COORDINATED ENERGY CONSERVATION IN AD HOC NETWORKS

Selma Ikiz, Vinit A. Ogale and Vijay K. Garg
{ikiz, ogale, garg }@ece.utexas.edu
Dept of Electrical and Computer Engineering
The University of Texas at Austin
Austin TX 78712

ABSTRACT

This paper presents a new power conservation scheme for multi-hop ad hoc networks. A virtual backbone consisting of special nodes (coordinators) is used for the power saving algorithm and routing. We present a novel algorithm for constructing a connected dominating set (CDS) that is used to construct and maintain the virtual backbone of the network. Our scheme includes a message history based variable sleeping time for the non-coordinators. Simulations indicate that our scheme results in better power conservation than other practical schemes discussed in the literature if the network has a sparse message density.

1 INTRODUCTION

An ad hoc network is a collection of wireless mobile hosts without any fixed infrastructure. In such a network each host can act as an intermediary and forward packets to the next hop in order to reach the final destination. Ad hoc mobile networks have far reaching applications due to their suitability for rapid deployment and inherent robustness.

Ongoing research addresses issues like routing, network management, QoS, Media Access Control (MAC) protocols, topology management, mobility and security. Routing is a fundamental issue for any network and, not surprisingly, is a very active topic of research in ad hoc networks. Also, in most mobile ad hoc networks, power seems to be a major constraining factor. Hence energy conservation is one of the key issues in any protocol or algorithm for ad hoc networks.

In this paper we focus on energy conserving dynamic backbone based routing technique for ad hoc networks. A backbone based scheme involves partitioning the network into coordinators and non-coordinators. The coordinator nodes are responsible for the routing within the network and hence need to be active. The set of the coordinator nodes constitutes the *backbone* of the network. In contrast, the non-coordinator nodes are responsible for only the packets sent by them or addressed to them and are allowed to enter very low power consuming *sleep* states. This can drastically increase the life of the non-coordinators. Since coordinator nodes can not sleep, it is important to dynamically update the backbone according to the power remaining in each node, if we wish to increase the overall life of the network. Our paper makes two important contributions. First, we present and prove a new algorithm for maintaining and constructing the backbone. Secondly, we propose a power saving protocol, which allows the nodes to sleep for varying amounts of time depending on the message history of that node. Our simulations show that the proposed scheme results in noticeable power saving compared to existing schemes.

2 BACKGROUND AND RELATED WORK

The wireless exchange of data between nodes strongly dominates other node functions including sensing and processing in terms of energy consumption [4][2]. Furthermore a node in the sleeping mode i.e. with their radios turned off; consumes significantly less energy than a node in the idle, transmit or receive mode. However, putting the node in sleep mode essentially disconnects it from the network and changes the network topology. Therefore instead of allowing all nodes to sleep, we force some nodes to become coordinators which never sleep. The coordinators have to be chosen such that each node is connected to at least one coordinator and the coordinators form a connected sub graph. Therefore we need to select a virtual backbone in the network.

The backbone consists of coordinators and all nodes are connected to at least one coordinator. Such a backbone also offers a scalable solution for routing. Only the coordinators participate in the routing and the non-coordinators simply send the data they want to transmit to a coordinator. The coordinators use a conventional ad hoc routing algorithm like DSR [5] or AODV [6] to find a route to the coordinator associated with the destination. The advantage with such backbone based routing is that route discovery becomes much simpler. Route discovery for most ad hoc routing schemes is very expensive in terms of time and number of messages required and the cost increases with the number of nodes.

The backbone formation can be considered to be a problem of determining a *connected dominating set* (CDS) in ad hoc networks. A *dominating set* (DS) of a graph G is a subset V_s of the vertex set V such that each node that does not belong to the subset V_s is adjacent to a node in V_s. A connected dominating set is a dominating set which induces a connected subgraph.

We know that the task of finding a *minimum connected dominating set* (MCDS) in an ad hoc network is NP-hard [3]. Fortunately, in an ad hoc network with high chances of link failures and topology changes due to mobility, constructing a minimum CDS is not the aim. Some redundancy in the backbone is desirable to increase reliability of the network and mitigate the backbone maintenance.

Current work in this field includes different algorithms to construct a CDS [1][7][10]. In [10] a scheme which factors power in construction of a CDS is proposed. However, in this scheme the nodes forming the backbone are not changed periodically. AS a result the coordinators spend more energy and die out much sooner than the other nodes.

Chen et. al.[4] suggest a protocol (*Span*) to form a CDS and change the coordinator nodes periodically. Span adaptively selects coordinators to form a backbone. It has several rules based on the node's remaining energy level and number of neighbors for coordinator announcement and withdrawal. It also assumes periodic broadcasting of `HELLO` messages that contain the node's status, its neighbors and each neighbor's status. Span uses an approach similar to the 802.11 ad hoc power-saving mode (PSM) that uses periodic beacons to synchronize nodes in the network. This synchronization is done at `MAC` layer. In the 802.11 PSM, beacon periods are divided into two time slots; the advertisement time and the advertised packet transfer time. Moreover, Span increases energy saving by adding another window only for non-coordinator nodes. For such nodes, there are three windows inside a beacon; `ATIM`, `NATIM`, and the rest. `ATIM` is used to transfer packet advertisements from node to node and coordinator to node. New advertisement window (`NATIM`) is used for packet transfers between these nodes. The remaining time is used by coordinators for packet transfer among themselves.

