reservation performance is observed. For the unidirectional case, the reservation algorithm dominates at lower network degrees, as in the simulation results, but under more connected conditions, the reservation algorithm performance suffers. This phenomenon is not well explained by the analysis applied to the simulation results. In the bidirectional case, we see a change in the performance of both algorithms at different network densities, but the results are too close to judge that one algorithm is outperforming the other. In both cases, the overall trend (higher density leading to lower connectivity ratio) is consistent with the simulation results. The regime where the Aloha algorithm outperforms the reservation algorithm in the unidirectional case remains unexplained.

6.2 Dynamic Metric: Link Durations

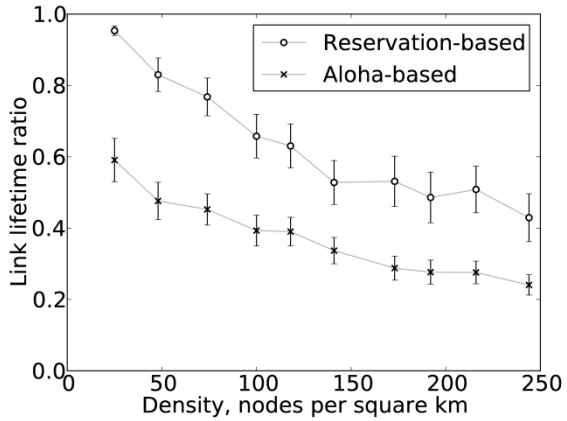

Figure 8: Mean *unidirectional* link lifetime ratio vs density in simulated environment.

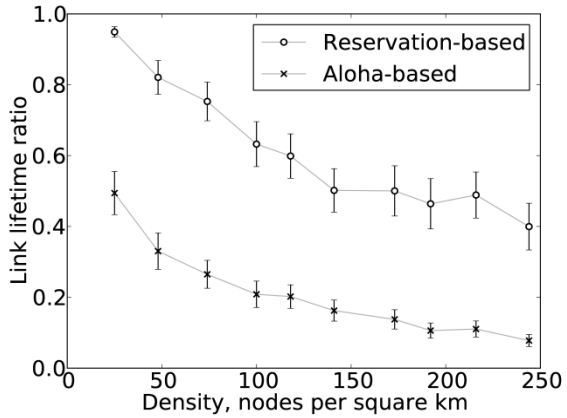

Figure 9: Mean *bidirectional* link lifetime ratio vs density in simulated environment.

The dynamic link duration test proceeds as follows:

1. Simulate 25 nodes for 60 seconds;
2. pick a node and a superframe at random;
3. pick one of the edges on the physical connectivity graph incident to that node at random; this is the link we will test;
4. count the number of consecutive superframes (forward and backward in time) in which this link is in the physical connectivity graph; this is the *physical link duration*;
5. count the number of superframes in which the link exists in the logical connectivity graph, either as a unidirectional link (the original node to the destination) or as a bidirectional link; this is the *logical link duration*;

6. the ratio of the logical link duration to the physical link duration is the *link lifetime ratio*. A link lifetime ratio of 0.8 indicates that the algorithm has scheduled a logical link 80% of the time a physical link is present. That is, if two nodes are within radio range for 10 s, they can exchange data for 8 s. A link lifetime ratio of 1 is the best possible case.

For the experimental results, the procedure is the same, with random node and superframe drawn from the experimental logs.

Figure 8 and Figure 9 show the mean link lifetime ratio vs. the density of the nodes in the simulated environment for 1250 simulations. The bars represent the 95% confidence interval for the estimate of the mean. The degree of the node is not well defined over many superframes, as the physical and logical connectivity change. While the static connectivity test could use the node degree as the independent variable, for the dynamic link duration test we use the node density as a surrogate. See Figure 2 for the relationship between the mean node degree and node density.

The dynamic performance in simulation also shows that the reservation-based algorithm outperforms the Aloha-based algorithm. Again, the Aloha-based algorithm is at a disadvantage, because its advertisement/connection request transactions build unidirectional links, not bidirectional links. At low densities, the ratio between the algorithms' performances for bidirectional links is roughly 2, which suggests that the unidirectional/bidirectional allocation difference dominates in this regime. But at higher densities, the difference between the two algorithms widens, which means other effects must be significant as well.

