Energy-Efficient Dynamic Source Routing in Ad-Hoc Wireless Networks

Elie Krevat and Arian Shahdadi
Massachusetts Institute of Technology
krevat, shahdadi@mit.edu

Abstract

We present Span-DSR and DSR-PSM, two power-saving algorithms designed for ad-hoc wireless networks using source routing. Span-DSR reduces power consumption and has desirable fairness properties, while maintaining a connected backbone of nodes to route packets. Span-DSR's power saving protocol increases system lifetime by a factor of two better than standard Dynamic Source Routing (DSR), while maintaining slightly higher loss rates under congestion. DSR-PSM is a simpler and purely reactive approach that also shows considerable power savings in comparison with standard DSR. However, DSR-PSM does not perform as well as Span-DSR with respect to energy efficiency, loss rate or fairness.

1. Introduction

One of the most significant issues in mobile inter-networking is power consumption. Although new hardware devices have larger energy sources and tend to consume less power, it is important that mobile inter-networking protocols and software are aware of power consumption. The goal is to control the hardware resources in a way that minimizes power usage while still providing constant or near-constant network availability.

The wireless network interface is one of the largest sources of power consumption in a mobile device. This is counter to the general usage pattern of wireless devices, where network access is often intermittent. Wireless devices tend to spend a great deal of their time in a network-idle state. One way of addressing this problem is to turn off or put to sleep those network interfaces that are idle for an extended period of time. This approach creates many complications, not the least of which are in terms of routing protocols, which generally assume that every reachable node will be available and ready to route packets if necessary.

The Span algorithm addresses these issues [4]. Span presents a power-saving scheme that allows access to the network over a connected backbone with approximately as much total capacity as possible in the network. Span does this by electing *coordinators* that stay awake for a period of time in order for packets to continue their routes through the network. Other nodes are allowed to sleep if they are not coordinators in order to limit their power consumption. Every node periodically broadcasts HELLO messages containing state information that allows the network to elect new coordinators if the need arises. Since the state information is a list of coordinators and neighboring nodes, Span has a similar stored state structure as routing table based protocols, and works well under such schemes.

Although Span can be used with little modification on routing table based protocols, one routing scheme that does not work well under Span is source routing, where route state is maintained primarily in the packet rather than at each host. Source routing does not work effectively with Span in its current form since nodes on a path can suddenly go to sleep and render a source route invalid. Although most implementations of source routing deal with this eventuality, the common ways of recovering routes involve some level of setup, which is detrimental to throughput and efficiency. Under Span, nodes wake up and sleep depending on parameters not related to their involvement in routes, and thus source routes would break frequently in a source routing scheme running Span unmodified.

Why choose source routing at all? Source routing has many advantageous qualities in comparison to other forms of routing, and it would be desirable to make the current implementation of Span work with source routing to take advantage of them. An important feature of source routing is that it is done entirely on-demand, allowing the network to scale well with the number of nodes. Source routes can easily be checked to guarantee loop-free routing behavior, and they require very little stored state at each node (since the state is concentrated in the packets). Perhaps most importantly, source routing is commonly used in current ad-hoc networks, and so adding power-saving features to current source routing implementations would have an immediate impact.

We present two approaches to solving the problem of adding power-saving features to source routing. The first scheme combines the aforementioned Span algorithm with source routing. We refer to this scheme as Span-DSR. The second scheme uses the power-saving mode of 802.11 to create a simple, reactive, power-saving algorithm for the sake of comparison. We refer to this scheme as DSR-PSM. These comparisons are presented as results from simulations under the *ns-2* network simulator. We compare a standard implementation of source routing without power-saving features with these two power-saving schemes in terms of average power savings and throughput. We also discuss our observations on the fairness of the two schemes.

The rest of this paper describes the Span-DSR and DSR-PSM schemes and evaluates their performance. Section 2 discusses related work in the field. Section 3 gives a basic overview of how the two schemes work in algorithmic form. Section 4 presents the implementation of these two schemes, written within the previous implementation of Span on top of an IEEE 802.11b MAC layer in the *ns-2* network simulator [1]. Section 5 presents a performance comparison of the two schemes with the standard CMU implementation of DSR for *ns-2* [2]. Section 6 discusses issues in Span-DSR and outlines future work. Section 7 presents our conclusions.