In this paper our scheme extends and improves Span. We use a new CDS selection algorithm presented in the next section. We also propose a new scheme which enables inactive nodes to sleep for longer periods thus improving the overall network life.

3 Network Model

In a typical ad hoc wireless network, each node can have a different transmission radius resulting in unidirectional links between nodes. Fig. 1 shows an example of the topology of a typical ad hoc wireless network.

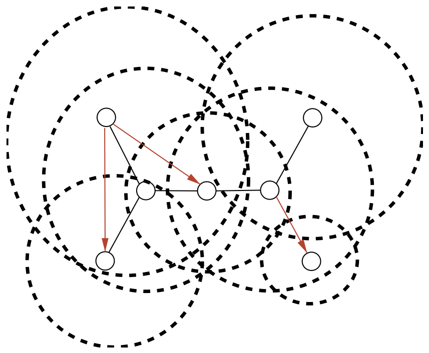

Figure 1. The topology of a typicalwireless ad hoc network in 2D.

For convenience, we simplify the underlying network topology by disregarding all the unidirectional links. Hence we consider two nodes to be connected if and only if there exists a bidirectional link between them. The topology of such a network can be modeled as a unit-disk graph (UDG) [1]. The simplified topology for the network in Fig.1 is shown in Fig.2.

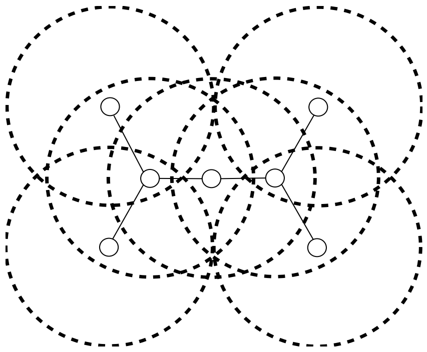

Figure 2.

Yet, a mobile object is positioned in a 3D world [13]. Hence, we model the underlying topology as a unit-sphere graph (USG). This is a realistic model, if the network topology is unknown and the nodes have identical functions. Generally in such cases, unidirectional links are not useful. The simplified 3D topology for the network is shown in Fig 3. Each node represents a vertex and we draw an edge between two nodes if and only if a bidirectional wireless link exists between the nodes. Note that the network topology is not static and may change frequently as nodes move, die or join the network.

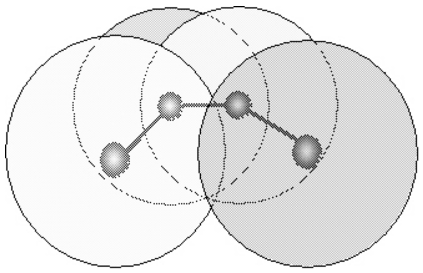

Figure 3. Model the topology of wireless ad hoc network by unit-sphere graphs.

4 Virtual Backbone Formation Algorithm

In this section we present our virtual backbone formation algorithm. Nodes apply the rules of the algorithm locally to form a CDS, which forms the virtual backbone of the network.

4.1 Notation

Let us represent the network as a connected graph $G = (V, E)$. The backbone consists of a smaller graph $G' \subset G$ such that G' remains connected. Let :

1. $N(v)$ be the *open neighbor set* of vertex v, i.e., $N(v) \equiv \{u \mid \{v, u\} \in E\}$,

2. $n(v)$ be the cardinality of $N(v)$

3. $p(v)$ be the power metric defined as $p(v) = E_r/E_t$ where E_t is the maximum amount of energy available at the node and E_r is the remaining energy at the node,

4. $ID(v)$ be the unique node identifier of v.

We construct a tuple $T(v) \equiv \langle N(v), p(v), n(v), ID(v)\rangle$. To compare two such tuples we use the following rule; $T(v) \prec T(u)$ if and only if $N(v) \subset N(u)$, or $N(v) \equiv N(u)$ and we compare the other elements lexicographically. The two tuples are equal if and only if all their elements are equal.
Clusters : Let v be a coordinator, then we say $c.v =$ `true`, and $N(v)$ is the *cluster* of v.
Each node in the network periodically sends a `HELLO` message containing a list of its neighbors. Each node maintains a list of its neighbors and each neighbor's neighbor list. The nodes in the neighbor table are marked as either coordinators or non-coordinators.

4.2 Activity I: initiate coordinator selection

4.2.1 Rule 1

If any two neighbors of node v are not connected, then node v declares itself as the coordinator. This algorithm is similar to the algorithm proposed by Wu and Li in [4][10].

4.3 Activity II : reduce the backbone size

4.3.1 Rule 1

A coordinator node v withdraws as a coordinator and sends `NLC` (No Longer a Coordinator) to its neighbors if there exist a node u i.e. $T(v) \prec T(u)$.

4.3.2 Rule 2

For the coordinator node *v* ::

1. Node v sends `RTW` (Request to Withdraw) to all neighboring coordinators including itself if :
   (a) all coordinator neighbors of node *v* are connected to each other directly or through a coordinator, and
   (b) all the non-coordinator neighbors of *v* are connected to at least one other coordinator other than *v*.
2. If node *v* receives `PTW` (Permission to Withdraw) from all neighboring coordinators, then node *v* withdraws as coordinator and sends `NLC` (No Longer a Coordinator) to all neighbors including itself.
3. If node *v* receives `RTW` from one or more nodes, then it waits till the end of the round and sends `PTW` to the node that has the minimum T value.

