

Unified Synchronously Scheduled Routing for Static Multi-hop Wireless Networks

Dan Wendlandt
dwendlan@cs.cmu.edu
Carnegie Mellon University

Austin McKinley
amckinle@cmu.edu
Carnegie Mellon University

Abstract

We present Unified Synchronously Scheduled Routing (USSR): a combined pseudo-MAC and Routing layer protocol for commodity 802.11 static multi-hop networks. Considering a community wireless scenario, our approach leverages a centralized network “gateway” to intelligently route traffic through the broadcast wireless mesh. The centralized gateway schedules nodes on non-overlapping channels in order to minimize inter-node interference. This approach increases the overall capacity of the networks and allows for improved multi-hop throughput and fairness for nodes several hops from the gateway compared to traditional routing protocols. We make an argument for why the increased complexity and overhead is reasonable and demonstrate that such an approach can provide improvements in properties central to making multi-hop wireless networks more useful in real-world deployments.

1 Introduction

The commoditization and proliferation of 802.11 wireless radio technology presents an exciting opportunity to vastly decrease the cost of providing Internet access to those currently not connected. The goal of this work is to someday be able to claim that *“In all of history, never before has so much performance been owed by so many, to so little spectrum”*. Consumer-grade 802.11a can exceed 20Mbps over the air, for hundreds of feet, at less than a tenth of a watt of radiated power. Large networks of these radios have several valuable properties: decentralization, path diversity, node failure independence, and ease of deployment. These properties largely stem from the broadcast nature of the wireless medium. However, this same broadcast nature leads to significant issues with interference between individual wireless nodes and external sources of noise. Many proposals have been made to solve this problem at the routing layer, by having nodes measure local interference or react to different traffic loads. One thing all these proposals share is the assumption that the network is decentralized in nature, or equivalently, that

no nodes will see an inordinately large amount of traffic compared to the others.

Our work seeks to exploit a scenario of growing importance that violates this traditional assumption: the community wireless network. In community wireless networks, a single (or very few) node(s) are responsible for providing the rest of the network with some “off-net” service, usually Internet access. In this usage case, all traffic will be destined either to or from these “gateway” nodes, providing a point of central knowledge and control from which to administer the network. Since the network already shares fate with the gateway because of its role in providing Internet access, relying on it for scheduling does little to reduce the resiliency of the network. We chose to exploit this property to build a network with a single omniscient gateway that makes decisions on behalf of other, less-informed nodes.

1.1 Scenario Assumptions

For the community wireless usage case, we make several assumptions about the overall network deployment.

1. **Node Hardware & Deployment** We assume only driver level access to commodity 802.11 hardware and minimal memory and processor requirements at the nodes themselves. A single gateway node requires computational resources comparable to a standard PC to carry out scheduling.

In community wireless scenarios, it is costly and complicated to orchestrate the placement of the participating nodes or to carefully install directed antennas. In most cases, people hosting nodes are not wireless experts, so we assume only omnidirectional antennas.

2. **Off-net Resources** This paper assumes that a wireless mesh is providing some kind of off-network resource to its members, usually Internet access through a wired connection reached via a gateway node. Our work focuses on getting data to and from this gateway and ignores larger issues of provisioning upstream bandwidth on a wired network.

Time-slot	Node A	Node B	Node C	Node D
1	1	1	2	2
2	-	1	1	-
3	1	1	2	2
4	-	1	1	-
5	1	1	1	1

Figure 1: An example schedule with four nodes being scheduled for five time-slots. Each value represents the channel a particular node will be active on during the corresponding time-slot. We use sequential integers to represent non-overlapping channels, as the actual channel numbers depend on the specific 802.11 protocol used. A dash indicates that the node is scheduled to be inactive.

2 USSR Design

In the USSR scheme, the centralized gateway performs “scheduling” based on its current knowledge of traffic flows in the network and recent measurements of both inter-node interference and the general quality and rate of links quality between nodes. In this section we describe our scheduling algorithm and how its design aims to improve multi-hop performance and fairness.

2.1 Scheduling Terminology & Assumptions

For our system we assume time synchronization between nodes to a precision of at most hundreds of microseconds. Considering the geographic proximity of the nodes, recent work suggests such a requirement is reasonable [24][25][26].

In our system, time is divided into “scheduling blocks” of several seconds each, for which the gateway creates and disseminates a master schedule to all nodes. Each scheduling block is divided up into many different “time-slots” of identical duration. Each slot is tens of milliseconds in length. A schedule is a table with a column for each node in the network, a row for each time-slot, and values that assign a node to a given channel for each time-slot. During a given time-slot, a node may also be assigned by the gateway to remain in an “off-state”, during which no transmissions are allowed.

We assume that the gateway is notified about all nodes in the network and that a centralized entity has configured each node with a unique identifier. We also assume a network whose nodes that may malfunction, but will never act maliciously by providing the gateway with false information or ignoring directives from the gateway. Finally, in the descriptions provided here we consider only traffic flows either to or from the gateway. Extending this scheme to handle inter-node traffic not including the gateway is straight-forward, but not discussed in the paper.

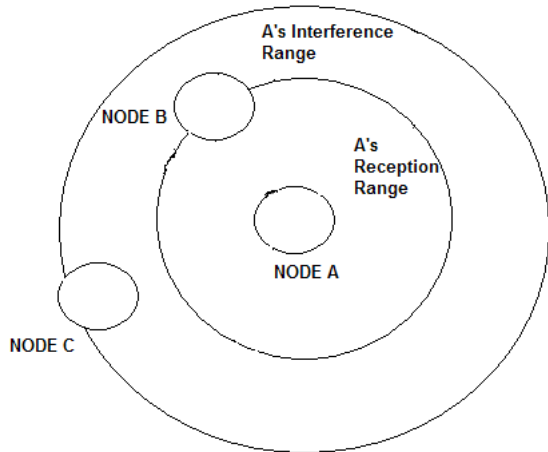


Figure 2: A simple three node topology demonstrating an idealized conception of the reception range as a small circle and the interference range as a larger circle. We use this topology in considering various interference detection mechanisms.

2.2 Inter-node Interference Detection

Fundamental to our goal of avoiding inter-node interference is the detection of which nodes cannot be transmitting while a particular node wishes to receive a packet without any interference. We formalize this notion by saying that all nodes that would interfere with packet reception at a node A if they were to transmit are in node A’s “conflict set”. By choosing this representation, we make the simplifying decision to treat interference as a binary quality, which improves the tractability of our algorithm as well as simplifying development and evaluation.