2. Related Work

The most common implementation of source routing is Dynamic Source Routing [6]. DSR is entirely "on-demand", that is it does not require constant information updates in order to build and main-

tain routes. Its two major services (route discovery and route maintenance) are only invoked when a network node requests them. Span, on the other hand, uses periodic broadcasts to maintain information and elect new coordinators. Nodes may go up or go to sleep irrespective of the routes they are currently servicing. This underlying election process, which currently runs irrespective of routes, would cause many broken source routes and create unnecessary overhead in route recovery.

The current implementation of Span uses geographic routing [7]. The main idea in geographic forwarding is that nodes forward packets toward what they believe to be the location of the destination node. Coupled with a location service (the Grid Location Service in this case), geographic forwarding protocols provide a very scalable and efficient substrate for packet forwarding in an ad-hoc network. Geographic forwarding does suffer from several drawbacks, most notably the phenomenon of "holes". Holes in the topology occur when a node knows of no other node closer to the source than itself, and when this node cannot reach the destination. Such a situation indicates that there is a gap in the geographical distribution of nodes in the topology. Such gaps are difficult to fix, and the scheme described above avoids them by returning an error to the source. Source routing, although somewhat less scalable and efficient than geographical forwarding, is more robust and requires less infrastructure (specifically the infrastructure for determining the position of a node).

One body of work compares several basic ad-hoc routing protocols, including DSR [3]. A major contribution of this research is a basic implementation of DSR for *ns-2*, on which our implementation was based. Their work concluded that DSR has several benefits, mainly a low packet loss rate and relatively low overhead as compared to other major routing protocols (DSDV, AODV and TORA). This work is several years old, but still gives a good baseline for comparison.

The details behind the energy consumption of network interfaces are important for power-saving schemes [5]. This research provides insight into common patterns of use in network interfaces, and gives specific numbers to general observations about power consumption in wireless devices. The authors distinguish between the power consumption of the device in different modes. The results for broadcast and promiscuous mode traffic are enlightening, and validate the optimiziations made by Span. Their specific observations apply to an 802.11 wireless interface, which we use in our simulations.

On-line power aware routing in ad-hoc wireless networks is another way of approaching the power saving problem [8]. This scheme takes particular care to route messages through the network efficiently, minimizing the energy used in communication. Their work is complementary to that of Span, which seeks to minimize energy usage by putting nodes in a sleeping state when they are not needed. Their work also presents a new routing scheme, called zone routing, that takes advantage of their approach to online power saving.

3. Protocol Design

3.1 Design Goals

The fundamental design goals of Span remain intact. These goals are to maintain global connectivity, preserve fairness and conserve energy, all without significant losses to throughput. It is important, however, to note the fundamental difference between the two schemes presented here. Span-DSR is, like Span, a *proactive* protocol, in the sense that it keeps nodes up at all times in order to maintain connectivity. DSR-PSM, however, was designed to be

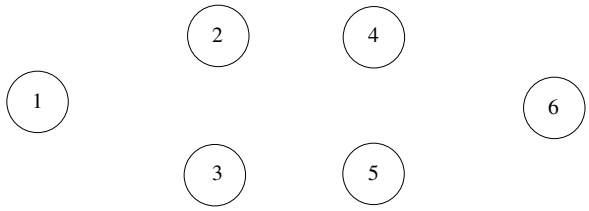

Figure 1: A simple ad-hoc topology example.

reactive, that is, it does not make nodes stay awake unless they are needed in order to provide connectivity. While these are apparently two divergent design goals, we attempt to minimize the proactive overhead of Span while constraining the bandwidth requirements of route maintenance in DSR by taking advantage of Span's connectivity properties.

3.2 Assumptions

Given the nature of a Span network, we made the following assumptions in designing Span-DSR. Since Span is a protocol that is present above the MAC and physical layers, we assumed the existence of a standard link-layer (such as the one in 802.11). All links were considered to be bi-directional with a link-level acknowledgment feature. Nodes also were assumed to have a promiscuous receive mode which can sniff other packets. Finally, we assumed only one homogeneous Span network without the complication of external networks.

In our design we assumed the existence of an 802.11 power-saving MAC layer. One of the features of 802.11 is an ad-hoc power saving mode. In power-saving mode, nodes use periodic beacon messages in order to synchronize. Beacon periods start with an ATIM (ad-hoc traffic indication message) window, during which all nodes must listen for pending traffic signals. Nodes that acknowledge a traffic signal must stay up for the entire beacon period, after which their packets are transmitted. The specification does not allow for data transmission during the ATIM window, reducing overall channel capacity.