4.4 ACTIVITY III : POWER-BASED COORDINATOR RE-SELECTION

A coordinator consumes more than thrice the energy consumed by a sleeping node [4]. Hence it is likely that the coordinators will die much before the non-coordinators, with the potential consequence of partitioning the network. To prevent this and maximize the overall network life, our algorithm allows a coordinator *v* to withdraw if its power metric $p(v)$ is lower than the power metric of its neighboring non-coordinator nodes.
For a coordinator node *v* ::

1. If the power metric p of all non-coordinator neighbors of node *v* is least 15% of $p(v)$ and all coordinators of *v* are connected to each other *directly or through another node*, then *v* sends a `RCR` (Request for Coordinator Re-selection) message to all the nodes on the alternative path and also to each of its neighbors including itself.
2. If node *v* receives a `RCR` from node *u*, then node v
   - waits till the end of the round for all `RCR`s,
   - sends `VTC` (Volunteer To be a Coordinator) to the coordinator with the minimum power metric,
   - sends a `CRI` to all other nodes, and
   - announces itself as the coordinator.
3. If node *v* receives a `VTC` from all its neighbors, it withdraws as a coordinator. Otherwise, on receiving a `CRI` it sends a `WCRR` (Withdraw Coordinator Re-selection Request) to all neighbors.
4. On receiving a `WCRR` a node, which had become a coordinator in response to the `RCR`, withdraws.

Each coordinator in the network periodically checks if it can withdraw by applying Activity III. Thus the algorithm tries to balance out the energy consumption amongst all the nodes in the network.

5 PROOF OF VALIDITY

Lemma 1 *The coordinators decided by Activity I form a connected dominating set if the underlying network is connected.*

Proof: (Outline) Assume that the coordinators do not form a CDS. Therefore, there exist two coordinators connected by a non-coordinator node. Now Activity I requires such a node to be a coordinator. Therefore the claim is true. ■

Claim 1 *In Activity II, Rule 2, if node* v *withdraws then no other coordinator in the same or adjoining cluster can withdraw in the same round i.e. if $c.v \wedge c.u \wedge (N(v) \cap N(u) = \Phi$ is true in this step, then $c.v \vee c.u$ is true in the next step.*

Proof: Rule 2 requires a coordinator to get a `PTW` message from all neighbors before withdrawing. Each neighbor can give a `PTW` to exactly one node in a round. Hence the claim follows. ■

Lemma 2 *The connectivity of the dominating set does not change due to Activity II.*

Proof: We show that if there exists a CDS before applying Activity II, then the connectivity is maintained after applying Activity II.
CASE 1: A node *v* withdraws due to Rule 1 implies that there exists a node u i.e. $(T(v) \prec T(u) \wedge c.v \wedge c.u)$. Therefore node *u* has a link to all the nodes connected by *v*. Since both cannot withdraw in the same round, the connectivity is maintained.
CASE 2: A node *v* withdraws due to Rule 2. After applying Rule 2, the connectivity of the network could change if

1. there is a non-coordinator neighbor of v that has no coordinator, or
2. there is no path between two coordinators in CDS.

The algorithm eliminates the first possibility, because the coordinator *v* withdraws only if all the non-coordinator neighbors are connected to at least one other coordinator. Node *v* also checks if all its coordinator neighbors are connected to each other directly or through an other coordinator. Hence, the backbone cannot be partitioned due to the withdrawal of *v*. ■

Lemma 3 *The connectivity of the dominating set in the network does not change due to Activity III.*

Proof: This proof is similar to the proof of Lemma 2. ■

Theorem 1 *Activities I, II and III applied in sequence result in the formation of a CDS in a connected graph.*

Proof: This follows from Lemmas 1, 2 and 3. ■

Theorem 2 *The distributed algorithm proposed in this paper have an approximation factor of $\frac{n}{4}$, and $O(m)$ message complexity in 3D environment, where n is the number of nodes and m is the number of links in the network.*

Proof: For every activity in the algorithm messages are exchanged locally, and no message spurs any other. Hence, the message complexity is $O(m)$. Next section proves the approximation factor. ■

6 Example

Fig.4 shows an example of using the proposed marking algorithm. Since each node keeps track of all its neighbors, it broadcasts its neighbor list and their states periodically. After this information exchange phase, every node will have information on all nodes with a radius of two units. In Fig.4 (a), node 1 will not mark itself as a coordinator node since two of its neighbor is directly connected. However, node 2 will mark itself as a coordinator seeing that node 5 and node 4 does not have a connection. Fig.4 (b) shows the resultant graph of the second phase. These coordinator nodes absolutely form a connected dominating set, but not the minimal one. After applying Rule 1 at the third phase, node 21 and 27 will withdraw from being a coordinator, and will be unmarked as shown in Fig.4(c). At phase four, coordinator nodes form their clusters, which are simply just their neighbors. Node 2 will check if all its coordinator neighbors are connected, and all its non-coordinator neighbors are connected to one more coordinator. After seeing that its coordinator neighbors, node 4 and node 9, are connected, and all its other neighbors have a coordinator other than itself, node 2 will decide to withdraw. Similarly, node 9, 13, 15, and 18 will decide to withdraw, and broadcast withdraw request. Assuming energy levels are equal, node 2 will get permission from node 9, while node 9 cannot get a permission from node 2. For node 13, 15, 18 and 19, node 11 will give permission to just one of them. Therefore, after the first round, graph will be as shown in Fig.4(d). At the beginning of the second round, node 9 will see that it cannot not decide to withdraw since node 8 does not have another coordinator. Likewise node 15 cannot decide. On the other hand, node 18 will still insist at withdrawing, and will get the permissions. Fig.4(e) shows the final graph.