Because node locations are static, we aim to determine conflict sets during short measurement periods called “conflict detection periods”. The data collected at each node during a conflict detection period would then be reported to the gateway and used for all scheduling blocks until the next conflict detection period occurs. The fidelity of such information over time will be dependant on environmental variables at the deployment, and is beyond the scope of this work. Below, we consider mechanisms for detecting inter-node interference with the goal of maximizing accuracy while minimizing this detection overhead. For all the scenarios described below, we assume all nodes are on a single shared channel during a detection period.

2.2.1 Detection within Reception Range

It is largely trivial for a node to detect potential interference from another node if the first node is able to receive packets transmitted from the interfering node. To collect this information, the network would agree upon a short time period in which no transmissions occur except for interference detection broadcasts. During this period, each node would choose several random times (subject to standard CSMA/CA) to broadcast a small “interference test message” containing its own node id. At the end of the conflict detection period each node sends the gateway a conflict set containing each node ID from which it received a broadcast.

2.2.2 Detection Beyond Reception Range

It is well-known in the networking field, however, that a node can often receive interference capable of corrupting incoming packets from nodes that are beyond its reception range. Previous work requiring inter-node detection [13] has taken the simple approach of assuming that all nodes within two hops of each other may cause interference, and others do not.

While this seems reasonable for an idealized wireless model with uniform and spherical reception and transmission ranges, measurements in outdoor networks exhibit significantly different connectivity patterns than would be expected from such idealized models [1]. As a result, we feel that a measurement approach to interference detection is necessary so that interference is neither over or under-estimated through the use of a simplified metric.

Before discussing several possibilities for interference detection beyond node reception ranges, we recognize that no interference detection scheme is likely to be able to perfectly predict inter-node interference. However, we argue that the sources of interference that have a more significant impact on the ability of our protocol to transmit data should also be those that are easier to detect using either the trivial in-transmission-range detection mechanism, or one of the beyond-range mechanisms described below.

Due to the unknown and unpredictable nature of the surrounding environment with respect to RF interference, for out of range interference detection we again prefer the use of a dedicated, and therefore more controlled, conflict detection period rather than attempting to infer interference during normal data transmission.

1. **Broadcasting with Increased Transmit Power** A technique nearly identical to the in-range detection mechanism described above may be viable if a node normally transmits at a power less than its maximum available transmit power. Since increased power yields larger reception ranges, a node would simply increase its transmit power to a level where its reception range is equivalent to its estimated interference

range. As in Figure 2, Node A would broadcast at an increased power, which would allow both Node B and Node C to successfully receive A’s transmission and add A to their conflict sets.

An evaluation of the roofnet architecture notes that their deployment was sufficiently dense such that the maximum transmission power of 200 mW was deemed necessary [8]. Such densities are also likely in cases where radios for multi-hop routing are co-located with radios for wireless access, since the access points must be dense enough to provide coverage to users with wireless cards, which often have significantly less transmit power than dedicated access points.

This scheme is attractive because of its simplicity, but requires experimental evaluation to determine if it is feasible to estimate a range of interference by the reception range at higher power levels.

2. Monitoring Background Noise Levels

Many, but not all, wireless cards expose readings for both received signal and background noise levels. Each radio has a power threshold for incoming transmissions. Signals coming in above this threshold can be interpreted as packets, but signals received below the threshold (ie: from nodes out of range) are assumed to be external noise. Thus, if a node receives a transmission from another node that is beyond its reception range, it may see a spike in its noise level compared to background noise with no hosts transmitting.

If we consider an extension to the “conflict detection period” described above for in-range interference, nodes could report to the gateway the time intervals when they transmitted broadcasts, and the time intervals when they saw noise spikes. The centralized gateway can correlate noise readings with packet broadcasts to detect interference. For example, Node C would not be able to receive Node A’s transmission, but may see a noise spike during the time while A is transmitting. This correlation could likely be assisted by past interference information and any data about the physical location of the nodes.

This detection scheme has the advantages of relatively low overhead, but requires routers to have chipsets which expose sufficiently frequent and precise noise information to the device driver.

3. Conflict Detection Period Packet-Loss Measurements

A more heavy-weight scheme for interference detection involves a conflict detection period in which brief per-node experiments are carried out to detect packet-loss. During these experiments, which last perhaps a few milliseconds each, Nodes A and B

with a reliable connection begin sending packets to one another at rapid and predetermined rate. During the middle 50% of the experiment, Node C broadcasts packets continuously. If Nodes A or B notice a loss receive rate significantly higher during the middle portion of their experiment, that receiving node can conclude that Node C does in fact interfere with its reception of packets.

While this scheme greatly increases in complexity as the number of nodes grows large, it may be possible to “prune” the total number of measurements that must be run by eliminating pairs with consistent link parameters. One highly desirable property of this mechanism is that rather than inferring the potential loss of data packets from some other metric, this technique measures packet-loss itself and as a result may be easier and less fragile.

We recognize that the first two schemes, while attractive because of their low overhead, require further evaluation. The third scheme, using actual measurements, is similar to previous work, [2] and as a result we believe such an approach is feasible.

2.3 Flows & Selecting Flow Paths

USSR uses a flow-based scheme to schedule traffic on the network. Each flow is an ordered pair of two node id’s, plus an estimated data rate. As discussed below, USSR schedules only long-lived, consistently high bandwidth flows. Thus, we make the assumption that the gateway can both recognize these flows and estimate a lower-bound on their actual bandwidth demand in order to add or remove a flow from its state table as needed.

In our implementation, all traffic coming from a particular node is considered to be a single flow, regardless of how many application-level flows it may represent. We believe this will give us larger flows that are less bursty and easier to schedule. Additionally, this fits with our notion of fairness, as one node with many flows should not be able to dominate other flows on the network.

Each flow has a 32-bit flow-id chosen randomly at boot by the source node. This value is used in place of the node’s IP address in the IP header of packets sent to and from the gateway.

The gateway chooses the best path for each flow, using the estimated transmission time (ETT) metric developed and improved in [3][15]. At a high level, this metric provides the ability to find the links least impacted by external sources of interference. The overhead of the algorithm includes regular probe broadcasts to estimate loss between all pairs of nodes. These probe loss rates are then used to calculate the expected number of retransmissions required to cross a single-hop path. The expected number of transmissions can be combined with the data-rate of the link to calculate the expected time of transmission for a single

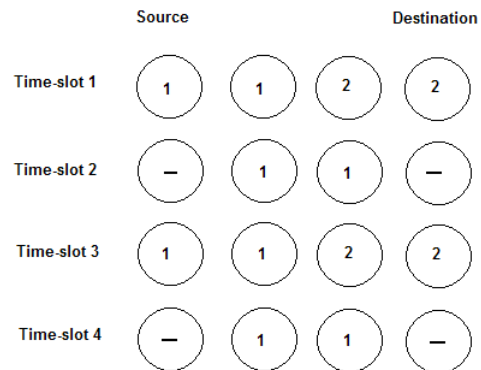


Figure 3: A simple 4 node chain topology, scheduled for four sequential time-slots in a simple flip-flop that is repeated twice.

hop. The ETT for an path is then the summation of the ETT metrics of each hop in the path.