Thus, nodes may not be able to immediately send packets. Packets are buffered at the previous hop until the next node receives them, or two beacon periods have passed, whichever comes first. At the end of every beacon period, packets that have been buffered for the two previous periods are dropped.

3.3 Span-DSR Design

In order to meet Span's goals, the topology of coordinators in a non-sparse Span network changes frequently. DSR, however, is most effective in a static network; when a route used by an active connection becomes invalid, the connection is broken and the source must generally broadcast a route request and await a route reply before reestablishing that connection. In our basic design, we chose not to resolve this inconsistency through Span, but rather through the mechanisms of DSR.

3.3.1 Tentative and Withdrawn Coordinators

One important feature of Span is the ability of coordinators to mark themselves as tentative. A tentative coordinator will announce his intention to withdraw to his neighbors, but will remain up until another node announces itself as a replacement coordinator. In the original Span scheme, tentative coordinators continue to route packets until such time as they either withdraw or are forced to clear their tentative status and remain a coordinator. This causes

some problems under DSR. Figure 1 presents a very simple topology under Span-DSR. Assume that node 4 intends to withdraw as a coordinator and announces itself as tentative. Node 4 will continue to forward packets as a tentative coordinator since the election process may take some number of seconds to finish. By allowing a tentative coordinator to route packets normally we avoid many performance complications involved in premature detection of broken routes (since tentative coordinators may very well stay on as coordinators). Assume that at some point in the future, node 5 will announce itself as a coordinator to replace node 4, and node 4 withdraws. Now, say that node 1 sends a packet along this path for node 6 with the original source route that includes node 4. Assuming node 2 is on the route, when this packet reaches node 2, it is clear that this node should send a route error back to the source. The route error mechanism works in this case just as it does in standard DSR. A route error will propagate along the path back to the source, invalidating cache entries.

It is not straightforward, however, how the node which comes before the withdrawn coordinator in the source route (node 2) should handle the data packet. In our original scheme, node 2 dropped all data packets intended for the withdrawn coordinator (node 4). As we will show in our performance summary, this decision led to a significant number of dropped packets. As a fix for this problem, node 2 now continues to forward data packets to node 4 even though node 4 is asleep. The few outstanding packets still in the network with the old source route will take a little longer to propagate through the sleeping node, but should not be dropped. When the source node receives the route error, it will stop sending packets on this route.

If the incoming packet is a route request or route reply, the withdrawn coordinator will drop it. By dropping these route setup packets, the tentative coordinator assures that it will not be involved in future source routes, since it is planning on sleeping and passing its coordinating duties to another node.

If the incoming packet is a route error, the withdrawn coordinator will forward it. If a route error has been received at a tentative node, then some other portion of the route has been broken, and so forwarding the error will invalidate outdated paths.

3.3.2 Coordinator Election

Changing the frequency of coordinator elections was also an important design optimization in order to achieve better performance under DSR. In particular, when a coordinator announces itself, it will tend to stay on as a coordinator for a longer period of time as compared to normal Span with geographic routing. This coordinator window period is based on the flow rate going through the node, as measured by the amount of data received over a constant size time window.

3.4 DSR-PSM Design

Span is by nature a proactive protocol, since it maintains a constantly available network backbone. Source routing schemes do not need this kind of constant availability, since they react to network traffic as needed. For this reason, we believe that source routing protocols can take advantage of the power saving features of MAC layers with very little difficulty.

The Span paper [4] presents several results for optimal function in the 802.11 MAC layer. Specifically, they use a beacon period of 200 ms and an ATIM window size of 40 ms, which we use unaltered in our scheme. These values result in fairly good throughput with a small loss rate. The paper also shows that 802.11 power-saving mode suffers from fairly long packet delivery latency (because of the beacon scheme) and that it does not save much power with routing protocols that rely heavily on broadcasts.

Since DSR has no need for broadcasts, our scheme can take advantage of the features of 802.11. All nodes begin asleep in 802.11 power-saving mode. When a node wishes to communicate with another node, it initiates a route request, which travels along the network normally through the sleeping nodes. This phase of the protocol suffers from long latencies because of 802.11 power-saving mode. We do, however. resolve this problem upon the completion of the route setup phase of DSR.