7 The approximation factor of CDS

Peng, Khaled and Ophir in [1] reinvestigate CDS algorithms in [10][8][9], and [7] and establish an approximation factor for one each of them. By using their approach, we show that our proposed algorithm has an approximation factor of n/4. When n is even, we consider the instance illustrated in Fig.5 (a). These nodes are evenly distributed over the the two horizontal sides of a unit squares. Each node has exactly m+1 neighbors, one in the opposite side and m of them are in the same horizontal side. Peng, Khaled and Ophir argues that, any MCDS consist of a pair of nodes lying in the same vertical segment [1]. For this to be true, they must assume nodes at the same horizontal level are strongly connected. If we consider only one of the horizontal sides, let's say v is selected to be a part of MCDS, then it means node v is the DS of that horizontal side. If node v is the DS by itself, then every other node at this horizontal side must be connected to it. Therefore m=n/2-1. Since every other node at the same horizontal side has exactly m neighbors, nodes at the same horizontal side are strongly connected.

Our algorithm, initially includes all nodes into CDS, then starts eliminating at phase3. At first round,

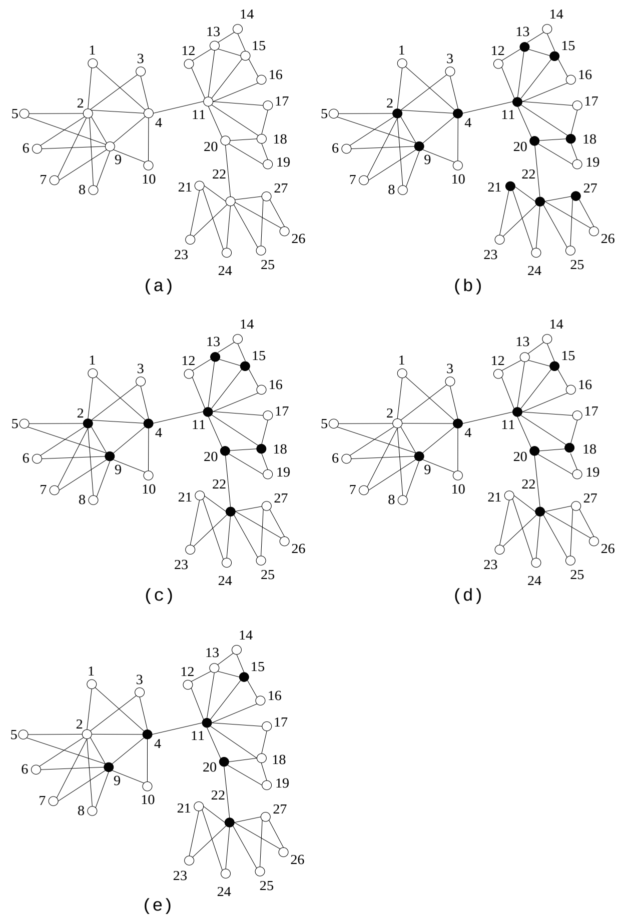

Figure 4. An example of marking process

every node wants to witdraw. It's obvious that nodes at the same horizontal side are connected. Let's say node v and u are neighbors and they do not belong to the same horizontal side. Clearly, every neighbor of node u has an other neighbor on the other side which are neighbors of v. Therefore, every neighbor of v is connected to u by an other coordinator. Because of strongly connected assumption of a horizontal side, at a round only one node can withdraw. At the end of round n/2, witdraw requests will cease with n/2 non-coordinator nodes. Therefore, our algorithm has an approximation factor of n/4. Similar argument is also applicable when n is odd. For this case one should consider the instance illustrated in Fig.5 (b).

7.1 Example of the approximation factor

Fig.6(a) shows a similar graph argued above. In Fig.6(a), node 1 decides to become a coordinator since node 7 and 1 are not connected, and marks itself. Sim-

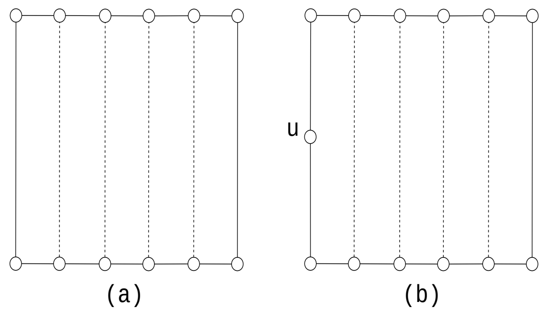

Figure 5. Instance for which the MCDS is 2

ilarly all nodes mark themselves. At the second phase none of the nodes withdraw as shown in Fig.6(b) and Fig.6(c). However, at phase 3 round 1, every node wants to withdraw, but node 0 will win assuming identical energy levels. Fig.6(d) shows the result of round 1. At round 2, still all coordinator nodes want to withdraw but node 2 will win. Fig.6(e) and (f) shows the result of round 2 and 3 consecutively. At round 4, only node 5,6, and 7 decide to withdraw and node 5 will win. At the end of round 4, no node wants to withdraw, and CDS is formed as shown in Fig.6(g).

