SSCH: Slotted Seeded Channel Hopping for Capacity Improvement in IEEE 802.11 Ad-Hoc Wireless Networks

Paramvir Bahl
Microsoft Research
One Microsoft Way
Redmond, WA 98052
bahl@microsoft.com

Ranveer Chandra*
Department of Computer Science
Cornell University
Ithaca, NY 14853
ranveer@cs.cornell.edu

John Dunagan
Microsoft Research
One Microsoft Way
Redmond, WA 98052
jdunagan@microsoft.com

ABSTRACT

Capacity improvement is one of the principal challenges in wireless networking. We present a link-layer protocol called Slotted Seeded Channel Hopping, or SSCH, that increases the capacity of an IEEE 802.11 network by utilizing frequency diversity. SSCH can be implemented in software over an IEEE 802.11-compliant wireless card. Each node using SSCH switches across channels in such a manner that nodes desiring to communicate overlap, while disjoint communications mostly do not overlap, and hence do not interfere with each other. To achieve this, SSCH uses a novel scheme for distributed rendezvous and synchronization. Simulation results show that SSCH significantly increases network capacity in several multi-hop and single-hop wireless networking scenarios.

Categories and Subject Descriptors

C.2.2 [**Computer-Communication Networks**]: Network Protocols

General Terms

Algorithms, Performance

Keywords

ad-hoc wireless networks, channel assignment, frequency diversity, pseudo-randomness, scheduling, medium access control

1. INTRODUCTION

The problem of supporting multiple senders and receivers in wireless networks has received significant attention in the past decade. One domain where this communication pattern naturally arises is fixed wireless multi-hop networks, such as community networks [1, 4, 6, 23]. Increasing the capacity of such wireless networks has been the focus of much recent research (e.g., [14, 20, 27]). A natural approach to increase the network capacity is to use frequency diversity [9, 31]. Commodity wireless networking hardware commonly supports a number of orthogonal channels, and distributing the communication across channels permits multiple simultaneous communication flows.

Channelization was added to the IEEE 802.11 standard to increase the capacity of infrastructure networks — neighboring access points are tuned to different channels so traffic to and from these access points does not interfere [9]. Non-infrastructure (i.e., ad-hoc) networks have thus far been unable to capture the benefits of channelization. The current practice in ad-hoc networks is for all nodes to use the same channel, irrespective of whether the nodes are within communication range of each other [4, 6].

In this paper, we propose a new protocol, Slotted Seeded Channel Hopping (SSCH), that extends the benefits of channelization to ad-hoc networks. Logically, SSCH operates at the link layer, but it can be implemented in software over an IEEE 802.11-compliant wireless Network Interface Card (NIC). The SSCH layer in a node handles three aspects of channel hopping (i) implementing the node's channel hopping schedule and scheduling packets within each channel, (ii) transmitting the channel hopping schedule to neighboring nodes, and (iii) updating the node's channel hopping schedule to adapt to changing traffic patterns. SSCH is a distributed protocol for coordinating channel switching decisions, but one that only sends a single type of message, a broadcast packet containing that node's current channel hopping schedule. Our simulation results show that SSCH yields a significant capacity improvement in ad-hoc wireless networks, including both single-hop and multi-hop scenarios.

The primary research contributions of our paper can be summarized as follows:

- We present a new protocol that increases the capacity of IEEE 802.11 ad-hoc networks by exploiting frequency diversity. This extends the benefits of channelization to ad-hoc networks. The protocol is suitable for a multi-hop environment, does not require changes to the IEEE 802.11 standard, and does not require multiple radios.

*Ranveer Chandra was supported, in part, by DARPA under AFRL grant RADC F30602-99-1-0532 and by AFOSR under MURI grant F49620-02-1-0233. Additional support was provided by Microsoft Corporation.

MobiCom'04, Sept. 26-Oct. 1, 2004, Philadelphia, Pennsylvania, USA.

- We introduce a novel technique, *optimistic synchronization*, for distributed rendezvous and synchronization. This technique allows control traffic to be distributed across all channels, and thus avoids *control channel saturation*, a bottleneck identified in prior work on exploiting frequency diversity [31].

- We introduce a second novel technique to achieve good performance for multi-hop communication flows. The *partial synchronization* technique allows a forwarding node to partially synchronize with a source node and partially synchronize with a destination node. This synchronization pattern allows the load for a single multi-hop flow to be distributed across multiple channels.

The rest of this paper is organized as follows: we provide background and motivate the problem in Section 2. In Section 3 we describe SSCH in detail, and in Section 4 we analyze its performance. We discuss design alternatives in Section 5, and we consider related work in Section 6. Finally, we discuss future work in Section 7, and conclude in Section 8. %

2. BACKGROUND AND MOTIVATION

In this paper, we will limit our discussion to the widely-deployed IEEE 802.11 Distributed Coordination Function (DCF) protocol [8]. We begin by reviewing some relevant details of this protocol. IEEE 802.11 recommends the use of a Request To Send (RTS) and Clear To Send (CTS) mechanism to control access to the medium. A sender desiring to transmit a packet must first sense the medium free for a DCF interframe space (DIFS). The sender then broadcasts an RTS packet seeking to reserve the medium. If the intended receiver hears the RTS packet, the receiver sends a CTS packet. The CTS reserves the medium in the neighborhood of the receiver, and neighbors do not attempt to send a packet for the duration of the reservation. In the event of a collision or failed RTS, the node performs an exponential backoff. For additional details, we refer the reader to [8].

The IEEE 802.11 standard divides the available frequency into orthogonal (non-overlapping) channels. IEEE 802.11b has 11 channels in the 2.4 GHz spectrum, 3 of which are orthogonal, and IEEE 802.11a has 13 orthogonal channels in the 5 GHz spectrum. Packet transmissions on these orthogonal channels do not interfere if the communicating nodes on them are reasonably separated (at least 12 inches apart for common hardware [9]).

Using only a single channel limits the capacity of a wireless network. For example, consider the scenario in Figure 1 where there are 6 nodes within communication range of each other, all the nodes are on the same channel, and 3 of the nodes have packets to send to distinct receivers. Due to interference on the single channel, only one of them, in this case node 3, can be active. In contrast, if all 3 orthogonal channels are used, all the transmissions can take place simultaneously on distinct channels. SSCH captures the additional capacity provided by these orthogonal channels.

We imposed three constraints on ourselves in the design of SSCH:

- SSCH should require only a single radio per node. Some of the previous work on exploiting frequency diversity has proposed that each node be equipped with multiple radios [9,33]. Multiple radios draw more power, and energy consumption continues to be a significant constraint in mobile networking scenarios. By requiring only a single standards-compliant NIC per node, SSCH faces fewer deployability hurdles than schemes with additional hardware requirements.

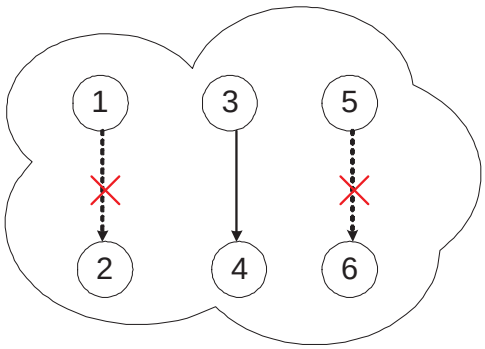

Figure 1: Only one of the three packets can be transmitted when all the nodes are on the same channel.

- SSCH should use an unmodified IEEE 802.11 protocol (including RTS/CTS) when not switching channels. Requiring standards-compliant hardware allows for easier deployment of this technology.

- SSCH should not cause a *logical partition*, which we define to occur when two nodes in communication range are unable to communicate. Because SSCH switches each NIC across frequency channels, different NICs may be on different channels most of the time. Despite this, any two nodes in communication range will overlap on a channel with moderate frequency (e.g., at least 10 ms out of every half second) and discovery is accomplished during this time. The mathematical properties of SSCH guarantee that this overlap always occurs.

SSCH exploits frequency diversity using an approach we call *optimistic synchronization*. SSCH is designed to make the common case be that nodes are aware of each other's channel hopping schedules, yet SSCH allows any node to change its channel hopping schedule at any time. If node A has traffic to send to another node B, and A knows B's hopping schedule, A will probably be able to quickly send to B by changing its own schedule. In the uncommon case that A does not know B's schedule, or A has out-of-date information about B, then the traffic incurs a latency penalty while A discovers B's new schedule. The SSCH design achieves this good common case behavior when SSCH is used with a workload where traffic patterns change (i.e., new flows are started) less often than hopping schedule updates are propagated. Because hopping schedule update propagation requires only tens of milliseconds, this is a good workload assumption for many wireless networking scenarios. Our performance evaluation in Section 4 gives absolute numbers for these qualitative claims.

SSCH is designed to work in a single-hop or multi-hop environment, and therefore SSCH must support multi-hop flows. We introduce the *partial synchronization* technique to allow one node B to follow a channel hopping schedule

that overlaps half the time with another node A, and half the time with a third node C; this is necessary for node B to efficiently forward traffic from node A to node C. Although it is trivially possible for node B to have a channel hopping schedule that is an interleaving of A and C's schedules, this leaves open how B will represent its schedule when a fourth node desires to synchronize with B. The design for channel hopping that we describe in Section 3.3 resolves this issue.

3. SSCH

SSCH switches each radio across multiple channels so that multiple flows within interfering range of each other can simultaneously occur on orthogonal channels. This results in significantly increased network capacity when the network traffic pattern consists of such flows.

SSCH is a distributed protocol, suitable for deployment in a multi-hop wireless network. It does not require synchronization or leader election. Nodes do attempt to synchronize, but lack of synchronization results in at most a mild reduction in throughput.

SSCH defines a *slot* to be the time spent on a single channel. We choose a slot duration of 10 ms to amortize the overhead of channel switching. At 54 Mbps (the maximum data rate in IEEE 802.11a), 10 ms is equivalent to 35 maximum-length packet transmissions. A longer slot duration would have further decreased the overhead of channel switching, but would have increased the delay that packets encounter during some forwarding operations. The *channel schedule* is the list of channels that the node plans to switch to in subsequent slots and the time at which it plans to make each switch. Each node maintains a list of the channel schedules for all other nodes it is aware of – this information is allowed to be out-of-date, but the common case will be that it is accurate. The good performance exhibited by SSCH (Section 4) validates this claim.