Unlike the use of the ETT metric in past work, the hop-by-hop values are reported to the gateway and used in a centralized routing calculation. The best paths for each flow-id are included in the schedules transmitted to each node, and are used there as described in Section 2.7.

2.4 Centralized Scheduling Algorithm

At the heart of the USSR scheme is the scheduling algorithm for assigning nodes to channels within each time-slot in order to achieve our goals of high multi-hop throughput and fairness for flows at different distances from the gateway. We first describe the notion of a “flip-flop path”, a basic building block in our mechanism, and then detail the iterative algorithm used to make assignments for each time-slot.

2.4.1 Flip-Flop Paths

Many previous schemes [12] [15] that considered the use of routing over multiple channels in mesh networks assume multiple radios at each node. This simplifies some issues with path selection, but adds a significant expense to the network, since a radio is a significant cost of a node.

Limiting ourselves to a single radio limits our ability to maintain end-to-end path dynamics. To understand this, consider a simple linear path of four hops as in Figure 3, with traffic flowing from node A to node D. While node A is transmitting to node B, node C cannot be transmitting to node D on the same channel, without conflicting with B. If we choose to avoid this problem by having C transmit on a different channel than B receives on, this means that

B and C will not be able to communicate packets to each other without switching channels.

Two factors complicate this issue. First is the issue of rendezvous: how do nodes B and C know which channel to switch to in order to communicate? Having a channel schedule at the resolution of individual packet transmissions is clearly both excessive and inefficient. Secondly, there is time-cost associated with switching a radio from one channel to another. On commodity wireless cards, a recent analysis of common chipsets benchmarked the cost of a channel switch at anywhere between 19 milliseconds for the Intersil Prism2 chipset to 5 milliseconds for the Atheros 512 chipset [21]. While these values are extremely high, an already available chipset [22] has a channel switch time of 150 microseconds and further research [23] demonstrates that this cost can be as low as 80 microseconds if the manufacturer chooses to optimize for fast switching.

Since per-packet channel switching seems unreasonable, we instead developed the notion of scheduling a pair of related flip-flop paths during different time-slots. For the example topology in Figure 3, during one time-slot nodes A and B would be on one channel, and nodes C and D would be on another. In a following time-slot, nodes B and C would be on the same channel while nodes A and D are silent. These flip-flop paths aim to emulate an end-to-end path, but without the negative effects of interference between nodes on the path.

2.4.2 Iterative Scheduling of Time-slots

We now consider the actual process of channel assignment for each time-slot. The gateway considers each time-slot individually in an iterative fashion in a way that schedules flip-flop paths for each flow with performance and fairness goals in mind.

For each time-slot, the gateway creates a randomly-ordered list of all flows, called the “flow constraining order,” which determines the priority of flows in the channel assignment process. While determining the optimally fair set of constraining orders is NP-complete [27], using a random ordering provides a simple mechanism to avoid consistent use of a pathological ordering and provides a probabilistic expectation of fairness.

Starting with the first flow in the constraining order, the gateway makes node channel assignments that provide a flip-flop path consistent with the best path calculated for that flow. When looking at subsequent flows, the gateway makes decisions according to the best-path calculated for these flows, but *constrained* by the fact that these decisions cannot interfere with any prior decisions made during this time-slot. This means two things:

1. Since all nodes have only one radio, each node can only be assigned to one channel during a time-slot

2. A node cannot be on the same channel as another node which contains that node in its conflict set.

As a result of these limitations, some flows will not be able to schedule an entire flip-flop path, but rather will only schedule some subset of the hop-pairs that collectively form the flip-flop path.

Note that many different flip-flop paths could exist for a single calculated best path. Furthermore, choosing one hop-pair in a flip-flop path can preclude other hop pairs, since each node can only be part of one hop-pair per time-slot. As a result, we use a simple priority scheme to make sure that a node always makes its hop-pair decisions using a scheme that prioritizes pairs that have been assigned less frequently. This heuristic recognizes that the overall throughput of a flow will only be as high as the throughput through its least frequently scheduled link.

As described so far, the scheduling algorithm treats all flows equally. In reality, the gateway wants to be careful to never over-schedule flows, as doing so leads to a channel configuration that does not efficiently use the limited resources of radio time and low-interference frequency ranges. As a result, we use the traffic demand to estimate how many time-slots a flow should consume in the constraining order, governed by min-max fairness considerations.

Currently the algorithm considers only a single best path for each flow. However, it is reasonable to extend the algorithm to consider multiple paths. In this case, flows later in the constraining order that have many or all of the nodes in their best path already constrained could potentially pick alternate paths or partial alternate paths to the destination. The key requirement for such an approach would be an accurate heuristic to calculate the value provided to a flow by different sets of hop-pair assignments.

2.5 Hybrid Scheduling

While long and persistent flows can be scheduled using the mechanism described above, it is clear that not all traffic on a mesh network can be scheduled. Firstly, there is a bootstrapping problem of informing the gateway about new flows. Secondly, data collected on current static mesh networks show that while the total number of bytes transmitted is dominated by large flows from applications such as Bittorrent, most connections are actually small and short-lived web flows [7]. It makes little sense to attempt to schedule such flows because of their unpredictable data-rate and duration.

As a result, the USSR scheme uses a form of hybrid scheduling where a portion of the total network time is dedicated to transmitting scheduled traffic on multiple non-interfering channels, and another portion of the time requires all nodes to be on the same channel and forward non-scheduled traffic using the ETT routing metric to pro-

vide a path to the gateway. We call these two types of time “scheduled time” and “common time,” respectively.

During common time, nodes prioritize the transmission of non-scheduled traffic. Realistically, short periods of common time would be mixed into the schedule at predictable intervals so that non-scheduled flows are not adversely affected by the latency of waiting for the next period of common time. In this way, our approach aims to use the performance of a single channel protocol as a baseline, and improve upon it by explicitly scheduling large flows.