7.2 Approximation factor for Peng's algorithm

Peng, Khaled and Ophir in [1] show that their algorithm has an approximation factor of 8. However, using the same example above, we will show that their approximation factor is also n/4 in 3D environment.

Assuming that node 0 starts the leader election algorithm, Fig.7(a) shows a possible spanning tree. If we apply the rules, node 0 will mark itself black and will send `DOMINATOR` message to node 1,2,3 and 7 as in Fig.. (b). After receiving a DOMINATOR message node 1 marks itself gray, and broadcasts `DOMINATEE` message. The result of this phase is shown in Fig.7(c). After receiving `DOMINATEE` messages from all its low-ranked neighbors, node 4,5, and 6 marks itself black, and broadcasts a `DOMINATOR` message as shown in Fig.7(d). Receiving a `DOMINATOR` message from its child, nodes 1,2, and 3 mark themselves black and broadcast a DOMINATOR message. Fig.7(e) and (f) shows the resultant spanning tree according to their algorithm, since all black nodes will check if they have a higher rank than all their neighbors and all its neighbors are black, it remarks itself gray. However, there is no such node at Fig.7(e). Therefore, their ap-

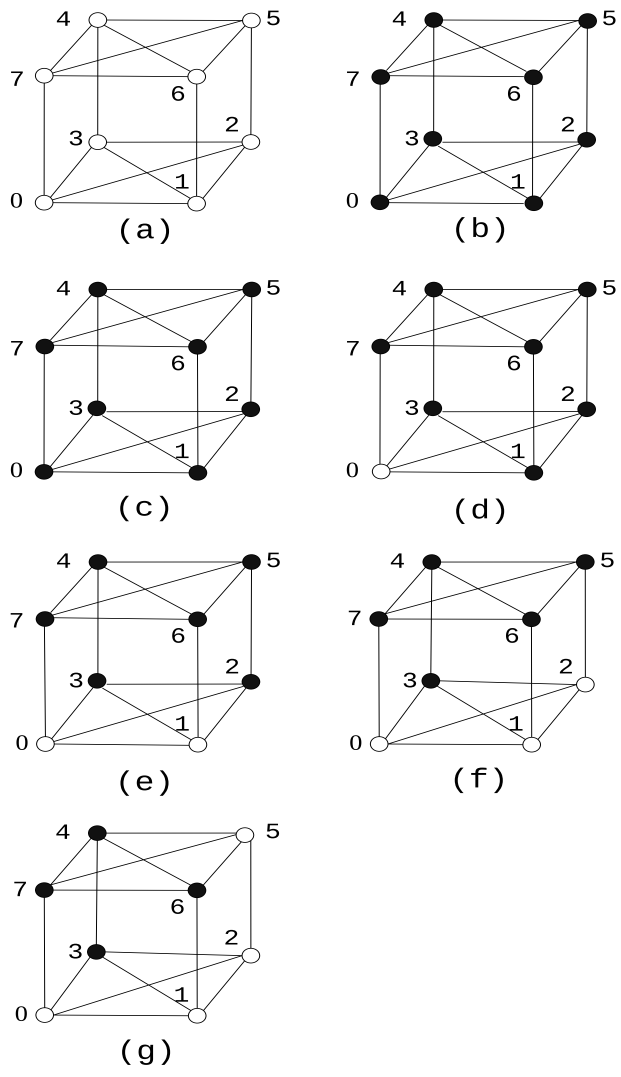

Figure 6. An example of marking process

proximation is (n-1)/2 for this case. If we improve their algorithm by relaxing their last rule by changing it as follows; If a black node's rank is greater or equal to all its neighbors and one of its lower ranked neighbors is black, it marks itself gray and broadcasts DOMINATEE message. With this improvement, Fig.7(g) is obtained as a final configuration which has n/4 as an approximation factor. We show that for the above case, n can be any even number. For the configuration in Fig.5 (a), a spanning tree in Fig.8 is possible. Applying the same argument above, it is clear that it has an approximation factor of $n/4$, or O(n).

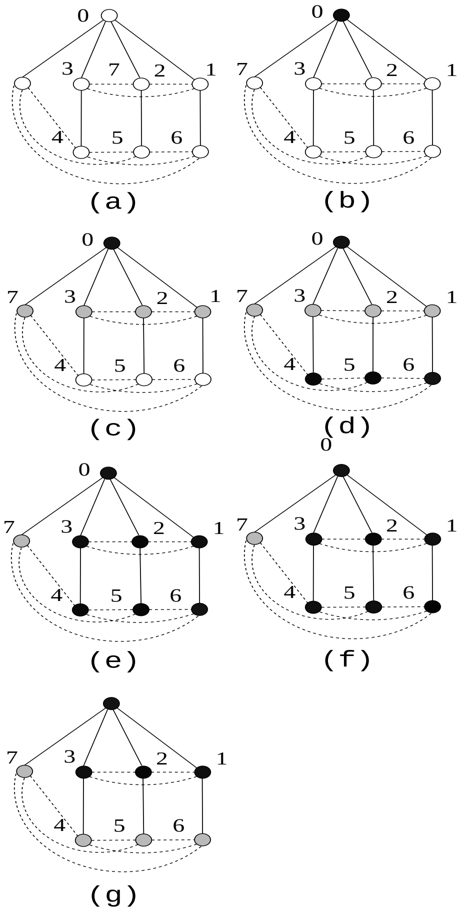

Figure 7. An example for color marking algorithm of Peng

8 Variable Sleeping Time Based On History

Our approach is based on Span's NATIM success, however, it furthers its effect by taking packet delivery history into account. We assume a virtual backbone is formed by coordinator nodes, and non-coordinator nodes are allowed to turn off their radio receivers. In Span's design NATIM is constant. To increase the power saving, we propose this value to be variable with an upper bound. The second improvement is non-coordinator nodes should use a two bit history for sleeping time. When a node observes two consecutive beacons without any packet advertisement, it decides to sleep through the next beaconing period. Fig.9 shows its transition graph.