We develop the SSCH protocol by first describing its assumptions about the underlying hardware and Medium Access Control (MAC) protocol (Section 3.1). We then describe the packet transmission attempts that are made by each node within a slot, and we refer to this as the *packet schedule* (Section 3.2). We then define the policy for updating the channel schedule and for propagating the channel schedule to other nodes (Section 3.3). We then describe the mathematical properties that guided SSCH's design (Section 3.4). Finally, we discuss implementation considerations for SSCH (Section 3.5).

3.1 Hardware and MAC Assumptions

We assume that all nodes are using IEEE 802.11a – SSCH could also be used with other MACs in the IEEE 802.11 family, but this evaluation is beyond the scope of our paper. IEEE 802.11a supports 13 orthogonal channels, and we assume no co-channel interference, a reasonable assumption for physically separated nodes [9]. We expect wireless cards to be capable of switching across channels. As we discuss in more detail at the beginning of Section 4, recent work has reduced this switching delay to approximately 80 μs ([3,17]). We assume that each wireless card contains only a single half-duplex single-channel transceiver.

We require that NICs with a buffered packet wait after switching for the maximum length of a packet transmission before attempting to reserve the medium. This prevents hidden terminal problems from occurring just after switching. This requirement about the hardware is not necessary if the NIC packet buffer can be cleared whenever the channel is switched.

3.2 Packet Scheduling

SSCH maintains packets in per-neighbor FIFO queues. Using FIFO queues maintains standard higher-layer assumptions about in-order delivery. The per-neighbor FIFO queues are maintained in a priority queue ordered by perceived neighbor reachability. At the beginning of a slot, packet transmissions are attempted in a round-robin manner among all flows. If a packet transmission to a particular neighbor fails, the corresponding flow is reduced in priority until a period of time equal to one half of a slot duration has elapsed – this limits the bandwidth wasted on flows targeted at nodes that are currently on a different channel to at most two packets per slot whenever a flow to a reachable node also exists. Packets are only drawn from the flows that have not been reduced in priority unless only reduced priority flows are available.

Because nodes using SSCH will often be on different channels, broadcast packets transmitted in any one slot are likely to reach only some of the nodes within physical communication range. The SSCH layer handles this issue through repeated link-layer retransmission of broadcast packets enqueued by higher layers. Although broadcast packets sent this way may reach a different set of nodes than if all nodes had been on the same channel, we have not found this to present a difficulty to protocols employing broadcast packets — in Section 4 we show that as few as 6 transmissions allows DSR (a protocol that relies heavily on broadcasts) to function well. This behavior is not surprising because broadcast packets are known to be less reliable than unicast packets, and so protocols employing them are already robust to their occasional loss. However, the SSCH retransmission strategy may not be compatible with all uses of broadcast, such as its use to do synchronization [15]. Also, deploying SSCH in an environment with a different number of channels might require the choice of 6 transmissions to be revisited. Finally, although retransmission increases the bandwidth consumed by broadcast packets, SSCH still delivers significant capacity improvement in the traffic scenarios we studied (Section 4).

An SSCH node with a packet to send may discover that a neighbor is not present on a given channel when no CTS is received in response to a transmitted RTS. However, the node may very well be present on another channel, in which case SSCH should still deliver the packet. To handle this, we initially retain the packet in the packet queue. Packets are dropped only when SSCH gives up on all packets to a given destination, and this dropping of an entire flow occurs only when we have failed to transmit a packet to the destination node for an entire cycle through the channel schedule. We will explain the meaning of a cycle through the channel schedule in Section 3.3, but with our chosen parameter settings the timeout is 530 ms. After a flow has been garbage collected, new packets with the same destination inserted in the queue are assigned to a new flow, and attempted in the normal manner.

This packet scheduling policy is simple to implement, and yields good performance in the common case that node schedules are known, and information about node availability is accurate. A potential drawback is that a node crash (or other failure event) can lead to a number of wasted RTSs

to the failed node. When added across channels, the number may exceed the IEEE 802.11 recommended limit of 7 retransmission attempts. In Section 4, we quantify the cost of such failures and show that it is small.

3.3 Channel Scheduling

We begin our description of channel scheduling by describing the data structure used to represent the channel schedule. We then describe the policy nodes use to act on their own channel schedule, the mechanism to communicate channel schedules to other nodes, and finally the policy nodes implement for updating or changing their own channel schedule.

The channel schedule must capture a given node's plans for channel hopping in the future, and there is obvious overhead to representing this as a very long list. Instead, we compactly represent the channel schedule as a current channel and a rule for updating the channel – in particular, as a set of 4 (channel, seed) pairs. Our experimental results show that 4 pairs suffice to give good performance (Section 4). We represent the (channel, seed) pair as (x_i, a_i). The channel x_i is represented as an integer in the range [0, 12] (13 possibilities), and the seed a_i is represented as an integer in the range [1, 12]. Each node iterates through all of the channels in the current schedule, switching to the channel designated in the schedule in each new slot. The node then increments each of the channels in its schedule using the seed,

$$x_i \leftarrow (x_i + a_i) \bmod 13$$

and repeats the process.

We introduce one additional slot to prevent logical partitions. After the node has iterated through every channel on each of its 4 slots, it switches to a *parity slot* whose channel assignment is given by $x_{parity} = a_1$. The term parity slot is derived from the analogy to the parity bits appended at the end of a string in some error correcting codes. The mathematical justification for this design is given in Section 3.4. We use the term *cycle* to refer to the 530 ms iteration through all the slots, including the parity slot.

In Figure 2, we illustrate possible channel schedules for two nodes in the case of 2 slots and 3 channels. In the Figure, node A and node B are synchronized in one of their two slots (they have identical (channel, seed) pairs), and they also overlap during the parity slot. The field of the channel schedule that determines the channel during each slot is shown in bold. Each time a slot reappears, the channel is updated using the seed. For example, node A's slot 1 initially has (channel, seed) = (1,2). The next time slot 1 is entered, the channel is updated by adding the seed to it mod 3 (mod 3 because in this example, there are 3 channels). The resulting channel is given by $(1 + 2) \bmod 3 = 0$.

Nodes switch from one slot to the next according to a fixed schedule (every 10 ms in our current parameter settings). However, the decision to switch channels may occur while a node is transmitting or receiving a packet. In this case we delay the switch until after the transmission and ACK (or lack thereof) have occurred.

Nodes learn each other's schedules by periodically broadcasting their seeds and the offset within this cycle through the channel schedule. We use the IEEE 802.11 Long Control Frame Header format to embed both the schedule and the node's current offset – this is discussed in more detail in Section 3.5. The SSCH layer at each node schedules one of

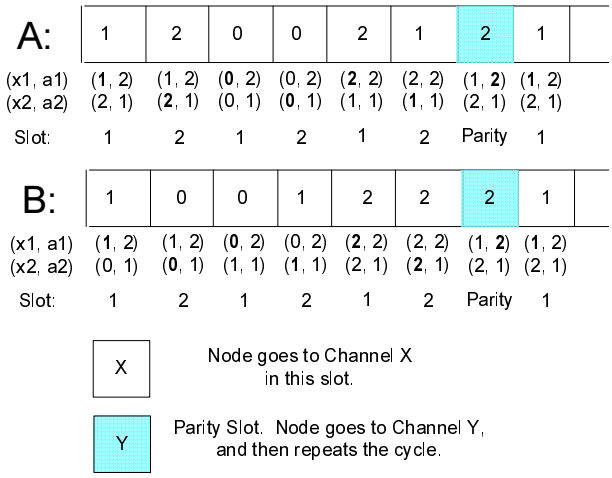

Figure 2: Channel hopping schedules for two nodes with 3 channels and 2 slots. Node A always overlaps with Node B in slot 1 and the parity slot. The field of the channel schedule that determines the channel during each slot is shown in bold.

these packets for broadcast once per slot.

Nodes also update their knowledge of other nodes' schedules by trying to communicate and failing. Whenever a node sends an RTS to another node, and that node fails to respond even though it was believed to be in this slot, the node sending the RTS updates the channel schedule for the other node to reflect that it does not currently know the node's schedule in this slot.

We now turn to the question of how a given node changes its own schedule. Schedules are updated in two ways: each node attempts to maintain that its slots start and stop at roughly the same time as other nodes, and that its channel schedule overlaps with nodes for which it has packets to send. We embed the information needed for this synchronization within the Long Control Frame Header as well. Using this information, a simple averaging scheme such as described by Elson et al [15] can be applied to achieve the loose synchronization required for good performance (Section 4 shows that a 100 μs skew in clock times leads to less than a 2% decrease in capacity).

At a high level, each node achieves overlap with nodes for which it has traffic straightforwardly, by changing part of its own schedule to match that of the other nodes. However, a number of minor decisions must be made correctly in order to achieve this high level goal.

Nodes recompute their channel schedule right before they enqueue the packet announcing this schedule in the NIC (and so at least once per slot). In a naive approach, this node could examine its packet queue, and select the (channel, seed) pairs that lead to the best opportunity to send the largest number of packets. However, this ignores the interest this node has in receiving packets, and in avoiding congested channels. An example of the kind of problem that might arise if one ignores the interest in receiving packets is given in Figure 3. Here, A synchronized with B, and then B synchronized with C in such a way that A was no longer synchronized with B. This could have been avoided if B had used its other slot to synchronize with C, as it would have

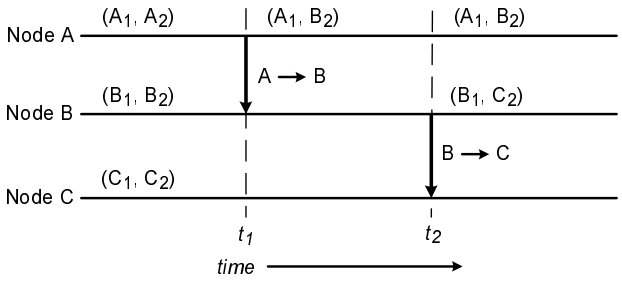

Figure 3: The problem with a naive synchronization scheme. Node A has two slots, with (channel, seed) pairs represented by A_1 and A_2; nodes B and C are similarly depicted. At time t_1, node A synchronizes with node B. Node B synchronizes with node C at time t_2, after which A and B are no longer synchronized.

if it considered its interest in receiving packets.

