Mesh networks are an attractive paradigm for wireless connectivity because they do not require any pre-existing infrastructure and are completely distributed. Despite this promise, however, mesh networks have failed to gain substantial popularity in actual use. Perhaps one of the reasons for this disappointing adoption rate is the scalability limitations faced by the naïve single-channel mesh network implementation. A simple derivation shows that a single-channel mesh network’s per-node throughput grows as $\lambda(n)L \propto \frac{1}{\sqrt{n \log n}}$. Many proposals for improving this bound have been advanced, often based on the idea of taking advantage of the several orthogonal wireless channels the 802.11 standard provides. The main difficulty with multi-channel mesh networks is assigning channels to nodes so as to maximize throughput without disconnecting the network. Ideally, of course, this assignment must be done in an efficient, distributed manner, as a distributed architecture is one of the mesh network’s primary advantages.

One interesting scheme for multi-channel mesh networks is Slotted Seeded Channel Hopping (SSCH), in which nodes cycle through all available channels according to advertised schedules. Schedules differ between nodes, thereby utilizing the orthogonal channels to allow parallel communication between node pairs. SSCH provides substantial performance benefits over single-channel mesh networks. Channel switching imposes overhead on nodes, however, and keeping different nodes synchronized (so that they all switch channels at the same time) could complicate implementation of SSCH. More fundamentally, SSCH adds overhead even during steady-state transmission. An ideal multi-channel mesh network protocol would coordinate channel assignment and then step back, allowing full utilization of the wireless channels. SSCH falls short of this goal, because the switching and clock drift overhead mentioned above persists during steady-state transmission.

---

1 See Problem Set 3 or lecture notes (Yang 2/5/2009 46).
3 Bahl, Chandra, Dunagan 9.
The intuition that a multi-channel mesh network protocol should only be active during the channel assignment phase motivates a new proposal. In this new protocol, one channel (likely a logical channel, see below) is designated a party channel to which all non-transmitting nodes listen. A node \( O \) wishing to transmit to node \( D \) sends an RTS packet on the party channel listing the channels free at \( O \)’s location. If \( D \) is not busy, it responds with a CTS packet listing a channel \( k \) from \( O \)’s RTS such that \( k \) is also free at \( D \) (if no such channel exists, \( O \) and \( D \) cannot communicate until a channel is freed). \( O \) and \( D \) then move to the agreed-upon channel and transmit as normal in an 802.11 network. Because the protocol ceases to operate once \( O \) and \( D \) have completed their RTS/CTS handshake, no overhead is introduced to steady-state transmissions. Moreover, the protocol is completely distributed, requiring neither a central controller nor network-wide clock synchronization (which SSCH does require). The channel assignment handshake is also not significantly more costly than in SSCH, since it essentially consists of only a standard RTS/CTS exchange.

The conceptual sketch of the protocol given above obviously leaves out several crucial details. One of the most pressing questions is how to implement the party channel. Simply devoting an entire 802.11 physical channel to coordination is prohibitively wasteful: 802.11b would lose about 33% of its capacity and 802.11a nearly 8% of its capacity. At the same time, the simplicity of the protocol depends on the party channel being in fact separate from the data channels (thus excluding from consideration RTS/CTS broadcasts or similar techniques). The party channel must therefore be a logical channel. Time-slot partitioning is unattractive because it either requires network-wide clock synchronization (the lack of which is a primary benefit of the proposed protocol) or adds protocol complexity. Hence, the best option is likely code-division multiplexing. CDMA not only avoids clock synchronization but also allows fine-tuning of the bandwidth devoted to the party channel (by adjusting relative chipping sequence lengths). Intuitively, the party channel should consume as little bandwidth as possible, but restricting the party channel bandwidth trades channel coordination time for transmission speed.

My goal with this project is to explore the implementation and efficiency of the proposed protocol, particularly with respect to the party channel issues discussed above. Having done very preliminary proof-of-concept simulations for Problem Set 3, the first step in the project would be to do a full simulation of the protocol (including multi-hop transmissions, which my earlier simulations neglected). This more realistic simulation would hopefully help to validate the prior
simulations’ conclusions about the protocol’s efficiency and give insight about how it performs in more realistic multi-hop scenarios. I would also like to examine through the simulation the consequences of using CDMA to create the party channel and the optimal chipping sequence lengths. Finally, comparing the simulation of the proposed protocol to SSCH could be instructive in testing the aforementioned intuitions about the shortcomings of SSCH. Time permitting, I would also like to try implementing and testing this protocol in real hardware. Useful as simulations often are, they cannot replace real results for evaluating a protocol. Implementing the protocol would also test the claim that the proposed protocol is in fact substantially easier to implement than SSCH (since it does not require clock synchronization or frequent channel hopping).