2.6 Information Dissemination

The discussions above assumed a mechanism to communicate several different types of information between the nodes and the gateway. Such information includes the transmission schedules calculated by the gateway, default routing paths for forwarding during common time, and link metrics resulting from the ETT broadcast probes. These different types of information have differing properties as to their sensitivity to delay and loss. As a result, we consider different use cases for each technique.

1. Piggy-backing Information on Data Packets

One straight-forward method for transferring information is to piggy-back it on data packets transported across the network. Using data packets means that packets will be reliably retransmitted until destination nodes receive them, thereby giving an assurance that data will be received and within an unbounded, but likely very short time-span.

The cost of appending a relatively small amount of additional information to a packet is small, and such an approach is commonly used in wireless protocols including the Srcr protocol used in roofnet. While such a process is ideal for nodes that are already either sending or relaying traffic to and from the gateway, nodes not on such paths will either require explicit messages sent during common time or will have to deal without data communication.

2. Inserting Information in ETT Broadcast Probe Packets

Another option is to include information in the probes that are periodically broadcast between nodes to detect link quality. In order to more accurately estimate actual loss rates for arbitrarily large packets, these probe packets often have bogus data portions to artificially increase packet size. However, since these packets are not reliably transmitted, they do not contain actual network data, and as a result we can use this space to communicate information that does not have strict time or reliability requirements. While this approach does a poor job of communicating information to and from nodes that have only

very poorly connected paths to the gateway, such nodes are also likely to be less important to the overall operation of the network, potentially making such a trade-off acceptable.

For our design, we use piggy-backing to disseminate schedules. Since this information has strict time requirements and most nodes that need to be scheduled are regularly forwarding packets to and from the gateway anyway. The gateway may need to send explicit schedule update messages to nodes in two circumstances. First, if a node is on a path but the gateway has no packets to send to or through that node. Secondly, if a node is in a schedule for the first time, due to a new flow or a new best-path for an existing flow. Both situations are likely to be rare, so we believe this overhead will be small.

We believe the two other types of information being disseminated, routing information for common time packets and link quality statistics, fall into a category that could be labeled “best effort” as the consequences of untimely delivery are significantly less damaging. As a result, we believe transmitting this information in a hop-by-hop fashion via the ETT broadcast probes will be sufficient, which introduces essentially no additional overhead.

For transferring default routing information, each node simply includes in its broadcast probe its best routing metric for reaching the gateway. To calculate its default path, a node considers all metrics received from its neighbors, as well as the quality of its links to those neighbors and decides on a default next hop and uses this new metric in its broadcasts. This approach is essentially a simplified notion of the Bellman-Ford algorithm, with the gateway as the only possible destination.

For transferring link quality information, each node keeps a cache of all link quality information it has seen within a recent time-period, and for each broadcast it randomly picks node pairs and the associated link quality metrics to include in its broadcast probe. The number of metrics included depends on the desired size of the broadcast packet, but since broadcasts are more likely to be received over high quality links this means that the gateway has more recent information about the higher quality paths. This is important since it is hop-pairs on these paths that are most likely to be included in best paths chosen by the gateway.

2.7 Packet Forwarding

The final design topic to discuss is the actual procedure by which each node uses the schedule to make packet forwarding decisions. The actions of a routing node are governed by the schedule and flow-id best-path information

sent by the gateway, as well as an internal timer that directs it when to move to the next time-slot.

Nodes keep per-flow-id queues and another queue for all traffic not part of a scheduled flow. Nodes also have a high priority queue for the transmission of any protocol specific messages.

Since only one best path exists from each node the gateway, there will only every be one next hop per time-slot. As a result, upon receiving the best paths from the gateway at the beginning of a scheduling block, each node parses the paths to create routing entries of the form (flow-id, next-hop) by searching through each flow-id's best path for its own node id, and noting the next-hop if its own id is found.

Before the beginning of each new time-slot, a node must figure out what flows it will be forwarding in the subsequent time-slot and what node will be the next hop. This is a simple matter of looking at the schedule to identify what channel it will be transmitting during the next time-slot and identifying what other nodes are also on the same channel during the time-slot. Because of the constraint guarantees of the scheduling algorithm, only one of these nodes can be within reception range of the node, so finding the next hop is as simple as choosing whatever node is both on the same channel and is the next hop for any routing entry. For each routing entry with this node as the next hop, the node sets the corresponding packet queues as active for the subsequent time-slot.

Note that all flows using this next hop are permitted to send, subject to weighted fair-queuing. This is useful because it increases the chances that the node will have data it needs to send to that next hop for the duration of the time-slot. Utilization is further improved by the fact that both sides of a hop-pair can transmit to the other during a time-slot. Since only two nodes are communicating, we believe it would be possible to replace RTS/CTS with a bit set in the header of the ACK message that indicates that the receiver would like to send a packet and that the sender should not immediately send another packet. We leave this optimization to future work.

Finally, due to the need for strict control over the MAC layer, USSR sends only a single packet to the MAC layer at a time, and instructs the MAC layer to return an error after a single failed packet transmission. This gives the routing layer maximum control over MAC layer retransmissions.

3 NS-2 Implementation of USSR

In this section we describe a simplified implementation of USSR within the network simulator ns-2 and the limitations this platform imposes on the the conclusions drawn from our evaluation in Section 4.

We implemented a simplified version of our USSR protocol in a modified version of ns-2.28, which includes a

port of the original CMU Monarch wireless extensions. We predominantly used standard ns-2 wireless configurations, slightly modified to allow channel switching for nodes.

3.1 Simplifications & Limitations

Due to time constraints, our NS-2 implementation of the USSR protocol included several simplifications compared to the design described in Section 3. Below we briefly discuss these simplifications as well as limitations inherent to the use of ns-2.

1. No Schedule Distribution Overhead

In our simulations, we did not deal with the distribution of schedules and best path information over the network. This simplified the development of the protocol, but also means that the overhead of piggybacking this information on data packets is not represented in simulations.

To develop a back-of-the-envelope calculation for the size of a schedule, we consider a network with 20 scheduled flows (recall that short flows are not scheduled) and 40 different nodes involved in initiating or forwarding these flows. If we assume values of 4 bits/channel-id, 16 bits/node-id, and 32 bits/flow-id we get a schedule of approximately 6.4Kbits and a best-path list of size 3.8Kbits, which must be disseminated every few seconds. We believe that such overhead is already reasonable, but point out that both entities can likely be significantly compressed due to the inherent replication in scheduling and paths, or by the use of delta's from the previous information sent to nodes. As a result, we do not believe adding the additional overhead would significantly change our results.