Whenever a coordinator does not get a reply back after two consecutive ATIM windows, it removes the neighbor node from its neighbor table and clear

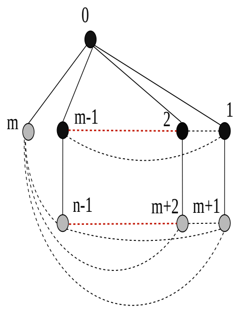

Figure 8. Color marking approximation with n nodes

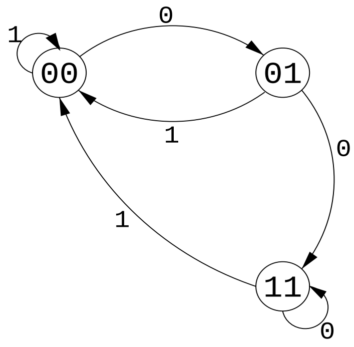

Figure 9. The state transition graph.

its buffer. This process is the same for an immediate neighbor node, except it sends the packet through a coordinator.

8.1 Simulation

For the aforementioned approach, we simulate the algorithm for the topology in Fig. 10.

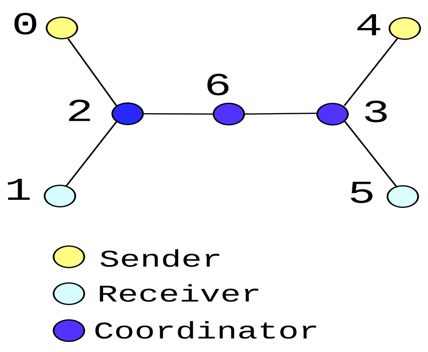

Figure 10. Ad hoc network toplology for simulation.

Node 0 sends packets to node 1 and node 5, while node 4 sends packets to node 1 and node 5. nodes 2, 3 and 6 are coordinators, and they buffer the packets if their neighbors are sleeping. Sender neighbors send directly to receiver neighbors if they are not sleeping, otherwise they send through their coordinators. We use the same beacon, ATIM and NATIM values given to be optimal in Span simulations [4]. A packet is fixed to 128 bytes, and number of packets per beacon period is limited to 100. The simulator randomly selects when the node receives the packets in the NATIM window. When the node receives the last packet destined to it, it gets into sleeping state without waiting the end of NATIM window. For power consumption, we use the values as shown in Table 1.[4].

Table 1: Power consumption of the Cabletron 802.11 network card in the Tx (tansmit), Rx (receive), Idle, and sleeping modes [4].

Tx	Rx	Idle	Sleeping
1400mW	1000mW	830mW	130mW

SPAN improves the power consumption of a leaf node greatly, however, when the coordinator to non-coordinator ratio increases, the improvement decreases.

Given a dense network and low message probability, our algorithm gives slightly better results. It improves the SPAN's energy saving up to 16 percentage(see Fig. 11).

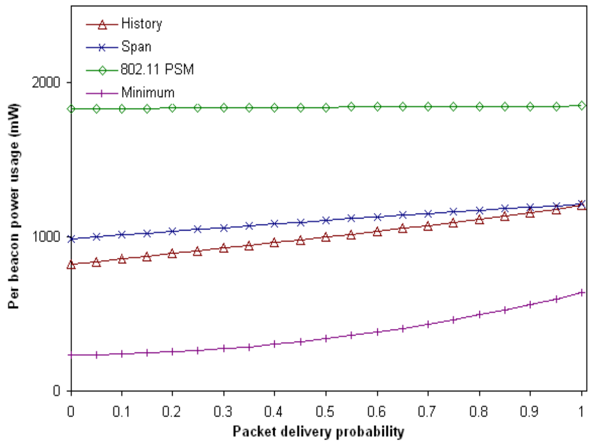

Figure 11. Energy consumption per node at different packet delivery probabilities.

Receiver nodes are the ones that are affected

most from this approach. Sender nodes are somewhat affected. Coordinator nodes are least affected, since from their perspective nothing changes.
Therefore we simulate the Span, 802.11 PSM, and our algorithm for only a receiver node with different packet receiving probabilities. We also calculate the possible minimum value, i.e. receiver is awake only when there is a message transfer, which is the absolute minimum power requirement.

Simulation results indicate that employing variable NATIM window and sleeping time ameliorates the power usage of a receiver Fig.12. It is a significant improvement over the 802.11 PSM, and a sensible improvement over Span. The advantages of our scheme are pronounced when the network a has high node density and low packet arrival probabilities.