To account for this node's interest in receiving packets, we maintain per-slot counters for the number of packets received during the previous time the slot was active (ignoring broadcast packets). Any slot that received more than 10 packets during the previous iteration through that slot is labeled a *receiving* slot; if all slots are receiving slots, any one is allowed to be changed. If some slots are receiving slots and some are not, only the (channel, seed) pair on a non-receiving slot is allowed to be changed for the purpose of synchronizing with nodes we want to send to.

To account for channel congestion, we compare the (channel, seed) pairs of all the nodes that sent us packets in a given slot with the (channel, seed) pairs of all the other nodes in our list of channel schedules. If the number of other nodes synchronized to the same (channel, seed) pair is more than twice as many as this node communicated with in the previous occurrence of the slot, we attempt to *de-synchronize* from these other nodes. De-synchronization just involves choosing a new (channel, seed) pair for this slot. In our experiments, this de-synchronization mechanism was both necessary and sufficient to prevent the nodes from all converging to the same set of (channel, seed) pairs.

The final constraints we add moderate the pace of change in schedule information. Each node only considers updating the (channel, seed) pair for the next slot, never for slots further in the future. If the previous set of criteria suggest updating some slot other than the next slot, we delay that decision. Given these constraints, picking the best possible (channel, seed) pair simply requires considering the choice that synchronizes with the set of nodes for which we have the largest number of queued packets. Additionally, the (channel, seed) pair for the first slot is only allowed to be updated during the parity slot – this helps to prevent logical partition, as will be explained in more detail in Section 3.4.

This strategy naturally supports nodes acting as sources, sinks, or forwarders. A source node will find that it can assign all of its slots to support sends. A sink node will find that it rarely changes its slot assignment, and hence nodes sending to it can easily stay synchronized. A forwarding node will find that some of its slots are used primarily for receiving; after re-assigning the channel and seed in a slot to support sending, the slots that did not change are more likely to receive packets, and hence to stabilize on their current channel and seed as receiving slots for the duration of the current traffic patterns. Our simulation results (Section 4) support this conclusion. We refer to the technique of enabling this synchronization pattern as *partial synchronization*.

3.4 Mathematical Properties of SSCH

Our discussion of the mathematical properties of SSCH will initially focus on the static case. The behavior of SSCH when channel schedules are not changing assures us that in a steady-state flow setting, nodes will rendezvous appropriately, in a sense that we make precise below. We will then expand our discussion to include the dynamics of channel scheduling in an environment where flows are starting and stopping.

The channel scheduling mechanism has three simultaneous design goals: allowing nodes to be synchronized in a slot, infrequent overlap between nodes that do not have data to send to each other, and ensuring that all nodes come into contact occasionally (to avoid a logical partition). To achieve these goals, we rely on a very simple mathematical technique, addition modulo a prime number.

Consider two nodes that want to be synchronized in a given slot. If they have identical (channel, seed) pairs for this slot, then clearly they will remain synchronized in future iterations (using the static assumption). Now consider two nodes that are not synchronized because they have different seeds. A simple calculation shows that these two nodes will overlap exactly one out of every 13 iterations in this slot (recall that 13 is the number of channels). This is the behavior we want from these nodes: they overlap regularly enough that they can exchange their channel schedules, but they are mostly on different channels, and so do not interfere with each other's transmissions.

Now consider the rare case that two nodes share identical seeds in every slot, but different channels accompany each seed – this has at most a 1 in $13^4 \approx 28,000$ chance of occurring for randomly chosen (channel, seed) pairs. In this case, the nodes will march in lock-step through the same set of channels in each slot, never overlapping. This would be problematic, and it is this situation that the parity slot prevents. To justify this claim, we consider two distinct situations. If both nodes enter their parity slot at the same time, then they overlap there because the parity channel is equal to the seed for the first slot for both nodes. With our chosen parameter settings of 10 ms per slot, 4 slots, and 13 channels, this overlap occurs once every 530 ms and lasts for 10ms. If their parity slots do not occur at the same time, then the first node's parity slot offers a fixed target for the slot in which the second node is changing channels, and again, the two nodes will overlap. This overlap occurs once every 7 seconds. Although both these cases will be rare, the SSCH time synchronization mechanism allows us to ignore the second case entirely – a relative clock skew of 5 ms or less is sufficient to guarantee that two parity slots overlap in time.

Now considering the dynamic case (and assuming clock synchronization to within 5 ms), we note that nodes are not permitted to change the seed for the first of their four slots except during a parity slot. Therefore they will always overlap in either the first slot or the parity slot, and hence will always be able to exchange channel schedules within a moderate time interval.

The use of addition modulo a prime to construct channel hopping schedules does not restrict SSCH to scenarios where the number of channels is a prime number. If one desired to use SSCH with a wireless technology where the number of channels is not a prime, one could straightforwardly use a larger prime as the range of x_i, and then map down to the actual number of channels using a modulus reduction. Though the mapping would have some bias to certain channels, the bias could be made arbitrarily small by choosing a sufficiently large prime.

A final point about the use of addition modulo a prime is that SSCH can be modified to require fewer bits to represent a node's schedule by reducing the number of choices for a seed. The only penalty to this reduction is increasing the protocol's reliance on the parity slot for avoiding logical partitions.

3.5 Implementation Considerations

When simulating SSCH in QualNet [5], we made two technical choices that seem to be relatively uncommon based on our reading of the literature. The first technical choice relates to how we added SSCH to an existing system, and the second relates to a little-utilized part of the IEEE 802.11 specification.

In order to implement SSCH, we had to implement new packet queuing and retransmission strategies. To avoid requiring modifications to the hardware (in QualNet, the hardware model) or the network stack, SSCH buffers packets below the network layer, but above the NIC device driver. To maintain control over transmission attempts, we configure the NIC to buffer at most one packet at a time, and to attempt exactly one RTS for each packet before returning to the SSCH layer. By observing NIC-level counters before and after every attempted packet transmission, we are able to determine whether a CTS was heard for the packet, and if so, whether the packet was successfully transmitted and acknowledged. All the necessary parameters to do this are exposed by the hardware model we used in QualNet. This also prevents head-of-line blocking from interfering with our desire to implement the SSCH transmission strategy.

For efficiency reasons, we choose to use the IEEE 802.11 Long Control Frame Header format to broadcast channel schedules and current offsets, rather than using a full broadcast data packet. The most common control frames in IEEE 802.11 (RTS, CTS, and ACK) use the alternative short format. The long format was included in the IEEE 802.11 standard to support inter-operability with legacy 1-Mbps and 2-Mbps DSSS systems [8]. The format contains 6 unused bytes; we use 4 to embed the 4 (channel, seed) pairs, and another 2 to embed the offset within the cycle (i.e., how far the node has progressed through the 530 ms cycle).

Lastly, we comment that the beaconing mechanism used in IEEE 802.11 ad-hoc mode for associating with a Basic Service Set (BSS) works unchanged in the presence of SSCH. A newly-arrived node can associate to a BSS as soon as it overlaps in the same channel with any already-arrived node.

%

4. SYSTEM EVALUATION

We simulate SSCH in QualNet and compare its performance with the commonly used single-channel IEEE 802.11a protocol. In Section 4.1, we present microbenchmarks quantifying the different SSCH overheads. In Section 4.2, we present macrobenchmarks on the performance of SSCH with a large number of nodes in a single hop environment. In Section 4.3, we extend the macrobenchmark evaluation to encompass mobility and multi-hop routing. Our results show that SSCH incurs very low overhead, and significantly outperforms IEEE 802.11a in a multiple flow environment.

Our simulation environment comprises a varying number of nodes in a $200m \times 200m$ area. All nodes in a single simulation run use the same MAC, either SSCH or IEEE 802.11a. We set all nodes to operate at the same raw data rate, 54 Mbps. We assume 13 usable channels in the 5 GHz band. SSCH is configured to use 4 seeds, and each slot duration is 10 ms. All seeds are randomly chosen at the beginning of each simulation run. The macrobenchmarks in Sections 4.2 and 4.3 are averages from 5 independent simulation runs, while the microbenchmarks in Section 4.1 are drawn from a single simulation run.

We primarily measure throughput under a traffic load of maximum rate UDP flows. In particular, we use Constant Bit Rate (CBR) flows of 512 byte packets sent every 50 μs. This data rate is more than the sustainable throughput of IEEE 802.11a operating at 54 Mbps.

For all our simulations, we modified QualNet to use a channel switch delay of 80 μs. This choice was based on recent work in solid state electronics on reducing the settling time of the Voltage Control Oscillator (VCO) [7]. Switching the channel of a wireless card requires changing the input voltage of the VCO, which operates in a Phase Locked Loop (PLL) to achieve the desired output frequency. The delay in channel switching is due to this settling time. The specification of Maxim IEEE 802.11b Transceivers [3] shows this delay to be 150 μs. A more recent work [17] shows that this delay can be reduced to 40-80 μs for IEEE 802.11a cards.

4.1 Microbenchmarks

We present microbenchmarks measuring the overhead of SSCH in several different scenarios. In Section 4.1.1, we measure the overhead during the successful initiation of a CBR flow. In Section 4.1.2, we measure the overhead on an existing session of failing to initiate a parallel CBR flow. In Section 4.1.3, we measure the overhead of supporting two flows simultaneously. In Section 4.1.4, we measure the overhead of continuing to attempt transmissions to a mobile node that has moved out of range. These scenarios cover many of the different dynamic events that a MAC must appropriately handle: a flow starting while a node is present, a flow starting while a node is absent, simultaneous flows where both nodes are present, simultaneous flows where one node moves out of range, etc. Finally, the scenario in Section 4.1.5 measures the overhead of SSCH with respect to a different kind of event, clock skew.

4.1.1 Overhead of Switching and Synchronizing

In this experiment, we measured the overhead of successfully initiating a CBR flow between two nodes within communication range of each other. The first node initiates the flow just after the parity slot. This incurs a worst-case delay in synchronization, because the first of the four slots will not be synchronized until 530 ms later.