2. Explicit Flow Notification

Our implementation also does not automatically detect flows, or decide when a flow should change from using only common time to being scheduled, or vice versa. Instead, we assume explicit notification of the gateway with information both about when the flow is starting and its rate.

3. Uniform Sized Flows of Constant Bit-rate Traffic

In our initial evaluation of USSR, we also sought to simplify the traffic load in order to make the analysis of our protocol more straightforward. As a result, we have only one standard flow size that can be scheduled. To accommodate this limitation, all of our traffic is constant bit-rate, and best effort, sent at a rate sufficient to saturate the wireless links. In this, we chose to explore the theoretical maximum performance of USSR as a layer 2 and 3 protocol instead of optimizing for transport layer throughput.

4. Simplified Interference Model

Finally, it is extremely difficult for a simulator to provide an interference model that is similar to noisy and unpredictable real-world environments. Even if ns-2 was capable of simulating such environments, it would still be unclear what types of environments to simulate, as every deployment is different. We note that ns-2 provides a more predictable interference environment than is likely to be available in the real-world, and for that reason we consider our evaluation to be only a first step.

4 Evaluation

With USSR, we hope to address two major problem areas: **Multihop Throughput and Fairness**. To evaluate performance, we consider three ns-2 topologies: straight lines of up to twenty nodes in length for measuring throughput and fairness, a small cluster of nodes for evaluating single hop contention, and three parallel lines of nodes for evaluating path interference. For all our experiments, our performance metric is packets successfully received at the gateway. These numbers are presented without any interpretation. Because our results are entirely simulator based, we chose to present only relative comparisons between USSR and other wireless protocols implemented in ns-2.

Unless otherwise noted, all simulations used identical flows of constant bit-rate, UDP traffic, with a packet size of 282 bytes, at a rate exceeding that needed to saturate the link. All simulations begin at 0.0 seconds, and end after at most 30 seconds.

There are many variables that affect the performance characteristics of USSR. Unless otherwise noted, all simulations used a slot time of 80 milliseconds, and 20 slots per scheduling block. Further, we assumed zero hardware latency to switch channels. Before presenting our results, we discuss these assumptions in more detail.

4.1 Experimental Assumptions

We choose reasonable values for the tunable parameters of USSR based on a preliminary evaluation and used them in all experiments. The two values that most affect performance are slot time, and channel switching delay. Increasing the slot time leads to better performance increase, because there are fewer packet drops caused by channel switching, but creates a throughput versus latency trade-off. Since we have not yet optimized the protocol to minimize losses at channel switching times, we choose to run the experiments with a high, but not totally unrealistic slot time of 80 milliseconds. We ran several experiments on a very simple four node topology and varied the slot time to test the change in throughput. The results of these adjustments can be seen in figure 5

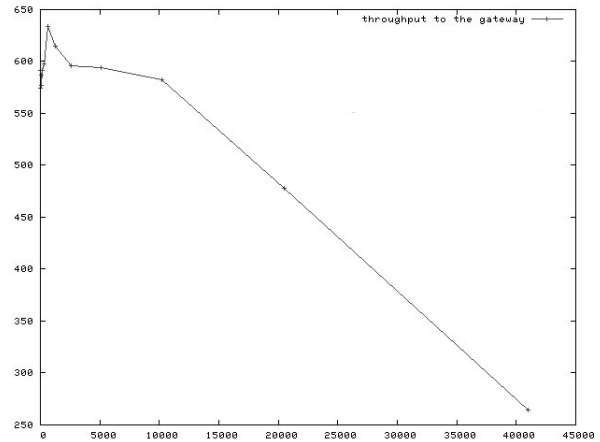


Figure 4: Throughput vs. Channel Switching Time

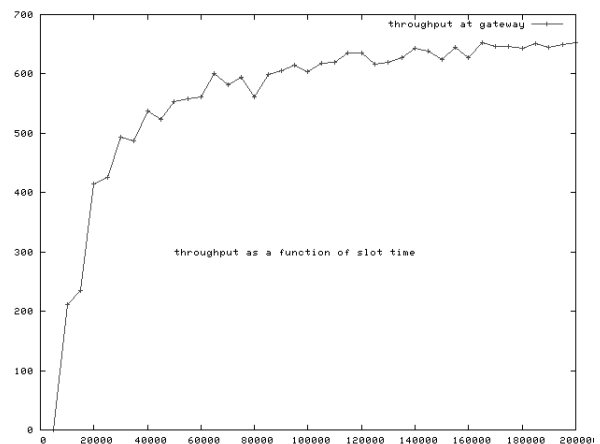


Figure 5: Throughput vs. Slot Time

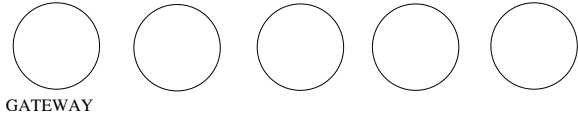


Figure 6: Simple linear topology to test multi-hop throughput. To save space we show a chain with only 5 nodes, though the test uses chains of between 1 and 20.

A similar experiment to explore the impact of the channel-switching time found that a delay of less than 10 milliseconds has almost no significant effect on performance at least for simple topologies. Thus, for the remainder of our evaluation we assume instantaneous channel switches and conduct our experiments with this parameter at zero.

4.2 Multihop Throughput

One of the goals of USSR is to improve the throughput of long multihop paths in wireless networks. This usage case is a critical component of community wireless networks, because it enables a small number of gateways to serve a large number of geographically diverse clients. We hope that with further tuning, a scheduled protocol like USSR could push multihop wireless to speeds significantly higher than existing consumer broadband alternatives.

To evaluate USSR on multihop throughput, we created a simple linear topology in ns-2 with each node within the reception range of its two immediate neighbors. We ran USSR against DSR over 20 simulation runs, each time running a single flow from one end of the topology to the other. Every simulation run added another hop to the path, increasing the length of the path. We measured total traffic received at the gateway over the course of the entire simulation run. This simple linear topology, of which a five node example is shown in Figure 6, demonstrates the flip-flop paths discussed earlier: two nodes at a time pair off on a channel and communicate, with the pairings changing every slot time. Both protocols have RTS/CTS disabled.

Our results show that as the number of hops increases, the performance benefits for centralized scheduling become more pronounced. Where DSR spends valuable transmission time colliding and backing off, USSR makes the most efficient use of valuable spectrum. See figure 7.