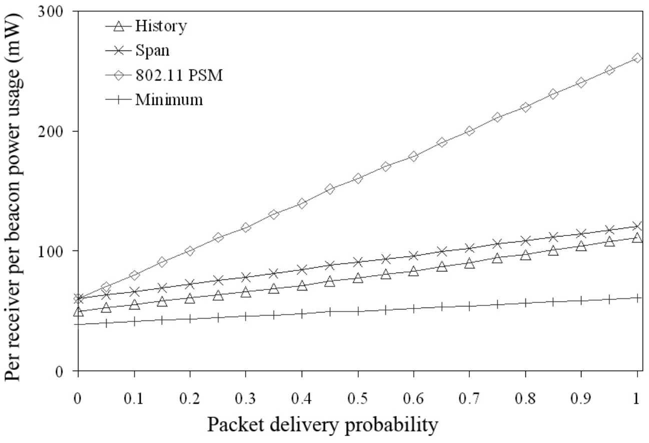

Figure 12. Comparison of energy consumption for a receiver.

8.2 Worst case analysis of the algorithm

The worst case for our algorithm arises when a receiver has to stay awake till the end of the NATIM window. In this section, we only compare our algorithm with SPAN by calculating the expected sleeping time of a receiver at a beaconing period. Here we show that our algorithm's performance is equal to Span's in the worst case.
Let's first define the notation. At the n^{th} beacon for a receiver,

1. A_n represents the arrival of a packet,
2. X_n represents the sleeping time,
3. S_n represents the state, where state = *sleep*, *awake*,
4. $P[A_n = 1] = p$ and $[P[A_n = 1] = 1 - p$, where p is the probability that a packet arrives,

Then, sleeping time formula for our algorithm is as follow;

$$X_n = \begin{cases} \text{beacon-NATIM}, & \text{if } A_n = 1 \wedge S_n = awake; \\ \text{beacon-ATIM}, & \text{if } A_n = 0 \wedge S_n = awake;; \\ \text{beacon}, & \text{otherwise}; \end{cases}$$

The sleeping time formula for Span is as follow;

$$X_n = \begin{cases} \text{beacon - NATIM}, & \text{if } A_n = 1 \\ \text{beacon - ATIM}, & \text{otherwise}; \end{cases}$$

Let's update the state transition graph of our according to the previous information. New state transition graph makes it easier to estimate the sleeping time for a given beacon period(see Fig.13).

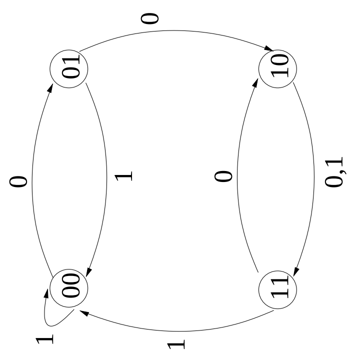

Figure 13. The detailed state transition diagram.

Let's convert new state transition graph to state transition diagram i.e. it includes the probabilities of the transitions for a given p (see Fig.14).

From the above state transition graph, it is easy to build the transition probability matrix P;

$$\begin{array}{c|cccc|} & 00 & 01 & 10 & 11 \\ 00 & p & 1\text{-}p & 0 & 0 \\ 01 & p & 0 & 1\text{-}p & 0 \\ 10 & 0 & 0 & 0 & 1 \\ 11 & p(1\text{-}p) & 0 & (1-p)^2 & 0 \end{array}$$

Let $\Pi = [\pi_{00}, \pi_{01}, \pi_{10}, \pi_{11}]$ be the vector of long-run probabilities for the states. This vector can

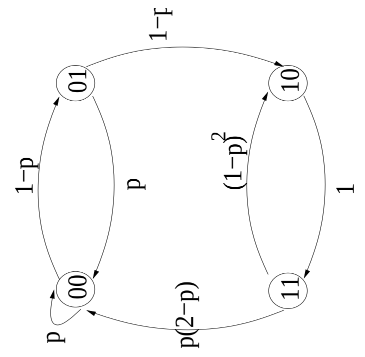

Figure 14. The state transition graph.

be found for a given transition matrix P as the unique stochastic vector solution to the eigenvector equation:

$$\Pi = \Pi . P$$

When we solve the above equetion, we get the following state probabilities;

$$\begin{aligned}
\pi_{00} &= \frac{p(2-p)}{p^3-2p+2} \\
\pi_{01} &= \frac{p(1-p)(2-p)}{p^3-2p+2} \\
\pi_{10} &= \frac{(1-p)^2}{p^3-2p+2} \\
\pi_{10} &= \frac{(1-p)^2}{p^3-2p+2}
\end{aligned} \tag{1}$$

It is easy to observe that the reciver node sleeps the following amount of time during a beaconing period;

$$\text{beacon-NATIM if state is 00} \tag{2}$$
$$\text{beacon-ATIM if state is 01 or 10} \tag{3}$$
$$\text{beacon if state is 11} \tag{4}$$
$$\tag{5}$$

Hence for our algorithm the expected sleeping time formula for a given packet arrival probability p can be calculated as;

$$\begin{aligned}
E[X_n] = \; & beacon \\
& - \frac{p(2-p)}{p^3-2p+2} NATIM \\
& - \frac{2(1-p)^2}{p^3-2p+2} ATIM
\end{aligned} \tag{6}$$

And, for Span the formula is;

$$E[X_n] = \; beacon - p.NATIM - (1-p)ATIM \tag{7}$$

Fig.15 shows the graph of expected sleeping time of a receiver according to SPAN and our algorithm for various packet arrival probabilities.