Once a route request has finished propagating, the sink node wakes up and builds a route reply. This route reply is sent along the discovered route. As sleeping nodes receive the reply, they pass it along if they are on the route and stay in sleep mode. Otherwise they drop the packet.

A node wakes up only when it receives a data packet which includes it in a valid source route. Originally we considered a scheme where nodes wake up upon receiving a route reply containing their address. DSR, however, sends multiple route replies from the sink in order to maintain cached routes along multiple paths. Therefore, this scheme would have caused nodes to stay up unnecessarily.

Nodes that are awake do not stay awake indefinitely. The duration of their activity is governed by a constant timeout in order to handle bursts of traffic. The timeout value is kept as a counter that is reset every time the node receives a packet with itself on the source route.

Note that there is no explicit mechanism for aggregation of routes in this scheme - every node may participate in a source route. If multiple connections try to create routes at approximately the same time, many nodes may stay up unnecessarily to service routes that close neighbors could also service. However, if new connections are initiated on a regular basis over a longer period of time then nodes that are awake should become involved in several data flows. Because they are awake, they will tend to propagate any route requests and replies faster than nodes that are in 802.11 power saving mode.

4. Implementation

An implementation of DSR exists for ns, written by the Monarch project (now joint between CMU and Rice). When implementing our two power-saving schemes as lightweight protocols for the ns-2 network simulator, we built on this DSR code and the reference implementation of Span [4].

4.1 Interaction with the 802.11 MAC layer

Both schemes rely on the presence of an underlying 802.11 MAC layer. The implementation used is that provided by the Monarch extensions, and supports all basic 802.11 MAC layer functionality including power-saving mode (an extension added by networking researchers at MIT). The MAC layer is used to insure the proper delivery of packets. and to flag the higher software layers when transmission fails. When nodes are awake, communication between them is straightforward, and packets are transmitted using the usual MAC layer checks. When nodes are sleeping, their packets are buffered by the sender until a MAC layer acknowledgment is received requesting transmission of the packets. While asleep, nodes still participate in basic 802.11 power-saving mode functions (such as synchronization across beacon periods). The MAC layer is accessed according to the standard Monarch DSR implementation, except that additional calls are made to cause nodes to sleep or wake up. Span-DSR and DSR-PSM will occasionally reach directly into the MAC layer's interface queue to acquire and reorder packets, usually in response to route errors.

4.2 Span-DSR: Modifications to Span

There were few modifications made to the Span protocol, since keeping Span's properties intact was a primary design goal. HELLO packets are sent regularly to maintain the correctness of nearest neighbor tables, and coordinator elections continue to take place with the standard delay function described for Span [4]. In order to implement the optimizations for smoother functioning under DSR, the coordinator window time, which decides when a coordinator announces itself as tentative, was increased proportional to the rate of flows which a node is servicing. Rate flows were calculated according to the number of data packets received during a time window of two seconds. In order to insure that the fairness properties of Span remain intact, a maximum window increase of 30 seconds was applied. Given this constant increase, an original Span window of t_{Span}, and a rate r, we calculate the new time window $t_{Span-DSR}$ as follows:

$$t_{Span-DSR} = t_{Span} + \max(r, 30)$$

4.3 Span-DSR: Modifications to DSR

In order to work properly with Span, the implementation of DSR was modified to recognize Span coordinators in order to determine whether to process route maintenance and data packets. All DSR related packets are classified as Span data packets. Source routes which include withdrawn coordinators are handled as described in Section 3.3.1. In this scenario, route errors may be sent even though the next node is reachable, while in the standard Monarch implementation route errors are handled through a callback function from the MAC layer. We used the *MobiCache* route cache, which implements a "path cache" organization that stores whole routes as opposed to a "link cache" protocol, which stores individual links but requires more CPU time to execute graph search algorithms.

4.4 DSR with 802.11 PSM: Modifications

As described in Section 3.4, modifications were made to DSR to utilize 802.11's Power Saving Mode. A node currently servicing one or more routes will go back to sleep when it does not receive a data packet after a timeout of 5 seconds. This value was determined to work well for our experiments, but more extensive experiments may determine a more optimal timeout (see Section 6). Because nodes begin in sleep mode, where the propagation time for packets is much longer, we turned off the *ring zero search* option that sends a non-propagating route request as the first action of a route discovery. Alternatively, we experimented with increasing the timeout value between the initiation of a ring zero search and a subsequent propagating route request, but this alternative created even longer latency delays.