This is an important result in the context of community wireless networks. Efficient multihop paths enable community wireless networks to scale without establishing an explicit hierarchy of routing paths. Additionally, better multi-hop paths can help in exploiting path diversity that was previously out of reach. The routing protocol can no longer consider more, longer hop paths that would otherwise be discounted for their heavy bandwidth penalty.

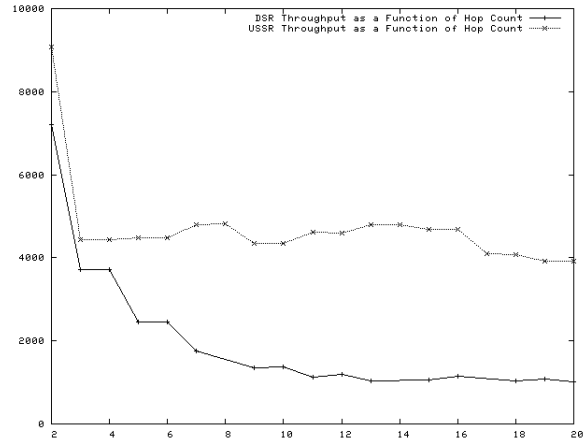


Figure 7: For this experiment, we conducted 20 trials per protocol, increasing the length of the path at each iteration, and measured throughput to the gateway. The x-axis is number of hops in the path.

4.3 Fairness and Quality of Service

In community wireless networks, throughput to "distant" nodes often suffers, since nodes closer to the destination on the same path probabilistically send more of their own packets than those being forwarded for others. Because of this, community wireless networks have fairness problems for paths more than a few hops away from the gateway. Not only is performance poor, but it's highly inconsistent. There is no way of implementing hard guarantees on layer 2 quality of service in contention-based networks. In USSR, we defined "fairness" in terms of flows: two flows of the same rate should have the same throughput to the gateway, regardless of the distance of the flow's source from the gateway. USSR's scheduling can enforce this type of guarantee during scheduling. This is in stark contrast to contention-based single channel networks.

Solving this problem was essential to our project goals. To make the scaling properties of multihop wireless paths useful, we needed to show that we could schedule longer paths to perform as well as their shorter hop counterparts. Our results indicate that we've accomplished exactly that, as seen in Figures 9 and 8.

To evaluate fairness, we used the linear topology discussed above in the multi-hop throughput section, and fixed the number of nodes at 15. We first start a flow 15 hops away from the gateway, and run it for 10 seconds. We then start a second flow, only 7 hops away from the gateway, and on the same path as the first flow. With USSR, the second flow cuts the throughput of the first flow exactly in half. Both flows receive the same throughput at the gateway, despite the drastic differences in hop count. The scheduler balances the resource demands of the distant flow against that of the close flow, and allocates time slots accordingly.

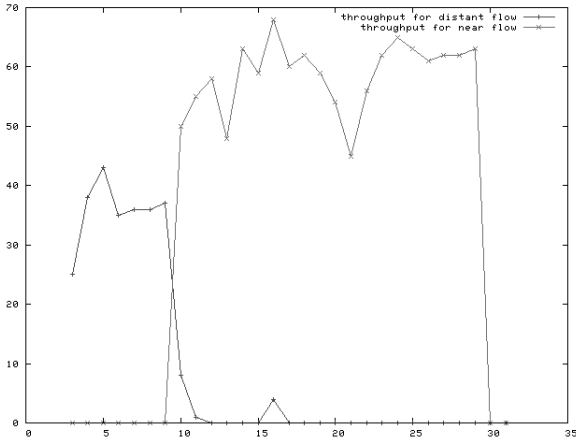


Figure 8: In this graph, we measure the throughput of two flows: one 15 hops from the gateway, and one seven hops from the gateway. With DSR, the near flow starves the distant flow.

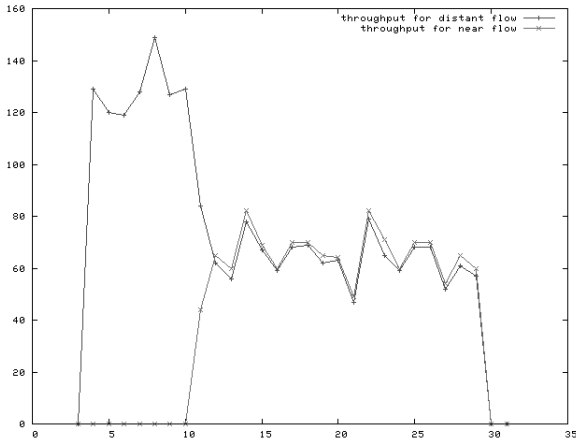


Figure 9: In this graph, we measure the throughput of two flows: one 15 hops from the gateway, and one seven hops from the gateway. With USSR, these two flows get exactly half of the gateway's resources. Note the difference in max values for the Y-axis, as overall throughput is better with USSR.

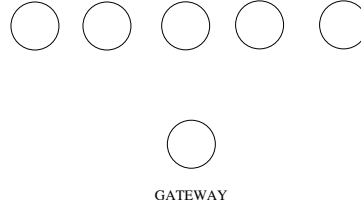


Figure 10: This topology is a very simple single hop scenario. All 6 nodes are within range of the gateway. For our experiments, each nodes starts a flow to the gateway at 0 seconds.

Using DSR, however, the throughput of the first flow drops to near zero once the second is started. Because the flows share a path, the demands of the closer second flow dominate those of the distant flow. We can see that even though cumulative throughput to the gateway is constant for both DSR and USSR, USSR makes far more balanced use of the rest of the network.

The fairness properties of USSR, combined with the multihop throughput improvements discussed above, make a compelling case for USSR in community wireless networks.

4.4 Single hop Contention

We originally envisioned USSR as a protocol suitable for use even in a traditional access point/client environment. We show that in a single hop environment, USSR is capable of performing almost as well as simple CSMA/CA in a single hop environment, despite it's channel switching overhead.

For this set of experiments, we ran USSR against stock 802.11, with RTS/CTS disabled. The topology as seen in figure 10 is a gateway and five nodes, all in the same interference range, to simulate one access point at multiple clients. In the experiment, we start five flows at the same time, one per node, and measure cumulative throughput at the gateway. To ensure link saturation, we increased the packet size to 1600 bytes, and cut the inter-packet interval down to .001 seconds.

As seen in figure 11, both protocols performed closely to the theoretical maximum performance of the topology. However, in our experiments, USSR was outperformed by a small, but non-negligible margin. We attribute this to some inadvertent packet drops during channel switching: if an ns-2 node is in the middle of a transmission during a scheduled channel switch, the packet is dropped, and retransmitted.