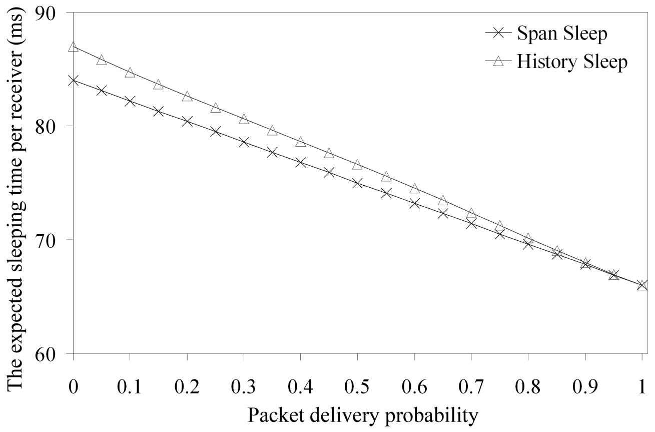

Figure 15. Comparison of expected sleeping time in a beacon.

Sender nodes are affected by our algorithm, if

1. the immediate sender node cannot get an ”HELLO” message from the receiver node, and
2. it has a packet to the receiver.

Then the receiver has to wake up the following beacon to check if the neighbor is still there. The overhead of this process is the receiver has to spend at most one more ATIM time. Let q be the probability that sender node has a packet for its immediate receiver, then the following formula gives the state probability that sender has to spend an extra ATIM.

$$\pi_{miss} = \frac{q\frac{(1-p)^2}{p^3-2p+2}}{1+q+q\frac{(1-p)^2}{p^3-2p+2}}, \text{where } q =< p$$

Then, we calculate the possible maximum value as $\pi_{miss} = 0.058$, and the expected power usage value of this overhead is 1.38 mW per beacon.

9 Conclusion

In this paper we have extended Span’s energy saving mode by using history and variable NATIM time.

Although, using CDS improves both the network's and node's life span, the gap between sleeping energy consumption and idle mode need some more research.

Wu and Li [10] proposed a distributed algorithm for approximating connected dominating sets in ad hoc networks that also appears to preserve the capacity. In a later paper with [7], they advance their algorithm by adding new rules and refining the existence ones. Our algorithm, however, elects fewer coordinators because it actively checks the redundancy locally whenever there's a local topological change. Therefore, in this paper, we also have established a better CDS forming algorithm for ad hoc networks, and proved it. Using a similar method of [1], we established an approximation factor for our algorithm.

A further research topic could be focusing on more in-depth simulation under different settings to see the effects of CDS construction algorithm in conjunction with variable sleeping time.

REFERENCES

[1] P. Wan, K.M. Alzoubi, O. Frieder, "Distributed Construction of Connected Dominating Set in in Wireless Ad Hoc Networks", in Proc. INFOCOM, 2002, pp. 1597-1604.

[2] Aleksi Penttinen, "Research On Ad Hoc Networking: Current Activity And Future Directions", http://keskus.hut.fi/opetus/s38030/k02/Papers/13-Aleksi.pdf

[3] B. N. Clark, C. J. Colbourn, and D. S. Johnson, Umt Disk Graphs, D F Crete Mrthemchc , X6: 165-177, 1990.

[4] B. Chen, K. Jamieson, H. Balakrishnan, R. Morris, "Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks", in Proc. ACM International Conference on Mobile Computing and Networking, Italy, July 2001

[5] David B Johnson and David A Maltz, "Dynamic Source Routing in Ad Hoc Wireless Networks", Mobile Computing, Kluwer Academic Publishers, 1996.

[6] C. Perkins, "Ad Hoc On Demand Distance Vector (AODV) Routing", draft-ietf-manet-aodv-00.txt, November 1997.

[7] J. Wu, M. Gao, I. Stojmenovic, "On calculating power-aware connected dominating sets for efficient routing in ad hoc wireless networks", Journal of Communication and Networks, March 2002.

[8] B. Das, V. Bharghavan, "Routing in ad-hoc networks using minimum connected dominating sets", in IEEE International Conference on Communications, Montreal, Canada. June 1997.

[9] B. Das, E. Sivakumar, and V. Bhargavan, "Routing in ad-hoc networks using a spine", in IEEE International Conferenceon Computer Communications and Networks, Las Vegas, 1997.

[10] Wu, J., and Gao, M. "On Calculating Power-Aware Connected Dominating Sets for Efficient Routing in Ad Hoc Wireless Networks", in Proc. of the 30th Annual International Conference On Parallel Processing,Valencia, Spain. Sept. 2001.

[11] I. Stojmenovic, M. Seddigh, J. Zunic, "Dominating sets and neighbor elimination-based broadcasting algorithms in wireless networks", IEEE Transactions on Parallel and Distributed Systems, 2002, pp. 14-25.

[12] Y. Xu, J. Heidemann, and D. Estrin, "Geography-informed Energy Conservation for Ad Hoc Routing", in MobiCom'2001.

[13] George Kao, Thomas Fevens and Jaroslav Opatrny, "Position-Based Routing on 3-D Geometric Graphs in Mobile Ad Hoc Networks", Proceedings of the 17th Canadian Conference on Computational Geometry (CCCG'05), pp.88-91.