Figure 4 shows the moving average over 20 ms of the throughput at the receiver node. The sender quickly synchronizes with the receiver on three of the four slots, as it should, and on the fourth slot after 530 ms. The fig-

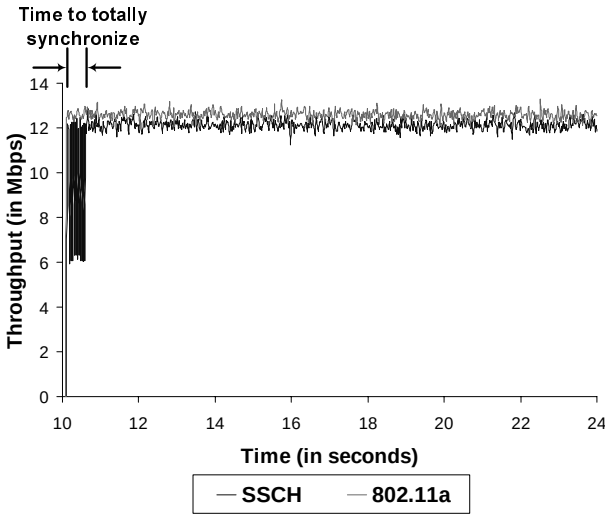

Figure 4: Switching and Synchronizing Overhead: Node 1 starts a maximum rate UDP flow to Node 2. We show the throughput for both SSCH and IEEE 802.11a.

ure shows the throughput while synchronizing (oscillating around 3/4 of the raw bandwidth), and the time required to synchronize. After synchronizing, the channel switching and other protocol overheads of SSCH lead to only a 400 Kbps penalty in the steady-state throughput relative to IEEE 802.11a. This penalty conforms to our intuition about the overheads in SSCH: a node spends 80 μs every 10 ms switching channels (80 μs/10 ms = .008), and then must wait for the duration of a single packet to avoid colliding with pre-existing packet transmissions in the new channel (1 packet/35 packets = .028). Adding these two overheads together leads to an expected cumulative overhead of 3.6%, which is in close agreement with the measured overhead of (400 Kbps/12 Mbps) = 3.3%.

Note that the throughput reaches a maximum of only 13 Mbps, although the raw data rate is 54 Mbps. This low utilization can be explained by the IEEE 802.11a requirement that the RTS/CTS packets be sent at the lowest supported data rate, 6 Mbps, along with other overheads [18].

4.1.2 Overhead of an Absent Node

SSCH requires more re-transmissions than IEEE 802.11 to prevent logical partitions. These retransmissions waste bandwidth that could have been dedicated to a node that was present on the channel. To quantify this overhead, we initiated a CBR flow between two nodes, allowed the system to quiesce, and then initiated a send from the first node to a non-existent node. Figure 5 shows the moving average over 80 ms of the throughput. The Figure shows that the sender takes 530 ms to timeout on the non-existent node. During this time the throughput drops by 550 Kbps, which is a small fraction (4.6%) of the total throughput.

4.1.3 Overhead of a Parallel Session

Our next experiment quantifies the ability of SSCH to fairly share bandwidth between two flows, and to quickly achieve this fair sharing. To measure this we start with Node 1 sending a maximum rate UDP flow to Node 2. At

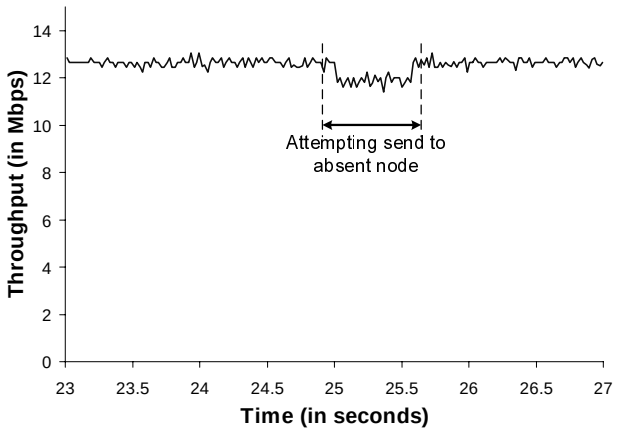

Figure 5: Overhead of an Absent Node: Node 1 is sending a maximum rate UDP flow to Node 2. Node 1 then attempts to send a packet to a non-existent node.

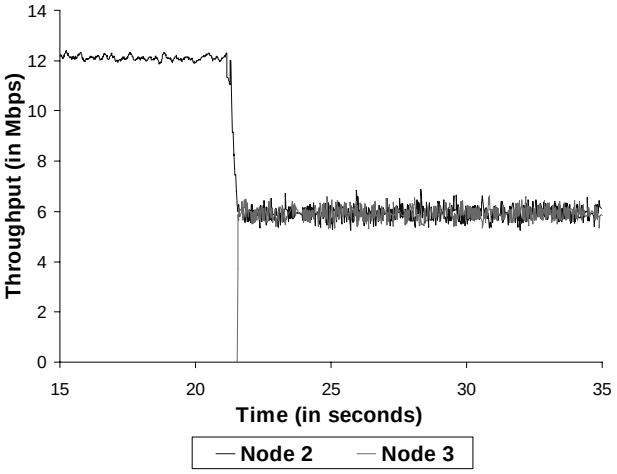

Figure 6: Overhead of a Parallel Session: Node 1 is sending a maximum rate UDP flow to Node 2. Node 1 then starts a second flow to Node 3.

21.5 seconds, Node 1 starts a second maximum rate UDP flow to Node 3. Figure 6 presents the moving average over 140 ms of the throughput achieved by both receivers. The bandwidth is split between the receivers nearly perfectly, and with no decrease in net throughput.

4.1.4 Overhead of Mobility

We now analyze the effect of mobility at a micro-level on the performance of SSCH. Ideally, SSCH should be able to detect a link breakage due to movement of a node, and subsequently re-synchronize to other neighbors. We show that SSCH can indeed handle this scenario with an experiment comprising 3 nodes and 2 flows, and in Figure 7 we present the moving average over 280 ms of each flow's throughput.

Node 1 is initially sending a maximum rate UDP flow to Node 2. Node 1 initiates a second UDP flow to Node 3 at around 20.5 seconds. This bandwidth is then shared between both the flows (as in the experiment of Section 4.1.3) until 30 seconds, when Node 3 moves out of the communi-

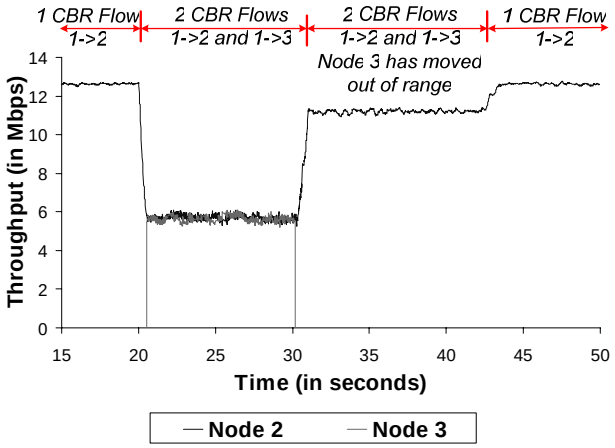

Figure 7: Overhead of Mobility: Node 1 is sending a maximum rate UDP flow to Node 2. Node 1 starts another maximum rate UDP flow to Node 3. Node 3 moves out of range at 30 seconds, while Node 1 continues to attempt to send until 43 seconds.

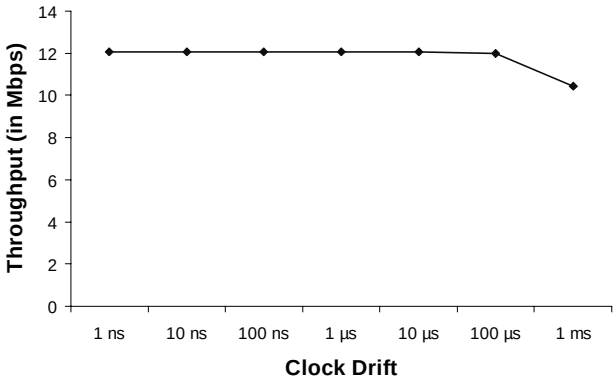

Figure 8: Overhead of Clock Skew: Throughput between two nodes using SSCH as a function of clock skew.

cation range of Node 1. Our experiment configures Node 1 to continue to attempt to send to Node 3 until 43 seconds, and during this time it continues to consume a small amount of bandwidth. In contrast, the experiment in Section 4.1.2 measured the overhead of enqueueing a single packet to an absent node. When the flow to Node 3 finally stops, Node 2's received throughput increases back to its initial rate.

4.1.5 Overhead of Clock Drift

As we described in Section 3.3, SSCH tries to synchronize slot begin and end times, though it is also designed to be robust to clock skew. In this experiment, we quantify the robustness of SSCH to moderate clock skew. We measure the throughput between two nodes after artificially introducing a clock skew between them, and disabling the SSCH synchronization scheme for slot begin and end times. We vary the clock skew from 1 ns (10^{-6} ms) to 1 ms such that the sender is always ahead of the receiver by this value, and present the results in Figure 8. Note the log scale on the x-axis.

The throughput achieved between the two nodes is not significantly affected by a clock skew of less than 10 μs. These practical values of clock skew are extremely small to impact the performance of SSCH. The drop in throughput is more for larger clock skews, although the throughput is still acceptable at 10.5 Mbps when the skew value is an extremely high 1 ms.

These results provide justification for the design choice we made of not requiring nodes to switch synchronously across slots, as described in Section 3.3. For example, a node will delay switching to receive an ACK, or to send a data packet if its channel reservation is successful. In the 100 node experiment described in Section 4.3.2, we measured the skew in channel switching times for a traffic pattern of 50 flows to be approximately 20 μs. Figure 8 shows that this is a negligible amount.

4.2 Macrobenchmarks: Single-hop Case

We now present simulation results showing SSCH's ability to achieve and sustain a consistently high throughput for a traffic pattern consisting of multiple flows. We first evaluate this using steady state UDP flows. We then extend our evaluation to consider a dynamic traffic scenario where UDP flows both start and stop. Finally, we study the performance of TCP over SSCH.