5. Performance Evaluation

To evaluate the performance of our power saving schemes with respect to energy-efficiency and throughput, we performed tests on randomly generated static topologies.

5.1 Simulation Environment

We evaluated our source-routing schemes in the *ns-2* [1] network simulator using the Monarch [2] extensions. Our implementation of DSR handled routing, while the MAC layer was an unmodified 802.11 layer that was capable of operation in (802.11 PSM) and out of (802.11) power-saving mode.

5.2 Energy Efficiency

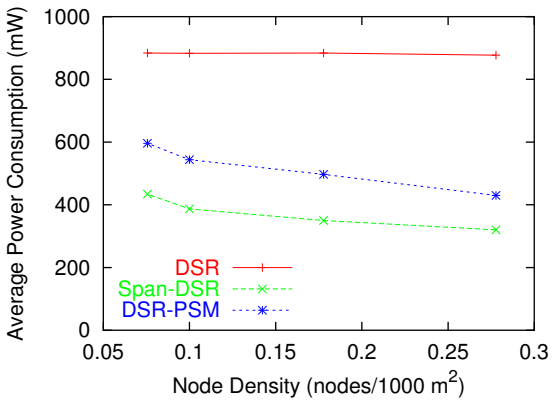

Figure 2: Energy-Efficiency Results for DSR, DSR-PSM and Span-DSR.

5.2.1 Simulation Setup

In order to evaluate our scheme in terms of energy efficiency, we simulated routing in variable size square grids (600 by 600, 750 by 750, 1000 by 1000 and 1150 by 1150) with a random arrangement of 100 nodes. The nodes used radios with a bandwidth of 2 Mbps and a nominal range of 250 meters.

In each simulation there were 20 flows between 10 source and 10 destination nodes. Each communicating node was a CBR source sending packets of size 128 bytes. The sources and sinks were distributed at random along two sides of the grid in order to assure that nodes must travel several hops to reach each other. The remaining nodes were placed with a random uniform distribution onto the rest of the grid. Nodes remained stationary throughout these simulations. Note also that these nodes were counted in addition to the 100 nodes mentioned previously, so that our simulations took place with 120 nodes (but were only meaningful for the 100 nodes in the middle of the grid).

The sources and sinks were always awake and sent constantly. This simplified the analysis and the simulation, although it is not a perfect model of actual flows in an ad-hoc network. Source and sink nodes never participated in coordinator elections for Span but were otherwise unrestricted.

5.2.2 Results

Our energy efficiency results are very promising for Span-DSR. As Figure 2 shows, Span-DSR saves over 2 times as much power as regular DSR. These numbers are similar to those in the original Span implementation with geographic routing. One difference between the two sets of numbers is that the overall power levels under DSR are about 70 mW higher. We surmise that this occurs due to the increased packet overhead of DSR as compared to geographical routing. The excessive number of packets sent out under DSR causes many nodes to be in send mode for extended periods of time, which consumes more power than simply idling while awake (nodes in their idle waking state consumed 831 mW of power while nodes in their idle sleeping state consumed 171 mW of power for comparison).

The energy efficiency results for the PSM scheme were somewhat surprising, and exposes a weakness in this design. The power consumption in this case is significantly higher than the Span-DSR case, even though the protocol was made to react to network traffic and does not require nodes to stay awake when there are no flows. After examining traces, we discovered that the DSR-PSM scheme

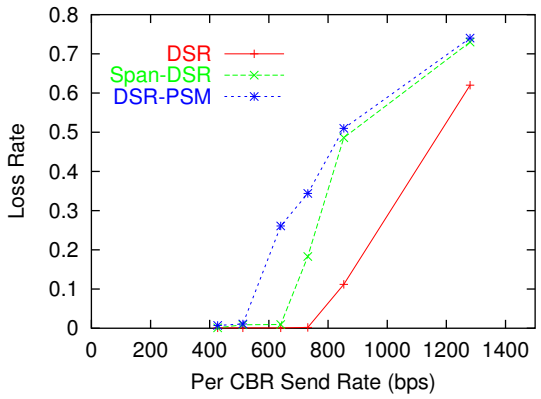

Figure 3: Loss Rate Results for DSR, DSR-PSM and Span-DSR.

was making many distinct routes between nodes, often unnecessarily involving nodes in routes that could be made more efficient with some level of route aggregation. Some of these routes were also excessively long because of the variable delays which occur between sleeping nodes that attempt to synchronize in 802.11 power-saving mode - the first route reply which is received at a source is not necessarily the shortest or most efficient, causing increased power consumption compared to the optimal route for a given network.

