Channel Assignment and Channel Hopping in IEEE 802.11
Operating Channels for 802.11b

Europe (ETSI)

channel 1
2400 2412 2442 2472 2483.5

US (FCC)/Canada (IC)

channel 1
2400 2412 2437 2462 2483.5

channel 1
22 MHz
Operating channels for 802.11a / US U-NII

**5150 MHz**

5180 5200 5220 5240 5260 5280 5300 5320 5350 MHz

16.6 MHz

**5725 MHz**

5745 5765 5785 5805 5825 MHz

16.6 MHz

center frequency = 5000 + 5*channel number [MHz]
SSCH: Slotted Seeded Channel Hopping for Capacity Improvement in IEEE 802.11 Ad-Hoc Wireless Networks

Victor Bahl, Ranveer Chandra, John Dunagan
Questions

• How to take advantage of channelization in multihop networks?

• Challenge:
  - Sender and receiver have to share a channel → all nodes on a multihop path use the same channel
Two Approaches

• Using multiple radios

• Using SSCH
SSCH

- **Goal:** Extend the benefits of channelization to ad-hoc networks

- **SSCH (Slotted Seeded Channel Hopping)**
  - Improve capacity in ad-hoc wireless multi-hop networks
  - Use a single radio
  - Do not use dedicated control channel
  - Do not require changes to 802.11
SSCH - Overview

- SSCH divides the time into equal sized slots and switches each radio across multiple orthogonal channels on the boundary of slots in a distributed manner.

- Main Aspects of SSCH
  - Channel Scheduling
    - Self-computation of tentative schedule
    - Communication of schedules
    - Synchronization with other nodes
  - Packet Scheduling within a slot
SSCH - Desired Properties

• **No Logical Partition:** Ensure all nodes come into contact occasionally so that they can communicate their tentative schedule.

• **Synchronization:** Allow nodes that need to communicate to synchronize.

• **De-synchronization:** Infrequently overlap between nodes with no communication.
Channel Scheduling - Self-Computation

- Each node uses (channel, seed) pairs to represent its tentative schedule for the next slot.
- Seed: [1, number of channels -1]. Initialized randomly.
- Focus on the simple case of using one pair.
- Update Rule:
  \[ \text{new channel} = (\text{old channel} + \text{seed}) \mod (\text{number of channels}) \]

<table>
<thead>
<tr>
<th>Channel Schedule</th>
<th>Seed</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A:</td>
</tr>
<tr>
<td></td>
<td>1 0 2 1 0 2 1 0</td>
</tr>
<tr>
<td></td>
<td>B:</td>
</tr>
<tr>
<td></td>
<td>0 1 2 0 1 2 0 1</td>
</tr>
</tbody>
</table>

Example: 3 channels, 2 seeds
Channel Scheduling – Logical Partition

• Are nodes guaranteed to overlap?
  - Same channel, same seed (always overlap)
  - Same channel, different seed (overlap occasionally)
  - Different channel, different seed (overlap occasionally)

• Special case: Nodes may never overlap if they have the same seeds and different channels

A: Seed = 1

| 0 | 1 | 2 | 0 | 1 | 2 | 0 | 1 | 2 |

B: Seed = 1

| 0 | 1 | 2 | 0 | 1 | 2 | 0 | 1 | 1 |
Channel Scheduling – Solution to Logical Partition

• Parity Slot
  - Every (number of channels) slots, add a parity slot. In parity slot, the channel number is the seed.
  - Do not allow the seed to change until the parity slot

A: Seed = 1

B: Seed = 1
Channel Scheduling - Communication of Schedules

- Each node broadcasts its tentative schedule (represented by the pair) once per slot
Channel Scheduling - Synchronization

- If node B needs to send data to node A, it adjusts its (channel, seed) pair to be the same as A.

<table>
<thead>
<tr>
<th>Seed</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td></td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td></td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

Flow starts

Sync starts upon the parity slot
Channel Scheduling - Channel Congestion

- It is likely various nodes will converge to the same (channel, seed) pair and communicate infrequently after that.
Channel Scheduling – Solution to channel congestion

• De-synchronization
• To identify channel congestion: compare the number of the synchronized nodes and the number of the nodes sending data. De-synchronize when the ratio $\geq 2$.
• To de-synchronize, simply choose a new (channel, seed) pair for each synchronized and non-sending nodes
Channel Scheduling – Synchronizing with multiple nodes

• Examples
  – a sender with multiple receivers
  – a forwarding node in a multi-hop network

• Solution: Use multiple seeds per node
  – Use one seed to synchronize with one node
  – Add a parity slot every cycle (= number of channels * number of seeds). The channel number of the parity slot is the first seed.
  – The first seed is not allowed to change until the parity slot.