4.2.1 Disjoint Flows

We first look at the number of disjoint flows that can be supported by SSCH. All nodes in this experiment are in communication range of each other, and therefore two flows are considered disjoint if they do not share either endpoint. Ideally, SSCH should utilize the available bandwidth on all the channels on increasing the number of disjoint flows in the system. We evaluate this by varying the number of nodes in the network from 2 to 30 and introducing a flow between disjoint pairs of nodes — the number of flows varies from 1 to 15.

Figure 9 shows the per-flow throughput, and Figure 10 shows the total system throughput. IEEE 802.11a performs marginally better when there is just one flow in the network. When there is more than one flow, SSCH significantly outperforms IEEE 802.11a.

An increase in the number of flows decreases the per-flow throughput for both SSCH and IEEE 802.11a. However, the drop for IEEE 802.11a is much more significant. The drop for IEEE 802.11a is easily explained by Figure 10, which shows that the overall system throughput for IEEE 802.11a is approximately constant.

It may seem surprising that the SSCH system throughput has not stabilized at 13 times the throughput of a single flow by the time there are 13 flows. However, this can be attributed to SSCH's use of randomness to distribute flows across channels. These random choices do not lead to a perfectly balanced allocation, and therefore there is still unused spectrum even when there are 13 flows in the system, as shown by the continuing positive slope of the curve in Figure 9.

4.2.2 Non-disjoint Flows

We now consider the case when the flows in the network are not disjoint – nodes participate as both sources and sinks, and in multiple flows. This scenario stresses SSCH's ability to efficiently support sharing among simultaneous

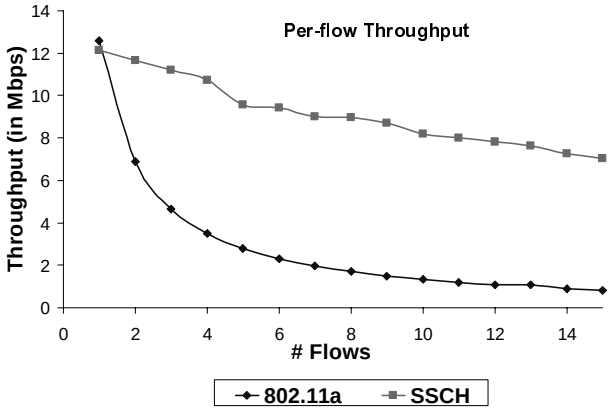

Figure 9: Disjoint Flows: The per-flow throughput on increasing the number of flows.

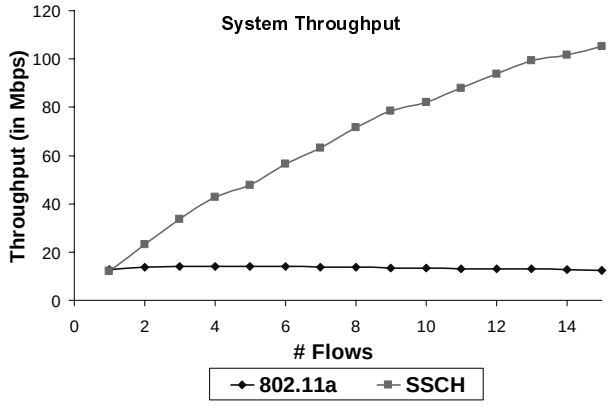

Figure 10: Disjoint Flows: The system throughput on increasing the number of flows.

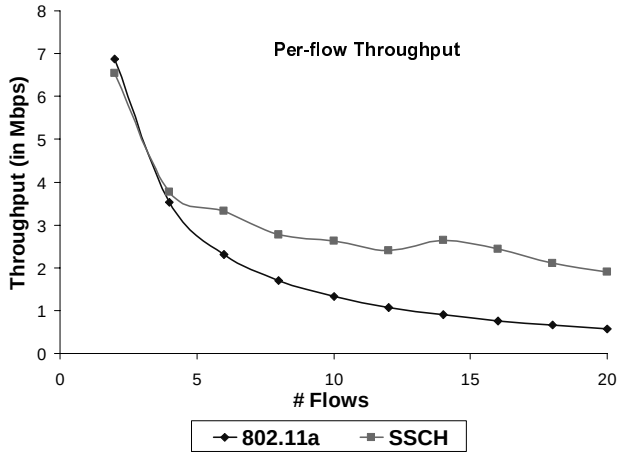

Figure 11: Non-disjoint Flows: The per-flow throughput on increasing the number of flows.

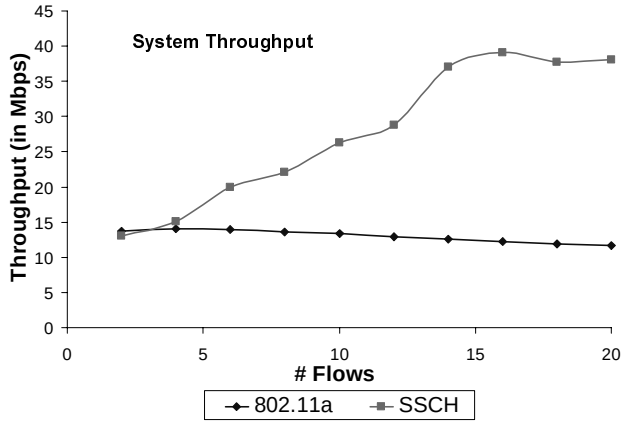

Figure 12: Non-disjoint Flows: The system throughput on increasing the number of flows.

flows that have a common endpoint. Each node in the network starts a maximum rate UDP flow with one other randomly chosen node in the network. We vary the number of nodes (and thus flows) from 2 to 20. As in the previous experiment, all nodes are within communication range of each other. We present the per-flow and system throughput for SSCH and IEEE 802.11a in Figures 11 and 12 respectively. The curves are not monotonic because variation in the random choices leads to some receivers being recipients in multiple flows (and hence bottlenecks). This lack of monotonicity persisted even after averaging over 5 simulation runs. As in the disjoint flow experiment, SSCH performs slightly worse in the case of a single flow, but much better in the case of a large number of flows.

4.2.3 Effect of Flow Duration

SSCH introduces a delay when flows start because nodes must synchronize. This overhead is more significant for shorter flows. We evaluate this overhead for maximum rate UDP flows with different flow lengths. In the first experiment the flow duration is chosen randomly between 20 and 30 ms, while for the second experiment it is between 0.5 and 1 second. In both the experiments, each node starts a flow with a randomly selected node, discards all packets at the end of the designated sending window, pauses for a second at the end of the flow, and then starts another flow with a new randomly selected node. The decision to discard enqueued packets at the end of the flow duration is designed to model a time-sensitive application. This process continues for 30 seconds. We run these experiments for both SSCH and IEEE 802.11a, and vary the number of nodes from 2 to 16.

Figure 13 presents the ratio of the average throughput over SSCH to the average throughput over IEEE 802.11a. For small numbers of sufficiently short-lived flows, IEEE 802.11a offers superior performance; short flows do indeed suffer from a more pronounced synchronization overhead. However, as soon as there are more than 4 simultaneous flows in the network, the ability of SSCH to spread transmissions across multiple channels leads to a higher total throughput than IEEE 802.11a in both the short and long flow scenarios.

4.2.4 TCP Performance over SSCH

We now study the behavior of TCP over SSCH. SSCH allows a node to stay synchronized to multiple nodes over

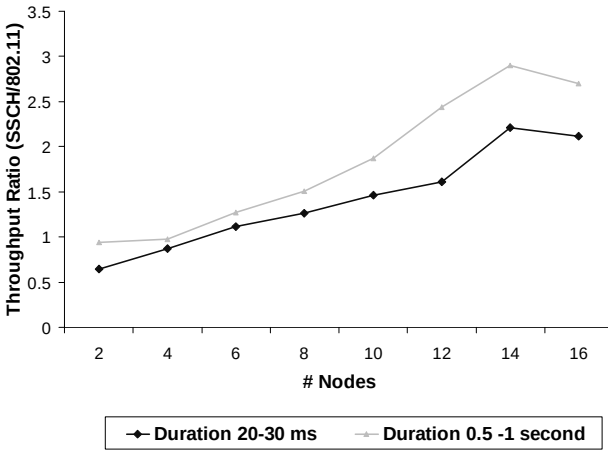

Figure 13: Effect of Flow Duration: Ratio of SSCH throughput to IEEE 802.11a throughput for flows having different durations.

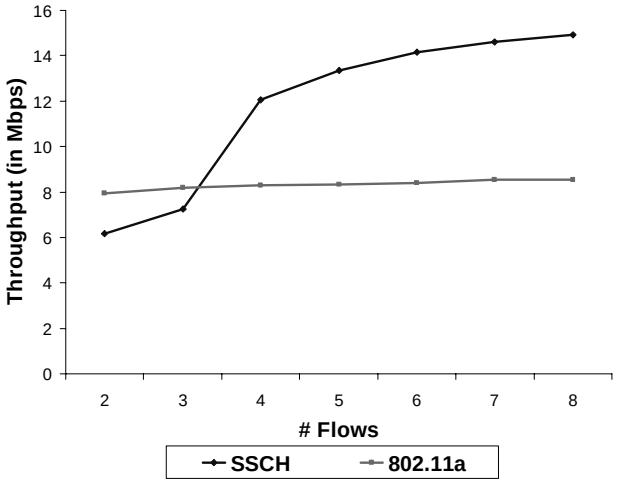

Figure 14: TCP over SSCH: Steady-state TCP throughput when varying the number of non-disjoint flows.

different slots. However, this might cause significant jitter in packet delivery times, which could adversely affect TCP. To evaluate this concern quantitatively, we run an experiment where we vary the number of nodes in the network from 2 to 9, such that all nodes are in communication range of one another. We then start an infinite-size file transfer over FTP from each node to a randomly selected other node. This choice to use non-disjoint flows is designed to stress the SSCH implementation by requiring nodes to be synchronized as either senders or receivers with multiple other nodes. In Figure 14 we present the resulting cumulative steady-state TCP throughput over all the flows in the network.