Another important result of the Span paper that we echo here is that at higher node densities power consumption is reduced. This is an obvious but important result, since in dense networks more nodes will be able to stay idle as other nodes service routes around them.

One final note is that the Span-DSR scheme went through several iterations due to bug fixes. The numbers presented here are for the final fixed version of Span-DSR. The differences in efficiency across versions were negligible.

5.3 Loss Rate and Throughput

5.3.1 Simulation Setup

We conducted a few preliminary loss rate analyses that involved a small network of 12 nodes with 2 sources receiving and sending. In this network we used 1280 byte packets and varied the send rate of the sources in order to measure the reaction of our schemes to congestion. This scheme was used to identify the cause of some problems we had initially due to a deficiency in the Span-DSR scheme. Specifically, we tested to see if allowing withdrawn coordinators to send outstanding packets would have an effect on the overall delivery rate.

Our final experimental setup allowed us to accurately measure the performance our three schemes in a fairly large network. The simulations were run on a 1000 by 1000 meter grid with 120 nodes as in the power case. Again, 20 of the nodes were sources/sinks and were placed in strips on either side of the grid, with the remaining 100 nodes placed randomly in the middle with a uniform distribution.

Each source sent 128 byte packets and we varied the packet send rate in order to measure the response of the system to congestion under the final version of each of the three schemes (DSR, DSR-PSM and Span-DSR).

5.3.2 Results

During our intermediate trials, we validated the fixes we made to Span-DSR for improved throughput. Specifically, by allowing a withdrawn coordinator to continue sending packets after it withdraws, we avoid a severe loss in throughput due to dropping the outstanding packets while a route error propagates back to the source. This effect is more severe because the sources are CBR sources, causing a significant number of outstanding packets during this time interval.

When we tested the final version of our code on the larger network, after eliminating a small bug that caused unnecessary packet drops at coordinators, our results were extremely interesting (see Figure 3). Using the DSR case as a baseline, we can see that neither Span-DSR nor DSR-PSM are optimal in terms of loss rate. Each scheme begins losing packets at a lower send rate than in regular DSR. In Span-DSR, this is most likely due to the latency caused by sending outstanding packets through a withdrawn (and sleeping) coordinator after an election. Thus, during congestion events, this latency will cause losses to accrue faster than in the DSR case where all nodes are awake and ready to route packets. The results for the DSR-PSM case can also be explained in this regard. Because the scheme has a heavy reliance on sending packets through nodes that are sleeping in power-saving mode, congestion events will cause packet drops earlier across paths with sleeping nodes. This can occur during periods of congestion because nodes have timed out and fallen asleep or because nodes are sleeping during the route setup phase of the protocol.

5.4 Fairness

Our static testing environment does not accurately evaluate the fairness of our two power saving approaches. Because we continue to initiate Span coordinator elections we expect that fairness should approximate that of the original Span. Since we decrease the frequency of coordinator elections based on traffic flow, given a network of equally powered nodes, the standard deviation of the time spent as coordinator for each node should increase slightly. Otherwise, Span's fairness properties remain intact.

The DSR-PSM scheme does not impose any level of fairness upon the network. Especially given our long running CBR connections without mobility, the same nodes will continue to service a route until they run out of energy. If mobility were introduced with randomly initiated connections and varying connection lengths, we would expect that nodes distributed around the center of a topology would be much more active than those along the edges. Additionally, we would expect to see some aggregation effects as described in Section 3.4, and active nodes would tend to service new routes. Additional fairness experiments are necessary to quantify these claims (see Section 6).

6. Future Work

There are still many opportunities to improve the efficiency of our two power saving schemes. Further study of Span-DSR parameters may determine more optimal values. For instance, we introduced a scheme to increase the coordinator window length but have not run extensive simulations to quantify the effects of this increase, especially with regards to throughput and fairness. We can further study the effects of the DSR-PSM sleep timeout value, the tendency for aggregation of routes through nodes which are already awake, and the existence of sub-optimal source routes due to the inconsistent propagation delays of route maintenance packets of nodes in power-saving mode.