The saturation effects at work in the connectivity tests are also significant in the dynamic case. Links can be broken by co-channel interference, if another pair of nodes begins transmitting at the same channel/slot as an existing link. Nodes that have many active links also have less vacant slots available to form new links. Saturation effects alone cannot explain the decreased performance at high density, however, since the Aloha-based algorithm's performance decreases significantly more than the reservation-based algorithm's performance.

The reservation-based algorithm benefits when advertisements are exchanged frequently, because information about connected neighbors is carried by the advertisement packets. The reactivity of the reservation-based algorithm therefore increases at higher densities, as nodes learn about possible new neighbors more quickly. Because the advertisements in the reservation-based algorithm carry more information than the advertisements in the Aloha-based protocol, the reservation-based algorithm gains relative performance at higher node densities.

Although the dynamic link survival time test can be applied to the experimental data, the experimental was conducted at essentially a single density condition. We therefore do not have values of the dynamic test at different densities, and cannot explore the density-link time relationship as in Figure 8 and Figure 9. The results of the dynamic link survival time test are summarized in Table 3.

Table 3. Link lifetime results from experimental data.

Algorithm	Link type	Mean lifetime ratio and 95% Confidence Interval
Aloha	unidirectional	0.80 ± 0.03
Reservation	unidirectional	0.79 ± 0.03
Aloha	bidirectional	0.80 ± 0.03
Reservation	bidirectional	0.82 ± 0.03

The dynamic lifetime test shows strong performance for both algorithms, under either the unidirectional or bidirectional case, with no statistically significant difference in the mean lifetime ratios. Although the value of the mean lifetime ratio is consistent with those observed in Figure 8 and Figure 9, having both the Aloha and reservation algorithms perform (practically) identically is inconsistent with our observations in the simulated system. A major difference between the two scenarios is the distribution of link lifetimes in the physical connectivity graph. In the simulated environment, the connectivity is highly dynamic, and the short simulation time (60 seconds) places an upper bound on the link lifetime. In the experimental setup, link lifetimes ranged from as short as 1 second to several hours long. When the connectivity is not as dynamic, the increased reactivity of the reservation-based algorithm is not an advantage, and the algorithms have similar performance.

7 Conclusions and Future Work

In this article, we present what is, to our knowledge, the first scheduling algorithm for Time Synchronized Channel Hopping networks which both is distributed and which copes with mobile networks. The two variant algorithms are based on an

advertisement and rendezvous scheme: nodes continuously advertise their presence to allow neighbor nodes to discover and contact one another. An inactivity threshold mechanism is used to tear down previously established links.

The algorithms are tuned for a network of 25 drifter nodes randomly moving inside a lake or river. Simulation results show, under realistic propagation and mobility models, the efficiency of the algorithms. The simulation results in Figures 5, 6, 9, and 10 support the conclusion that the reservation-based algorithm outperforms the Aloha-based algorithm in practically all density conditions. Experimental results (Figures 7 and 8, Table 4) do not show a significant advantage to one algorithm or the other; the major difference between the experimental setup and the simulated system was the rate at which links formed and dropped, which suggests that in an environment with highly dynamic connectivity, including networks of mobile nodes, devoting additional resources to neighbor discovery and coordination pays off.

The goal of the scheduling algorithms presented in this article is to establish two-way connections between neighbor nodes, subject to the constraints of the superframe structure and the physical connectivity. We did not make assumptions about what kind of data is sent over the links; the latency, throughput, and reliability requirements are not specified. These scheduling algorithms could be adapted to meet either pre-determined or dynamic provisioning requirements. For example, a pair of nodes that need to exchange a large amount of data might wish to schedule more than one transmission slot per superframe.

Many wireless sensor network applications are highly energy-constrained. Our scheduling algorithms, as described here, require the radios to constantly either receive or transmit. This may consume too much power for some applications. An obvious modification is to reduce the duty cycle of the Aloha coordination activities; the algorithms could be implemented exactly as written, while only performing Aloha listen/transmit actions on a subset of the idle slots. The obvious tradeoff is between the energy consumed for Aloha coordination versus the reactivity of the network to changes in the physical connectivity graph. Further work will focus on characterizing the rate of change of the connectivity graph, and determining a method for balancing power consumption and reactivity. Comparing the performance of these algorithms to previously proposed algorithms like TRAMA will also yield insight into the tradeoffs made when designing algorithms for static versus mobile connectivity.

Acknowledgments. The authors thank Kevin Weekly and Carlos Oroza for their assistance during the experiment. Andrew Tinka acknowledges the support of NSERC.