Figure 14 shows that the TCP throughput for a small number of flows is lower for SSCH than the throughput over IEEE 802.11a. However, as the number of flows increases, SSCH does achieve a higher system throughput. Although TCP over SSCH does provide higher aggregate throughput than over IEEE 802.11a, the performance improvement is

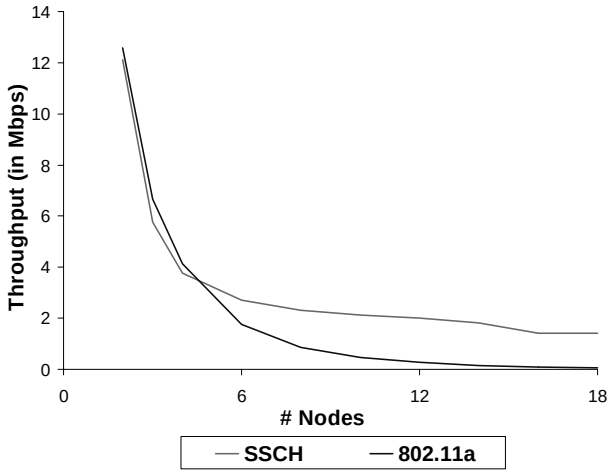

Figure 15: Multi-hop Chain Network: Variation in throughput as chain length increases.

not nearly as good as for UDP flows. This shows that jitter due to SSCH does have an impact on the performance of TCP. A more detailed analysis of the interaction between TCP and SSCH, and modifications to support better interactions between TCP and SSCH, is a subject we plan to address in our future work.

4.3 Macrobenchmarks: Multi-hop Case and Mobility

We now evaluate SSCH's performance when combined with multi-hop flows and mobile nodes. We first analyze the behavior of SSCH in a multi-hop chain network. We then consider large scale multi-hop networks, both with and without mobility. As part of this analysis, we study the interaction between SSCH and MANET routing protocols.

4.3.1 Performance in a Multi-hop Chain Network

IEEE 802.11 is known to encounter significant performance problems in a multi-hop network [34]. For example, if all nodes are on the same channel, the RTS/CTS mechanism allows at most one hop in an $A-B-C-D$ chain to be active at any given time. SSCH reduces the throughput drop due to this behavior by allowing nodes to communicate on different channels. To examine this, we evaluate both SSCH and IEEE 802.11a in a multi-hop chain network.

We vary the number of nodes, which are all in communication range, from 2 to 18. We initiate a single flow that encounters every node in the network. Although more than 4 nodes transmitting within interference range of each other would be unlikely to arise from multi-hop routing of a single flow, it could easily arise in a general distributed application. Figure 15 shows the maximum throughput as the number of nodes in the chain is varied. We see that there is not much difference between SSCH and IEEE 802.11a for flows with few hops. As the number of hops increases, SSCH performs much better than IEEE 802.11a since it distributes the communication on each hop across all the available channels.

4.3.2 Performance in a Multi-hop Mesh Network

We now analyze the performance of SSCH in a large scale multi-hop network without mobility. We place 100 nodes

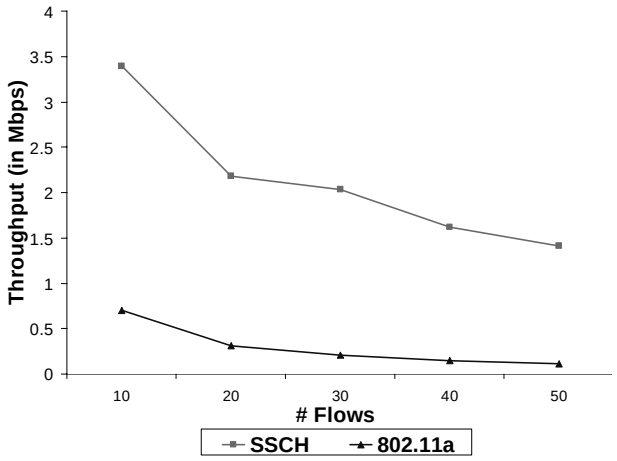

Figure 16: Mulithop Mesh Network of 100 Nodes: The per-flow throughput on varying the number of flows in the network.

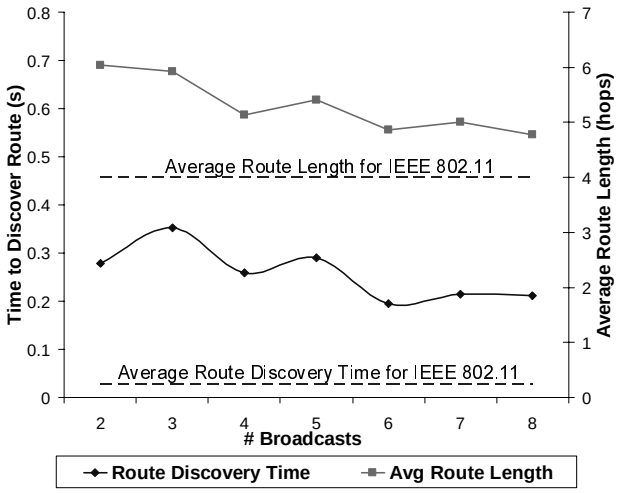

Figure 17: Impact of SSCH on Unmodified MANET Routing Protocols: The average time to discover a route and the average route length for 10 randomly chosen routes in a 100 node network using DSR over SSCH.

uniformly in a 200×200 m area, and set each node to transmit with a power of 21 dBm. The Dynamic Source Routing (DSR) [22] protocol is used to discover the source route between different source-destination pairs. These source routes are then input to a static variant of DSR that does not perform discovery or maintain routes. We vary the number of maximum rate UDP flows from 10 to 50. We generate source-destination pairs by choosing randomly, and rejecting pairs that are within a single hop of each other.

We present the per-flow throughput in Figure 16. Increasing the number of flows leads to greater contention, and the average throughput of both SSCH and IEEE 802.11a drops. For every considered number of flows, SSCH provides significantly higher throughput than IEEE 802.11a. For 50 flows, the inefficiencies of sharing a single channel are sufficiently pronounced that SSCH yields more than a factor of 15 capacity improvement.

4.3.3 *Impact of Channel Switching on MANET Routing Protocols*

Previous work on multi-channel MACs has often overlooked the effect of channel switching on routing protocols. Most of the proposed protocols for MANETs rely heavily on broadcasts (e.g., DSR [22] and AODV [28]). However, neighbors using a multi-channel MAC could be on different channels, which could cause broadcasts to reach significantly fewer neighbors than in a single-channel MAC. SSCH addresses this concern using a broadcast retransmission strategy discussed in Section 3.2.

We study the behavior of DSR [22] over SSCH in the same experimental setup used in Section 4.3.2, with 100 nodes in a 200 m×200 m area. However, we reduce the transmission power of each node to 16 dBm to force routes to increase in length (and hence to stress DSR over SSCH). We select 10 source-destination pairs at random, and we use DSR to discover routes between them. In Figure 17 we compare the performance of DSR over SSCH, when varying the SSCH broadcast transmission count parameter (the number of consecutive slots in which each broadcast packet is sent once).

Figure 17 shows that the performance of DSR over SSCH improves with an increase in the broadcast transmission count. The DSR Route Request packets see more neighbors when SSCH broadcasts them over a greater number of slots. This increases the likelihood of discovering shorter routes, and the speed with which routes are discovered. However, there seems to be little additional benefit to increasing the broadcast parameter to a value greater than 6. We attribute the slight bumpiness in the curves to the stochastic nature of DSR, and its reliance on broadcasts.

Comparing SSCH to IEEE 802.11a, we see that the SSCH discovers routes that are comparable in length. However, the average route discovery time for SSCH is much higher than for IEEE 802.11a. Because each slot is 10 ms in length, broadcasts are only retransmitted once every 10 ms, and this leads to a significantly longer time to discover a route to a given destination node. We believe that this latency is a fundamental difficulty in using a reactive protocol such as DSR with SSCH. We plan to explore the interaction of other proactive and hybrid routing protocols with SSCH in the future.

4.3.4 *Performance in Multi-hop Mobile Networks*

We now study the impact of mobility in a network using DSR over IEEE 802.11a and SSCH. In this experiment, we place 100 nodes randomly in a square and select 10 flows. Each node transmits packets at 21 dBm. Node movement is determined using the Random Waypoint model. In this model, each node has a predefined minimum and maximum speed. Nodes select a random point in the simulation area, and move towards it with a speed chosen randomly from the interval. After reaching its destination, a node rests for a period chosen from a uniform distribution between 0 and 10 seconds. It then chooses a new destination and repeats the procedure. In our experiments, we fix the minimum speed at 0.01 m/s and vary the maximum speed from 0.2 to 1.0 m/s. Although we have studied SSCH at higher speeds, the results are not significantly different. We performed this experiment using two different areas for the nodes, a $200m \times 200m$ area and a $300m \times 300m$ area. We refer to

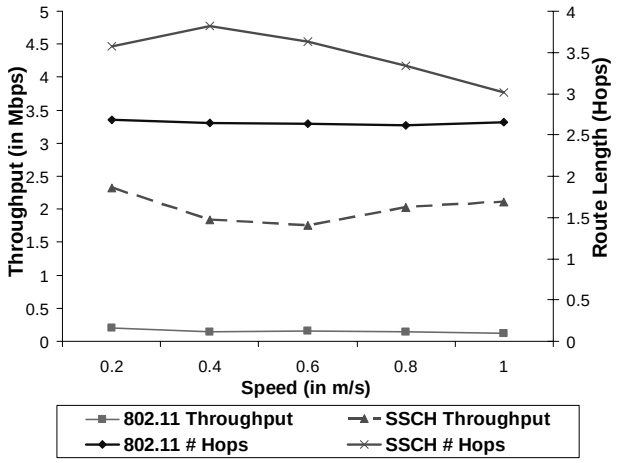

Figure 18: Dense Multi-hop Mobile Network: The per-flow throughput and the average route length for 10 flows in a 100 node network in a $200m \times 200m$ area, using DSR over both SSCH and IEEE 802.11a.