In the context of our stated goals, this result is slightly problematic. It is easy to imagine a large USSR topology being reduced to a pathological case where the most significant bottleneck is at the gateway's radio, and performance suffers network wide. We believe that although USSR has strong scaling properties, its performance may

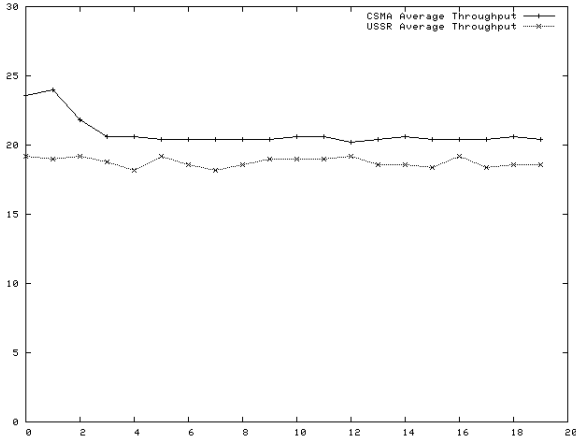


Figure 11: We measured the throughput of each flow in our single-hop topology, and averaged them. In this usage case, simple CSMA/CA outperformed USSR by a small margin.

still limited by maximum throughput at the gateway. For this reason, we discuss the use of multi-radio nodes, particularly the gateway, in Section 6.

4.5 Conflicting Parallel Paths

The ideal performance case for USSR is multiple flows over multiple paths. To simulate this in ns-2, we took three copies of our linear topology and stacked them on top of each other, with flows going from one end of the line to the other, just like before.

This topology was chosen as a somewhat artificial demonstration of the ideal case for a USSR-like scheme. We stagger the start times of the flows: one begins at 0, another at 5, and another at 15. We then measured the throughput of each flow to its respective gateway. In this experiment, we expected to significantly outperform single channel CSMA/CA, and the results show exactly that in figure 12. USSR discovers the conflict sets accurately, and schedules each "line" of nodes with alternating pairs of channels to avoid interference and collisions. Utilizing only two channels, we completely cover the topology and saturate the links of all the nodes, while minimizing collisions and packet drops. This results in a substantial performance boost, on the order of 400% or more over DSR. We believe that this result shows the real strength of USSR. Multiplexing network traffic over multiple channels allows the wireless network to emulate point-to-point links, while maintaining the path flexibility of a broadcast medium.

This result combines the throughput and fairness results already discussed. USSR keeps all three flows at comparable rates for the duration of the simulation, and maintains the same high initial throughput even after adding more flows. At no point in the simulation does DSR ap-

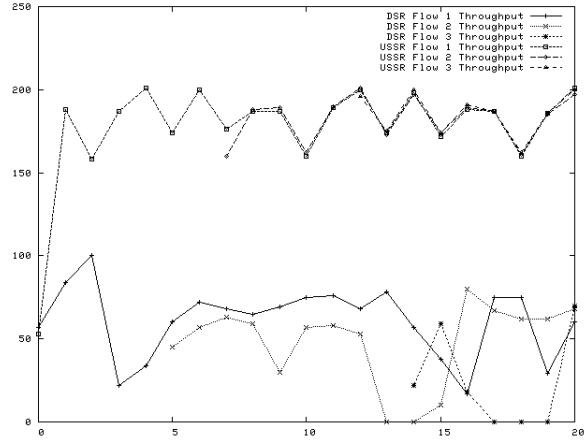


Figure 12: This shows the throughput over time of six different flows across two different simulation runs, one under DSR and one under USSR. In both runs, the flows were started at identical times: 0, 5, and 15 seconds. Each flow crosses the entire length of the topology to reach its distant neighbor. The top three lines are USSR, the bottom three are DSR.

proach the performance of USSR. This is a significant result, as it indicates that USSR’s performance benefits can be strongly felt at realistic hop counts. This parallel topology was run with a hop count of 6 nodes. Throughput numbers like these further indicate that with a scheduled protocol, community wireless networks with sufficient density could realize substantial scalability improvements.

5 Related Work

With a recent surge in interest that multi-hop 802.11 networks can provide a cost effective solution for last-mile Internet access, announcements of community wireless test-beds [8][28] and full-scale deployments [4][6] [5] [14] have grown increasingly common. As a result, routing protocols aimed at providing high throughput on static topologies with potentially significant variance in link quality are proposed in [3] [7][15][16], including the ETT routing metric USSR uses in its scheduling algorithm.

Research has also explored the ability of routing protocols or MAC layers to exploit the frequency diversity of 802.11’s non-overlapping channels. Previous MAC-layer systems [19][18][29] operate in a distributed fashion, with a heavy focus on how nodes “rendezvous” on a particular channel with a node they have packets to send.

Work on an Overlay Mac Layer (OML) [13], like our work, attempts to schedule node transmission into time-slots and gives more time-slots to nodes that must carry multi-hop traffic in addition to packets they themselves

originate. However, OML uses only a single channel and therefore does not have to deal with the channel rendezvous issue. Additionally, OML does not specify a particular routing policy and assumes nodes interfere with all other nodes within k hops, where k is a small integer.

Building on previous work at Stony Brook [27], Hyacinth system [12], in a fashion similar to USSR, seeks to exploit knowledge about traffic load pass traffic over a static mesh topology on multiple channels. However, Hyacinth assumes multiple multi-radios per node and does not actually route traffic in an interference aware way. Likewise, the system runs in a distributed manner and does not seek to enforce any fairness guarantees.

The Transit Access Point (TAP) [17] project first developed a notion of fairness that is similar to the model used by USSR, stating that multi-hop routing entities should receive approximately the same rate of service to an Internet gateway regardless of the number of hops it takes to reach the gateway.

A significant amount of work has taken a graph-theoretic approach to considering interference between nodes in a wireless network, including [11][20]. Work out of Microsoft in this area [9] showed that an omniscient scheduler with knowledge about traffic load and interference conflicts should allow for significant throughput gains on a network.

Finally, in the area of real-world measurements of interference in static mesh networks, [2] considered the estimation of inter-node interference in an indoor wireless network. while [1] looked at the overall interference environment in the outdoor Roofnet network.

6 Future Work

Our results with USSR are quite preliminary in nature, and previous discussions of both the design and evaluation have outline areas for improvement and further study. Additionally, we recognize that implementation in a real-world testbed is a critical step in determining whether our approach for both interference detection and scheduling is feasible for real deployments.