Green slots are generated by seed 1
Yellow slots are generated by seed 2

| 1 | 2 | 2 | 1 | 0 | 0 | 1 | 1 | 2 | 2 | 1 | 0 | 0 |
Channel Scheduling - Partial Synchronization

Flow starts

Sync the second seed only
Packet Scheduling - Main Idea

• Send packets to receivers in the same channel and delay sending packets to receivers in other channels
Packet Scheduling - Basic Scheme

• Within a slot, a node transmits packets in a round robin fashion among all flows
• For a single flow, the packet is transmitted in FIFO order
• Failed transmission causes the relevant flow to be inactive for half a slot. An inactive flow does not participate the transmission unless there are no active flows.
Packet Scheduling - Absent Destination

- **Problem:** The destinations are in other channel
- **Solution:** Retransmission
  - Broadcast: 6 transmission
  - Unicast: Until successful or the cycle ends
- **Question:** Can SSCH distinguish
  - Destinations in other channels?
  - Failure because of bad channel condition or node crash
  - Collision
Evaluation

- Simulate in QualNet
- 802.11a, 54Mbps, 13 orthogonal channels
- Slot switch time = 80 μs
- 4 seeds per node, slot duration = 10ms
- UDP flows: CBR flows of 512 bytes sent every 50 μs (enough to saturate the channel)
Evaluation - Throughput (UDP)

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

Figure 11: **Non-disjoint Flows**: The per-flow throughput on increasing the number of flows.
Evaluation – Multi-hop Mobile Networks

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.
Future Work

• Implementation over actual hardware
• Interaction with proactive routing protocols
• Interoperability with non-SSCH nodes
• Interaction with auto-rate adaptation scheme
• Interaction with TCP
• Study power consumption
Distributed Topology Control for Power Efficient Operation in Multihop Wireless Ad Hoc Networks

Roger Wattenhofer, Li Li, Paramvir Bahl, Yi-Min Wang
Evaluation - Broadcast

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.
Introduction and Motivation

• Network lifetime limited by battery power

• Two choices
  - Increase battery power
  - Energy-efficient algorithms
Goal

- **Minimize transmission power while maintaining network connectivity**
  - Fully distributed algorithm
  - Use only local information
  - Simple to execute (feasible for sensors to run)
Cone-based Algorithm

• **Cone-based topology control algorithm**
  - Designed for multihop wireless ad hoc networks in 2-D

• **Phase 1**
  - Neighbor discovery process

• **Phase 2**
  - Redundant edge removal without disconnecting networks
Phase 1

- Each node $u$ beacons with increasing power $p$, starting from min power
  - If node $u$ discovers a new neighbor $v$, put $v$ into $N(u)$
- Stop when for any cone with angle $\alpha$, $u$ has least one neighbor $v$ or $u$ hits max power
- To ensure symmetry
  - If node $u$ puts $v$ in its neighbor set, then node $v$ also puts $u$ in its neighbor set
Phase 2

- Two nodes $v, w$
  - $v, w$ in $N(u)$ and $w$ in $N(v)$
  - $p(u,v) \leq p(u,w)$
  - $p(u,v) + p(v,w) \leq p(u,w)$

- Remove $w$ from $N(U)$ (and $u$ from $N(w)$)
Phase 2 (Cont.)

- Two nodes $v, w$
  - $v, w$ in $N(u)$ and $w$ in $N(v)$
  - $p(u,v) \leq p(u,w)$
  - $p(u,v) + p(v,w) \leq q \cdot p(u,w)$ where $q \geq 1$
- Remove $w$ from $N(U)$ (and $u$ from $N(w)$)
Phase 2 (Cont.)

Which edge should be removed to minimize power usage?
Phase 2 (Cont.)

$u$ transmitting to $v$

$30 < 35$

remove edge $u,v$
Phase 1

- Each node $u$ beacons with growing power $p$
  - If node $u$ discovers a new neighbor $v$, put $v$ into $N(u)$

- Stop when for any cone with angle $\alpha$, $u$ has at least one neighbor $v$ or $u$ hits max power

- Question: what is largest $\alpha$ that preserves network connectivity?
Main Result

• Let $G'$ be the connectivity graph when each node uses max power
• Let $G$ be the graph after applying phase 1 with $\alpha \leq 2\pi/3$
• If $G'$ is connected $\Rightarrow G$ is connected
Simulation and Results

- 100 nodes
- Placed randomly in 1500 by 1500 rectangle
- Two-ray propagation model for terrestrial communications
Simulation and Results

(e) R&M [3]  (f) Max Power
Simulation and Results

(a) Phase 1 Only $\alpha = 2\pi/3$

(b) Phase 1 Only $\alpha = \pi/2$
Simulation and Results

(c) Cone Based $\alpha = 2\pi/3$  
(d) Cone Based $\alpha = \pi/2$
Simulation and Results

Fig. 5. Network lifetime
Simulation and Results

Fig. 6. Average node degree over time
Simulation and Results

<table>
<thead>
<tr>
<th>Node Degree</th>
<th>Phase 1 Only</th>
<th>Cone Based</th>
<th>R&amp;M [3]</th>
<th>Max Power</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average</td>
<td>$\alpha = \frac{2\pi}{3}$</td>
<td>$\alpha = \frac{\pi}{2}$</td>
<td>$\alpha = \frac{2\pi}{3}$</td>
<td>$\alpha = \frac{\pi}{2}$</td>
</tr>
</tbody>
</table>

**TABLE I**

*Average degree of different topology control algorithms*