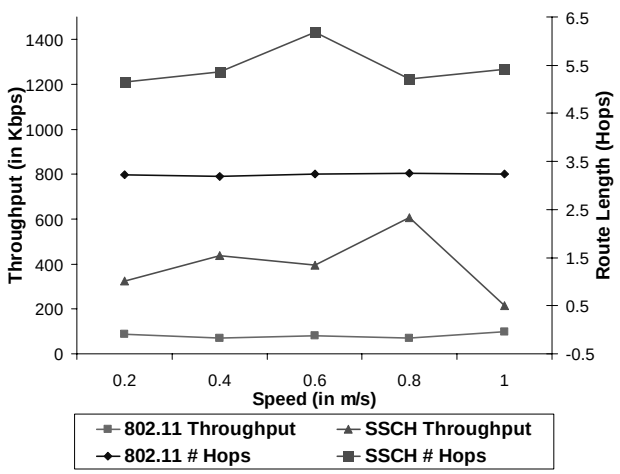

Figure 19: Sparse Multi-hop Mobile Network: The per-flow throughput and the average route length for 10 flows in a 100 node network in a $300m \times 300m$ area, using DSR over both SSCH and IEEE 802.11a.

the smaller area as the dense network, and the larger area as the sparse network – the average path is 0.5 hops longer in the sparse network. For all these experiments, we set the SSCH broadcast transmission count parameter to 6.

Figure 18 shows that in a dense network, SSCH yields much greater throughput than IEEE 802.11a even when there is mobility. Although DSR discovers shorter routes over IEEE 802.11a, the ability of SSCH to distribute traffic on a greater number of channels leads to much higher overall throughput. Figure 19 evaluates the same benchmarks in a sparse network. The results show that the per-flow throughput decreases in a sparse network for both SSCH and IEEE 802.11a. This is because the route lengths are greater, and it takes more time to repair routes. However, the same qualitative comparison continues to hold: SSCH causes DSR to discover longer routes, but still leads to an overall capacity improvement.

DSR discovers longer routes over SSCH than over IEEE 802.11a because broadcast packets sent over SSCH may not reach a node's entire neighbor set. Furthermore, some optimizations of DSR, such as promiscuous mode operation of nodes, are not as effective in a multi-channel MAC such as SSCH. Thus, although the throughput of mobile nodes using DSR over SSCH is much better than their throughput over IEEE 802.11a, we conclude that a routing protocol that takes the channel switching behavior of SSCH into account will likely lead to even better performance.

5. DISCUSSION

In this Section we discuss alternative designs for SSCH within the constraints that we enumerated in Section 2. We will discuss prior work related to SSCH in detail in Section 6.

SSCH distributes the rendezvous and control traffic across all the channels. One straightforward alternative scheme, which still only requires one radio, is to use one of the channels as a control channel, and all the other channels as data channels (e.g., [21]). Each node must then somehow split its time between the control channel and the data channels.

Such a scheme will have difficulty in preventing the control channel from becoming a bottleneck. Suppose that two nodes exchange RTS/CTS on the control channel, and then switch to a data channel to do transmission. Unless all other nodes were also on the control channel during the RTS/CTS exchange, these two nodes will still need to do an RTS/CTS on this channel in order to avoid the hidden terminal problem. The two nodes should wait to even do the RTS/CTS until after an entire packet transmission interval has elapsed, because another pair of nodes might have also switched to this channel, orchestrating that decision on the control channel during a time that the first pair of nodes were not on the control channel. In order to amortize this startup cost, the nodes should have several packets to send to each other. However, while any one node remains on a data channel, any other node that desires to send it a packet must remain idle on the control channel waiting for the node it desires to reach to re-appear. If the idle node on the control channel chooses not to wait, and instead switches to a data channel with another node for which it has traffic, it may repeatedly fail to rendezvous with the first node, leading to a significant imbalance in throughput and possibly a logical partition.

The problems with a dedicated control channel may be solvable, but it is clear that a straightforward approach with un-synchronized rendezvous presents several difficulties. If one instead tried to synchronize rendezvous on the control channel, the control channel could again become a bottleneck simply because many nodes simultaneously desire to schedule packets on that channel.

6. RELATED WORK

We divide the prior work relevant to SSCH into three categories: prior uses of pseudo-random number generators in wireless networking, channel switching for reasons other than capacity improvement, and alternative approaches to exploiting frequency diversity. In the first category, we find that pseudo-random number generators have been used for a variety of tasks in wireless networking. For example, the SEEDEX protocol [29] uses pseudo-random generators to avoid RTS/CTS exchanges in a wireless network. Nodes

build a schedule for sending and listening on a network, and publish their seeds to all the neighbors. A node will attempt a transmission only when all its neighbors (including the receiver) are in a listening state. Assuming relatively constant wireless transmission ranges, this protocol also helps in overcoming the hidden and exposed terminal problem. The TSMA protocol [11, 12] is a channel access scheme proposed as an alternative to ALOHA and TDMA, for time-slotted multihop wireless networks. TSMA aims to achieve the guarantees of TDMA without incurring the overhead of transmitting large schedules in a mobile environment. Each node is bootstrapped with a fixed seed that determines its transmission schedule. The schedules are constructed using polynomials over Galois fields (which have pseudo-random properties), and the construction guarantees that each node will overlap with only a single other node within a certain time frame. The length of the schedule depends on the number of nodes and the degree of the network. Porting these schedules to a multichannel scenario, where the number of channels is fixed, remains an open problem, and even such a porting would not meet the SSCH goal of supporting traffic-driven overlap. Redi et al. [13] use a pseudo-random generator to derive listening schedules for battery-constrained devices. Each device's seed is known to a base station, which can then schedule transmissions for the infrequent moments when the battery-constrained device is awake. Although pseudo-random generators have been used for a number of tasks (as this survey of the literature makes clear), to the best of our knowledge, SSCH is the first protocol to use a pseudo-random generator to construct a channel hopping schedule.

The second category of prior work is channel switching for reasons other than capacity improvement. MultiNet [10] is the main piece of work that we are aware of in this category. MultiNet allows a NIC to periodically hop between two channels, enabling a single wireless NIC to connect to two logically distinct networks, such as an AP network and an ad-hoc network. MultiNet is designed to provide new functionality: simultaneous connectivity to distinct networks using a single NIC. In contrast, SSCH is designed to yield capacity improvement within a single ad-hoc network.

The third category of prior work we define encompasses all prior approaches to increasing network capacity by exploiting frequency diversity. This is a significant body of work. The first division we make in this body of work is between research that assumes a single NIC capable of communicating on a single channel at any given instance in time, and research that assumes more powerful radio technology, such as multiple NICs [9, 30] or NICs capable of listening on many channels simultaneously [21, 26], even if they can only communicate on one. Our work falls in to the former category; the SSCH architecture can be deployed over a single standards-compliant NIC supporting fast channel switching.

Dynamic Channel Assignment (DCA) [33] and Multi-radio Unification Protocol (MUP) [9] are both technologies that use multiple radios (in both cases, two radios) to take advantage of multiple orthogonal channels. DCA uses one radio on a control channel, and the other radio switches across all the other channels sending data. Arbitration for channels is embedded in the RTS and CTS messages, and is executed on the control channel. Although this scheme may fully utilize the data channel, it does so at the cost of using an entire radio just for control. MUP uses both radios for data and control transmissions. Radios are assigned to orthogonal channels, and a packet is sent on the radio with better channel characteristics. This scheme gives good performance in many scenarios. However, it still only allows the use of as many channels as there are radios on each physical node. From our perspective, the key drawback to both DCA and MUP is simply that they require the use of multiple radios. Recently, commercial products have appeared that claim the ability to place multiple radios on a single NIC [2]. It is still not known whether these products will ever achieve as many radios on a NIC as there are available channels, nor what their power consumption will be.

A straightforward way to view the different potential gains of SSCH compared to a true multiple radio design is to consider two distinct sources of bottleneck in a single-radio, single-channel system: the saturation of the channel, and the saturation of any particular radio. Conceptually, SSCH significantly increases the channel bandwidth, without increasing the bandwidth of any individual radio. In contrast, a true multiple radio design increases both. A specific example of this difference is that a node using MUP (a true multiple radio design) can simultaneously send and receive packets on separate channels, while a node using SSCH can only perform one of these operations at a time.

We next turn our attention to work assuming more powerful radio technology than is currently technologically feasible. HRMA [35] is designed for frequency hopping spread spectrum (FHSS) wireless cards. Time is divided into slots, each one of which corresponds to a small fraction of the time required to send a packet, and the wireless NIC is on a different frequency during each slot. All nodes are required to maintain synchronized clocks, where the synchronization is at the granularity of slot times that are much shorter than the duration of a packet. Each slot is subdivided in to four segments of time for four different possible communications: HOP-RESERVED/RTS/CTS/DATA. The first three segments of time are assumed to be small in comparison with the amount of time spent sending a segment of the packet during the DATA time interval. To the best of our knowledge, a FHSS wireless card that supports this type of MAC protocol at high data rates is not commercially available.

Another line of related work assumes technology by which nodes can concurrently listen on all channels. For example, Nasipuri et al [26] and Jain et al [21] assume wireless NICs that can receive packets on all channels simultaneously, and where the channel for transmission can be chosen arbitrarily. In these schemes, nodes maintain a list of free channels, and either the sending or receiving node chooses a channel with the least interference for its data transfer. Wireless NICs do not currently support listening on arbitrarily many channels, and we do not assume the availability of such technology in the design of SSCH.

We finally consider prior work that only assumes the presence of a single NIC with a single half-duplex transceiver. The only other approach that we are aware of to exploiting frequency diversity under this assumption is Multichannel MAC (MMAC) [31]. Like SSCH, MMAC attempts to improve capacity by arranging for nodes to simultaneously communicate on orthogonal channels. Briefly, MMAC operates as follows: nodes using MMAC periodically switch to a common control channel, negotiate their channel selections, and then switch to the negotiated channel, where they contend for the channel as in IEEE 802.11. This scheme raises

several concerns that SSCH attempts to overcome. First, MMAC extends IEEE 802.11 Power Save Mode (PSM) for ad-hoc networks. This implies a relatively stringent reliance on clock synchronization, which is particularly hard to provide in multi-hop wireless networks [19]. In contrast, SSCH does not require tight clock synchronization because SSCH does not have a common control channel or a dedicated neighbor discovery interval. Secondly, synchronization traffic in MMAC can be a significant fraction of the system traffic, and the common synchronization channel can become a bottleneck on system throughput. SSCH addresses this concern by distributing synchronization and control traffic across all the available channels. A third concern with MMAC is that it can lead to packet delays of hundreds of milliseconds for even a single hop. MMAC switches channels every 100 ms, so a node with packets for two different destinations will have to wait at least 100 ms to send traffic to one of them whenever the two destinations decide to use different channels. In contrast, SSCH performs well while switching channels every 10 ms. A fourth concern with MMAC is that it does not specify how to support broadcasts, which are required by most MANET routing protocols (e.g., DSR). SSCH addresses this using a broadcast retransmission strategy that we experimentally validated to be compatible with DSR.