References

1. *Quadratic Programming based data assimilation with passive drifting sensors for shallow water flows.* **Tinka, A., et al.** 8, August 2010, International Journal of Control, Vol. 83, pp. 1686-1700.

2. **IEEE.** *IEEE Standard for Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements. Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications* . 2006.

3. **IEEE P802.15 Working Group for Wireless Personal Area Networks (WPANs).** *IEEE P802.15.4e/D0.01 Draft Standard for Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements - Part 15.4: Wireless Medium Access Control (MAC) and Physical Lay.* 2009.

4. *Versatile Low Power Media Access for Wireless Sensor Networks.* **Polastre, J., Hill, J. and Culler, D.** ACM Press, 2004. Second ACM Conference on Embedded Networked Sensor Systems (SenSys). pp. 95-107.

5. *X-MAC: A Short Preamble MAC Protocol for Duty-Cycled Wireless Sensor Networks.* **Buettner, M., et al.** ACM Press, 2006. 4th international conference on embedded networked sensor systems (SenSys).

6. *Ultra-Low Duty Cycle MAC with Scheduled Channel Polling.* **Ye, W., Silva, F. and Heidemann, J.** ACM Press, 2006. 4th ACM Conference on Embedded Networked Sensor Systems (SenSys). pp. 321-334.

7. *PEDAMACS: power efficient and delay aware medium access protocol for sensor networks.* **Ergen, S.C. and Varaiya, P.** 7, 2006, IEEE Transactions on Mobile Computing, Vol. 5.

8. *Energy-Efficient, Collision-Free Medium Access Control for Wireless Sensor Networks.* **Rajendran, V., Obraczka, K. and Garcia-Luna-Aceves, J. J.** 1, 1995, Wireless Networks, Vol. 12.

9. *Dozer: Ultra-Low Power Data Gathering in Sensor Networks.* **Burri, N., von Rickenbach, P. and Wattenhofer, R.** 2007. Information Processing in Sensor Networks.

10. *Protocols for Self-Organization of a Wireless Sensor Network.* **Sohrabi, K., et al.** 5, 2000, IEEE Personal Communications, Vol. 7.

11. *Reliability Through Frequency Diversity: Why Channel Hopping Makes Sense.* **Watteyne, T., Mehta, A. and Pister, K.** ACM Press, 2009. 6th ACM International Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks (PE-WASUN).

12. *Mitigating Multipath Fading Through Channel Hopping in Wireless Sensor Networks.* **Watteyne, T., et al.** 2010. IEEE International Conference on Communications (ICC).

13. *TSMP: Time Synchronized Mesh Protocol.* **Doherty, L. and Pister, K.** 2008. Parallel and Distributed Computing and Systems (PDCS).

14. *Channel-Specific Wireless Sensor Network Path Data.* **Doherty, L., Lindsay, W. and Simon, J.** 16th International Conference on Computer Communications and Networks (ICCCN). pp. 89-94.

15. **HART Communication Foundation.** *HART Field Communication Protocol Specifications.* 2008. Standard Revision 7.1.

16. **Gustafsson, D.** *WirelessHART - implementation and evaluation on wireless sensors.* 2009. Master's thesis, Kungliga Tekniska hogskolan.

17. *WirelessHART: Applying wireless technology in real-time industrial process control.* **Song, J., et al.** 2008. IEEE Real-Time and Embedded Technology and Applications Symposium. pp. 377-386.

18. **ISA.** *Wireless Systems for Industrial Automation: Process Control and Related Applications.* 2009. ISA-100.11a-2009.

19. *Radio Channel Characterization for Moderate Antenna Heights in Forest Areas.* **Oestges, C., Vollacieros, B. Montenegro and Vanhoenacker-Janvier, D.** 8, October 2009, IEEE Transactions on Vehicular Technology, Vol. 58, pp. 4031-4035.

20. **Texas Instruments, Inc.** *CC2420, 2.4 GHz IEEE 802.15.4 / ZigBee-Ready RF Transceiver (Rev. B).* 2007. Data Sheet SWRS041B.

21. **University of California, Berkeley.** *OpenWSN project.* http://openwsn.berkeley.edu.

Appendix

Pseudocode listings for the two proposed algorithms are given below. The Aloha-based algorithm is described in Algorithm 1. The reservation-based algorithm has different behaviors during time slot 0 and other slots; its slot 0 behavior is given in Algorithm 2, while the behavior at other times is given in Algorithm 3.