Beyond these obvious next steps, in this final section, we want to briefly touch on a few areas that we believe could yield interesting results to build upon our and others' work in this field.

Firstly, we would like to explore the trade-offs between single-radio nodes and multi-radio nodes. We believe that high contention points such as the gateway could clearly benefit from having multiple radios, and our centralized scheduling provides a simple mechanism for rendezvous. Additionally, having flip-flop paths on long chains produces a latency that depending on slot-time may lead to undesirable latency. Nodes upgraded with multiple radios instantly double their throughput, because each radio can be placed on a different channel. When communicat-

ing with another multiple radio node, the radios could be paired off for full duplex connectivity.

Another topic we would like to explore related to our work is how adjusting or scheduling the transmit power of the nodes may be useful in reducing interference and improving throughput. Likewise would would like to explore algorithms for an Internet gateway to predict network demand based on past usage. Finally, to replace our purely greedy scheduling, we would like to explore the potential for an auction-based scheme run at the gateway to "price" paths based on their demand by all flows in the network. Our centralized scheme makes such an approach significantly easier and we feel that more complicated topologies could benefit significantly from this approach.

7 Conclusion

Our goal in creating the USSR protocol was to explore how the network knowledge of a centralized gateway entity could be used to improve multi-hop throughput and fairness in a static wireless network. While our work remains preliminary, we believe we have outlined and offered initial algorithms for many of the key issue in performing centralized scheduling in a static multi-hop environment. Furthermore, our results on relatively simple topologies indicate that our scheme is successful in improving throughput on long multi-hop paths and offers substantially improved inter-flow fairness compared to traditional wireless routing protocols. Additionally, the use of multiple channels offers a significant boost in overall network capacity when multiple paths exist within interference range of the eachother. We argue that our assumptions, while significantly demanding with respect to previous routing protocols, offer benefits significant enough to merit further study on how such a centrally scheduled protocol could be implemented on a real-world network. As deployments of static multi-hop networks continue to become more popular as a technology for last-mile Internet access, we believe our work provides a promising direction to providing high throughput and fairness within these environments.

References

- [1] D. Aguago, J. Bicket, S. Biswas, G. Judd, and R. Morris. Link-level Measurements from an 802.11 Mesh Network. In SIGCOMM, 2004.
- [2] J. Padhye, S. Agarwal, V. Padmanabhan, L. Qiu, A. Rao, B. Zill . Estimation of Link Interference in Static Multi-hop Wireless Networks. In Internet Measurement Conference (IMC), 2005.

- [3] D. De Couto, D. Aguayo, J. Bicket, and R. Morris. High-throughput Path Metric for Multi-hop Wireless Routing. In MOBICOM, 2003.
- [4] Google Blog: Wi-Fi in Mountain View. <http://googleblog.blogspot.com/2005/11/wi-fi-in-mountain-view.html>.
- [5] Washingtonpost.com : New Orlean's New Connection . <http://www.washingtonpost.com/wp-dyn/content/article/2005/11/28/AR2005112801773.html>
- [6] USAToday.com : Biggest Wi-Fi cloud is in rural Oregon. http://www.usatoday.com/tech/products/services/2005-10-16-oregon-wi-fi_x.htm
- [7] J. Bicket, D. Aguayo, S. Biswas, and R. Morris. Architecture and Evaluation of an Unplanned 802.11b Mesh Network. In MOBICOM, 2005.
- [8] MIT roofnet. <http://www.pdos.lcs.mit.edu/roofnet>.
- [9] K. Jain, J. Padhye, V. Padmanabhan, and L. Qiu. Impact of Interference on Multi-hop Wireless Network Performance. In MOBICOM, 2003.
- [10] M. Kodialam and T. Nandagopal. Characterizing Achievable Rates in Multi-hop Wireless Networks: The Joint Routing and Scheduling Problem. In MOBICOM, 2003.
- [11] M. Marina and S. Das. A Topology Control Approach for Utilizing Multiple Channels in Multi-Radio Wireless Mesh Networks.
- [12] A. Raniwala and T. Chiueh. Architecture and Algorithms for an IEEE 802.11-Based Multi-Channel Wireless Mesh Network.
- [13] A. Rao and I. Stoica. An Overlay MAC Layer for 802.11 Networks.
- [14] R. van Drunen, J. Koolhaas, H. Schuurmans, and M. Vijn. Building a Wireless Community Network in the Netherlands.
- [15] R. Draves, J. Padhye, and B. Zill. Routing in Multi-Radio, Multi-Hop Wireless Mesh Networks. In MOBICOM, 2004.
- [16] S. Biswas, and R. Morris. ExOR: Opportunistic Multi-Hop Routing for Wireless Networks. In SIGCOMM, 2005.
- [17] V. Gamberoza, B. Sadeghi, and E. Knightly. End-to-End Performance and Fairness in Multihop Wireless Backhaul Networks. In MOBICOM, 2004.
- [18] P. Porwal and M. Papadopouli. On-demand Channel Switching for Multi-channel Wireless MAC protocols.
- [19] P. Bahl, R. Chandra, and J. Dunagan. SSCH: Slotted Seeded Channel Hopping for Capacity Improvements in IEEE 802.11 Ad-Hoc Wireless Networks. In MOBICOM, 2004.
- [20] Y. Xue, B. Li, K. Nahrstedt. Optimal Resource Allocation in Wireless Ad Hoc Networks: A Price-based Approach.
- [21] I. Ramani and S. Savage. SyncScan: Practical Fast Handoff for 802.11 Infrastructure Networks
- [22] Maxim 2.4GHz 802.11b Zero-IF Transceivers. <http://pdfserv.maxim-ic.com/en/ds/MAX2820-MAX2821.pdf>.
- [23] 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), Oct. 2003.
- [24] J. Elson and D. Estrin. Time Synchronization for Wireless Sensor Networks. In PDPS Workshop on Parallel and Distributed Computing Issues in Wireless Networkings and Mobile Computing, pages 186-186.
- [25] K. Romer. Time synchronization in ad hoc networks. In Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing, pages 173-182. ACM Press, 2001.
- [26] U. Schmid and K. Schossmaier. Interval-based clock synchronization. Real-Time Systems, 12(2): 173-228, 1997.
- [27] A. Raniwala, K. Gopalan, T. Chiueh. Centralized Channel Assignment and Routing algorithms for Multi-channel Wireless Mesh Networks. ACM Mobile Computing & Comm Review (MC2R), April 2004.
- [28] J. Camp, E. Knightly, and W. Reed. Developing and Deploying Multihop Wireless Networks for Low-Incoming Communities.
- [29] J. 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.