Although this survey does not cover all related work, it does characterize the current state of the field. At the level of detail in this section, prior work such as CHMA [32] is similar to HRMA [35], and MAC-SCC [25] and the MAC protocols implicit in the work of Li et al [24] and Fitzek et al [16] are similar to DCA [33]. However, a final related channel hopping technology that is worth mentioning is the definition of FHSS channels in the IEEE 802.11 [8] specification. At first glance, it may seem redundant that SSCH does channel hopping across logical channels, each one of which (per the IEEE 802.11 specification) may be employing frequency hopping across distinct frequencies at the physical later. The IEEE 802.11 specification justifies this physical layer frequency hopping with the scenario of providing support for multiple Basic Service Sets (BSS's) that can coincide geographically without coinciding on the same logical channel. In contrast, SSCH does channel hopping so that any two nodes can coincide as much or as little of the time as they desire. This is also at the heart of the difference between SSCH and past work on channel-hopping protocols where nodes overlap a fixed fraction of the time [12] – the degree of overlap between any two nodes using SSCH is traffic-dependent.

7. FUTURE RESEARCH

SSCH is a promising technology. In our future work, we plan to investigate how SSCH will perform when implemented over actual hardware, and is subject to the normal environmental vagaries of wireless networks, such as unpredictable variations in signal strength. As part of this implementation effort, we also plan to evaluate how metrics reflecting environmental conditions, such as ETX [14], can be integrated into SSCH.

Our results in Section 4.3.3 show that existing routing protocols do not give the best performance over SSCH. In particular, we find that the time to discover a route can be quite large in a reactive routing protocol being run over SSCH. In the future, we plan to more thoroughly evaluate routing over SSCH (as opposed to classical single channel routing), and to explore a wider variety of proactive and hybrid routing protocols over SSCH.

There are at least four additional topics that would also need to be addressed before SSCH can be deployed. One is interoperability with nodes that are not running SSCH. Another is the evaluation of power consumption under this scheme. We have not attempted to evaluate the energy cost of switching channels, nor have we attempted to enable a power-saving strategy such as in the IEEE 802.11 specification for access-point mode. A third topic of investigation is the evaluation of SSCH in conjunction with auto-rate adaptation mechanisms. A fourth topic is a more detailed evaluation of the interplay between SSCH and TCP.

8. CONCLUSION

We have presented SSCH, a new protocol that extends the benefits of channelization to ad-hoc networks. This protocol is compatible with the IEEE 802.11 standard, and is suitable for a multi-hop environment. SSCH achieves these gains using a novel approach called optimistic synchronization. We expect this approach to be useful in additional settings beyond channel hopping.

We have shown through extensive simulation that SSCH yields significant capacity improvement in a variety of single-hop and multi-hop wireless scenarios. In the future, we look forward to exploring SSCH in more detail using an implementation over actual hardware.

Acknowledgment

The authors would like to thank Ken Birman for his insightful comments on early drafts of this paper.

9. REFERENCES

[1] Bay Area Wireless Users Group, http://www.bawug.org.
[2] Engim, http://www.engim.com/.
[3] Maxim 2.4GHz 802.11b Zero-IF Transceivers. http://pdfserv.maxim-ic.com/en/ds/MAX2820-MAX2821.pdf.
[4] MIT RoofNet, http://www.pdos.lcs.mit.edu/roofnet/.
[5] QualNet, http://www.qualnet.com/.
[6] Seattle Wireless, http://www.seattlewireless.net/.
[7] Tracking Advances in VCO Technology, http://pdfserv.maxim-ic.com/en/an/AN1768.pdf.
[8] IEEE 802.11b/D3.0, Wireless LAN Medium Access Control(MAC) and Physical (PHY) Layer Specification: High Speed Physical Layer Extensions in the 2.4 GHz Band, 1999.
[9] A. Adya, P. Bahl, J. Padhye, A. Wolman, and L. Zhou. A Multi-Radio Unification Protocol for IEEE 802.11 Wireless Networks. In *IEEE International Conference on Broadband Networks (Broadnets) 2004*.
[10] R. Chandra, P. Bahl, and P. Bahl. MultiNet: Connecting to Multiple IEEE 802.11 Networks Using a Single Wireless Card. In *IEEE Infocom 2004*.
[11] I. Chlamtac and A. Farago. Making Transmission Schedules Immune to Topology Changes in Multi-Hop Packet Radio Networks. *IEEE/ACM Transactions on Networking*, 2(1):23–29, February 1994.

[12] I. Chlamtac and A. Farago. Time-Spread Multiple-Access (TSMA) Protocols for Multihop Mobile Radio Networks. *IEEE/ACM Transactions on Networking*, 5(6):804–812, December 1997.

[13] I. Chlamtac, C. Petrioli, and J. Redi. Energy-Conserving Access Protocols for Identification Networks. *IEEE/ACM Transactions on Networking*, 7(1):51–61, February 1999.

[14] D. D. Couto, D. Aguayo, J. Bicket, and R. Morris. A High-Throughput Path Metric for Multi-Hop Wireless Routing. In *ACM MobiCom 2003*.

[15] J. Elson, L. Girod, and D. Estrin. Fine-Grained Network Time Synchronization using Reference Broadcast. In *USENIX OSDI 2002*.

[16] F. Fitzek, D. Angelini, G. Mazzini, and M. Zorzi. Design and performance of an enhanced IEEE 802.11 MAC protocol for multihop coverage extension. *IEEE Wireless Communications*, 10(6):30–39, December 2003.

[17] F. Herzel, G. Fischer, and H. Gustat. An Integrated CMOS RF Synthesizer for 802.11a Wireless LAN. *IEEE Journal of Solid-state Circuits*, 18(10), October 2003.

[18] M. Heusse, F. Rousseau, G. Berger-Sabbatel, and A. Duda. Performance Anomaly of 802.11b. In *IEEE Infocom 2003*.

[19] L. Huang and T.-H. Lai. On the scalability of IEEE 802.11 ad hoc networks. In *ACM MobiHoc 2002*.

[20] K. Jain, J. Padhye, V. Padmanabhan, and L. Qiu. Impact of Interference on Multi-hop Wireless Network Performance. In *ACM MobiCom 2003*.

[21] N. Jain and S. R. Das. A Multichannel CSMA MAC Protocol with Receiver-Based Channel Selection for Multihop Wireless Networks. In *IEEE International Conference on Computer Communications and Networks (IC3N) 2001*.

[22] D. Johnson, D. Maltz, and J. Broch. DSR: The Dynamic Source Routing Protocol for Multihop Wireless Ad Hoc Networks. In C. Perkins, editor, *Ad Hoc Networking*, chapter 5, pages 139–172. Addison-Wesley, 2001.

[23] R. Karrer, A. Sabharwal, and E. Knightly. Enabling Large-scale Wireless Broadband: The Case for TAPs. In *ACM HotNets 2003*.

[24] J. Li, Z. J. Haas, M. Sheng, and Y. Chen. Performance Evaluation of Modified IEEE 802.11 MAC for Multi-Channel Multi-Hop Ad Hoc Network. In *IEEE International Conference on Advanced Information Networking and Applications (AINA) 2003*.

[25] Y. Li, H. Wu, D. Perkins, N.-F. Tzeng, and M. Bayoumi. MAC-SCC: Medium Access Control with a Separate Control Channel for Multihop Wireless Networks. In *IEEE International Conference on Distributed Computing Systems (ICDCS) Workshop 2003*.

[26] A. Nasipuri and S. R. Das. Multichannel CSMA with Signal Power-Based Channel Selection for Multihop Wireless Networks. In *IEEE Vehicular Technology Conference (VTC) 2000*.

[27] J. Padhye, R. Draves, and B. Zill. Routing in Multi-radio, Multi-hop Wireless Mesh Networks. In *ACM MobiCom 2004*.

[28] C. Perkins, E. Belding-Royer, and S. Das. Ad hoc On-Demand Distance Vector (AODV) Routing. In *IETF RFC 3561*, July 2003.

[29] R. Rozovsky and P. Kumar. SEEDEX: A MAC Protocol for Ad Hoc Networks. In *ACM MobiHoc 2001*.

[30] E. Shih, P. Bahl, and M. Sinclair. Wake on Wireless: An event driven power saving strategy for battery operated devices. In *ACM MobiCom 2002*.

[31] J. So and N. H. Vaidya. Multi-Channel MAC for Ad Hoc Networks: Handling Multi-Channel Hidden Terminals Using a Single Transceiver. In *ACM MobiHoc 2004*.

[32] A. Tyamaloukas and J. J. Garcia-Luna-Aceves. Channel-Hopping Multiple Access. In *IEEE International Communications Conference (ICC) 2000*.

[33] S.-L. Wu, C.-Y. Lin, Y.-C. Tseng, and J.-P. Sheu. A New Multi-Channel MAC Protocol with On-Demand Channel Assignment for Mobile Ad Hoc Networks. In *International Symposium on Parallel Architectures, Algorithms and Networks (I-SPAN) 2000*.

[34] S. Xu and T. Saadawi. Does the IEEE 802.11 MAC Protocol Work Well in Multihop Wireless Ad Hoc Networks? *IEEE Communications Magazine*, pages 130–137, June 2001.

[35] Z. Tang and J. J. Garcia-Luna-Aceves. Hop-Reservation Multiple Access (HRMA) for Ad-Hoc Networks. In *IEEE Infocom 1999*.