Algorithm 1: Aloha-based Scheduling

```
for each slot i in 0..L-1
   S[i] = Aloha
   N[i] = NULL
   C[i] = NULL
end for
loop
   Go to the next slot i
   if S[i] == Aloha
      if uniform(0,1) < 0.1
         Find the set {j} of all other slots with state S[j] == Aloha
         for each of these slots
            S[j] = Receive Connection Request
            C[j] = uniform(1,15)
         end for
         Send Advertisement with slots and channels {(j,C[j])}, on channel 0
      else
         Listen for an Advertisement on channel 0
         if Advertisement {(j,C[j])} received
            Find own set of slots {k} which are of state S[k] == Aloha
            if {k} ∩ {j} is not empty
               Choose common slot n in {k} ∩ {j} randomly
               S[n]= Transmit Connection Request
               C[n] set to the receiving channel, read from Advertisement
               N[n] set to the node that sent the Advertisement
            end if
         end if
      end if
   else if S[i]==Receive Connection Request
      Listen for a Connection Request to self on channel Ci
      if valid Connection Request received
         Send Acknowledgment
         S[i] = Receive Data
         N[i] set to the ID of the requesting node
      else
         S[i] = Aloha
      end if
   else if S[i] == Transmit Connection Request
      Send Connection Request on channel C[i] to node N[i]
      if Acknowledgment received
         S[i] = Transmit Data
      else
         S[i] = Aloha
         N[i] = NULL
      end if
   else if S[i] == Receive Data or S[i] == Transmit Data
      if no successful communication for 5 consecutive superframes
         S[i] = Aloha
         N[i] = NULL
      end if
   end if
end loop
```

Algorithm 2: Reservation-based Scheduling, initialization and slot 0 behavior

```
for each slot i in 0..L-1
   S[i] = Aloha
   N[i] = NULL
end for
C[0] = uniform_integer(0,15)
P = {}
D = {}
loop
   Go to the next slot i
   if i == 0
      if P is not empty and uniform(0,1) < 0.1
         Choose (j,c) randomly from neighbors of interest in P
         Transmit Advertisement to node j on channel c
         if Acknowledgment received
            set state of all advertised slots to S[k] = Receive Connection Request
         end if
      else
         Listen for an Advertisement on channel C[0]
         if Advertisement received
            Send Acknowledgment
            If neighbor of interest, choose common slot n (similar to Algorithm 1)
            S[n] = Transmit Connection Request
            N[n] = the ID of the node that sent the Advertisement
            C[n] = the receiving channel for that slot in the Advertisement
         end if
      end if
   else
      execute Algorithm 3
   end if
end loop
```

Algorithm 3: Reservation-based Scheduling, behavior for slots other than 0

```
if S[i] == Aloha
   if uniform(0,1) < 0.1
      Find the set {j} of all other even slots with S[j] == Aloha
      for each j
         S[j] = Receive Connection Request
         C[j] = uniform_integer(1,15)
      end for
      Send Advertisement listing {(j,C[j])} and all tuples in D on channel 0
   else
      Listen for an Advertisement on channel 0
      if Advertisement {(j,C[j])} received
         Add new possible neighbors to P using the information in the Advertisement
         Find own set of slots {k} with S[k] == Aloha
         if {j} ∩ {k} is not empty
            Choose common slot n in {j} ∩ {k} randomly
            S[n] = Transmit Connection Request
            N[n] = the ID of the node that sent the Advertisement
            C[n] = the receiving channel for that slot in the Advertisement
         end if
      end if
   end if
else if S[i] == Receive Connection Request
   Listen for a Connection Request for self on channel C[i]
   if valid Connection Request received
      Send Acknowledgment
      S[i] = Receive Data; S[i+1] = Transmit Data
      N[i] and N[i+1] = the ID of the requesting node
   else
      S[i] = Aloha
   end if
 else if S[i] == Transmit Connection Request
   Send Connection Request on channel C[i] to node N[i]
   if Acknowledgment received
      S[i] = Transmit Data; S[i+1] = Receive Data
      Put (N[i], C[i]) in D
      Remove N[i] from P if present
   else
      S[i] = Aloha
   end if
else if S[i] == Receive Data or S[i] == Transmit Data
   if no successful communication for 5 consecutive superframes
      S[i] = Aloha
      move N[i] from D to P
      N[i] = NULL
   end if
end if
```