Another improvement to Span-DSR would involve a fast handoff of routes between a withdrawn coordinator and its replacement, eliminating the latency of routing through a sleeping node and wait-

ing for a route error to propagate to the source. There are many issues with this approach, including the fact that a new coordinator may not be able to take over all routes (connectivity may have been restored by two announcing coordinators instead of one). Another simple way to avoid the latency of routing through a withdrawn coordinator would be to make the coordinator stay up until all outstanding packets are cleared and a new route has been found.

One significant area of work still to be done is in terms of simulations. We would like to measure the effects of mobility on the two schemes, and also classify them in terms of latency and fairness. More simulations exploring power savings and total capacity may reveal further research issues.

The largest potential for improvement in power savings will probably not come from changes in the Span parameters. Instead, an application-aware power-saving scheme in the MAC layer, which conforms to the end-to-end argument [9], may provide substantial improvements. Specifically, the overhead of beacon period synchronization may be reduced for packet transmissions from coordinators to sleeping nodes, since packets are naturally queued at coordinators until they are retrieved.

Finally, as our experiments have confirmed the effectiveness of the Span power saving scheme applied to a source routing protocol, converting the Span code into a system library will improve its viability for use in future wireless network applications. In terms of *ns-2*, we would like to make Span modular enough to simulate it under many other routing protocols (such as DSDV, AODV, TORA and zone routing) in order to characterize its behavior under each scheme.

7. Conclusion

As ad-hoc wireless networks become more prevalent, power consumption will become an important issue for these devices. Although portable networked devices are more powerful and efficient than ever before, the software on these devices must be power aware in order to give users high system lifetimes and better quality of service.

The two protocols presented here attempt to solve this problem by building on previous work, in particular the Span scheme [4]. Our first scheme, Span-DSR, modified Span in order to make source routing more effective under it's coordinator election scheme. Our second scheme, DSR-PSM, used the 802.11 MAC layer power-saving mode in order to build a power-saving protocol more in line with the reactive nature of source routing.

We found that Span-DSR works very well, saving on average about 2 times as much power as standard DSR while maintaining high levels of throughput and fairness. This result was achieved even though Span is *proactive* while DSR is *reactive*. In spite of this fundamental difference, Span performs as well with source routing as it does with geographical routing. We also found the DSR-PSM scheme does not perform as well as Span, even though it only uses network resources when needed. This poor performance is due to the simplicity of the protocol, in that it does not explicitly deal with route aggregation or fairness, nor does it effectively overcome latency issues in 802.11 PSM. These results indicate that further research into optimizing these two protocols, as well as creating a more effective power-saving MAC layer, could produce significant gains in energy-efficiency for ad-hoc wireless networks.

8. Acknowledgments

The authors would like to thank Benjie Chen for his assistance in debugging the ns-2 code and especially for his guidance in understanding and extending Span to work with DSR. Additionally, we would like to thank Hari Balakrishnan for helping us to generalize a previous scheme to reduce Span-DSR coordinator election frequency into our current time window increase algorithm.

9. References

[1] The ucb/lbnl/vint network simulator-ns (version 2).

[2] Cmu monarch wireless and mobility extensions to ns-2, October 1999.

[3] BROCH, J., MALTZ, D. A., JOHNSON, D. B., HU, Y.-C., AND JETCHEVA, J. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Mobile Computing and Networking* (1998), pp. 85–97.

[4] CHEN, B., JAMIESON, K., BALAKRISHNAN, H., AND MORRIS, R. Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks. *ACM Wireless Networks 8* (2002).

[5] FEENEY, L., AND NILSSON, M. Investigating the energy consumption of a wireless network interface in an ad hoc networking environment. In *IEEE INFOCOM* (2001).

[6] JOHNSON, D. B., MALTZ, D. A., HU, Y.-C., AND JETCHEVA, J. G. The Dynamic Source Routing protocol for mobile ad hoc networks, March 2001.

[7] LI, J., JANNOTTI, J., DE COUTO, D. S. J., KARGER, D. R., AND MORRIS, R. A scalable location service for geographic ad hoc routing. In *Proceedings of the 6th ACM International Conference on Mobile Computing and Networking (MobiCom '00)* (Boston, Massachusetts, August 2000), pp. 120–130.

[8] LI, Q., ASLAM, J., AND RUS, D. Online power-aware routing in wireless ad-hoc networks, August 2001.

[9] SALTZER, J., REED, D., AND CLARK, D. D. End-to-end arguments in system design. In *ACM Transactions on Computer Systems,* (November 1984).